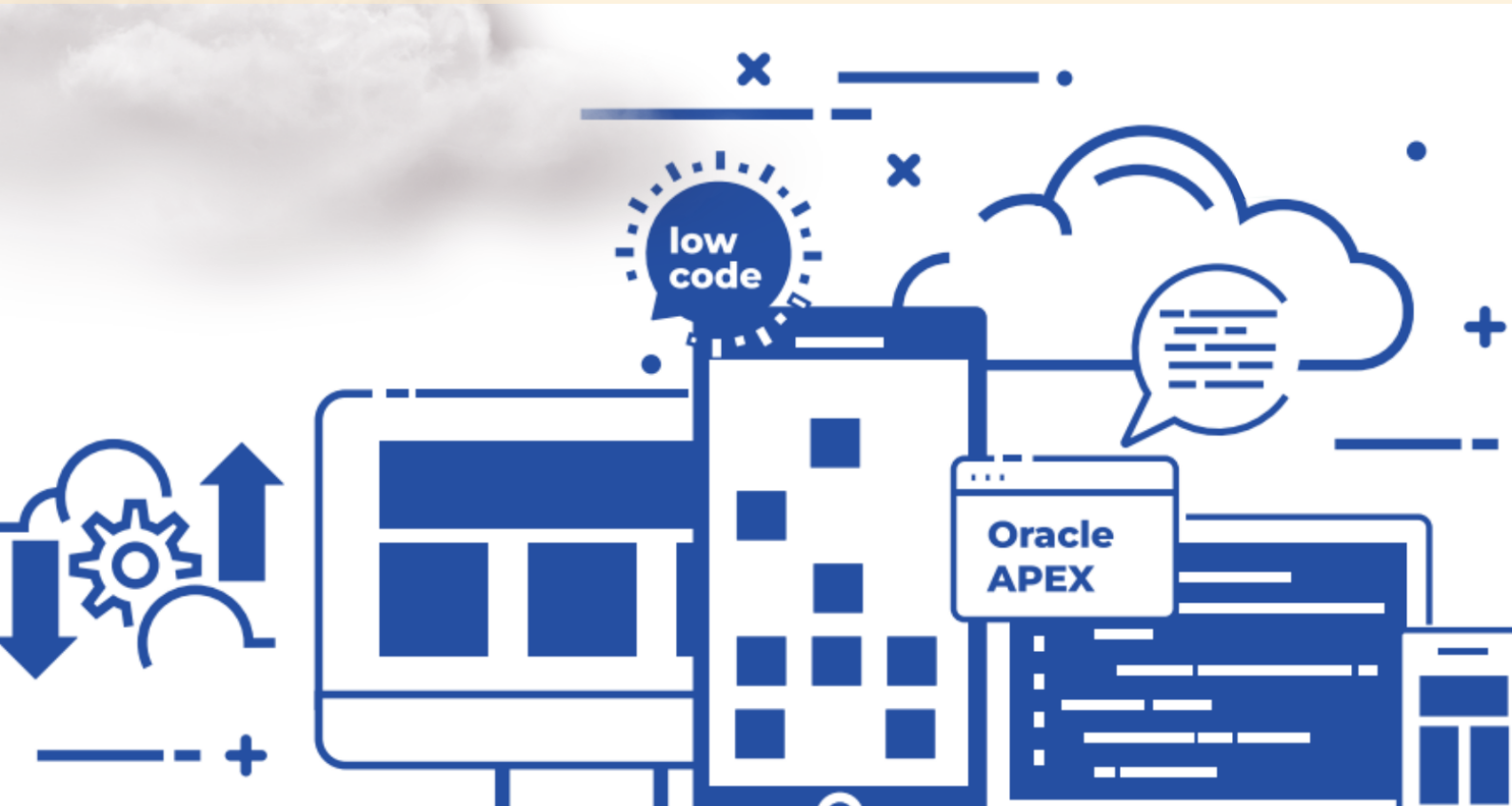
Funded by the European Union

BEE WITH APEX

E-knjiga
Malokodno programiranje z APEX-om

Praktični vodič za izboljšanje zaposljivosti

Projektna skupina BeeAPEX



Avtorske pravice © 2023 Projektna skupina BeeAPEX

IZDAJATELJ: LABORATORIJ ZA KAKOVOST IN TESTIRANJE PROGRAMSKE OPREME, UM FOV

BEEAPEX.EU

To je licencirana vsebina po Creative Commons Attribution-NonCommercial 4.0 License (v nadaljevanju "Licenca"). Vsebino je dovoljeno uporabljati izključno v skladu z Licenco. Kopija Licence je dostopna na povezavi https://creativecommons.org/licenses/by-nc-sa/4.0. Razen v primeru, da obstaja zakonska ali pogodbena obveza, je programska oprema po tej Licenci distribuirana po načelu "TAKA, KO JE", BREZ KAKRŠNIHKOLI ZAGOTOVIL ALI POGOJEV, bodisi, da so ti izraženi implicitno ali ekspilicitno. Preberite Licenco za specifična jezikovna dovoljenja in omejitve.

1. izdaja, 7. december 2023

Kazalo

Zahvala 27

Predgovor 29

I Kako delate v APEX-u?

1 Kako začnete z Oracle APEX? 35

Vjeran Strahonja, Dijana Oreški; prevod A. Baggia

1.1 Kaj je Oracle APEX in zakaj ga uporabljamo? 35
1.1.1 Kaj je Oracle APEX? 35
1.1.2 Kaj je aplikacija? 35
1.1.3 Kako Oracle APEX pomaga pri razvoju aplikacij? 35
1.1.4 Cikel razvoja aplikacij 36
1.2 Kako začeti z Oracle APEX? 42
1.2.1 Kakšen je nivo vašega znanja? 42
1.2.2 Uporaba lokalno nameščene instance Oracle APEX 43
1.2.3 https://apex.oracle.com 43
1.2.4 Virtual Box Appliance / Virtualni računalnik 43
1.2.5 APEX docker 44
1.2.6 APEX instanca v Oracle Cloud Infrastructure 46
1.2.7 APEX instanca v Oracle Academy 46
1.3 Vprašanja 46
1.4 Odgovori 46

2 Kako pripravite bazo podatkov? 49

E. Kapsammer, W. Retschitzegger, W. Schwinger; prevod U. Rajkovič

2.1 Kako modelirati podatke 50
2.1.1 Logični model podatkov 50
2.1.2 Relacijski model podatkov 54
2.1.3 Normalizacija 59

2.2 Kako upravljati shemo relacijske BP - SQL-DDL . . . 60
2.2.1 Samodejno generiranje tabel z uporabo ODM . . . 61
2.2.2 Samodejno generiranje tabel z uporabo Quick SQL . . . 63
2.2.3 Ročno ustvarjanje tabel z uporabo SQL-DDL . . . 65
2.2.4 Ročno upravljanje tabel z orodjem ORACLE Object Browser . . . 67
2.3 Kako upravljati podatke - SQL-DML . . . 68
2.3.1 Upravljanje podatkov z uporabo SQL-DML . . . 68
2.3.2 Upravljanje podatkov z ORACLE Object Browser (OB) . . . 69
2.3.3 Upravljanje podatkov z uporabo Quick SQL . . . 69
2.4 Kako poizvedovati po podatkih - SQL-DQL . . . 70
2.4.1 Poizvedovanje po podatkih z uporabo SQL-DQL . . . 70
2.4.2 Poizvedovanje po podatkih z uporabo orodja ORACLE Query Builder (QB) . . . 75
2.5 Izgradnja podatkovne plasti - širša slika . . . 78
2.6 Vprašanja . . . 78
2.7 Odgovori . . . 79

3 Kako navigirate v APEX-u? . . . 81
E. Kapsammer, W. Retschitzegger, W. Schwinger; prevod U. Rajkovič . . .

3.1 Uvod . . . 81
3.2 Proces razvoja spletne aplikacije z APEX . . . 81
3.3 Začetni meni programa APEX . . . 82
3.4 Pregled App Builderja - ustvarjanje in upravljanje aplikacij . . . 83
3.5 Izdelava aplikacije - trije primeri uporabe . . . 84
3.6 Izdelava aplikacije - lastnosti, strani, funkcije in nastavitve . . . 85
3.7 Izbera vrste strani . . . 87
3.8 Vzdrževanje in spreminjanje strani - Page Designer . . . 89
3.9 Zagon aplikacije ali strani . . . 89
3.10 Vprašanja . . . 91
3.11 Odgovori . . . 91

4 Kako izmenjujete podatke v APEX-u? . . . 93
E. Kapsammer, W. Retschitzegger, W. Schwinger; prevod A. Baggia . . .

4.1 Uvoz in izvoz podatkov z “Data Workshop” . . . 94
4.1.1 Uvoz podatkov . . . 94
4.1.2 Koraki uvoza . . . 96
4.1.3 Izvoz podatkov . . . 100
4.2 Uvoz in izvoz podatkov z “Object Browser” . . . 101
4.3 Izvoz rezultata SQL poizvedbe . . . 102
4.4 Izvoz podatkov iz poročila . . . 103
4.5 Izmenjava podatkov z RESTful Services . . . 104
4.5.1 REST Arhitekturni vzorec . . . 105
4.5.2 Nastavitve sheme baze podatkov za RESTful dostop . . . 106
4.5.3 Moduli | Predloge | Upravljavci virov . . . 107
4.5.4 Uporaba “AutoREST” . . . 109

4.6 Vprašanja **110**
4.7 Odgovori **111**

5 Kako izdelate prvi osnutek aplikacije? **113**
A. Angeioplastis, G. Myllis, A. Tsimpiris, D. Varsamis; prevod A. Baggia

5.1 Zakaj potrebujemo aplikacije? **113**
5.2 Nastavitev okolja v ORACLE APEX **113**
5.3 Priprava primera v SQL Workshop **117**
5.3.1 Quick SQL 120
5.3.2 SQL Script 121
5.4 Podatki v aplikaciji **124**
5.5 Upravljanje z aplikacijo **127**
5.6 Nadzor dostopa **127**
5.7 Dopolnilno študijsko gradivo **133**
5.7.1 Izvožena aplikacija 133
5.7.2 Video vodiči 133
5.8 Vprašanja **133**
5.9 Odgovori **134**

6 Kako uredite poročila? **135**
Zuzana Žillová, Erik Malina, Matej Grochal, Andrej Staník, Andrea Meleková, Michal Kvet, Miroslav Potočár; prevod A. Baggia

6.1 Poročilo **135**
6.2 Klasično poročilo **144**
6.3 Poročilo s preklopom stolpcev **146**
6.4 Interaktivno poročilo **146**
6.5 Vprašanja **155**
6.6 Odgovori **155**

7 Kako uredite obrazce? **157**
Veronika Šalgová, Michal Kvet, Miroslav Potočár; prevod A. Baggia

7.1 Vrste obrazcev? **157**
7.2 Interaktivna mreža **157**
7.3 Obrazec na osnovi tabele **160**
7.4 Glavni obrazec s podrobnostmi **160**
7.5 Vprašanja **163**
7.6 Odgovori **163**

8 Kako poročila spremenite v grafikone? **165**
Ivan Pastierik, Michal Kvet, Miroslav Potočár; prevod A. Baggia

8.1 Grafikon **165**
8.2 Izdelava stolpičnega grafikona **166**

8.3 Dodajanje filtrov na stolpični grafikon . . . 169
8.4 Razvrščanje v stolpičnemu grafikonu . . . 174
8.5 Izdelava različnih grafikonov . . . 177
8.6 Vprašanja . . . 179
8.7 Odgovori . . . 180

9 Kako urejate menije? . . . 181
Veronika Šalgová, Michal Kvet, Miroslav Potočár; prevod A. Baggia . . .

9.1 Kako urejamo menije . . . 181
9.2 Stranski meni . . . 181
9.3 Meni na vrhu strani . . . 182
9.4 Veliki meni . . . 182
9.5 Urejanje menijskih seznamov . . . 182
9.6 Vprašanja . . . 183
9.7 Odgovori . . . 183

10 Kako sodelujete v timu? . . . 185
P. Staniszewski, M. Sońta, A. Kierzkowski; prevod R. Leskovar . . .

10.1 Sodelovalno ustvarjanje znanja je bistvo razvoja malokodnih rešitev . 185
10.2 Povezanost preko spleta . . . 185
10.3 Tehnološka podkovanost za sodelovanje . . . 185
10.4 Opis funkcij . . . 187
10.4.1 Zaklepanje strani . . . 187
10.4.2 Komentarji . . . 187
10.4.3 Opcije za gradnjo aplikacije . . . 188
10.4.4 Razvoj v timu . . . 189
10.4.5 Odziv . . . 192
10.4.6 Standardizacija . . . 193
10.5 Zaključki . . . 194
10.6 Vprašanja . . . 195
10.7 Odgovori . . . 195

11 Kako izkoristite galerijo aplikacij in vtičnikov? . . . 197
Vjeran Strahonja, Dijana Oreški; prevod R. Leskovar . . .

11.1 Kako namestite vzorčne in zagonske aplikacije? . . . 197
11.2 Zagonske aplikacije . . . 198
11.3 Vzorčne aplikacije . . . 199
11.4 Vtičniki, ang. Plug-ins . . . 201
11.5 Vprašanja . . . 203
11.6 Odgovori . . . 204

12 Kako upravljate paketne in večjezične aplikacije? . . . 205
R. Leskovar, U. Rajkovič, A.Baggia; prevod U. Rajkovič

12.1 Aplikacija in paketna aplikacija . . . 205
12.2 Aplikacija . . . 206
12.2.1 Področje uporabe . . . 206
12.2.2 Ustvarjanje tabel . . . 206
12.2.3 Vstavljanje podatkov . . . 207
12.2.4 Ustvarjanje aplikacije . . . 208
12.3 Paketna aplikacija . . . 212
12.4 Večjezična aplikacija . . . 216
12.5 Dodatno učno gradivo . . . 220
12.5.1 Izvožene aplikacije . . . 220
12.5.2 Video vodiči . . . 220
12.6 Vprašanja . . . 221
12.7 Odgovori . . . 221

II Razvoj aplikacije v APEX-u

13 Intranetne novice za zaposlene . . . 225
R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

13.1 Poslovni vidik primera . . . 225
13.2 Definicija problema . . . 225
13.3 Primeri uporabe . . . 226
13.3.1 Opis primerov uporabe . . . 226
13.3.2 Delno strukturiran opis . . . 226
13.3.3 Diagram primera uporabe . . . 226
13.4 Model podatkov . . . 230
13.4.1 Opis modela podatkov . . . 230
13.4.2 Logični model podatkov . . . 230
13.4.3 Relacijski model podatkov . . . 230
13.5 Uporabniški vmesniki . . . 231
13.6 Dopolnilno študijsko gradivo . . . 234
13.6.1 Izvožena aplikacija . . . 234
13.6.2 Video vodiči . . . 234
13.7 Vprašanja . . . 234
13.8 Odgovori . . . 234

14 GreenDi - katalog rastlin . . . 237
Vjeran Strahonja, Dijana Oreški, Darko Andročec, Ana Kutnjak; prevod R. Leskovar

14.1 Definicija problema . . . 237
14.2 Primeri uporabe . . . 237
14.2.1 Opis primera uporabe . . . 238
14.2.2 Delno strukturiran opis . . . 238
14.2.3 Delno strukturiran opis . . . 238

14.2.4 Diagram primera uporabe . . . 238
14.3 Model podatkov . . . 240
14.3.1 Opis modela podatkov . . . 240
14.3.2 Logični model podatkov . . . 240
14.3.3 Relacijski model podatkov . . . 241
14.4 Uporabniški vmesniki . . . 242
14.5 Dopolnilno študijsko gradivo . . . 243
14.5.1 Izvožena aplikacija . . . 243
14.5.2 Video vodiči . . . 244
14.6 Vprašanja . . . 244
14.7 Odgovori . . . 244

15 GreenDi - avtorizacija in upravljanje uporabnikov . . . 245
VJERAN STRAHONJA; PREVOD R. LESKOVAR . . .
15.1 Poslovni vidik primera . . . 245
15.2 Definicija problema . . . 245
15.3 Primeri uporabe . . . 246
15.3.1 Opis primera uporabe . . . 246
15.3.2 Delno strukturiran opis . . . 246
15.3.3 Diagram primera uporabe . . . 246
15.4 Model podatkov . . . 246
15.4.1 Opis modela podatkov . . . 246
15.4.2 Logični model podatkov . . . 246
15.4.3 Relacijski model podatkov . . . 248
15.5 Uporabniški vmesniki . . . 248
15.6 Dopolnilno študijsko gradivo . . . 249
15.6.1 Izvožena aplikacija . . . 249
15.6.2 Video vodiči . . . 249
15.7 Vprašanja . . . 249
15.8 Odgovori . . . 249

16 Sistem malih inovacij . . . 251
R. LESKOVAR, U. RAJKOVIČ, A.BAGGIA; PREVOD R. LESKOVAR . . .
16.1 Poslovni vidik primera . . . 251
16.2 Definicija problema . . . 251
16.3 Primeri uporabe . . . 251
16.3.1 Opis primera uporabe . . . 251
16.3.2 Delno strukturiran opis . . . 252
16.3.3 Diagram primera uporabe . . . 252
16.4 Model podatkov . . . 252
16.4.1 Opis modela podatkov . . . 252
16.4.2 Logični model podatkov . . . 252
16.4.3 Relacijski model podatkov . . . 256

16.5 Uporabniški vmesniki . . . 257
16.6 Dopolnilno študijsko gradivo . . . 259
16.6.1 Izvožena aplikacija . . . 259
16.6.2 Video vodiči . . . 260
16.7 Vprašanja . . . 260
16.8 Odgovori . . . 260

17 Vodenje poslovnih procesov . . . 261
R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

17.1 Poslovni vidik primera . . . 261
17.2 Priprava za delo z vtičnikom Flows for APEX . . . 264
17.2.1 Namestitev Flows for APEX . . . 264
17.2.2 Namestitev aplikacije Sample Process Flow . . . 264
17.2.3 Preberite in vadite na priloženih primerih . . . 264
17.3 Definicija problema . . . 265
17.4 Primeri uporabe . . . 265
17.4.1 Opis primera uporabe . . . 265
17.4.2 Delno strukturiran opis . . . 265
17.4.3 Diagram primera uporabe . . . 268
17.5 Model delovnega toka . . . 268
17.6 Model podatkov . . . 271
17.6.1 Opis modela podatkov . . . 271
17.6.2 Logični model podatkov . . . 271
17.6.3 Relacijski model podatkov . . . 271
17.6.4 Skript Quick SQL za generiranje skripta v SQL . . . 272
17.6.5 Sekvence in dve shranjeni funkciji . . . 273
17.7 Uporabniški vmesniki . . . 274
17.7.1 Seznam vrednosti (List of values) v deljenih komponentah (Shared Components) 275
17.7.2 Vtičniki v deljenih komponentah (Shared Components) . . . 276
17.7.3 Komercialist . . . 276
17.7.4 Vodja proizvodnje . . . 280
17.7.5 Vodja financ . . . 282
17.7.6 Izvršni direktor . . . 284
17.8 Povezovanje aplikacije in vtičnika Flows for APEX . . . 286
17.9 Definiranje uporabniških vlog (user roles) . . . 288
17.10 Testiranje in odpravljanje napak . . . 288
17.11 Dopolnilno študijsko gradivo . . . 289
17.11.1 Izvožena aplikacija . . . 289
17.11.2 Video vodiči . . . 290
17.12 Vprašanja . . . 291
17.13 Odgovori . . . 291

18 GreenDi – menjava rastlin in semen . . . 293
Vjeran Strahonja, Dijana Oreški, Darko Andročec, Ana Kutnjak; prevod R. Leskovar

18.1 Poslovni vidik primera . . . 293

18.2 Definicija problema . . . 293
18.3 Primeri uporabe . . . 294
18.3.1 Opis primera uporabe . . . 294
18.3.2 Delno strukturiran opis . . . 294
18.3.3 Diagram primera uporabe . . . 294
18.4 Model podatkov . . . 294
18.4.1 Opis modela podatkov . . . 294
18.4.2 Logični model podatkov . . . 296
18.4.3 Relacijski model podatkov . . . 296
18.5 Uporabniški vmesniki . . . 297
18.6 Dopolnilno študijsko gradivo . . . 298
18.6.1 Izvožena aplikacija . . . 298
18.6.2 Video vodiči . . . 298
18.7 Vprašanja . . . 298
18.8 Odgovori . . . 298

19 Sistem za ocenjevanje knjig . . . 299
A. Kutnjak, L. Hrustek, A. Baggia and R.Leskovar; prevod R. Leskovar . . .

19.1 Poslovni vidik primera . . . 299
19.2 Definicija problema . . . 299
19.3 Primeri uporabe . . . 300
19.3.1 Opis primera uporabe . . . 300
19.3.2 Delno strukturiran opis . . . 300
19.3.3 Diagram primera uporabe . . . 300
19.4 Model podatkov . . . 302
19.4.1 Opis modela podatkov . . . 302
19.4.2 Logični model podatkov . . . 302
19.4.3 Relacijski model podatkov . . . 302
19.4.4 Skript v SQL . . . 303
19.4.5 Skript Quick SQL . . . 304
19.5 Uporabniški vmesniki . . . 305
19.5.1 Administrator . . . 305
19.5.2 Uporabnik . . . 307
19.6 Definiranje vlog uporabnikov . . . 309
19.7 Dopolnilno študijsko gradivo . . . 309
19.7.1 Izvožena aplikacija . . . 309
19.7.2 Video vodiči . . . 310
19.8 Vprašanja . . . 310
19.9 Odgovori . . . 310

20 Materialna kosovnica in kalkulacija stroškov . . . 311
R. Leskovar, U. Rajković, A.Baggia; prevod R. Leskovar . . .

20.1 Poslovni vidik primera . . . 311
20.2 Definicija problema . . . 311

20.3 Primeri uporabe . . . 312
20.3.1 Opis primera uporabe . . . 312
20.3.2 Delno strukturiran opis . . . 312
20.3.3 Diagram primera uporabe . . . 315
20.4 Model podatkov . . . 315
20.4.1 Opis modela podatkov . . . 315
20.4.2 Vpeljava poslovnih pravil v model podatkov . . . 316
20.4.3 Logični model podatkov . . . 316
20.4.4 Relacijski model podatkov . . . 316
20.4.5 Objekti v APEX-u . . . 318
20.5 Uporabniški vmesniki . . . 321
20.6 Dopolnilno študijsko gradivo . . . 324
20.6.1 Izvožena aplikacija . . . 324
20.6.2 Video vodiči . . . 324
20.7 Vprašanja . . . 324
20.8 Odgovori . . . 324

21 Vodenje prehrane in diete . . . 327
A. Angeioplastis, G. Myllis, A. Tsimpiris, D. Varsamis; prevod R. Leskovar

21.1 Poslovni vidik primera . . . 327
21.2 Definicija problema . . . 327
21.3 Primeri uporabe . . . 327
21.3.1 Opis primera uporabe . . . 327
21.3.2 Delno strukturiran opis . . . 328
21.3.3 Diagram primera uporabe . . . 328
21.4 Model podatkov . . . 329
21.4.1 Opis modela podatkov . . . 329
21.4.2 Logični model podatkov . . . 329
21.4.3 Relacijski model podatkov . . . 330
21.4.4 Skript Quick SQL za generiranje skripta v SQL . . . 331
21.5 Uporabniški vmesniki . . . 334
21.5.1 Izdelava nadzorne plošče . . . 338
21.5.2 Razvijalci in uporabniki aplikacije . . . 342
21.6 Dopolnilno študijsko gradivo . . . 345
21.6.1 Izvožena aplikacija . . . 345
21.6.2 Video vodiči . . . 345
21.7 Vprašanja . . . 345
21.8 Odgovori . . . 346

22 Razporejanje uradnih ur . . . 347
J. Mańko, M. Sońta, R. Leskovar; prevod R. Leskovar

22.1 Poslovni vidik primera . . . 347
22.2 Definicija problema . . . 348
22.3 Primeri uporabe . . . 348
22.3.1 Opis primera uporabe . . . 348
22.3.2 Delno strukturiran opis . . . 349

22.3.3 Diagram primera uporabe . . . 349
22.4 Model podatkov . . . 351
22.4.1 Opis modela podatkov . . . 351
22.4.2 Logični model podatkov . . . 351
22.4.3 Relacijski model podatkov . . . 352
22.4.4 Skript v Quick SQL . . . 353
22.4.5 Skript v SQL za kreiranje tabel . . . 353
22.4.6 Pomoč pri pisanju poizvedb - Query builder . . . 353
22.5 Uporabniški vmesniki . . . 354
22.5.1 Uporabniški vmesniki za vodstvo . . . 355
22.5.2 Uporabniški vmesniki za študentsko pisarno . . . 355
22.5.3 Uporabniški vmesniki za študente . . . 357
22.5.4 Uporabniški vmesnik za predavatelje . . . 359
22.6 Dopolnilno študijsko gradivo . . . 362
22.6.1 Izvožena aplikacija . . . 362
22.6.2 Video vodiči . . . 364
22.7 Vprašanja . . . 364
22.8 Odgovori . . . 364

23 Primer telekomunikacijskih storitev . . . 365

Veronika Šalgová, Jozef Kostolný, Michal Mrena, Michal Kvet, Miroslav Potočár; prevod R. Leskovar . . .

23.1 Poslovni vidik primera . . . 365
23.2 Definicija problema . . . 365
23.3 Primeri uporabe . . . 365
23.3.1 Opis primera uporabe . . . 366
23.3.2 Delno strukturiran opis . . . 366
23.3.3 Diagram primerov uporabe . . . 366
23.4 Model podatkov . . . 368
23.4.1 Opis modela podatkov . . . 368
23.4.2 Logični model podatkov . . . 368
23.4.3 Relacijski model podatkov . . . 368
23.5 Preverjanje pristnosti uporabnika in uporabniške vloge . . . 369
23.6 Uporabniški vmesniki . . . 371
23.6.1 Zasnova aplikacije . . . 372
23.7 Skripti . . . 380
23.8 Ustvarjanje domače strani . . . 382
23.9 Ustvarjanje strani stranke . . . 383
23.10 Ustvarjanje upraviteljske strani . . . 385
23.11 Dopolnilno študijsko gradivo . . . 386
23.11.1 Izvožena aplikacija . . . 386
23.11.2 Video vodiči . . . 386
23.12 Vprašanja . . . 386
23.13 Odgovori . . . 387

24 Najem vozila . . . 389
A. Angeioplastis, G. Myllis, A. Tsimpiris, D. Varsamis; prevod R. Leskovar . . .
24.1 Poslovni vidik primera . . . 389
24.2 Definicija problema . . . 389
24.3 Primeri uporabe . . . 390
24.3.1 Opis primera uporabe . . . 390
24.3.2 Delno strukturiran opis . . . 390
24.3.3 Diagram primera uporabe . . . 390
24.4 Model podatkov . . . 392
24.4.1 Opis modela podatkov . . . 392
24.4.2 Logični model podatkov . . . 392
24.4.3 Relacijski model podatkov . . . 393
24.4.4 Skript v SQL . . . 393
24.5 Uporabniški vmesniki . . . 394
24.6 Dopolnilno študijsko gradivo . . . 398
24.6.1 Izvožena aplikacija . . . 399
24.6.2 Video vodiči . . . 399
24.7 Vprašanja . . . 399
24.8 Odgovori . . . 399
Literatura . . . 401
Članki . . . 401
Knjige . . . 401
Stvarno kazalo . . . 403

Slike

1.1 Cikel razvoja aplikacije. 36
1.2 Kanvas poslovnega modela ((ó)) 38
1.3 Osnovni koncepti diagrama primera uporabe. 39
1.4 Generalizacija / specializacija igralcev. 40
1.5 Meje sistema ter posebna tipa povezav "include" in "extend". 40
1.6 Diagram primera uporabe za sistem rezervacij v hotelu. 41

2.1 SUBP kot hrbtenica spletnih aplikacij. 49
2.2 Logični model (ER diagram) našega kadrovskega primera. 50
2.3 Upravljanje logičnega modela z ODM. 52
2.4 Upravljanje atributov z ODM. 53
2.5 Upravljanje povezav z ODM. 54
2.6 Shema relacijske BP in shema tabele. 55
2.7 Postopek generiranja sheme relacijske BP iz logičnega modela. 56
2.8 Samodejno generirana shema relacijske BP. 56
2.9 Elementi grafičnega uporabniškega vmesnika za upravljanje stolpcev v RM. . . . 58
2.10 Elementi grafičnega uporabniškega vmesnika za upravljanje RM. 58
2.11 Pregled korakov normalizacije. 59
2.12 Kategorije stavkov SQL. 60
2.13 Izvoz sheme relacijske BP v skripto DDL. 62
2.14 Nalaganje in izvajanje skripte SQL. 63
2.15 Dostop do Quick SQL prek SQL Workshop 64
2.16 Ustvarjena skripta SQL, ki temelji na Quick SQL. 65
2.17 SQL Command Editor. 66
2.18 Stavek SQL za ustvarjanje tabele `Departments`. 66
2.19 Stavek SQL za brisanje tabele `Departments`. 67
2.20 Stavek SQL za spreminjanje tabele `Departments` 67
2.21 Upravljanje tabel z ORACLE Object Browser. 67
2.22 Stavek SQL za vpisovanje novih podatkov v tabelo `Departments`. 68
2.23 Stavek SQL za povečanje plače vseh zaposlenih za 10% v tabeli `Employees`. . . . 68
2.24 Stavek SQL za povečanje plače zaposlenega s priimkom "Miller". 68
2.25 Stavek SQL hkrati spreminja več atributov v tabeli `Employees`. 68
2.26 Stavek SQL izbriše vse podatke iz tabele `Employees`. 69

2.27 Stavek SQL izbriše le nekatere podatke iz tabele Employees. 69
2.28 Vstavljanje naključnih podatkov s Quick SQL. 70
2.29 Stavek SQL, ki prikliče vse podatke iz tabele Employees. 70
2.30 Del rezultata poizvedbe za priklic vseh podatkov iz tabele EMPLOYEES. 71
2.31 Stavek SQL za priklic vseh podatkov iz tabele Employees, urejenih po plačah. . . 71
2.32 Del rezultata poizvedbe za priklic vseh podatkov iz tabele Employees, urejenih po plačah. 71
2.33 Stavek SQL za priklic izbranih podatkov iz tabele Employees. 72
2.34 Del rezultata poizvedbe za priklic izbranih podatkov iz tabele Employees. 72
2.35 Stavek SQL za priklic izbranih podatkov iz tabele Employees z uporabo funkcije TO_CHAR(). 72
2.36 Del rezultata poizvedbe za priklic izbranih podatkov iz tabele Employees z uporabo funkcije TO_CHAR(). 73
2.37 Stavek SQL za združevanje podatkov iz tabel Employees in Departments. 73
2.38 Del rezultata poizvedbe za združevanje podatkov iz tabel Employees in Departments. 74
2.39 Stavek SQL, ki prešteje vse zaposlene in izračuna vsoto njihovih plač iz tabele Employees. 74
2.40 Izpis števila zaposlenih in vsote plač iz tabele Employees. 74
2.41 Stavek SQL za grupiranje podatkov po plačah in razvrščanje rezultatov po številu zaposlenih z določeno plačo, na podlagi podatkov iz tabele Employees. 75
2.42 Rezultat grupiranja podatkov po plačah in razvrščanja rezultatov po številu zaposlenih z določeno plačo, na podlagi podatkov iz tabele Employees. 75
2.43 Prikaz programa ORACLE Query Builder ("QB"). 76
2.44 Uporaba programa Query Builder za oblikovanje poizvedb z uporabo Join. . . . 77
2.45 Pregled možnosti za upravljanje podatkovne plasti. 78

3.1 Splošni razvojni proces v APEX-u. 82
3.2 Domača stran delovnega okolja Oracle APEX. 83
3.3 Pregled v App Builder. 84
3.4 Čarovnik za ustvarjanje aplikacije: Trije načini ustvarjanja aplikacije. 85
3.5 Lastnosti in strani. 86
3.6 Lastnosti in nastavitve. 86
3.7 Domača stran aplikacije - pogled razvijalca. 87
3.8 Ustvarjanje strani. 88
3.9 Page Designer. 89
3.10 Zaganjanje celotne aplikacije. 90
3.11 Zaganjanje posamezne strani. 91

4.1 Izmenjava podatkov v Oracle APEX. 94
4.2 Uporaba Data Workshop za uvoz podatkov - dostop preko SQL Workshop. 95
4.3 Uporaba Data Workshop za uvoz podatkov - dostop preko App Builder. 96
4.4 Zagotavljanje vira podatkov. 97
4.5 Prenos podatkov v novo tabelo. 97
4.6 Prenos podatkov v obstoječo tabelo. 99
4.7 Izvoz (Odlaganje) podatkov. 100
4.8 Čarovnik za izvoz podatkov. 101
4.9 Uvoz / izvoz podatkov iz tabele s pomočjo Object Browser. 102
4.10 Izvoz podatkov, ki so rezultat SQL poizvedbe. 103
4.11 Izvoz podatkov iz poročila. 104
4.12 RESTful Services za izmenjavo podatkov – Osnovna arhitektura. 105

4.13 Shemi na bazi podaktov omogočimo RESTful dostop. 107
4.14 Povezave med različnimi gradniki ORDS RESTful storitev. 108
4.15 Primer RESTful Services za tabelo Employees. 109
4.16 Omogočanje REST storitve v Object Browser. 110

5.1 Odprite spletno stran in izberite možnost prijave. 114
5.2 Začetni koraki pridobivanja delovnega prostora v Oracle APEX. 114
5.3 Zaključni koraki pri zahtevi za dodelitev brezplačnega delovnega prostora v Oracle APEX. 115
5.4 Potrditev zaključka procesa. 115
5.5 Elektronsko sporočilo za aktivacijo računa. 116
5.6 Nastavitev novega gesla za delovni prostor v Oracle APEX. 116
5.7 Delovni prostor Oracle APEX. 117
5.8 Logični model baze podatkov za primer Running HR. 118
5.9 Relacijski podatkovni model za primer Running HR. 119
5.10 Vpišite Quick SQL kodo. 121
5.11 Zagon SQL skript. 123
5.12 Začetek procesa kreiranja aplikacije. 123
5.13 Pogovorno okno Create App from Script. 124
5.14 Spletna aplikacija je kreirana. 125
5.15 Okolje App builder. 125
5.16 Stran za prijavo v spletno aplikacijo. 126
5.17 Okolje razvite aplikacije. 126
5.18 Okolje za upravljanje z uporabniki. 127
5.19 Primer nadzora dostopa. 128
5.20 Dodajanje novih uporabnikov. 129
5.21 Dodajanje vlog uporabnikom. 130
5.22 Nastavitve dostopnih pravic do gumba Create. 131
5.23 Dostop in pravice vloge Contributor. 132
5.24 Dostop in pravice vloge Reader. 133

6.1 Izbira vrste strani. 136
6.2 Kreiranje strani s poročilom. 137
6.3 Prikazovanje. 137
6.4 Lastnosti regije. 138
6.5 Različni viri podatkov. 138
6.6 Izgled (ang. Layout). 139
6.7 Razdelek Appearance. 139
6.8 Razdelek Templates. 140
6.9 Razdelek Advanced. 140
6.10 Razdelek Header and Footer. 140
6.11 Razdelek Server-side Condition. 141
6.12 Določanje lastnosti Read Only. 141
6.13 Razdelek Security, Server Cache and Customization. 142
6.14 Razdelek Pagination. 142
6.15 Določanje števila naloženih prikazanih vrstic. 142
6.16 Nastavitev možnosti prenosa in tiskanja. 143
6.17 Osrednji del okna za oblikovanje strani. 143
6.18 Različne oblike stolpcev. 144
6.19 Nastavitve stolpca. 144

6.20 Lastnot prikaza Format Mask. 144
6.21 Prilagoditev možnosti prikaza končnemu uporabniku. 145
6.22 Možnost urejanja stolpca. 145
6.23 Gumb za prikaz spletne strani. 145
6.24 Primer klasičnega poročila. 146
6.25 Možnost spreminjanja vidnosti stolpca. 146
6.26 Interaktivno poročilo. 147
6.27 Iskalna plošča. 147
6.28 Iskanje. 148
6.29 Gumb Actions. 148
6.30 Izbira stolpcev za prikaz. 149
6.31 Filtriranje po stolpcih. 149
6.32 Filtriranje po vrsticah. 150
6.33 Možnosti dela s podatki. 150
6.34 Sortiranje podatkov. 151
6.35 Funkcija Aggregate. 151
6.36 Funkcija Flashback. 151
6.37 Funkcija Control Break. 152
6.38 Rezultat Control Break funkcije. 152
6.39 Funkcija označevanja. 152
6.40 Rezultat označevanja. 153
6.41 Omejevanje števila prikazanih vrstic na strani. 153
6.42 Graf v interaktivnem poročilu. 153
6.43 Funkcija Group By. 154
6.44 Shranjevanje poročila. 154
6.45 Pogovorno okno za prenos poročila. 155

7.1 Poročilo Interactive Grid s prikazom podatkov o kupcih. 157
7.2 Možnosti urejanja prikaza na interaktivni mreži. 158
7.3 Gumb za skrivanje stolpca. 158
7.4 Stolpci za prikaz. 158
7.5 Možnost zamrznitve stolpca. 159
7.6 Sortiranje podatkov v stolpcu. 159
7.7 Možnost združevanja podatkov Aggregate. 159
7.8 Vrste grafov. 160
7.9 Obrazec za vnos podatkov o kupcu 160
7.10 Primer obrazca s podobrazcem v zloženem načinu. 161
7.11 Oblika obrazca z podpobrazcem - Side by Side. 162
7.12 Oblika obrazca s podobrazcem - prva stran, Drill Down 162
7.13 Oblika obrazca s podobrazcem - druga stran, Drill Down. 163

8.1 Podatkovni model za prikazani primer razvoja aplikacije. 165
8.2 Vstavljanje regije za grafikone. 166
8.3 Izgled strani po vstavljanju regije za grafikone. 166
8.4 Drevesni pregled, ki kaže napako v elementu grafikona. 167
8.5 Nastavitev vrste o prodaji. 167
8.6 Rezultat poizvedbe SQL. 168
8.7 Grafikon, ki se prikaže ob zagonu aplikacije. 168
8.8 Spremenjen naziv x osi. 169

8.9 Pripravljen grafikon prodaje. 169
8.10 Dodajanje elementa Checkbox Group. 170
8.11 Nastavitev osnovnih lastnosti elementa Checkbox Group. 170
8.12 Nastavitve seznama vrednosti za element Checkbox Group. 171
8.13 Nastavljanje privzetih vrednosti za element Checkbox Group. 171
8.14 Povezovanje regije Sales Chart z elementom Checkbox Group. 172
8.15 Določanje dinamične aktivnosti za element P1_PRODUCTS_CHOICE. 172
8.16 Nastavitev osnovnih lastnosti dinamične aktivnosti ob spreminjanju. 173
8.17 Nastavitev lastnosti ob osveževanju. 173
8.18 Dinamična aktivnost za Sales Chart v drevesnem pogledu. 174
8.19 Aplikacija z vključenim filtrom za izbiro izdelkov. 174
8.20 Izbira in postavitev elementa Select List. 175
8.21 Regija Sales Chart z dodanim elementom Select List. 175
8.22 Nastavitev seznama vrednosti za element Select List. 175
8.23 Nastavitev statičnih vrednosti pri elementu Select List. 176
8.24 Dodajanje dinamične aktivnosti za osveževanje pri elementu Select List. 176
8.25 Nastavitev elementa Order By za regijo Sales Chart. 176
8.26 Nastavitev ukazov Order By za razvrščanje pri elementu Select List. 177
8.27 Aplikacija z vključenim razvrščanjem. 177
8.28 Spreminjanje vrste grafikona v obliko Line with Area. 178
8.29 Seznam vrednosti za filtriranje izpisa prodaje po letih pri elementu Select List. . 178
8.30 Nastavitev poizvedbe SQL ter povezovanje stolpcev v regiji Monthly Sales Chart. 179
8.31 Stran z grafičnim prikazom Monthly Sales Chart. 179

9.1 Razširjen stranski meni. 181
9.2 Meni z zavihki na vrhu strani. 182
9.3 Meni na vrhu strani. 182
9.4 Zgornji veliki meni. 182
9.5 Lastnosti navigacijskega menija. 183

10.1 Zaklepanje strani v APEX-u. 187
10.2 Zaklepanje strani, ki jo trenutno urejate. 187
10.3 Dodajanje komentarja. 188
10.4 Določitev opcije strani kot komentirana ali izvedljiva. 189
10.5 Funkcija Team Development. 190
10.6 Aktiviranje funkcije Team Development. 190
10.7 Prilagoditev funkcije Team Development. 191
10.8 Komunikacija o težavi v funkciji Team Development. 191
10.9 Dovoljenje za odzive uporabnikov (v *Application definition*. 192
10.10 Oddajanje odziva v aplikaciji. 193
10.11 Menu "Utilities". 194

11.1 Seznam vzorčnih aplikacij v APEX-u. 197
11.2 Zagonske aplikacije v galeriji. 198
11.3 Namestitev aplikacije Team Calender. 199
11.4 Vzorčne aplikacije. 200

12.1 Čarovnik za uvoz in izvoz aplikacije. 205
12.2 Pretvarjanje Quick SQL v ukaze SQL. 207
12.3 Skripta CH12INSERT vstavi podatke v tri tabele in izvede transakcijo. 208

12.4 Ustvarjanje aplikacije iz skripte. 208
12.5 Izbira imena aplikacije in lastnosti. 209
12.6 Izbira skupnih komponent. 209
12.7 Ustvarjanje seznama vrednosti od začetka. 210
12.8 Opredelitev imena in vrste: CH12_LOV_COMPETENCE_DESCRIPTION je dinamičen. 210
12.9 Vnos ukaza SQL SELECT. 210
12.10 Določimo ime in tip seznama vrednosti: CH12_LOV_COMPETENCE_LEVEL je statična (ang. static). 211
12.11 Določimo ime in tip seznama vrednosti: CH12_LOV_COMPETENCE_LEVEL je statična. 211
12.12 Izberite stran 7 (Ch12 Job Competences). 212
12.13 Izberemo stran 7 (Ch12 Job Competences). 212
12.14 Izberemo Supporting Objects. 213
12.15 Nastavljanje predpogojev. 213
12.16 Nastavitev preverjanja obstoja treh tabel. 213
12.17 Nastavitev poziva za preimenovanje aplikacije. 214
12.18 Nastavitev namestitvenih skript. 214
12.19 Nastavitev skript za odstranjevanje aplikacije. 214
12.20 Uvoz aplikacije v drug delovni prostor. 215
12.21 Licenčna pogodba. 215
12.22 Preimenovanje uvožene aplikacije. 216
12.23 Dodajanje vloge uporabniku. 216
12.24 Določitev jezikov za prevajanje. 217
12.25 Besedilo, ki ga je mogoče prevesti. 217
12.26 Izvoz nizov besedila za določen jezik in stran. 218
12.27 Prevod označenih "ciljnih" nizov v vrsticah 48, 52, 56, 60, 72 in 76. 218
12.28 Nalaganje prevajalskih datotek XLDIFF. 218
12.29 Udejanjanje sprememb in objava. 219
12.30 Končna objava prevoda aplikacije. 219
12.31 Nastavitev primarnega jezika aplikacije. 219
12.32 Prevedena stran. 219

13.1 Diagram primerov uporabe. 226
13.2 Logični model podatkov. 230
13.3 Relacijski model podatkov. 231
13.4 Pregledna plošča uporabnika z vlogo založnika. 231
13.5 Uporabniški vmesnik za objavljanje novic s prilogami. 232
13.6 Pregledna plošča uporabnika z vlogo bralca. 232
13.7 Uporabniški vmesnik za branje novic. 233
13.8 Pregledna plošča uporabnika z vlogo skrbnika. 233
13.9 Urejanje vlog uporabnikov. 234

14.1 Diagram primerov uporabe. 238
14.2 Logični model podatkov. 241
14.3 Relacijski model podatkov. 242
14.4 Interaktivno poročilo (interactive grid) in obrazec za rastline. 243
14.5 Public part – open browsing of plant data. 243

15.1 Diagram primera uporabe - avtorizacija in upravljanje uporabnikov na platformi GreenDi. 246

15.2 Logični model podatkov. 248
15.3 Relacijski model podatkov. 248
15.4 Zgodovina sprememb podatkov uporabnika. 249

16.1 Diagram primera uporabe. 252
16.2 Logični model podatkov. 256
16.3 Relacijski model podatkov. 256
16.4 Pregledna plošča za zaposlenega. 257
16.5 Oddajanje ideje s prilogami. 257
16.6 Organizacijska struktura podjetja. 258
16.7 Vpogled v oddane inovacijske ideje. 258
16.8 Pregledna plošča ocenjevalca. 259
16.9 Stran za ocenjevanje idej. 259

17.1 Integracija Flows for APEX z aplikacijo APEX. 263
17.2 Diagram primerov uporabe. 268
17.3 Delovni tok - obdelava povpraševanja - diagram BPMN. 268
17.4 Ustvarjanje primerka (instance) delovnega toka. 269
17.5 Ustvarjanje primerka delovnega toka CH17. 269
17.6 Zagon primerka delovnega toka. 269
17.7 Zaključek prve naloge v primerku delovnega toka. 270
17.8 Izvajanje nalog v paralelnih vratih. 270
17.9 Zaključek vseh nalog v primerku delovnega toka. 270
17.10 Logični model podatkov. 271
17.11 Relacijski model podatkov. 272
17.12 Domača stran aplikacije "CH17 Business Process Management". 275
17.13 Komercialist - začetek primerka delovnega toka. 276
17.14 Komercialist - poročilo o poteku. 276
17.15 Komercialist - poročilo o delovnem toku določenega primerka. 277
17.16 Komercialist - poročilo o delovnem toku za izbrani primerek. 277
17.17 Komercialist - poročilo o povpraševanjih. 278
17.18 Komercialist - seznam dokumentov. 278
17.19 Komercialist - nalaganje dokumenta za povpraševanje 17. 279
17.20 Komercialist - seznam naloženih dokumentov po osvežitvi. 279
17.21 Komercialist - poročilo za določen primerek po nalaganju treh dokumentov. 280
17.22 Komercialist - poročilo v delovnem toku za določen primerek po nalaganju treh dokumentov. 280
17.23 Vodja proizvodnje - poročilo o delovnem toku. 281
17.24 Vodja proizvodnje - diagram BPMN - stanje primerka. 281
17.25 Vodja proizvodnje - vnos ocene za povpraševanje. 281
17.26 Vodja proizvodnje - ocenjevanje zaključeno, primerek čaka na finančno oceno. 282
17.27 Finančni direktor - poročilo o delovnem toku. 282
17.28 Finančni direktor - diagram BPMN - stanje primerka. 283
17.29 Finančni direktor - vnos ocene za povpraševanje. 283
17.30 Finančni direktor - ocenjevanje zaključeno, primerek čaka na poslovno ocenjevanje. 284
17.31 Izvršni direktor - diagram BPMN- stanje primerka. 284
17.32 Izvršni direktor - poročilo o delovnem toku. 285
17.33 Izvršni direktor - vnos poslovne ocene. 285
17.34 Izvršni direktor - poslovno ocenjevanje je zaključeno, primerek je zaključen. . 286

17.35 Prikaz stanja primerka na diagramu BPMN. 287
17.36 Nastavitev polj na strani. 287
17.37 Dokončanje koraka v Flows for Apex. 288
17.38 Določitev vloge in uporabniške vloge v meniju Application Access Control. . 288
17.39 Dodelitev pravice "create job privilege" delovnemu prostoru. 289
17.40 Uvoz delovnega toka CH17 v Flows for APEX - korak 1. 290
17.41 Uvoz delovnega toka CH17 v Flows for APEX - korak 2. 290

18.1 Diagram primera uporabe - izmenjava rastlin in semen. 294
18.2 Logični model podatkov. 296
18.3 Relacijski model podatkov. 297
18.4 Interaktivno poročilo o ponudbah. 297
18.5 Obrazec za vnos sporočil. 297

19.1 Diagram primera uporabe. 300
19.2 Logični model podatkov. 302
19.3 Relacijski model podatkov. 303
19.4 Generiranje aplikacije s pomočjo skripta CH19CREATEINSERT - korak 1. 305
19.5 Generiranje aplikacije s pomočjo skripta CH19CREATEINSERT - korak 2. 306
19.6 Skrbnik - dodajanje knjige. 307
19.7 Skrbnik - dodajanje kategorije. 307
19.8 Registracija uporabnika. 308
19.9 Brskanje po knjigah in dodajanje recenzije. 308
19.10 Obrazec za komentiranje recenzije. 309

20.1 Diagram primera uporabe. 315
20.2 Logični model podatkov. 316
20.3 Definiranje unikatnega indeksa v Oracle SQL Data Modeler. 317
20.4 Relacijski model podatkov. 317
20.5 Generiranje skripta SQL s pomočjo orodja Quick SQL. 319
20.6 Domača stran aplikacije. 322
20.7 Upravljanje osnovnih podatkov - poročilo. 322
20.8 Upravljanje osnovnih podatkov - obrazec. 322
20.9 Upravljanje strukturnih podatkov - poročilo. 323
20.10 Upravljanje strukturnih podatkov - obrazec. 323
20.11 Stran za izračun materialen kosovnice. 324

21.1 Diagram primera uporabe. 329
21.2 Logični model podatkov. 330
21.3 Relacijski model podatkov. 331
21.4 Quick SQL in kreiranje skriptav v SQL. 332
21.5 Primer izdelave in izvedbe skripta. 334
21.6 Čarovnik za kreiranje nove aplikacije. 335
21.7 Aplikacija z nazivom "Nutrition application". 336
21.8 Stran z recepti. 336
21.9 Stran s sestavinami. 337
21.10 Stran s sestavinami recepta. 337
21.11 Stran z izračunom prehranskih vrednosti v receptu. 338
21.12 Ustvarjanje nadzorne plošče. 339
21.13 Prilagoditev vsebine in izgleda nadzorne plošče. 340

21.14 Določitev poizvedbe v SQL za polje Source. 342
21.15 Ustvarjanje novega uporabniškega računa. 343
21.16 Kreiranje uporabnškega računa in določitev vlog v sami aplikaciji. 344
21.17 Dodelitev dostopa do strani samo določeni skupini (vlogi) uporabnikov. 345

22.1 Diagram primerov uporabe. 351
22.2 Logični model podatkov. 352
22.3 Relacijski model podatkov. 352
22.4 Model podatkov, ki je opisan s skriptom Quick SQL. 353
22.5 Primer uporabe funkcije Query Builder. 354
22.6 Domača stran aplikacije. 355
22.7 Nadzorna plošča za vodstvo. 355
22.8 Interaktivno poročilo o uradnih urah za študentsko pisarno. 356
22.9 Obrazec za študentsko pisarno - prerazporejanje uradnih ur. 356
22.10 Interaktivno poročilo o uradnih urah za študentsko pisarno po prerazporeditvi. 357
22.11 Koledar študentovih prijav na uradne ure. 357
22.12 Koledar ponujenih uradnih ur vseh učiteljev vpisanega študijskega programa. 358
22.13 Prijava na uradne ure preko koledarja - izbira koledarskega termina. 358
22.14 Prijava na uradne ure preko koledarja - izbira namena. 358
22.15 Prijava na uradne ure preko koledarja - prikaz uspešne prijave. 359
22.16 Preverjanje prijave z menujsko izbiro "View my appointments". 359
22.17 Detajlno interaktivno poročilo vseh študentovih uradnih ur. 359
22.18 Detajno poročilo o učiteljevih uradnih urah. 360
22.19 Učiteljev koledar uradnih ur z vidnimu imeni študentov, njihovimi jeziki in nameni prijave. 360
22.20 Prerazporeditev uradnih ur z nič prijavljenimi - pogled na koledarju. 361
22.21 Prerazporeditev uradnih ur z nič prijavljenimi - vnos novih datumov. 361
22.22 Rezultat prerazporeditve uradnih ur z nič prijavljenimi - pogled na koledarju. 361

23.1 Diagram primerov uporabe. 367
23.2 Logični model podatkov. 368
23.3 Relacijski model podatkov. 369
23.4 Stranka - pregledna plošča. 373
23.5 Stranka - pregledna plošča – modalno okno računa. 374
23.6 Stranka - pregledna plošča - račun v obliki PDF. 374
23.7 Stranka - pregledna plošča - statistika porabljenih minut. 374
23.8 Stranka - pregledna plošča - statistika SMS. 375
23.9 Stranka - pregledna plošča - statistika porabe prenosa podatkov. 375
23.10 Upravitelj - pregledna plošča – dodajanje stranke. 376
23.11 Upravitelj - pregledna plošča – upravljanje strank. 377
23.12 Upravitelj - pregledna plošča – pregled strank. 377
23.13 Upravitelj - pregledna plošča – izvoz podatkov o strankah. 378
23.14 Upravitelj - pregledna plošča – izvoz podatkov o strankah v preglednico. . . . 378
23.15 Skrbnik - pregledna plošča. 379
23.16 Pristajalna stran. 379
23.17 Prijavna stran. 379
23.18 Regije v telesu strani. 382
23.19 Seznam možnih vsebin v regiji. 382
23.20 HTML v statični regiji strani. 382

24.1 Diagram primera uporabe. 390
24.2 Logični model podatkov za projekt "Car Rental Project" 392
24.3 Relacijski model podatkov za projekt "Car Rental Project" 393
24.4 Prijava v aplikacijo. 395
24.5 Domača stran aplikacije. 395
24.6 Obrazec za avtomobile (Cars) 396
24.7 Nastavitev stolpca s fotografijo. 397
24.8 Tabela s podatki o strankah. 398
24.9 Podatki o najemu. 398



Tabele

13.1 Opis primera uporabe: objavljanje novice na intranetu. 227
13.2 Opis primera uporabe: branje intranetne novice 228
13.3 Opis primera uporabe: upravljanje intranetne aplikacije. 229

14.1 Opis primera uporabe: brskanje po katalogu rastlin 239

15.1 Opis primera uporabe: avtorizacija in upravljanje uporabnikov. 247

16.1 Opis primera uporabe: oddajanje ideje 253
16.2 Opis primera uporabe: pregledovanje idej. 254
16.3 Opis primera uporabe: ocenjevanje ideje. 255

17.1 Opis primera uporabe: priprava dokumentacije za povpraševanje. 266
17.2 Opis primera uporabe: ocenitev proizvodnega vidika povpraševanja. 266
17.3 Opis primera uporabe: ocenitev finančnega vidika povpraševanja. 267
17.4 Opis primera uporabe: ocenitev poslovnega vidika povpraševanja. 267

18.1 Opis primera uporabe: izmenjava rastlin in semen. 295

19.1 Opis primera uporabe: ocenjevanje knjig 301

20.1 Opis primera uporabe: poročanje in vzdrževanje osnovnih podatkov. 312
20.2 Opis primera uporabe: poročanje in vzdrževanje podatkov o sestavi. 313
20.3 Opis primera uporabe: izračun kosovnice. 314

21.1 Opis primera uporabe: vodenje receptov in sestavin ter izračun hranilnih vrednosti 328

22.1 Opis primera uporabe: prerazporeditev uradnih ur učitelja. 349
22.2 Opis primera uporabe: prijava študenta na uradne ure. 350

23.1 Opis primera uporabe: Dodaj storitev. 366
23.2 Opis primera uporabe: Prikaži stanje storitev. 367
23.3 Podatki, shranjeni v tabeli CH23_Person. 370

24.1 Opis primera uporabe: upravljanje kupcev in avtov 391

Zahvala

Projektni partnerji in projektni tim BEE APEX:

- Univerza Maribor Fakulteta za organizacijske vede,
- Univerza Zagreb Fakulteta za organizacijo in informatiko,
- Univerza Žilina,
- Univerza Kozminski,
- Mednarodna grška univerza in
- Univerza Johannes Kepler

se zahvaljujemo za finančno podporo European Commission z akcijo Erasmus + Better Employability for Everyone with APEX (projekt ID 2021-1-SI01-KA220-HED-000032218), ki ga financira program Erasmus+ programme Evropske unije.

Podpora Evropske komisije za izdelavo te publikacije ne pomeni odobritve vsebine. Ta odraža samo stališča avtorjev. Komisija ne more biti odgovorna za kakršno koli uporabo informacij, ki jih publikacija vsebuje.

Posebej se zahvaljujejemo tudi:

- gospodu *Darku Jurekoviću, programskemu vodji* Oracle Academy za njegovo nenehno pomoč pri širjenju rezultatov projekta in
- gospodu *Aljažu Maliju, direktorju podjetja* THE RIGHT THING SOLUTIONS za dragocene nasvete o APEX-u pred in med projektom.

Projektna skupina BeeAPEX



Predgovor

Dobrodošli pri raziskovanju Oracle Application Express (APEX) – intuitivne in zmogljive razvojne malokodne platforme za ustvarjanje podatkovno vodenih spletnih aplikacij. Ta učbenik je zasnovan z namenom, da vas obogati s kompetencami za izkoriščanje celotnega potenciala Oracle APEX in izdelavo vrhunskih aplikacij za reševanje resničnih poslovnih izzivov.

Del I v učbeniku je naslovljen "Kako delate v APEX-u?". Obravnava osnovne vidike tega orodja. V dvanajstih poglavjih boste potovali po vsebinah, ki vam bodo omogočile razumevanje jedrnih konceptov, pridobitev študijskega razvojnega okolja ter spoznavanje različnih funkcij pri gradnji robustnih aplikacij. Vsako poglavje obravnava specifično temo, zagotavlja jasna navodila ter vabi k preskusu za boljšo učno izkušnjo.

Poglavje 1: "Kako začnete z Oracle APEX?" razloži kaj je, za kaj se uporablja in več načinov za pripravo študijskega okolja za učenje in razvijanje aplikacij.

Poglavje 2: "Kako pripravite bazo podatkov?" podaja uvod v modeliranje podatkov, upravljanje baze podatkov, manipuliranje podatkov ter poizvedovanje. Za začetnike so predstavljeni vsi pomembno koncepti modeliranja podatkov, ki je obvezna veščina.

Poglavje 3: "Kako navigirate v APEX-u?" popelje bralca po funkcijah APEX-a, ki omogočajo razvoj aplikacij, generiranje in prilagajanje različnih spletnih strani in njihovih komponent. Izvajanje in testiranje aplikacije je samo en zavihek oddaljeno od razvojnega okolja.

Poglavje 4: "Kako izmenjujete podatke v APEX-u?" daje vpogled v zmogljivosti izvažanja in uvažanja podatkov. Poglavje prikazuje izmenjav z datotekami, preglednicami in tudi s pomočjo storitve RESTful.

Poglavje 5: "Kako izdelate prvi osnutek aplikacije?" vabi k preskusu razvojnih zmogljivosti platforme APEX. Ugotovili boste, da po odločitvi glede vaših podatkov, lahko nemudoma generirate prikupno in uporabno aplikacijo brez programiranja. Razloži vam kako kontrolirano dodeljujete dostopne pravice različnim vlogam uporabnikov s čarovnikom.

Poglavje 6: "Kako uredite poročila?" vodi skozi različne vpoglede v podatke od klasičnih poročil do interaktivnih poročil. APEX ima vgrajene funkcije, ki končnemu uporabniku omogočajo prilagajanje brez programiranje ali posredovanja razvijalca.

Poglavje 7: "Kako uredite obrazce?" vas uvede v tri splošne tipe spletnih obrazcev, vključno z zahtevnejšimi, ki imajo glavo in podrobnosti. Prilagajanje in generiranje spletnih obrazcev ne bo zahtevalo nobenega programiranja.

Poglavje 8: "Kako poročila spremenite v grafikone?" vas popelje po funkcijah APEX-a, ki

omogočajo prikaz podatkov v obliki različnih grafikonov naravnost iz besedilnih poročil.

Poglavje 9: “Kako urejate menije?” predstavlja različne navigacijske elemente v APEX-u.

Poglavje 10: “Kako sodelujete v timu?” daje vpogled v zmogljivosti APEX-a za podporo dela skupine, kajti zelo redko se zgodi, da na razvoju aplikacije dela zgolj en razvijalec.

Poglavje 11: “Kako izkoristite galerijo aplikacij in vtičnikov?” prikazuje moč APEX-a za ponovno uporabo dobrih vzorov.

Poglavje 12: “Kako upravljate paketne in večjezične aplikacije?” vas popelje na pot distribucije vaše aplikacije v drugo APEX-ovo okolje z uporabniki, ki govorijo druge jezike.

Del I pokriva tudi vsebine, ki so ključne za varnost aplikacij, strategije namestitev in pripravo za produkcijsko uporabo.

Del II tega učbenika vas popelje onkraj osnov in predstavi dvanajst privlačnih poslovnih primerov od besednega opisa, preko vseh tehničnih podrobnosti do rešitve - aplikacije. Vsak primer je skrbno dokumentiran, da zagotovi celovito razumevanje razvoja aplikacij z vidika poslovanja, podatkov in uporabniškega vmesnika. Ta del vključuje primere aplikacij za podjetja:

- intranetne novice za zaposlene,
- sistem malih inovacij,
- vodenje poslovnih procesov z delovnimi tokovi,
- kalkulacija materialne kosovnice,
- sistem za ocenjevanje knjig,
- vodenje prehrane in diete,
- razporejanje uradnih ur,
- zaračunavanje storitev v telekomunikacijskem podjetju,
- najemanje vozil

za skupine, društva:

- katalog rastlin,
- izmenjava rastlin

ter splošno uporabno avtorizacijo in upravljanje uporabnikov. V vsakem poslovnem primeru so vključeni:

- Poslovni pogled na primer: pregled poslovne situacije.
- Definicija problema: iskanje odgovora na kdo in zakaj ima nekdo glavobol.
- Primeri uporabe: predstavljene so tri vrste opisov: besedni, polstrukturirani in grafični za pripravo dokumentacije primerov uporabe v UML.
- Logični in relacijski podatkovni model: APEX ima vse funkcije za zagon aplikacije iz novih podatkovnih struktur, uporabo in spreminjanje obstoječih, združevanje z drugimi orodji za modeliranje podatkov in podporo za vnaprejšnjega ali obratnega inženiringa. Prizadevanja razvijalcev, da bi zagotovili ustrezne dele podatkov in povezave med njimi ter upoštevanje poslovnih potreb, so osnova za oblikovanje uporabniških vmesnikov.
- Aplikacijski vmesniki: učbenik ponuja HTML strani, poročila, obrazce, polja, menije, gumbe in hiperpovezave, ki materializirajo poslovno situacijo, rešitev poslovnega problema, primere uporabe in podatke z mislijo na končnega uporabnika.
- Dodatno učno gradivo: za izboljšanje, pospešitev in pomoč na razvojni poti boste našli povezave do izvoženih paketnih aplikacij, skriptov, podatkov in video vodičev za vsako poglavje. Ti viri vam bodo zagotovili praktične vpoglede, kar vam bo omogočilo, da okrepite svoje znanje in ga neposredno uporabite v projektih iz resničnega sveta.

Ne glede na to, ali ste izkušen razvijalec, ki želi še povečati svoje kompetence, ali začetnik, ki želi raziskati svet APEX-a, je ta učbenik vaš dober vodnik. Upamo, da ga bodo tudi tisti, ki se ne učijo IT, našli kot neprecenljivega spremljevalca na poti do obvladovanja Oracle APEX in gradnje inovativnih aplikacij, ki imajo pozitiven učinek.

Učbenik in dodatno študijsko gradivo so zasnovani za približno 75 ur napora študenta (3 ECTS). Upamo, da bodo omogočili različne načine študija:

- tečaj ali predmet, pri katerem učitelj predava in izvaja laboratorijske vaje,
- mešano učenje (to je del vodi učitelj, del samostojno preštudira študent) in tudi
- popolnoma samostojen študij.

Glede na osnovno znanje študentov in razpoložljiv čas za izvajanje tečaja/predmeta lahko učitelji enostavno sestavijo nabor poglavij, ki ustrezajo učnim situacijam, kot so: izvenštudijski tečaji/predmeti, poletne šole, časovno omejeni dogodki za predstavitve malokodnega pristopa za vse študente (ne samo v IT ali računalničarje) in praktično usposabljanje v različnih panogah industrije.

Za razvoj učbenika in dopolnilnih študijskih gradiv smo uporabljali verziji APEX-a 22 in 23. Avtorji smo prepričani, da bodo razloženi in uporabljeni koncepti ter jedrne tehnologije zelo koristile študentom tudi v prihodnjih verzijah APEX-a.
Uživajte v uporabi čarovnikov in malokodnega programiranja!

red. prof. dr. Robert Leskovar
vodja projekta BeeAPEX, predstojnik Katedre za informatiko Fakultete za organizacijske vede UM

Kako delate v APEX-u?

1 **Kako začnete z Oracle APEX?** 35
Vjeran Strahonja, Dijana Oreški; prevod A. Baggia

2 **Kako pripravite bazo podatkov?** 49
E. Kapsammer, W. Retschitzegger, W. Schwinger; prevod U. Rajkovič

3 **Kako navigirate v APEX-u?** 81
E. Kapsammer, W. Retschitzegger, W. Schwinger; prevod U. Rajkovič

4 **Kako izmenjujete podatke v APEX-u?** . 93
E. Kapsammer, W. Retschitzegger, W. Schwinger; prevod A. Baggia

5 **Kako izdelate prvi osnutek aplikacije?** 113
A. Angeioplastis, G. Myllis, A. Tsimpiris, D. Varsamis; prevod A. Baggia

6 **Kako uredite poročila?** 135
Zuzana Žillová, Erik Malina, Matej Grochal, Andrej Staník, Andrea Meleková, Michal Kvet, Miroslav Potočár; prevod A. Baggia

7 **Kako uredite obrazce?** 157
Veronika Šalgová, Michal Kvet, Miroslav Potočár; prevod A. Baggia

8 **Kako poročila spremenite v grafikone?** 165
Ivan Pastierik, Michal Kvet, Miroslav Potočár; prevod A. Baggia

9 **Kako urejate menije?** 181
Veronika Šalgová, Michal Kvet, Miroslav Potočár; prevod A. Baggia

10 **Kako sodelujete v timu?** 185
P. Staniszewski, M. Sońta, A. Kierzkowski; prevod R. Leskovar

11 **Kako izkoristite galerijo aplikacij in vtičnikov?** 197
Vjeran Strahonja, Dijana Oreški; prevod R. Leskovar

12 **Kako upravljate paketne in večjezične aplikacije?** 205
R. Leskovar, U. Rajkovič, A.Baggia; prevod U. Rajkovič



1. Kako začnete z Oracle APEX?

VJERAN STRAHONJA, DIJANA OREŠKI; PREVOD A. BAGGIA

1.1 Kaj je Oracle APEX in zakaj ga uporabljamo?

1.1.1 Kaj je Oracle APEX?

Oracle APEX (znan tudi kot APEX ali Oracle Application Express) je razvojna platforma za malokodno programiranje, ki jo je razvila korporacija Oracle. Oracle APEX uporabnikom omogoča razvoj in uporabo aplikacij prek spletnega vmesnika. Združuje nabor orodij za ustvarjanje različnih vrst aplikacij, ki jih sestavljajo programi za delo s podatki v bazi podatkov, vključno z vnosom, shranjevanjem, prikazovanjem, spreminjanjem, izračunavanjem in logičnimi operacijami s podatki itd. Uporaba platforme Oracle APEX omogoča razvoj lastnih aplikacij za različne namene, ki vključujejo generiranje podatkovne baze na podlagi niza strukturiranih podatkov ali podatkovnih modelov, neposredno delo s podatki v podatkovni bazi z uporabo jezika SQL, ustvarjanje uporabniških vmesnikov, generiranje različnih tabelaričnih in grafičnih poročil ter uporabo spletnih aplikacij na podatkovnih bazah Oracle.

1.1.2 Kaj je aplikacija?

Aplikacija je skupek računalniških programov za določen namen, ki delujejo na osnovi podatkov pridobljenih iz baze podatkov. Še posebej nas zanimajo poslovne aplikacije, torej tiste, katerih programi se uporabljajo za obdelavo podatkov o poslovnih procesih in transakcijah. Razvoj aplikacije, tudi preproste, je zapleten miselni in ustvarjalni proces, ki zahteva določena znanja in veščine. Za razvoj aplikacije potrebujemo algoritemski način razmišljanja (algoritemska pismenost). Gre za sposobnost konceptualizacije, zavedanje, da v neki situaciji obstaja algoritem, sposobnost njegove uporabe, znanje o oblikovanju in uporabi algoritma ter sposobnost kritičnega vrednotenja algoritmov. Poleg tega pa potrebujemo tudi znanja in spretnosti ustvarjanja kakovostnih zapisov v programskem jeziku (kodiranje), testiranja in odpravljanja napak ter drugih znanj, ki jih običajno imenujemo programiranje.

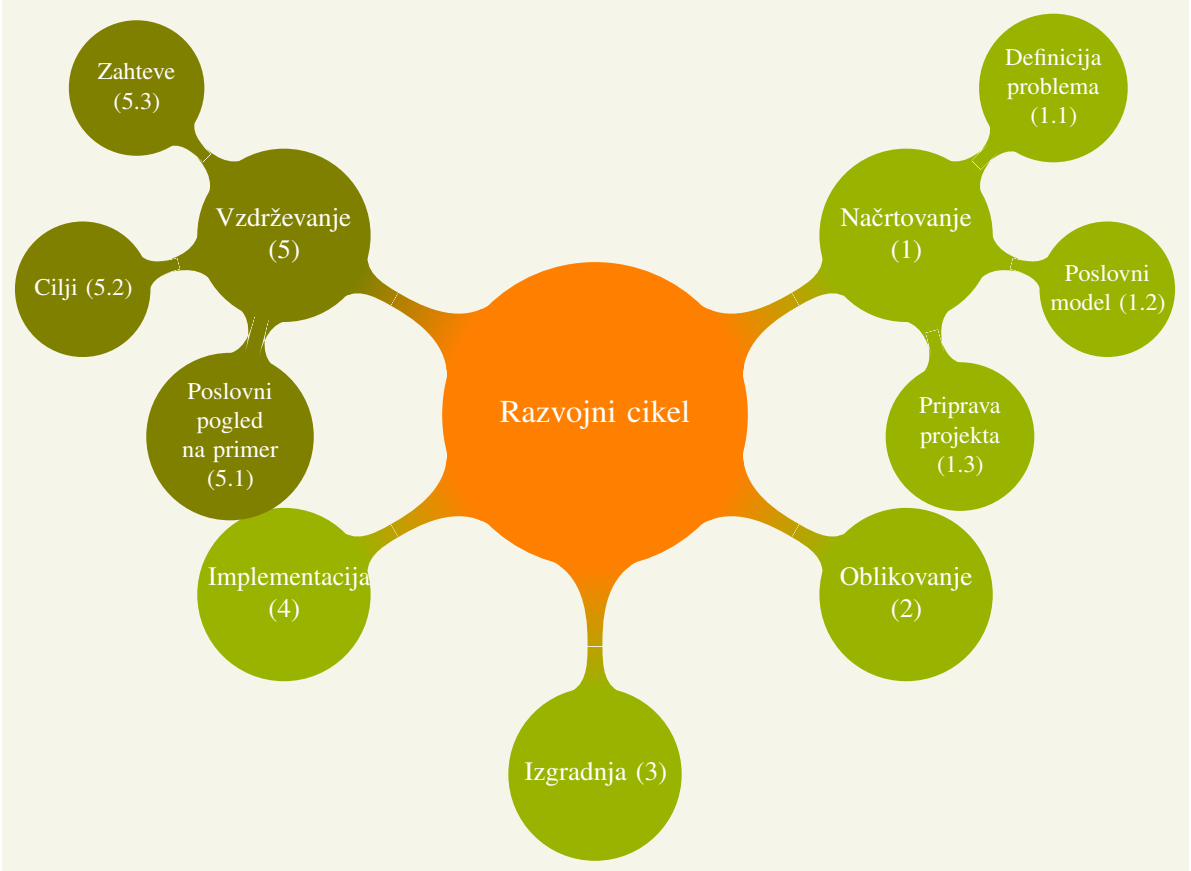
1.1.3 Kako Oracle APEX pomaga pri razvoju aplikacij?

Pri razvoju računalniških programov se ne moremo izogniti algoritemskemu načinu razmišljanja. Uporaba platform in orodij za programiranje z malo / brez kode, kot je Oracle APEX, omogoča neprogramerjem enostavno programiranje brez podrobnega poznavanja programskega jezika in naprednih veščin programiranja. Tako se lahko osredotočijo na problem in algoritem, ne pa na

programski jezik in tehnologijo. Pri razvoju aplikacij nas še posebej zanima hiter, uporabniku prijazen razvoj aplikacij za dostop, poenotenje, analizo in prikaz podatkov iz odprtih podatkov in/ali podatkovnih baz podjetij. Odprti podatki so prosto dostopni podatki, do katerih lahko vsakdo dostopa, jih uporablja in deli. Uporabljamo jih lahko brez omejitev (ali z nekaterimi), jih delimo z drugimi ali uporabljamo za ustvarjanje novih rešitev. Odprte podatke objavljajo vlade, podjetja in druge organizacije, ki želijo spodbujati preglednost in sodelovanje. Danes je na voljo veliko lahko dostopnih virov meteoroloških, prometnih, geoloških zdravstvenih in podobnih podatkov, pa tudi statističnih podatkov o izobraževanju, zdravju, gospodarstvu itd. Podatkovne baze podjetij vsebujejo podatke o poslovnih transakcijah in druge podatke, ki nastanejo kot rezultat poslovnih procesov. Vizualizacijo, analizo, integracijo in različne poglede na te podatke je mogoče uporabiti za sprejemanje poslovnih odločitev ali doseganje nekaterih drugih koristi. Poleg razvoja uporabniškega vmesnika se Oracle APEX uporablja tudi za razvoj prototipa. Prototip služi kot osnova za prikaz želenega delovanja programske rešitve, ki spodbuja komunikacijo o funkcionalnosti in izgledu predvidene rešitve med končnimi uporabniki in razvijalci. Prototip odraža bistvene značilnosti sistema. Ko je dodelan, ga lahko dodamo obstoječi rešitvi, ali pa služi kot podlaga za izdelavo sistema v drugem programskem jeziku, orodju ali platformi.

1.1.4 Cikel razvoja aplikacij

Ne glede na to, kakšno aplikacijo razvijamo in kakšen metodološki okvir, razvojna orodja ali platformo uporabljamo, gre razvojni cikel aplikacije skozi več faz. Te faze so pri različnih metodoloških pristopih različne. Enega od možnih vzorcev razvojnega cikla aplikacije prikazuje slika 1.1.



Slika 1.1: Cikel razvoja aplikacije.

Kot prikazuje slika, cikel razvoja aplikacije vključuje naslednj faze ali procese:

1. Načrtovanje (definicija problema in projekta)
   - definicija in analiza poslovnega modela
   - specifikacija zahtev
   - določanje osnovnih procesov in podatkov
   - odločitve glede tehnologije in arhitekture
   - planiranje razvoja
2. Oblikovanje aplikacije
   - analiza in specifikacija zahtev
   - logično oblikovanje aplikacije (primeri uporabe in logični podatkovni model)
3. Izgradnja aplikacije
   - oblikovanje baze podatkov
   - razvoj (prototipa) aplikacije
   - testiranje
4. Implementacija aplikacije

Načrtovanju sledijo oblikovanje, izgradnja in implementacija. Te faze ali procese lahko izvajamo zaporedno ali iterativno (evolucijsko, inkrementalno), tako da gremo skozi celoten razvojni cikel večkrat in v vsaki iteraciji dodamo del funkcionalnosti.

Načrtovanje aplikacije Cikel razvoja aplikacije se začne z načrtovanjem. Pri preprostih aplikacijah je lahko začetna informacija za načrtovanje aplikacije opredelitev problema, dokument, v katerem je jasno naveden problem. Kdo ima problem? Zakaj je to pravi problem? Kakšne so posledice problema? To je lahko besedilo v nekaj stavkih ali na več straneh. V primeru kompleksnih poslovnih aplikacij pa pripravimo podrobne specifikacije, kot so npr:

- Poslovni pogled na primer opisuje, kako ljudje v organizaciji dojemajo situacije pred razvojem aplikacije ali kaj pričakujejo od aplikacije.
- Cilji razvoja aplikacije opisujejo, kaj želi aplikacija doseči v poslovnem, tehnološkem in drugem smislu, ter kazalnike uspeha.
- Poslovni model opisuje vrednote, segmente strank, odnose s strankami in kanale, ključne partnerje, dejavnosti in vire, strukturo in vire prihodkov in stroškov itd.
- Specifikacija zahtev za programsko opremo opredeljuje zahteve glede funkcije, obnašanja, delovanja, tehnologije itd.
- Načrt projekta opisuje obseg in faze prihodnjega projekta, dejavnosti (naloge) ki se izvajajo v posameznih fazah, vhode in izhode posameznih dejavnosti, podatke o tem, kako se bodo izvajale, vire, ki so potrebni, medsebojno povezanost dejavnosti in njihovo trajanje, stroške, tveganja itd.

Za izdelavo vsake od naštetih specifikacij obstajajo metode, ki so opisane v literaturi. Alexander Osterwalder [6] je na primer razvil metodo Business Model Canvas (BMC), ki se uporablja za razvoj, izboljšanje in dokumentiranje novih in obstoječih poslovnih modelov. Kot je prikazano na sliki 1.2, gre pravzaprav za vizualno shemo devetih gradnikov, ki opisujejo ponudbo vrednosti, ki jo zagotavlja organizacija, njene stranke, odnose in kanale z njimi ter ključne partnerje, dejavnosti in vire ter stroškovno strukturo in tokove prihodkov.



Slika 1.2: Kanvas poslovnega modela ([6])

.

Medtem ko je BMC namenjen spodbujanju razumevanja, analize in oblikovanja poslovnega modela ter zlasti tega, kaj mora podpirati aplikacija, ki jo razvijamo, je specifikacija zahtev za programsko opremo (angl. Software Requirements Specification, SRS) dokument, ki opisuje, kaj bo programska oprema počela in kako naj bi delovala. Opisuje funkcionalnost, zmogljivost in druge značilnosti, ki jih mora imeti aplikacija, da bo podpirala poslovni model in potrebe vseh uporabnikov.

Oblikovanje aplikacije Od prvih računalnikov in računalniških programov do danes je bilo razvitih na stotine različnih metodoloških pristopov, metod in tehnik oblikovanja aplikacij. Metodologija oblikovanja programov se je spreminjala s tehnološkim razvojem. Razvoj programov z malo / nič kode je pristop, ki daje prednost preprostim metodam, ki so razumljive uporabnikom, ki niso profesionalni razvijalci programske opreme. Vendar pa razvoj aplikacij zahteva nekaj načrtovanja pred generiranjem programa. Kot smo že omenili, je računalniška aplikacija skupek programov, ki delajo s podatki na bazi podatkov. Da bi lahko razvili aplikacijo, moramo zasnovati tako programe kot podatkovno bazo. Če baza podatkov že obstaja, moramo vedeti, kako jo uporabljati, predvsem pa moramo razumeti podatkovni model. Za potrebe razvoja programov z malo / nič kode bomo predstavili najnujnejše metode oblikovanja. Hkrati je to osnova oblikovanja izdelava modelov, ki imajo tudi grafični prikaz. Uporabili bomo naslednje metode:

- Model primera uporabe (angl. Use Case, UC) določa povezave med aplikacijo in okoljem.

- Entitetno relacijski (ER) model opisuje strukturo podatkov na logičnem nivoju.

Primer uporabe bomo razložili v nadaljevanju tega poglavja, entitetno relacijski model pa bomo predstavili v poglavju 2.

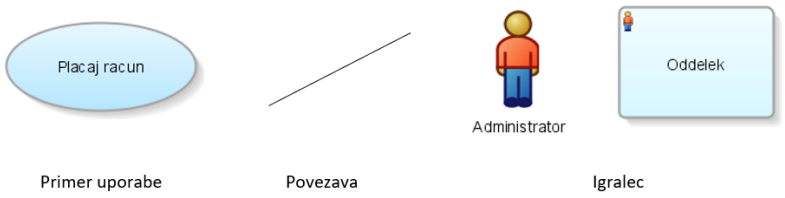
Model primera uporabe (UC) Primeri uporabe niso metoda razvoja programskih tešitev. Uporabljamo jih za opis povezav med sistemom (aplikacijo) in okoljem (uporabniki ali drugimi sistemi). Način prikaza je zelo enostaven in tako razumljiv tudi neprofesionalnim uporabnikom. Primeri uporabe so del Unified Modeling Language (UML), najbolj pogosto uporabljenega jezika za modeliranje, ki se uporablja v svetu programskih inženirjev. Tovrsten jezik zagotavlja standarden način vizualizacije oblikovanja nekega sistema. O UML in njegovih avtorjih, ki skrbijo za njegov razvoj in standardizacijio, Object Management Group (OMG) lahko najdemo veliko literature. UML vključuje 13 metod in diagramskih tehnik, s pomočjo katerih modeliramo strukturo, obnašanje in iterakcije programske rešitve. Mi bomo, kot smo že omenili, uporabili samo eno od teh diagramskih tehnik, torej Primere uporabe.

Model primera uporabe je sestavljen iz dveh delov:

- Specifikacije (besedni opis) primera uporabe
- Diagrama primera uporabe

Najprej si bomo ogledali diagram primera uporabe. Ta diagram prikazuje kaj sistem počne z vidika opazovalca zunaj sistema. Podrobnosti delovanja sistema nas ne zanimajo.

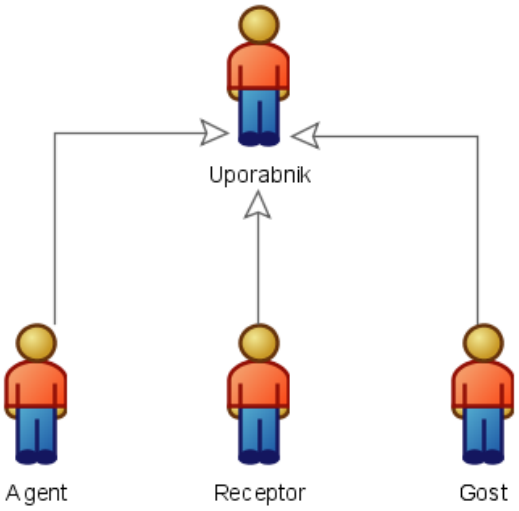
Slika 1.3 prikazuje osnovne koncepte diagrama primera uporabe.



Slika 1.3: Osnovni koncepti diagrama primera uporabe.

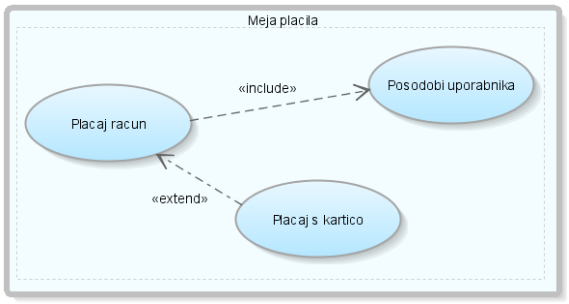
Primer uporabe predstavlja niz dejanj, ki jih lahko izvede sistem (npr. komponenta programske opreme) v interakciji z zunanjimi udeleženci. Gre za zgodbo, ki opisuje, kako akterji uporabljajo sistem za doseganje določenih ciljev ali izvajanje nalog. Predstavlja abstraktno nalogo, ki ima svoj namen. Povezava (angl. Association) povezuje udeležence v komunikaciji, na primer akterja in primer uporabe, ter predstavlja njihovo interakcijo in odnos med sistemom in okoljem (vedenje). Igralec (angl. Actor) predstavlja niz vlog, ki na enak način komunicirajo s sistemom, na primer razred uporabnikov, nek zunanji sistem itd. Igralec je nekdo zunaj obravnavanega sistema, ki je v interakciji z njim, vendar ni njegov del. Lahko je živo bitje (uporabnik, pacient, pilot) ali drug sistem (obračunski sistem, banka, prevoznik).

Diagram lahko prikazuje tudi klasifikacijsko strukturo (glej sliko 1.4), tj. posploševanje/specializacijo pojmov, na primer igralcev. Vsi podrejeni razredi (otroci) podedujejo lastnosti in obnašanje od nadrejenega razreda (starša), kar pomeni, da od starša podedujejo tudi povezave do primerov uporabe. Hkrati lahko posamezen podrejeni razred vstopi v komunikacijo, v katero ne vstopajo niti nadrejeni razred niti drugi podrejeni razredi.



Slika 1.4: Generalizacija / specializacija igralcev.

Slika 1.5 prikazuje meje sistema kot pravokotnik z imenom sistema znotraj katerega so locirani primeri uporabe.

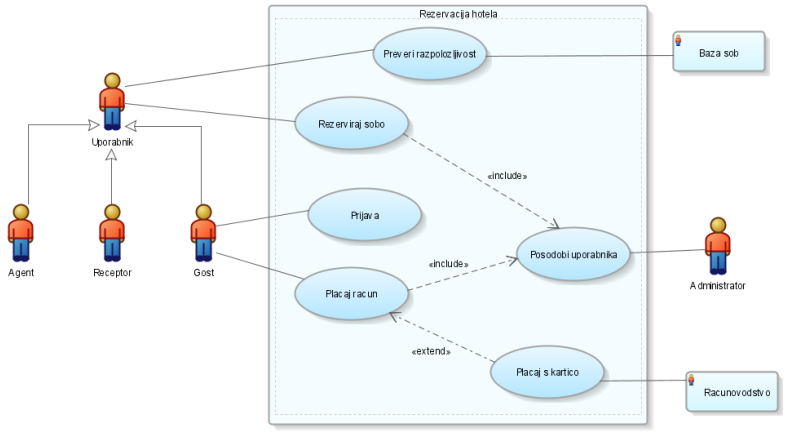


Slika 1.5: Meje sistema ter posebna tipa povezav “include” in “extend”.

UC diagrami ne omogočajo prikaza zaporedja ali poteka dela. Lahko pa uporabimo dva tipa povezav med primeri uporabe, ki jih prikazuje slika 1.5:

- “include” je povezava med osnovnim primerom uporabe in vključenim primerom uporabe, ki označuje, da osnovni primer uporabe vsebuje vedenje vključenega primera uporabe. Na ta način sta funkcionalnost in obnašanje, ki se pogosto uporabljata, ločena v primeru uporabe, ki ga bomo po potrebi vključili v druge primere uporabe. Upoštevajte, da osnovni primer uporabe ni popoln brez vključitve.
- “extend” je povezava iz primera uporabe, ki je razširitev osnovnega primera uporabe, kar pomeni, da je vedenje osnovnega primera uporabe razširjeno z vedenjem razširitve. Hkrati razširitev ni del osnovnega primera uporabe, prav tako ta komunikacija ne bi smela vedno obstajati in delovati brezhibno.

Uporabo konceptov, ki jih uporabljamo v diagramu primera uporabe predstavljamo na primeru sistema rezervacij v hotelu, ki je predstavljen na sliki 1.6.



Slika 1.6: Diagram primera uporabe za sistem rezervacij v hotelu.

Opazujemo obnašanje sistema Rezervacija hotela. Dogajanje v sistemu je znotraj meja pravokotnika. Okolica sistema je zunaj pravokotnika. Znotraj meja sistema je šest primerov uporabe, prikazanih z elipsami z imeni. Zunaj sistema so igralci, ki komunicirajo s sistemom, tj. njegovimi primeri uporabe, in ta komunikacija je prikazana s povezovalnimi črtami, asociacijami. Vsak uporabnik je igralec, ki komunicira s primerom uporabe Preveri razpolozljivost. Ta primer uporabe komunicira z zunanjim akterjem Baza sob. To je nekakšen zunanji sistem rezervacij, ki ima v svoji bazi podatkov informacije o sobah in rezervacijah. Vsak uporabnik lahko rezervira sobo. Ta UC vključuje še en UC, Posodobi uporabnika. Klasifikacijska struktura igralca Uporabnik je prikazana na levi strani. Agent, Receptor in Gost so vrsta Uporabnika. V prikazanem primeru lahko samo Gost komunicira z UC Prijava in Plačaj račun. UC Posodobi uporabnika je vključen v Plačaj račun, kar pomeni, da je mogoče podatke o uporabniku posodobiti tudi v okviru funkcionalnosti plačila računa. UC Plačaj s kartico razširja funkcionalnost Plačaj račun, kar pomeni, da je mogoče med plačilom računa priklicati plačilo s kartico. UC Plačaj s kartico komunicira z zunanjo storitvijo za plačevanje s kreditnimi karticami CC billing. Administrator lahko komunicira s funkcijo UC Posodobi uporabnika in posodablja podatke o uporabniku. Kot smo že omenili, je model primera uporabe sestavljen iz dveh delov:

- Specifikacije (besedni opis) primera uporabe
- Diagrama primera uporabe

Specifikacija primera uporabe opisuje scenarije in notranjo logiko primerov uporabe, začetna stanja (predpogoji) in končna stanja, vmesnike, sistemska sporočila, specifikacijo obdelave napak in izjem in podobno. Običajno se za specifikacijo primera uporabe uporablja neke vrste predloga, ki jo uporabljamo tudi v tej knjigi.

Izgradnja, implementacija in vzdrževanje Izdelava aplikacije pomeni izdelavo programa in vzpostavitev fizične podatkovne baze. Za to bomo uporabili platformo Oracle APEX, kot bo pojasnjeno v nadaljevanju knjige. Prva različica aplikacije, ki deluje in jo je mogoče preizkusiti, je pravzaprav prototip aplikacije. Ta prototip je mogoče nadgraditi in razširiti. Glede na to, kaj se bo s prototipom zgodilo pozneje, razlikujemo med dvema vrstama prototipov: Nadgradljiv prototip se izboljšuje in nadgrajuje do stanja, ki je primerno za uporabo. Prototip, ki ga je mogoče "zavreči", služi kot ilustracija funkcionalnosti in model, na podlagi katerega bo aplikacija zgrajena, običajno na drugi platformi. Če nameravamo prototip aplikacije uporabljati, ga moramo implementirati, to pomeni namestiti v produkcijsko okolje, izobraziti uporabnike, pripraviti bazo podatkov in podobno.

Po fazi implementacije sledi faza uporabe in vzdrževanja, ki vključuje popravljanje in odstranjevanje napak, izboljšanje delovanja in manjše razširitve funkcionalnosti. Če so potrebne večje spremembe, se začne nov razvojni cikel aplikacije.

1.2 Kako začeti z Oracle APEX?

Razvoj programskih rešitev je zapleten predvsem zaradi programiranja, ki ga vključuje. Malokoden pristop k razvoju programskih rešitev odpravlja prav to komponento. Omogoča namreč razvoj naprednih aplikacij, ki so funkcionalne in hkrati kompleksne, brez da bi napisali eno samo vrstico kode. Oracle Application Express je ena boljših rešitev za malokodni razvoj aplikacij. Oracle APEX lahko uporabljamo na različne načine. Uporabljamo ga kjerkoli, v lokalnem okolju ali v oblaku. Pomembno je le, da je na voljo baza podatkov. Oracle APEX lahko namestimo na fizični, namenski strežnik, virtualni računalnik ali kot tako imenovan docker. Slednja rešitev je še posebej priljubljena pri razvijalcih Oracle APEX, saj jo je mogoče zagnati na prenosnem računalniku, na avtobusu ali letalu. Poleg tega lahko uporabljamo tudi Exadata, superzmogljiv fizični strežnik APEX, ki ga Oracle ponuja med svojimi oblačnimi storitvami. Vse te možnosti opisujemo v nadaljevanju, hkrati pa prikažemo, kako dostopate do posameznih okolij.

1.2.1 Kakšen je nivo vašega znanja?

Z Oracle APEX lahko začnete na več načinov. Kateri način je za vas najboljši, pa določa vaša trenutna raven znanja in spretnosti pri delu z Oracle APEX in podobnimi orodji. Predpostavimo, da imate posebne domenske kompetence v katerem koli organizacijskem procesu (npr. prodaja, zaposlovanje, proizvodnja, posojanje, trženje, pravne zadeve, zdravstvo, gradbeništvo, izobraževanje) in naslednje ravni digitalnih veščin:

- popoln začetnik: vešč dela s spletnim brskalnikom, brez kakršnihkoli izkušenj s programiranjem; priporočilo: uporabite lokalno nameščen Oracle APEX, apex.oracle.com ali Oracle Academy
- začetnik: vešč dela s spletnim brskalnikom, nekaj izkušenj s programiranjem npr. uporaba formul v preglednicah; drugi programski jeziki; priporočilo: uporabite lokalno nameščen Oracle APEX, apex.oracle.com ali Oracle Academy
- razvijalec začetnik: vešč dela s spletnim brskalnikom, nekaj izkušenj s programiranjem; poznavanje konceptov baza podatkov, tabela, stolpec v tabeli, primarni ključ, tuj ključ; osnovno znanje poizvedovanja, vnašanja, spreminjanja in brisanja podatkov; priporočilo: uporabite lokalno nameščen Oracle APEX, apex.oracle.com, Oracle Academy ali pa virtualni strežnik VBox
- napredni razvijalec: vešč dela s spletnim brskalnikom; izkušnje s programiranjem shranjenih postopkov, funkcij in sistema za upravljanje baz podatkov; srednji nivo znanja pisanja poizvedb, vnašanja, spreminjanja in brisanja podatkov; priporočilo: uporabite lokalno nameščen Oracle APEX, apex.oracle.com, Oracle Academy, virtualni strežnik VBox ali APEX Docker
- na poti profesionalnega razvijalca: poznavanje razvoja z PL/SQL in JavaScript; priporočilo: uporabite lokalno nameščen Oracle APEX, apex.oracle.com, APEX docker ali Oracle Cloud Infrastructure (OCI) APEX instance

Od vaše uporabniške vloge in lokacije Oracle APEX je odvisno, kako se boste prijavili in uporabljali aplikacijo. Oracle APEX je lahko nameščen lokalno v vašem računalniku ali v gostujočem okolju, kot je storitev Oracle Cloud. Glede na vrsto namestitve se postopek prijave razlikuje. Uporabniki morajo ustvariti delovni prostor, dodati uporabnike Oracle APEX in se nato prijaviti v delovni prostor, preden začnejo razvijati ali nameščati aplikacije. Več uporabnikov lahko sodeluje pri isti namestitvi Oracle APEX s souporabo delovnega prostora, pri čemer lahko ohranijo zasebnost svojih objektov, podatkov in aplikacij. Če spletni brskalnik podpira JavaScript, se lahko uporabniki prijavijo v delovni prostor in dostopajo do domače strani Oracle APEX. Vsak delovni prostor ima posebno ime in ID. V storitvah Oracle APEX Administration Services lahko delovni prostor ročno ustvari skrbnik instance ali pa uporabniki pošljejo zahteve za kreiranje. Za nadzor celotne instance Oracle APEX uporabljamo neodvisen program Oracle APEX Administration Services.

1.2.2 Uporaba lokalno nameščene instance Oracle APEX

V tem razdelku je opisano, kako namestiti Oracle APEX v lokalnem okolju. Namestitev Oracle APEX vključuje več korakov:

- **Izberite Full ali Runtime Environment**. Način Full Environment omogoča popoln dostop do razvojnega okolja App Builder. Za produkcijsko okolje, kjer želite izvajati aplikacije, ki se ne bodo pogosto spreminjale, je dobra izbira izvedbeno okolje (angl. Runtime Environment). V tem primeru Oracle APEX omogoča namestitev izvedbene verzije Oracle APEX, ki jo uporabimo za testiranje in končno uporabo instanc v produkciji. Ker razvijalci ne morejo pomotoma posodobiti produkcijskega programa v izvajalni instanci, to izvajalno okolje zmanjša nameščeni odtis in pravice ter poveča varnost aplikacije. Uporabniki lahko izvajajo produkcijske aplikacije v izvajalnem okolju Oracle APEX, vendar to okolje nima spletnega vmesnika za upravljanje. Izvedbeno okolje je varnejše, ker vsebuje samo komponente, ki so potrebne za izvajanje programa.
- **Namestitev** programa - S prenosom .zip datoteke, ki ste jo našli na Oracle APEX strani, namestite Oracle APEX, nato pa prenesite in namestite Oracle REST Data Services (ORDS).

1.2.3 https://apex.oracle.com

V brskalniku odprite spletno stran. Zahteva brezplačnega delovnega prostora je najhitrejši način, da začnete z delom v Oracle APEX. Samo nekaj sekund vam bo vzelo, da se pridružite skupnosti in delovni prostor za razvoj aplikacij je pripravljen. Ta pristop ne zahteva posebnega računa v oblaku in je brezplačen. Za začetek z Oracle APEX:

- Vpišite naslov v spletni brskalnik: apex.oracle.com
- Kliknite gumb “Start for Free Today”
- na spletni strani potem izberite “Request a free Workspace”
- Vpišite svoje ime, e-naslov, določite ime za svoj delovni prostor in nato izberite “Next”
- Izpolnite vprašalnik in vpišite zakaj zahtevate delovni prostor. Preberite pogoje uporabe, sprejmite pogoje uporabe in izberite “Next”
- Na svoj elektronski naslov boste prejeli elektronsko sporočilo Oracle APEX s povezavo do registracije in namestitve gesla

1.2.4 Virtual Box Appliance / Virtualni računalnik

S spletnim brskalnikom odprite spletno stran na povezavi. Testing, prerequisites, APEX version is behing the latest. Prenesite Oracle Virtual Box in Extension manager. Nastavite Oracle Virtual Box in Extension manager. Uvozite datoteko s končnico .ova. Zaženite namensko napravo (angl. appliance).

Oracle APEX Development VM je virtualni računalnik, ki je pripravljen za uporabo takoj, ko ga uvozimo v VirtualBox. Oraclovi vnaprej pripravljeni virtualni računalniki so dostopni z Oracle uporabniškimi računi. Vse kar morate storiti je to, da si namestite Oracle Virtual Box (Free Virtual Machine Client), nato pa uvozite katerokoli namensko napravo (Pre-Built VM). Na spletnem naslovu je na voljo več različnih pripravljenih namenskih naprav.

Za namestitev virtualnega računalnika sledite naslednjim korakom:

- Prenesite in namestite Oracle VM VirtualBox na vaš računalnik.
- Prenesite datoteke (priporočamo uporabo “Download Manager”)
- Uvozite virtualni računalnik: Z izbiro File > Import Appliance sprožite Import Wizard. Izberite Choose... za pregled direktorija z datotekami in izberite OTN_Developer_Day_VM.ova. Nato z Next potrdite začetek uvoza virtualnega računalnika. Med uvozom boste morali sprejeti licence. Ko boste zaključili z uvozom, boste videli onemogočen ’Oracle Developer Days (Powered Off).
- Testirajte virtualni računalnik: Ko zaključite z uvozom, dvokliknite na OTN Developer Days VM. S klikom OK potrdite zapiranje Virtualbox informacijskih dialogov. Ko pridete do

Enterprise Linux 6 zaslona, se lahko prijavite (uporabniško ime in geslo je oracle.) Počakajte, da se proces zaključi; okolje je pripravljeno, ko se odpre terminalsko okno, ki ga lahko zaprete. Ko zaključite z delom na gostujočem virtualnem računalniku, ga lahko zaprete tako, da izberete System > Shut Down, s čimer izključite gostujoči virtualni računalnik.

Ključne prednosti uporabe virtualnih računalnikov so:

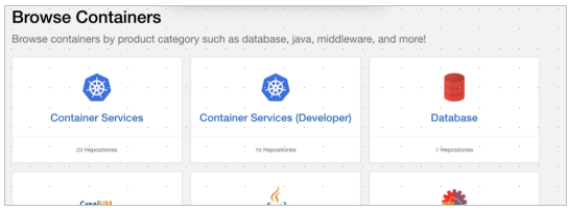
- Uporaba nove Oracle programske opreme brez zapletenih postopkov namestitve ali vključevanja svetovalcev.
- Preskušanje brez izpostavljanja vašega računalnika.
- Brezplačni vodeni vodiči, ki jih je pripravil Oracle, v obliki "Hands On Labs".

1.2.5 APEX docker

APEX docker dobimo na Oracle spletni strani. Z brskalnikom odprite to povezavo.

Oracle APEX lahko namestite kot vsebnik (angl. container) Docker. Dockerji so zelo priročni za razvijalce, saj jih je mogoče namestiti na osebne in prenosne računalnike. Priporočamo, da docker APEX pridobite samo iz zanesljivih virov. Namestitev je običajno hitra in preprosta. Za učenje dela z bazo podatkov Oracle in APEX različica baze in APEX-a ni pomembna, za produkcijsko uporabo pa bodite pozorni na različice. V nadaljevanju tega poglavja je opisan eden od možnih načinov za lokalno namestitev programa APEX docker na računalnik. Pri tem postopku kloniramo uradno sliko Oracle docker, ki vsebuje razvojna orodja.

1. V prvem koraku namestimo docker na računalnik. Prenesemo ga iz spletne strani: `https://docs.docker.com/desktop/install/windows-install/`
2. V drugem koraku se registriramo za brezplačen račun na Oracle Container Registry: `https://container-registry.oracle.com/ords/f?p=113:10` Docker image se prenese iz registra (glej sliko 2).



3. V tretjem koraku odpremo terminalsko okno in dostopamo do Oracle registry s predhodno kreiranim uporabnikom.

```
docker login container-registry.oracle.com
```

Priporočeno je, da kreiramo omrežje v okolju dockerja, da lahko komuniciramo z ostalimi dockerji z uporabo imena gostitelja (angl. hostname)

```
docker network create ords-database-network
```

Zdaj smo pripravili vse potrebno in lahko izvedemo ukaz za zagon baze podatkov Oracle Database XE v dockerju na našem računalniku.

```
docker run -d --name testapex --hostname database --network=ords-database-network -p 1
```

Bodite pozorini, da smo uporabili ime paramentra (angl. parameter name), ime gostitelja (angl. hostname) in omrežje (angl. network). Razlaga:

- Prva oznaka -d bo zagnala vsebnik (angl. container) v ločenem načinu.
- Parameter –name določa ime vsebnika.
- -p preslika vrata 1521 na gostujočem računalniku v vrata 1521 na vsebniku, da se lahko povežemo z bazo podatkov.

- –network poveže vsebnik v omrežje, ki smo ga kreirali.
- –hostname bo določil ime strežniku z bazo podatkov.
- Zadnji parameter je slika, ki jo želimo uporabiti za zagon vsebnika.

Če se želite povezati z sqlplus lahko uporabite ta ukaz:

```
docker exec -it -u oracle testapex sqlplus / as sysdba
```

Z ukazom *show pdbs* boste preverili status pdbs. S spodnjim ukazom lahko spremenite privzeto geslo (v tem primeru bo novo geslo Welcome1!!). Pred izvedbo tega ukaza zagotovite, da je pdb XEPDB1 odprta za branje in pisanje).

```
docker exec testapex ./setPassword.sh Welcome1!!
```

Za namestitev Apex for Oracle XE instalacije v dockerju, izvedite naslednje korake. Iz terminalske seje potegnite image ords.

```
docker login container-registry.oracle.com
```

Nato izvedite pull ukaz za prenos ords image.

```
docker pull container-registry.oracle.com/database/ords:latest
```

Za nastavitev Apex instalacije kreirajte direktorij.

```
mkdir ~/APEX
```

Informacije o nizu vstavite v datoteko v tem imeniku.

```
echo 'CONN_STRING=sys/Welcome1##@database:1521/XEPDB1' >
~/APEX/conn_string.txt
```

Upoštevajte, da sta bila uporabljena parameter ime gostitelja Oracle Database XE docker in storitev pdb za povezavo z nizom. Spremenljivko CONN_STRING opredelite na naslednji način. Zapišite kot je navedeno spodaj, brez enojnih navednic:

```
CONN_STRING=user/password@host:port/service_name
```

Zdaj lahko zaženemo docker.

```
docker run --rm --name apex
-v /Users/lbindi_it/APEX:/opt/oracle/variables
--network=ords-database-network -p 8181:8181
container-registry.oracle.com/database/ords:latest
```

Tudi tukaj smo uporabili enak parameter network. Za pregled namestitve lahko odpremo drugo terminalsko sejo in izvedemo spodnji ukaz:

```
docker run - rm - name apex
-v /Users/lbindi_it/APEX:/opt/oracle/variables
--network=ords-database-network -p 8181:8181
container-registry.oracle.com/database/ords:latest
```

Zamenjamo geslo za APEX_PUBLIC_USER uporabnika na bazi podatkov. Za to se moramo v Oracle XE docker prijaviti preko sqlplus.

```
sqlplus sys/Welcome1##@//localhost:1521/XEPDB1 as sysdba
```

Nastavimo geslo.

```
alter user APEX_PUBLIC_USER identified by Welcome1##;
```

Zdaj se lahko prijavimo v APEX okolje z brskalnikom na lokalnem računalniku.

```
http://localhost:8181/ords
```

1.2.6 APEX instanca v Oracle Cloud Infrastructure

Oracle Cloud Infrastructure (OCI) ponuja malokodni razvoj aplikacij z Oracle APEX na avtonomni infrastrukturi, ki vnaprej konfigurirana in pripravljena za uporabo. OCI zagotavlja skalabilnost, varnost, visoko razpoložljivost in možnost globalnega dostopa preko regionalnih podatkovnih centrov v oblaku. Postopek kreiranja instance APEX Service v OCI je prikazan v nadaljevanju. Koraki za dostop do OCI pa so naslednji:

1. **Pridobi OCI račun**. Uporabnik mora imeti dostop do računa OCI, da lahko uporablja storitev APEX OCI. V primeru, da ima organizacija sklenjeno prodajno pogodbo z družbo Oracle, uporabnik lahko uporabi ta račun OCI. V tem primeru stopite v stik z administratorjem OCI v vašem podjetju. V kolikor ste individualni uporabnik, začetnik, razmislite o prijavi na Oracle Cloud Free Tier. Ta poleg brezplačne začetne dodelitve kreditov v oblaku ponuja tudi brezplačno najemno razmerje in račun OCI brez izteka veljavnosti. Med poskusnim obdobjem lahko kredite uporabite za nakup plačljivih storitev OCI, kot je storitev APEX. Če želite po izteku poskusnega obdobja ali po izteku brezplačnih kreditov (kar se zgodi prej) še naprej uporabljati storitev APEX, morate račun nadgraditi v plačljivo verzijo ter kupiti več kreditov. Sicer pa bo po preteku vaš račun OCI spremenil lahko uporabljal samo storitve OCI, ki imajo na voljo vedno brezplačno obliko (angl. Always free). Oracle svetuje, da račun spremenite v plačljivega in kupite več kreditov med poskusnim obdobjem ali po njem. Postopek prijave začnite tako, da si ogledate možnost Oracle Cloud Infrastructure Free Tier. Za začetek prijave obiščite spletno mesto `https://signup.oraclecloud.com/`.
2. **Prijava v OCI konzolo.** S spletnim brskalnikom se postavite na OCI Console Sign-In Page. Vnesite uporabniško ime in geslo, nato pa še ime vašega oblačnega računa (znan tudi kot "tenancy name"). Elektronsko sporočilo z dobrodošlico, ki ga prejmete, vsebuje tako vaše uporabniško ime kot tudi ime oblačnega računa.

1.2.7 APEX instanca v Oracle Academy

Oracle Academy omogoča institucionalnim članom in njihovim študentom dostop do Oracle Application Express (APEX) za praktične vaje v oblaku. Za reševanje laboratorijskih nalog in prakitičnih vaj v okviru tečajev Database Foundations, Database Design and Programming with SQL, Programming with PL/SQL, in Oracle Application Express—Application Development Foundations, Oracle Academy ponuja učiteljem instance pripravljene za uporabo v učilnici z do 99 študenti. Za dostop do Oracle APEX preko Oracle Academy se morajo uporabniki prijaviti v Member Hub. Če niso člani Oracle Academy, pa se morajo za dostop do materialov prijaviti kot Institutional Member (brezplačno). Dostop je mogoč preko spletne strani `https://academy.oracle.com/en/oa-web-overview.html`.

1.3 Vprašanja

1. Katere so faze v ciklu razvoja aplikacije in katere aktivnosti se izvajajo v tretji fazi?
2. Katera od možnosti dela z Oracle Application Express je najbolj priporočljiva za popolnega začetnika?
3. Katere prednosti prinaša uporaba Oracle APEX instance v Oracle Cloud Infrastructure?

1.4 Odgovori

1. Faze v razvoju aplikacije so: (i) načrtovanje (definicija problema in projekta), (ii) oblikovanje aplikacije, (iii) izgradnja aplikacije, (iv) implementacija aplikacije. Aktivnosti v tretji fazi (izgradnja aplikacije) so: oblikovanje baze podatkov, razvoj (prototipa) aplikacije in testiranje.
2. Popolnemu začetniku priporočamo uporabo lokalno nameščene instance Oracle APEX, apex.oracle.com ali Oracle Academy.
3. Oracle Cloud Infrastructure (OCI) ponuja APEX razvojno okolje kot avtonomno infrastrku-

turo, ki je vnaprej pripravljena in primerna za takojšnjo uporabo. OCI zagotavlja skalabilnost, visoko razpoložljivost in globalni dostop preko regionalnih podatkovnih centrov v oblaku.

2. Kako pripravite bazo podatkov?

E. KAPSAMMER, W. RETSCHITZEGGER, W. SCHWINGER; PREVOD U. RAJKOVIČ

Predpostavimo, da delate za *Walmart* in *Sistemi za upravljanje podatkovnih baz* (SUBP) še ne obstajajo. Nadalje predpostavimo, da vas prosijo, da izdelate *spletno aplikacijo*, ki lahko shrani, prikliče, vizualizira in nadalje obdela vsako prodajo v vsaki od 10.500 Walmartovih trgovin skupaj z informacijami o oddelkih, zaposlenih in njihovih delovnih mestih, kar bi lahko predstavljalo petabajte podatkov. Naloga, ki je očitno ni mogoče obvladati, saj se soočate s številnimi izzivi:

- Koliko datotek in diskov potrebujete za shranjevanje?
- Kako najdete podatke?
- Kako zagotovite ustrezen odzivni čas?
- Kako omogočite spremembe in hkratni dostop do podatkov?
- Kako preprečite nepooblaščen dostop do podatkov?

Če uporabite SUBP kot hrbtenico svoje spletne aplikacije (glejte sliko 2.1), so ti izzivi obvladljivi, saj SUBP zagotavlja ustrezno funkcionalnost za njihovo obvladovanje.



Slika 2.1: SUBP kot hrbtenica spletnih aplikacij.

Sistem baze podatkov se sestoji iz:

1. programske opreme za pomoč pri upravljanju in obdelavi velikih zbirk podatkov, ki jih lahko uporablja spletna aplikacija, imenovana sistemi za upravljanje baz podatkov (SUBP SUBP)

2. sistema za shranjevanje samih podatkov, imenovano (), ki zagotavlja osnovo, na kateri temelji spletna aplikacija.

V nadaljevanju so podrobneje obravnavani glavni koncepti, potrebni za izgradnjo baze podatkov (kratica BP ali angleška kratica DB), tj. *razvoj BP*. Čeprav gre na koncu za "*ustvarjanje ustreznih tabel v BP, shranjevanje podatkov, manipuliranje z njimi in poizvedovanje po njih*", je treba za dosego tega cilja predstaviti nekaj ključnih pojmov.

V podpoglavju 2.1 bomo opisali postopek modeliranja podatkov, ki jih želimo shraniti v BP, z uvedbo dveh različnih *nivojev abstraktnosti*. , ki obsega *logični model* in *fizični model* v smislu *relacijskega modela*[1]. V nadaljevanju, v podpoglavju 2.2, se osredotočamo na mehanizme in orodja, značilne za SUBP, za *ustvarjanje podatkovne strukture BP*, kot jih predpisuje postopek relacijskega modeliranja podatkov. Na podlagi tega podpoglavje 2.3 obravnava vprašanja shranjevanja in manipulacije podatkov, podpoglavje 2.4 pa je namenjen obravnavi poizvedovanja po shranjenih podatkih. Na koncu, podpoglavje 2.5 vsebuje povzetek vseh možnosti in orodij, ki so na voljo za izgradnjo BP kot osnovnega nivoja spletne aplikacije.

Skozi celotno poglavje je uporabljen *delovni primer*, ki ga je navdihnil zgoraj predstavljeni primer uporabe trgovske verige Walmart, s poudarkom na njegovem *Človeških virih (HR)* v smislu *zaposlenih*, *oddelkov* in *delovnih mest*. ORACLE uporablja ta primer v predstavitvene namene že od prve različice svojega SUBP leta 1979.

2.1 Kako modelirati podatke

Prvi korak pri udejanjanju BP spletne aplikacije je, da se osredotočimo na dejanske podatke, ki jih je potrebno shraniti. Ker so ti podatki lahko precej obsežni, kot smo videli v zgoraj opisanem primeru Walmart, je smiselno podatke najprej obravnavati na bolj abstraktnem nivoju, da se izognemo številnim tehničnim podrobnostim, ki se nanašajo na samo izvedbo BP. Na to se osredotoča logični model, kot je obravnavan v oddelku 2.1.1. Nato je na bolj konkretni in s tem za SUBP specifičnem nivoju predstavljen relacijski model v razdelku 2.1.2.

2.1.1 Logični model podatkov

Uvod. *Logični (podatkovni) model* opisuje "stvari" iz *realnega sveta* (tj. *področje problema*), o katerih želi npr. organizacija, kot je Walmart, zbirati in obdelovati podatke. Logični modeli so vizualno opisani z diagrami, v praksi pogosto s tako imenovanimi *Entitetno-Relacijskimi diagrami (ER diagrami)*, brez upoštevanja posebnosti določenega SUBP. Prednosti tega vizualnega prikaza so številne, na primer večja razumljivost in lažje komuniciranje znotraj razvojne skupine in s stranko.

Za modeliranje ER diagramov so na voljo številni različni grafični formalizmi in orodja. Osredotočili se bomo samo na *ORACLE's SQL Developer Data Modeler (ODM)*. Slika 2.2 prikazuje ER diagram našega kadrovskega primera, ki bo podrobneje razložen v nadaljevanju.



Slika 2.2: Logični model (ER diagram) našega kadrovskega primera.

[1]Relacijski model podatkov, ki ga opisuje to poglavje, ni edin možen pristop k udejanjanju fizičnega modela BP, temveč poznamo tudi druge oblike predstavitve podatkov (npr. "podatkovni modeli"), kot sta *objektno usmerjeni podatkovni model* in *podatkovni modeli NoSQL*.

Enak primer je prikazan na slikah 2.3, 2.4 in 2.5, vendar tokrat znotraj ORACLE ODM. Vsi opisani koncepti so opremljeni z opombami, ki pojasnjujejo najpomembnejše grafične elemente in simbole.

ER diagrami - entitete, atributi, relacije oz. povezave *ER diagrami* opisujejo pojme, ki nas zanimajo pri našem problemu, in so sestavljeni iz (i) *entitet* (npr. oddelkov, zaposlenih, delovnih mest), (ii) *atributov* (npr. "departmentName" in "locationName" za oddelke ali "lastName" in "salary" za zaposlene), njihovih značilnosti in (iii) *povezav* (npr. oddelek "vključuje" več zaposlenih) med entitetami.

Entitete. *Entitete* poimenujemo s samostalniki, za vsako entiteto drugo ime, in so na diagramu predstavljeni kot zaobljeni pravokotniki. Za poimenovanje se uporabljata množinska in edninska oblika, vendar je v zadnjem času bolj zaželena edninska oblika (standardizirana tudi v standardu ISO 11179-5).

Atributi. Atributi so vedno povezani z entitetami, tj. ne morejo obstajati samostojno. V nasprotju z entitetami je *atribut* "*atomski*", kar pomeni, da ga ni mogoče dodatno strukturirati kot entiteto, saj je predstavljen kot samostalnik v ednini. Vsakemu atributu je mogoče dodeliti določeno *zalogo vrednosti* (tudi *podatkovni tip*), ki označuje "naravo"dovoljenih vrednosti. Najpogostejši podatkovni tipi so številčni (`NUMERIC`), znakovni (`VARCHAR`) in datumsko-časovni (`DATE/TIME`).

Poleg opisa entitete lahko nekateri atributi služijo tudi za *identifikacijo posameznega primerka* entitete (npr. določenega zaposlenega ali določenega oddelka). Takšni identifikacijski atributi delujejo kot tako imenovani *Primarni ključ (PK)*. Za oddelek se lahko npr. kot PK opredeli umetni oz. nadomestni identifikator (ki mora biti številčen, npr. "departmentID") ali kombinacija obstoječih atributov, kot sta "departmentName"in "location", kar tvori tako imenovani *sestavljen PK*. Priporočljivo je opredeliti PK za vsako entiteto, saj v nasprotnem primeru npr. zaposlenih ali oddelkov z istim imenom pri pridobivanju podatkov ni mogoče razlikovati. Tako mora biti vsaka vrednost atributa PK unikatna in seveda mora obstajati.

Na koncu naj omenimo, da se lahko poleg zalog vrednosti in PK za določene atribute določijo tudi druge tako imenovane *omejitve* vrednosti podatkov, npr. `NOT NULL`, tj. "obvezno", da se zahteva obstoj vrednosti (npr. za atribut plača).

Povezava. *Povezava* povezuje (najpogosteje) dve različni entiteti, vizualno je prikazan s črto med pravokotnikoma entitet in je poimenovan z glagolskim izrazom nad črto. Ločimo tri različne vrste razmerij, ki se se razlikujejo po *številu primerkov entitet*, ki so lahko del razmerja na vsakem koncu povezave, imenovanem "*kardialnost*" (predstavlja drugo vrsto omejitve podatkov, ki jo je mogoče določiti):

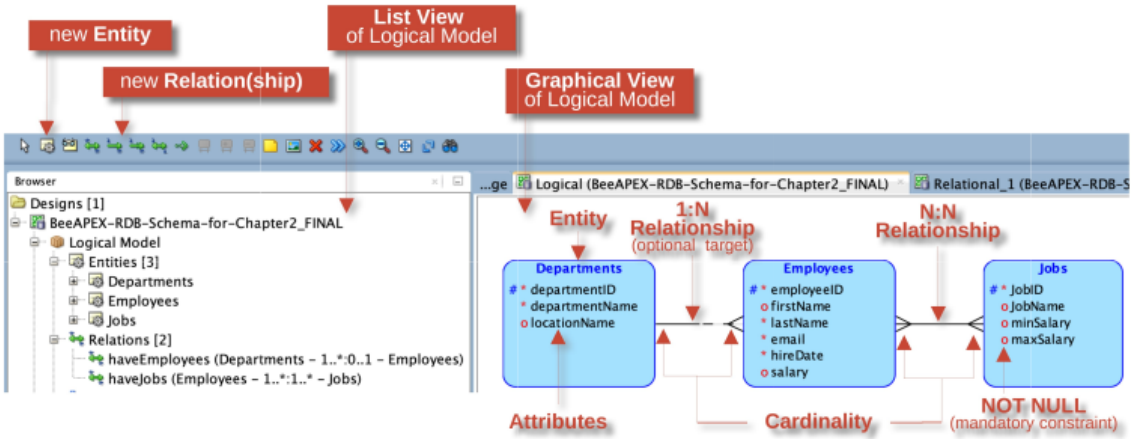
- **Ena-proti-mnogo (1:N):** En primerek entitete je lahko povezan z več primerki entitet na "mnogo strani" povezave (npr. en oddelek ima veliko zaposlenih, tj. povezava "haveEmployees"), entiteta na "mnogo strani" povezave (zaposlen) pa je lahko povezana samo z eno entiteto na "eni strani" (oddelek). Takšno povezavo imenujemo *relacija Izvor-cilj* ali *Starš-otrok*, pri čemer izvorna entiteta na strani "ena" deluje kot "starš", poljubno število ciljnih entitet na strani "mnogo" pa kot njeni "otroci". V navedenem primeru oddelek predstavlja starševsko entiteto, zaposleni pa otroke.
- **Mnogo-proti-mnogo (M:N):** En primerek izvorne entitete je lahko povezan z *več primerki ciljne entitete* in en primerek ciljne entitete je lahko povezan z *več primerki izvorne entitete*. Primer so zaposleni, ki imajo delovna mesta (tj. povezava "haveJobs"), saj ima lahko zaposlen več kot eno delovno mesto, določeno delovno mesto pa ima verjetno več zaposlenih. Opozoriti je treba, da bo razmerje "M:N" vedno ustvarilo dve relaciji "1:N", ko bomo nadaljevali z modeliranjem baze podatkov.
- **Ena-proti-ena (1:1):** En primerek entitete je lahko povezan tudi s samo *enim primerom entitete* na drugem koncu relacije in obratno (npr. oseba "živi" na določenem naslovu in ta naslov je naslov samo te osebe). Velja opozoriti, da se tovrstne relacije zaradi kompaktnejše in enostavnejše obdelave običajno predstavlja znotraj ene entitete (npr. entiteta osebe za-

jema tudi atribute naslova, kot sta “ulica” in “mesto”) in zato jih v nadaljevanju ne bomo obravnavali.

- **Izbirnost:** Za vsako od navedenih treh različnih razmerij je mogoče na vsakem koncu povezave določiti tudi, ali je *obstoj vsaj enega primerka entitete obvezen ali neobvezen*. Če je na primer ustanovljen nov oddelek, se lahko zgodi, da v tem oddelku še ni zaposlenih, tj. povezava “haveEmployees” mora biti opredeljena kot neobvezna. V ODM to prikažemo s črtkano črto.

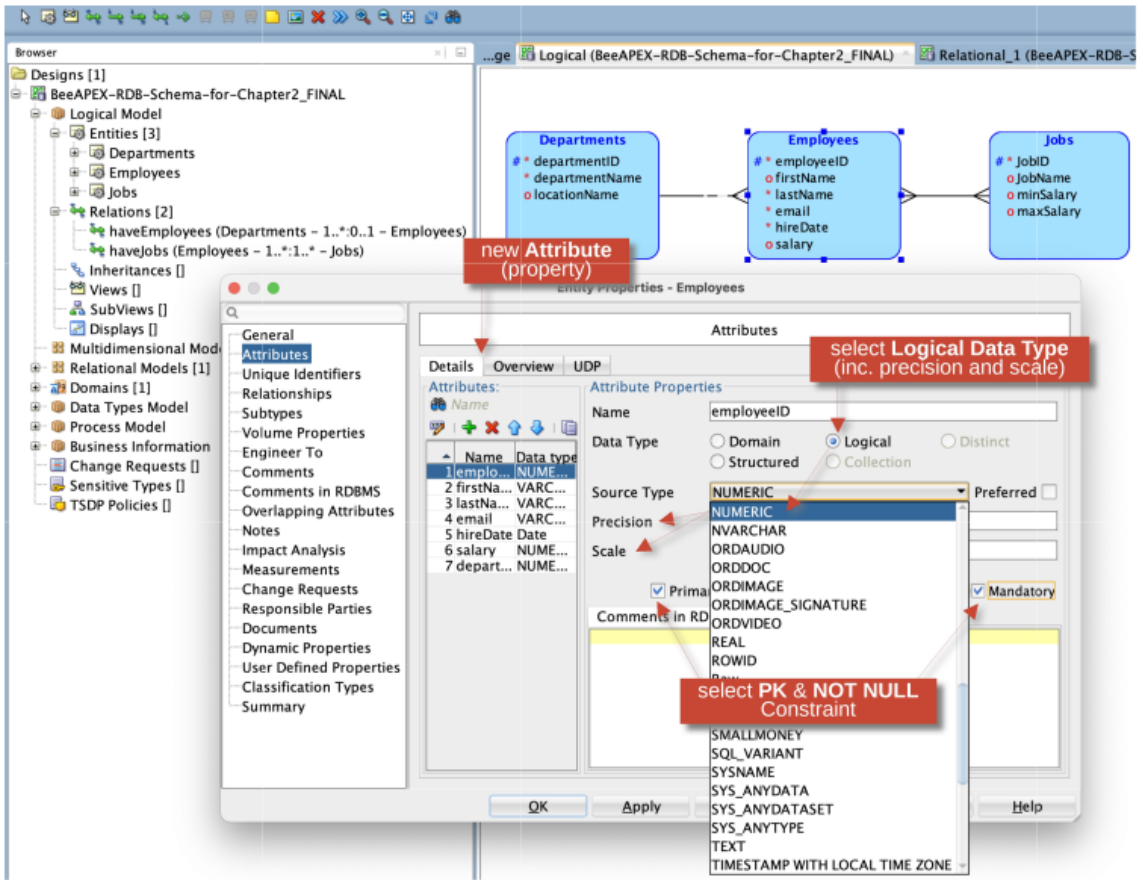
Povezave so v ODM poimenovane kot “relacije”, ki jih ne smemo zamenjevati z “relacijami” v “relacijskem modelu podatkov”, ki so pravzaprav sinonim za “tabele” (glejte podpoglavje 2.1.2). Zato se še naprej držimo bolj običajnega izraza “povezave”.

Grafični uporabniški vmesnik - elementi ODM za upravljanje ER-diagramov. Na naslednjih slikah so predstavljeni najpomembnejši elementi grafičnega uporabniškega vmesnika ODM. Prva slika 2.3 prikazuje, kako definiramo nove entitete in nove povezave ter vizualizacijo logičnega modela v obliki seznama in grafičnega prikaza. V grafičnem prikazu so najpomembnejše grafične oznake in simboli opremljeni z opombami.



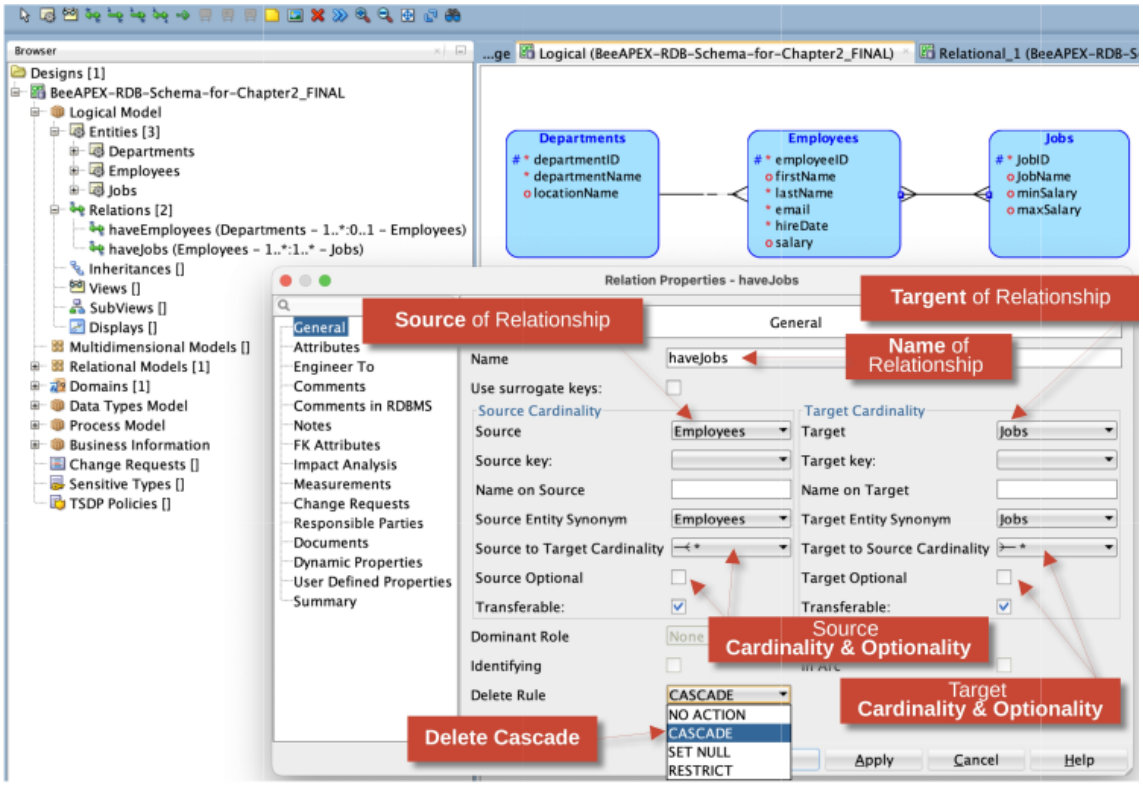
Slika 2.3: Upravljanje logičnega modela z ODM.

Slika 2.4 prikazuje pogovorno okno za definiranje atributov, ki ga lahko odprete z dvojnim klikom na entiteto.



Slika 2.4: Upravljanje atributov z ODM.

Slika 2.5 prikazuje pogovorno okno za definiranje povezav, ki ga lahko sprožite z dvojnim klikom na določeno vrstico, ki predstavlja povezavo.

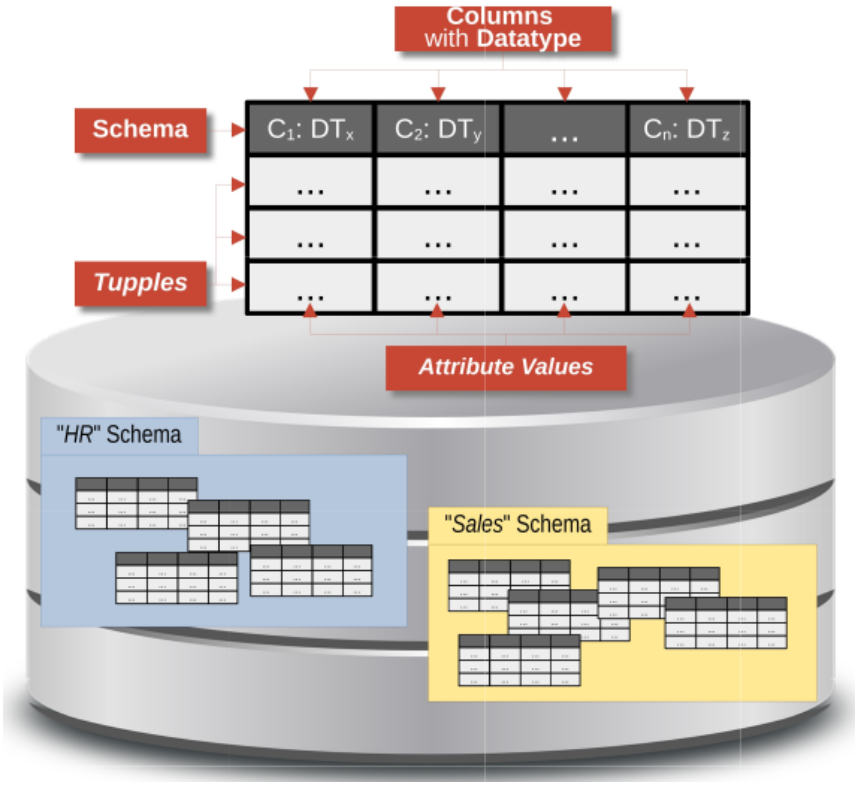


Slika 2.5: Upravljanje povezav z ODM.

2.1.2 Relacijski model podatkov

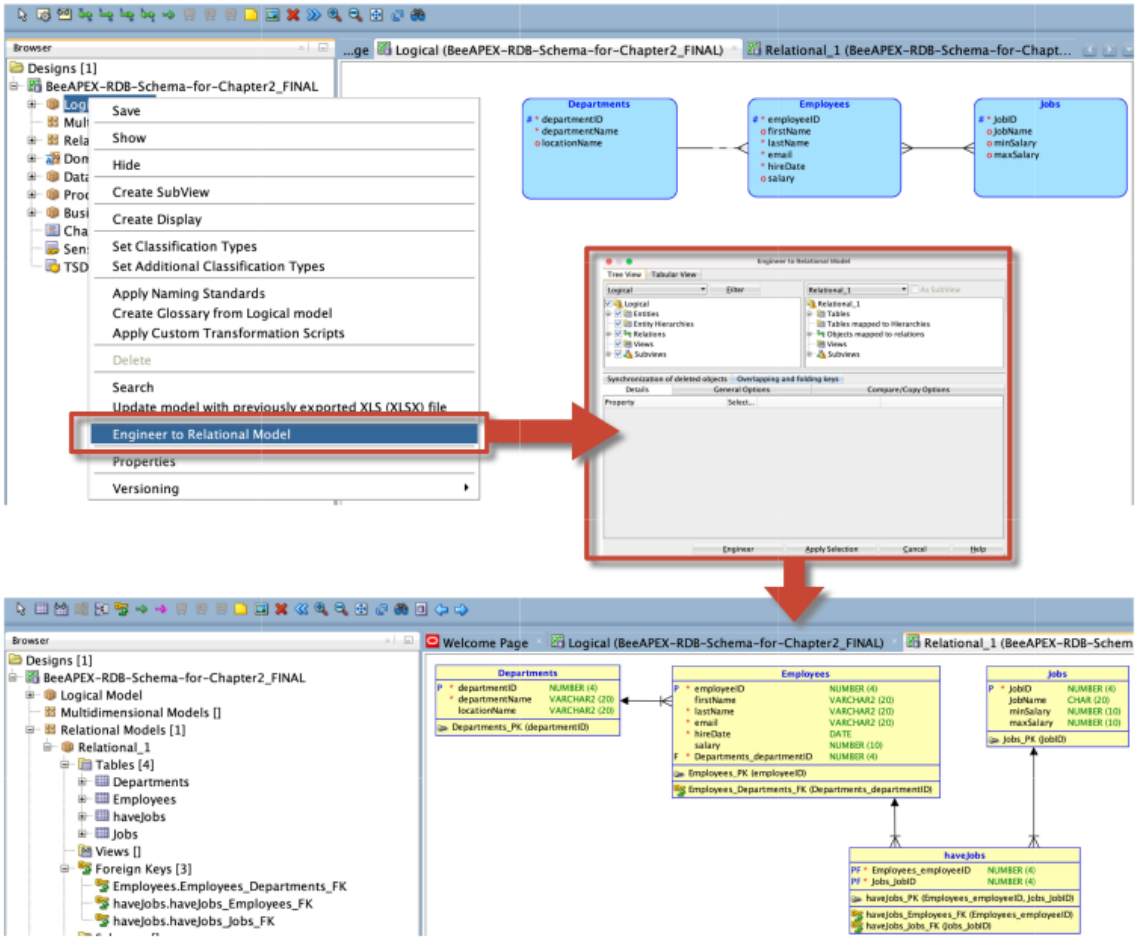
Shema BP. Logični model v smislu sheme ER, ki je neodvisen od izbranega SUBP, je osnova za vzpostavitev strukture BP, imenovane *shema BP*. Znotraj določenega SUBP je mogoče ustvariti in shraniti poljubno število različnih shem za različne uporabnike oz. področja, npr. eno za upravljanje podatkov o kadrih in drugo shemo za upravljanje podatkov o prodaji izdelkov. Shema deluje podobno kot mapa v datotečnem sistemu in omogoča združevanje podatkov za določen namen.

Shema relacijske BP. *Shema relacijske BP (R-BP)* uporablja *relacijski model (RM)* kot osnovni formalizem za opis strukture podatkov. RM je precej preprost, saj predstavlja shemo DB kot zbirko *relacij* oziroma *tabel*, ki spominjajo na koncept entitet. Tabele so neposredno izpeljane iz entitet. Podobno kot entiteta je tabela sestavljena iz imena, *stolpcev* (podobno kot *atributi*), njihovih podatkovnih tipov in drugih omejitev. Vse tabele skupaj predstavljajo *shemo R-BP* (glejte sliko 2.6). Shema tabele služi za shranjevanje dejanskih podatkov, medtem ko se vsaka vrstica v tabeli imenuje "*zapis*", ki je sestavljen iz povezanih podatkov, shranjenih v stolpcih, in tako predstavlja dejanski *primerek entitete*indexprimerek entitete v smislu *relacijskih* (*tabelaričnih*) podatkov.



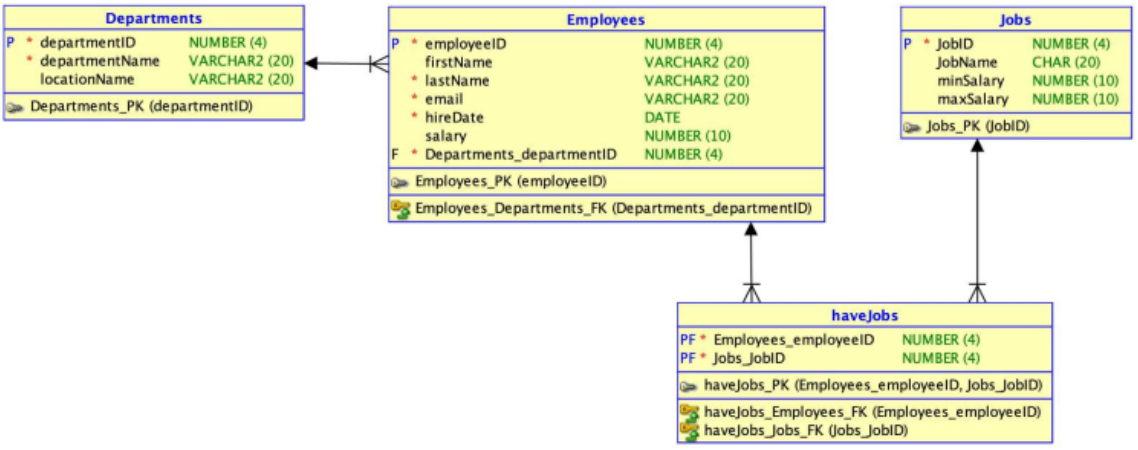
Slika 2.6: Shema relacijske BP in shema tabele.

Zanimivo je, da *ODM* omogoča samodejno generiranje sheme R-BP iz logičnega modela v obliki tabel, kar odraža entitete našega logičnega modela, skupaj z vsemi atributi in omejitvami posameznih entitet. To izvedemo z uporabo menija *Engineer to Relational Model*, kot je prikazano na sliki 2.7. Dobljena shema R-BP je grafično prikazana z uporabo nekakšnega formaliziranega ER-diagrama, ki temelji na nekoliko drugačni - bolj "tabelarično usmerjeni notaciji" v primerjavi z ER-diagramom logičnega modela - in prikazuje tudi podatkovne tipe vseh atributov.



Slika 2.7: Postopek generiranja sheme relacijske BP iz logičnega modela.

Vrnimo se k našemu primeru, ki ga predstavlja relacijski model na sliki 2.8.

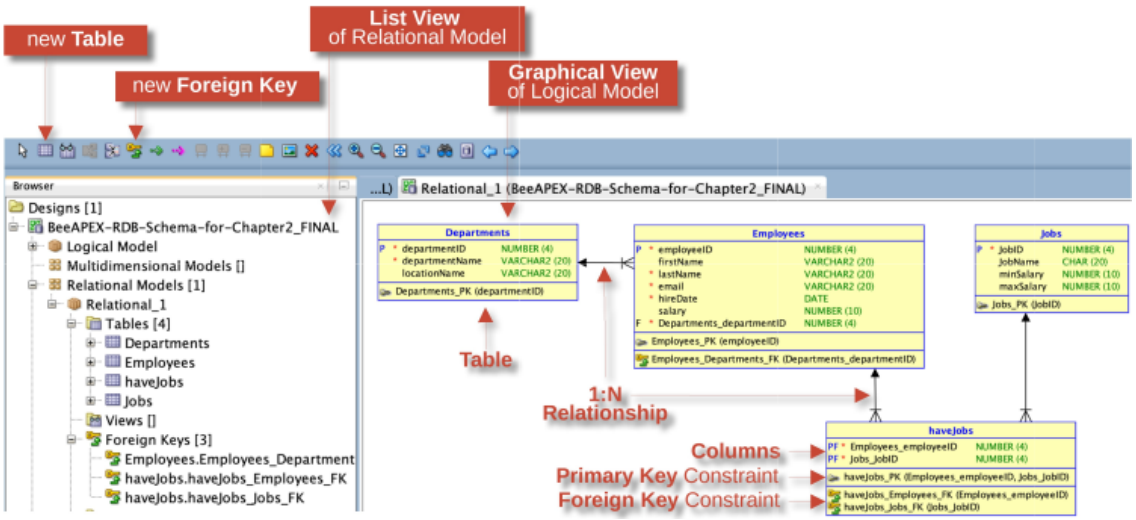


Slika 2.8: Samodejno generirana shema relacijske BP.

Med postopkom generiranja relacijskega modela iz logičnega modela se poleg preoblikovanja obstoječih entitet in atributov v tabele in njihove stolpce vsaki tabeli samodejno dodajo primarni ključi (glejte npr. "Departments_PK (departmentID)". Poleg tega so povezave med entitetami znotraj logičnega modela samodejno predstavljena v tabelah z uporabo tako imenovanih *tujih ključev (ang. Foreign key - FK)* na način:

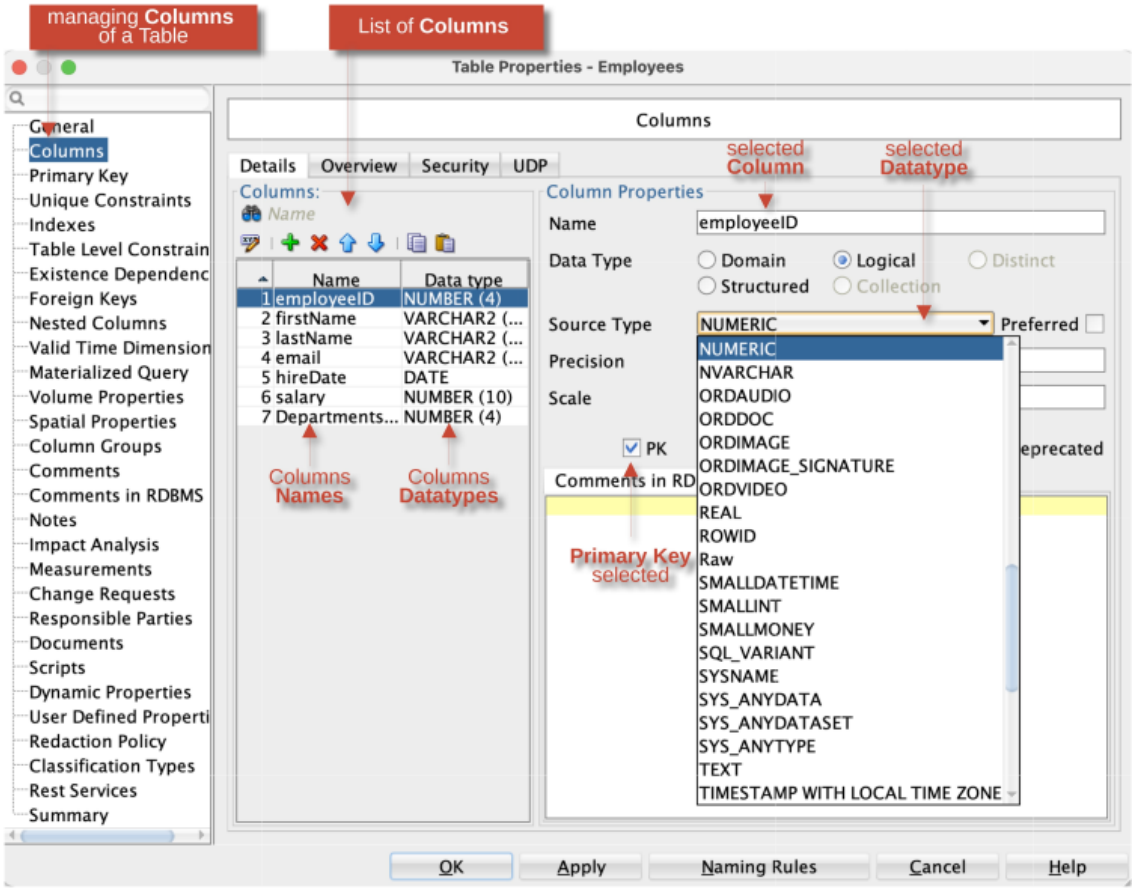
- **Povezava ena-proti-mnogo (1:N):** V tabeli na strani povezave "mnogo" (tj. tabela otrok v razmerju starš-otrok) je dodan atribut, ki je določen kot FK in deluje kot referenca na vrednosti PK v povezavi na strani "ena" (tj. tabela starš). Če obstaja npr. zaposlen z ID 4711, mora biti v atributu FK povezanega oddelka prisotna tudi vrednost 4711. Posledično mora biti atribut FK enakega podatkovnega tipa kot PK. V našem primeru se samodejno ustvarita FK "Departments_departmentID"in FK z imenom *Employees_Departments_FK*. Posledično SUBP ob vsaki obdelavi podatkov v BP, ki se nanaša na omenjeno povezavo, zagotavlja *konsistentnost* med PK in FK na obeh straneh povezave, tj. skladnost podatkov o povezavi. Če se npr. v tabelo Employees vstavi nov zaposlen, SUBP zagotovi, da je ta dodeljen obstoječemu oddelku. Če poskušate izbrisati oddelek, SUBP ne izbriše oddelka, če obstajajo povezani zaposleni. Takšno privzeto delovanje je mogoče spremeniti z uporabo tako imenovane *Cascade-Delete Option* pri določanju FK, kar povzroči samodejno brisanje (morda "odpuščanje") vseh zaposlenih, povezanih z izbrisanim oddelkom.
- **Povezava mnogo-proti-mnogo (M:N):** Povezava med dvema entitetama se realizira z uvedbo nove, tretje tabele, ki deluje kot nekakšna *tabela preslikav*, ki hrani sklice na PK povezanih entitet kot dva FK. V našem primeru zaposlenih in delovnih mest se samodejno ustvari tretja tabela z imenom "haveJobs", ki hrani povezave med zaposlenimi in njihovimi delovnimi mesti. PK te tabele je sestavljen iz PK obeh povezanih tabel, tj. "haveJobs_PK (Employees_employeeID, Jobs_jobID)". Vsak atribut tega PK se uporablja tudi kot FK, ki se navezuje na PK povezane tabele (tj. "haveJobs_Employees_FK (Employees_employeeID)" in "haveJobs_Jobs_FK (Jobs_JobID)". Takšna vmesna tabela lahko vsebuje še dodatne atribute, ki podrobneje opredeljujejo razmerje, npr. "startDate" in "endDate" delovnega mesta.
- **Izbirnost:** Privzeto je za vsako povezavo obstoj vsaj enega izvornega in enega ciljnega primerka neobvezen. Če je obvezen, je treba atributom FK dodeliti omejitev NOT NULL. V našem primeru vsi trije FK potrebujejo omejitev NOT NULL, ker (i) zaposleni morajo biti povezani z oddelkom (ii) in vsaj enim delovnim mestom ter (iii) mora biti delovno mesto povezano z vsaj enim zaposlenim.

Grafični uporabniški vmensik - elementi ODM za upravljanje relacijskega modela. Naslednje slike prikazujejo pregled najpomembnejših elementov grafičnega vmesnika *ODM* za upravljanje sheme R-BP, ki so precej podobni zgoraj omenjenim elementov za upravljanje logičnega modela. Slika 2.9 prikazuje, kako definiramo nove tabele in nove FK ter vizualizacijo RM v obliki seznama in grafičnega prikaza. V grafičnem prikazu so najpomembnejše grafične oznake in simboli opremljeni z opombami.



Slika 2.9: Elementi grafičnega uporabniškega vmesnika za upravljanje stolpcev v RM.

Slika 2.10 prikazuje pogovorno okno za definiranje stolpcev, ki ga lahko sprožite z dvojnim klikom na tabelo. Dialog je v veliki meri podoben dialogu za upravljanje atributov, ki je prikazan na sliki 2.4.



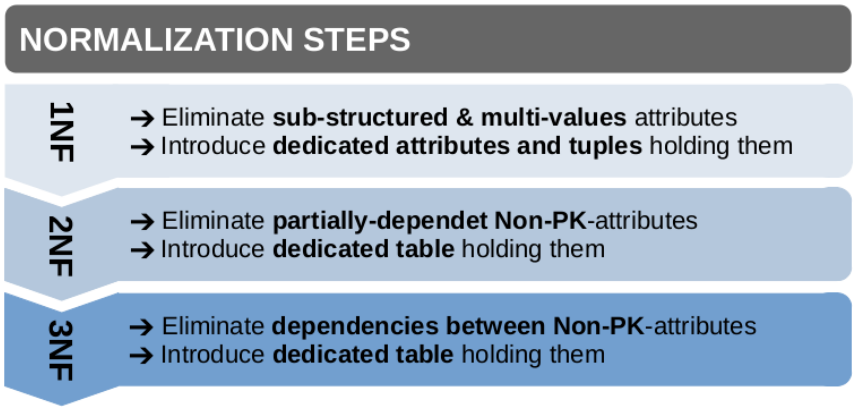
Slika 2.10: Elementi grafičnega uporabniškega vmesnika za upravljanje RM.

Shema R-BP, neodvisna od fizičnega shranjevanja podatkov. Čeprav je shema R-BP, ki temelji na relacijskem modelu, že bolj konkretna in specifična za izbran SUBP v primerjavi z

logičnim modelom (npr. “izvedba” povezav na osnovi koncepta FK), se moramo zavedati, da še vedno ne zajema kompleksnih mehanizmov shranjevanja BP (t.j. "*fizični*"*nivo*), npr. koliko datotek se uporablja za shranjevanje tabel ali na katerih strežnikih se shranjujejo podatki. Dodatna pomembna prednost abstrakcije, ki ne zajema fizičnega shranjevanja, je, da je mogoče izdelati nove sheme R-BP ali spremeniti obstoječe, ne da bi morali skrbeti za fizično organizacijo podatkov na pomnilnih medijih.

2.1.3 Normalizacija

Shemo R-BP, ki je bila ustvarjena na podlagi logične sheme, pogosto lahko še izboljšamo, da bolje dosega *merila kakovosti*, ki veljajo za relacijske baze podatkov. Tako si želimo, da bo BP idealna za *data read access* v smislu poizvedb po BP in da optimalno podpirala *data manipulation*, torej vstavljanje, posodabljanje in brisanje podatkov. Čeprav se zdi, da si navedena cilja nasprotujeta, imamo na voljo koncept “proces normalizacije”. Normalizacija opisuje sistematičen proces v treh korakih: *prva normalna forma*, *druga normalna forma* in *tretja normalna forma*, ki vodijo do izboljšanih shem R-BP (Pregled normalnih form prikazuje slika 2.11). V nadaljevanju so podrobneje opisane tri normalne forme na našem primeru.



Slika 2.11: Pregled korakov normalizacije.

Prva normalna forma. Prva normalna forma zahteva, da je vsak atribut tabele “atomarni”, kar pomeni, da niso dovoljene niti (i) *podstrukture* (npr. če bi namesto našega preprostega atributa “locationName” alternativni atribut “location” vseboval vrednosti o ulici, poštni številki, mestu in državi) niti (ii) *več vrednosti* (npr. atribut “location” lahko vsebuje tudi več naslovov določenega oddelka. Podstrukture znotraj vrednosti se lahko odpravijo tako, da se podstruktura predstavi z več atomarnimi atributi (npr. za vsak del naslova), več vrednostni pa tako, da se za vsako vrednost uporabi nov zapis (tj. vsak naslov se zapiše v ločeno vrstico).

Ena glavnih *prednosti* prve normalne forme je, da so lahko poizvedbe bolj specifične, saj je npr. mogoče ločeno dostopati do vsakega dela naslova in vsakega (zapisa) izmed več naslovov.

Druga normalna forma. Druga normalna forma zahteva, (i) da so tabele v *prvi normalni formi* in (ii) da so v primeru sestavljenega PK (tj. sestavljenega iz dveh ali več atributov), vrednosti opisnih atributov (tj. atributov, ki niso zajeti v PK) *unikatno določene samo s celotnim sestavljenim PK*. Predpostavimo, da tabela Departments vsebuje sestavljen PK z atributoma ’departmentName’ in ’locationName’. Ta PK pravilno določa vrednost atributa "annualRevenue", ki ni PK, saj seveda domnevamo, da je vrednost "annualRevenue"odvisna od oddelka in lokacije. Vrednosti drugega atributa, ki ni PK, “locationAddress”, pa bi bile že odvisne od dela PK, in sicer od imena lokacije. Da bi odpravili to delno odvisnost, moramo *razbiti* tabelo Departments tako, da informacije o lokaciji izločimo v ločeno tabelo “Locations” in obe tabeli povežemo prek FK.

Ena glavnih prednosti druge normalne forme je, da preprečuje *shranjevanje redundantnih oz.*

odvečnih podatkov, saj je v našem primeru določen naslov shranjen samo enkrat in se lahko ponovno uporabi za različne oddelke na istem naslovu. S tem se ne zmanjšuje le prostor za shranjevanje, temveč se v prvi vrsti preprečujejo tudi nepravilnosti v podatkih, tako imenovane *anomalije*, npr. če se obdelave podatkov ne izvedejo na vseh podvojenih podatkih (npr. v primeru selitve več oddelkov na drug naslov je treba naslov spremeniti le enkrat). Pomembno je tudi to, da pri tabelah, ki imajo enostavne (nesestavljene) PK (npr. številčni ID za vsako tabelo), shema R-BP "samodejno" izpolnjuje drugo normalno formo, saj delne odvisnosti ne morejo obstajati.

Tretja normalna forma. Tretja normalna forma zahteva, (i) da je tabela v *drugi normalni formi* in (ii) da je vsak *atribut, ki ni PK, enolično identificiran samo s PK* in ne s katerim koli drugim opisnim atributom, kar vodi do *tranzitivnih odvisnosti*. Na primer, če bi tabela "Departments" vsebovala poleg PK "departmentID" tudi dva opisna atributa "locationName" in "locationAddress", potem bi bil "locationAdress" že enolično identificiran z "locationName" (če predpostavimo, da je vsako "locationName" povezano z različnim naslovom), česar tretja normalna forma ne dovoljuje. Da bi odpravili takšno situacijo, je potrebno informacije o lokaciji izločiti v ločeno tabelo "Locations" in jo prek ustreznega FK povezati s tabelo Departments.

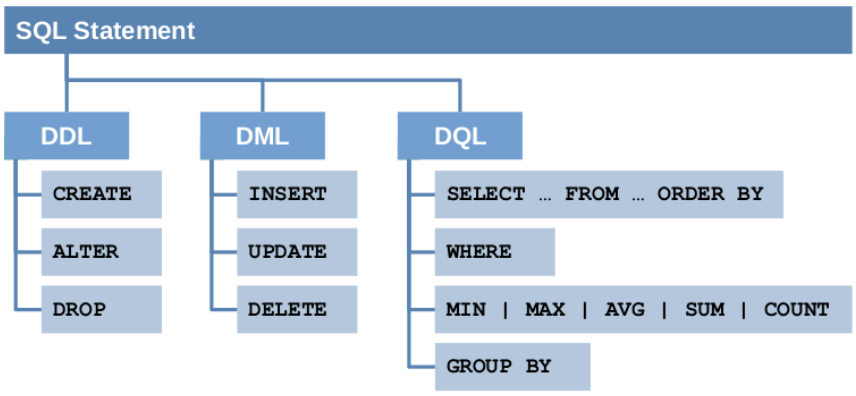
Prednost je podobna kot pri drugi normalni formi - zmanjšanje redundance in s tem morebitnih nepravilnosti v primeru obdelave podatkov.

2.2 Kako upravljati shemo relacijske BP - SQL-DDL

Do sedaj smo grafično modelirali shemo relacijske BP na različnih abstraktnih nivojih kot pri *malokodnem razvoju (ang. low-code development)*. Zanima nas, kako je mogoče te grafične modele "programsko" implementirati v relacijsko BP tako, da jih je bo mogoče uporabiti za upravljanje dejanskih podatkov spletne aplikacije. Vprašanja, na katera bomo odgovorili v nadaljevanju, so torej naslednja:

- Kako povedati BP, da naj *izdela zahtevane tabele*, najbolje iz grafičnih modelov, in ali imamo kakšne druge možnosti, če takšni grafični modeli ne obstajajo?
- Kako izgleda "programski jezik" SUBP, ki ustvarja tabele?

SQL = DDL | DML | DQL. Najprej odgovorimo na drugo vprašanje. Jezik, ki se uporablja za ustvarjanje tabel, in tudi za druge namene obdelave podatkov, se imenuje *standardiziran strukturiran poizvedovalni jezik (SQL)*. Ponuja široko paleto različnih *"programskih stavkov BP"*, ki jih lahko glede na njihov namen razvrstimo v različne kategorije (glejte sliko 2.12), najpomembnejši so: (i) stavki za upravljanje sheme R-BP (npr, ustvarjanje, spreminjanje in brisanje tabel), zajeti v *DDL (Data Definition Language)*, ki je osrednja tema tega poglavja, (ii) stavki za obdelavo podatkov znotraj sheme (npr. vstavljanje, posodabljanje in brisanje podatkov znotraj tabel), imenovani *DML (Data Manipulation Language)*, ki bodo obravnavane v poglavju 2.3, in (iii) stavki za poizvedovanje, imenovani *DQL (Data Query Language)*, ki so obravnavane v poglavju 2.4.



Slika 2.12: Kategorije stavkov SQL.

Pet načinov ustvarjanja tabel. Skladno z orodjem podjetja ORACLE, obstaja pet različnih možnosti za ustvarjanje tabel, tj. implementacijo sheme R-BP (za pregled glejte tudi sliko 2.45):

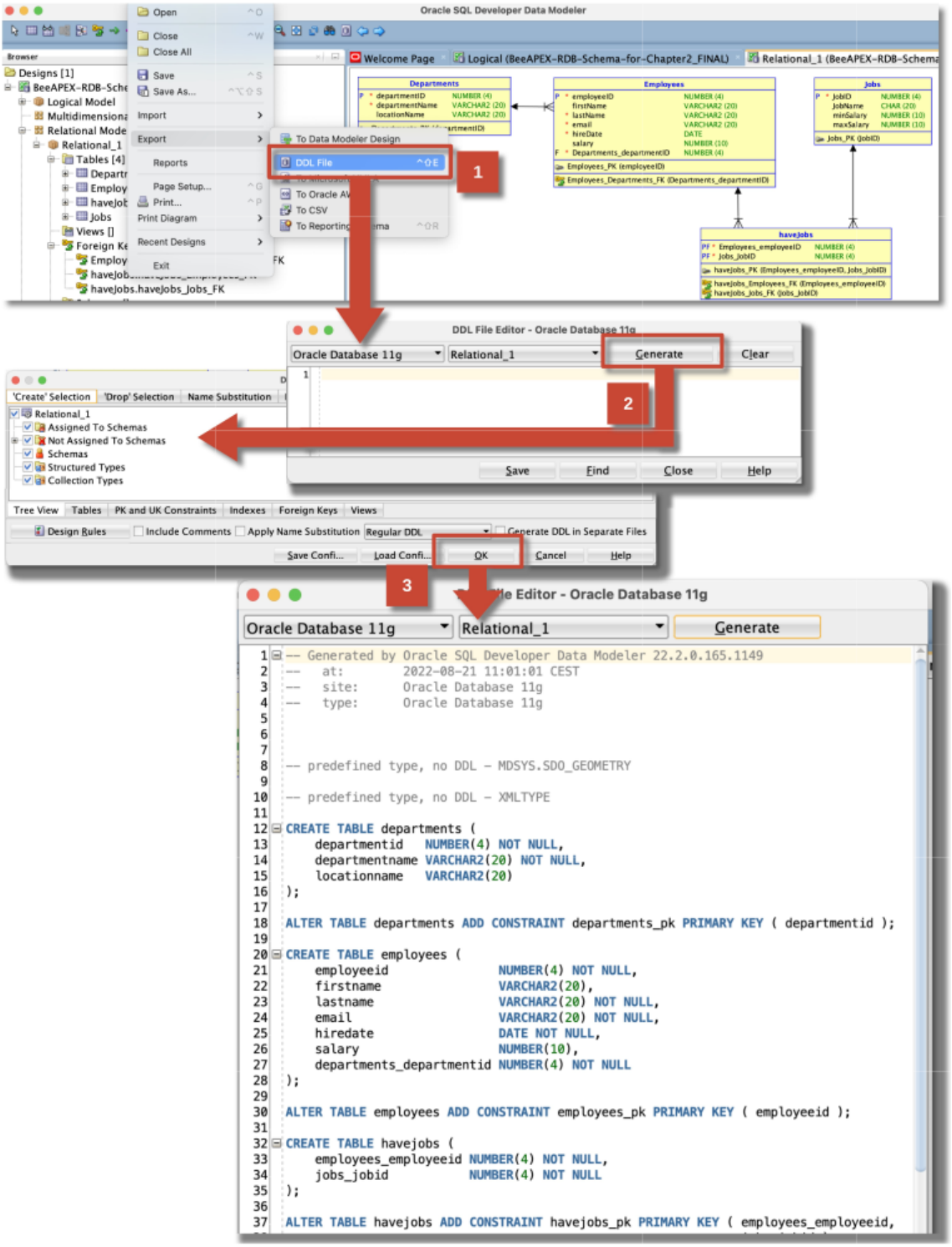
1. **Samodejno generiranje tabel z uporabo ODM.** To je možnost, ki zahteva najmanj truda, saj zadostuje, da sprožimo *avtomatsko generiranje tabel* z nekaj kliki na gumbe *z uporabo ODM* (prim. podpoglavje 2.2.1).
2. **Samodejno generiranje tabel z uporabo Quick SQL.** Druga možnost je uporaba "kratkega zapisa" SQL, ki ga zagotavlja orodje ORACLE Quick SQL (prim. podpoglavje 2.2.2). To je odlična izbira, če ni na voljo grafičnih (logičnih oz. fizičnih) modelov sheme R-BP ali če je zahtevano hitro testiranje.
3. **Ročna izdelava tabel z uporabo SQL-DDL.** Seveda obstaja tudi možnost programskega določanja tabel z uporabo običajnega jezika SQL, kar zagotavlja prednosti, saj je mogoče vsako podrobnost v zvezi s specifikacijo tabele konfigurirati glede na potrebe in ni odvisna od včasih neoptimalnih postopkov samodejnega kreiranja tabel (glejte podpoglavje 2.2.3).
4. **Ročno ustvarjanje tabel z uporabo ORACLE Object Browser (OB).** Uporabnikom, ki ne poznajo jezika SQL, brskalnik objektov omogoča ustvarjanje novih tabel in po potrebi njihovo spreminjanje na podlagi obrazca (prim. podpoglavje 2.2.4).
5. **Samodejno ustvarjanje tabel z uporabo ORACLE Data Workshop.** Obstaja še ena možnost samodejnega ustvarjanja tabel, ki pa se lahko uporabi le, če so podatki že shranjeni v zunanjih datotekah (npr. v datotekah EXCEL). Uporabimo lahko orodje ORACLE Data Workshop. Ta možnost je podrobneje opisana v poglavju 4.

V nadaljevanju so podrobneje obravnavane prve štiri možnosti.

2.2.1 Samodejno generiranje tabel z uporabo ODM

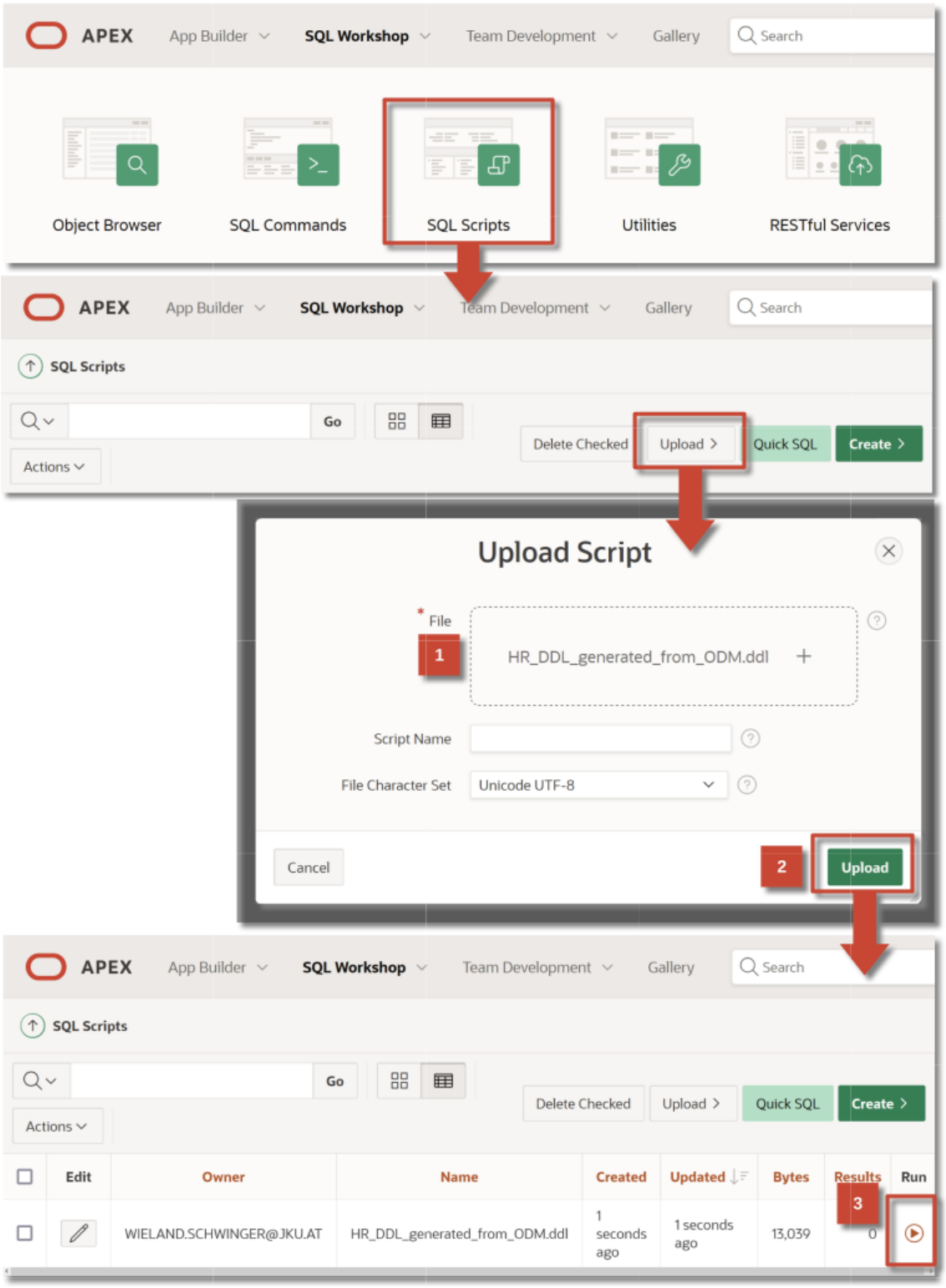
Predstavljena možnost ustvarjanja tabel zahteva najmanj truda, saj zadostujeta dva koraka. Predpogoj je, da je bila shemo R-BP razvila v *ODM*.

Najprej izvozimo shemo R-BP v *SQL-script* (glejte sliko 2.13, koraki 1 do 3), ki je pravzaprav preprosta besedilna datoteka, ki vsebuje vse potrebne stavke SQL-DDL, da se ustvarijo tabele v R-SUBP. Sintaksa teh stavkov SQL je podrobneje opisana v podpoglavju 2.2.3.



Slika 2.13: Izvoz sheme relacijske BP v skripto DDL.

Nato poženemo skripto SQL, ki naroči R-SUBP, da naj ustvari prazne tabele v svojem pomnilniku. Kot je prikazano na sliki 2.14, je potrebno skripto naložiti prek ORACLE SQL Workshop (glejte 1-2) in jo izvesti (glejte 3).



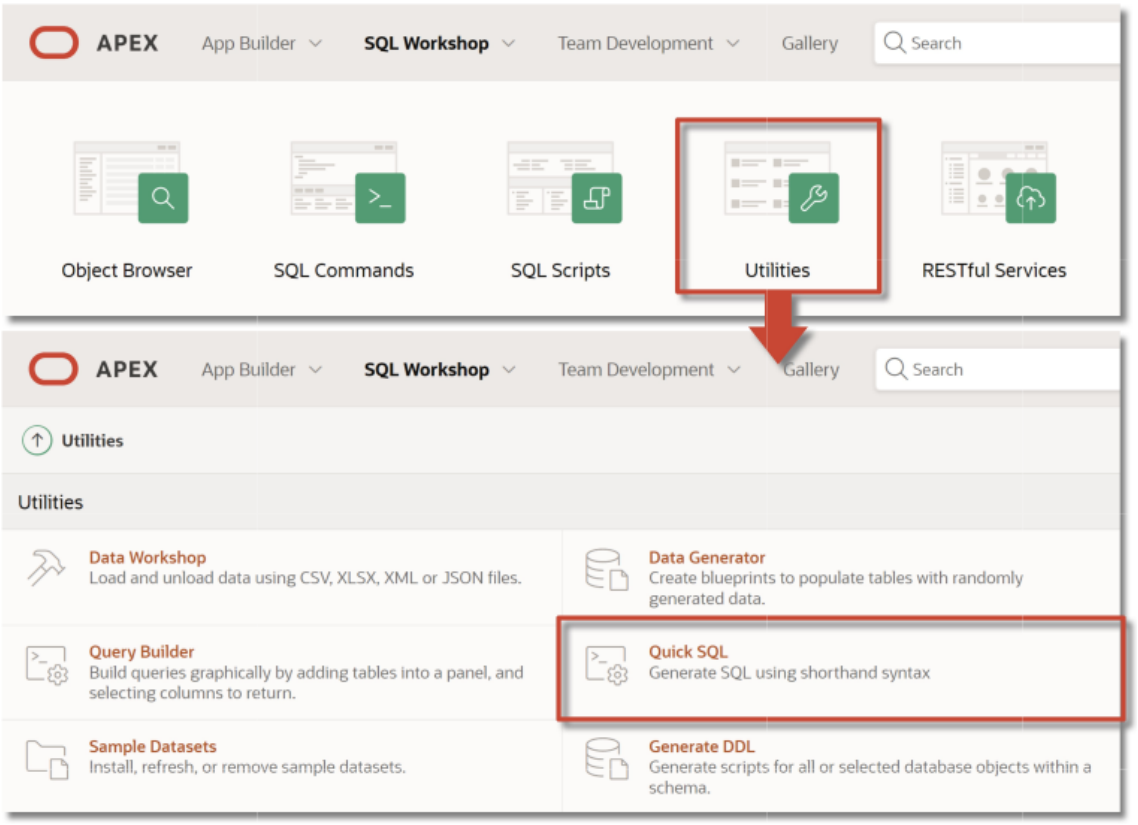
Slika 2.14: Nalaganje in izvajanje skripte SQL.

2.2.2 Samodejno generiranje tabel z uporabo Quick SQL

Pri gradnji BP spletne aplikacije je zelo priporočljivo upoštevati različne ravni abstrakcije v smislu logičnih in fizičnih relacijskih modelov. Kljub temu *Quick SQL* zagotavlja preprosto in intuitivno alternativo za ustvarjanje tabel, ki je precej enostavnejša od jezika SQL, na primer za namene hitrega testiranja. Quick SQL nadalje omogoča samodejno ustvarjanje ustrezne skripte SQL. Ko je

skripta SQL ustvarjena, jo je mogoče prilagoditi in dopolniti ter na koncu izvesti, da se ustvarijo tabele. Ponovno želimo poudariti, da čeprav je program Quick SQL namenjen poenostavljanju procesa ustvarjanja tabel, nikakor ne nadomešča procesa modeliranja podatkov (glejte podpoglavje 2.1).

Do urejevalnika ORACLE Quick SQL je mogoče dostopati prek "Utilities" v SQL Workshop (prim. sliko 2.15.



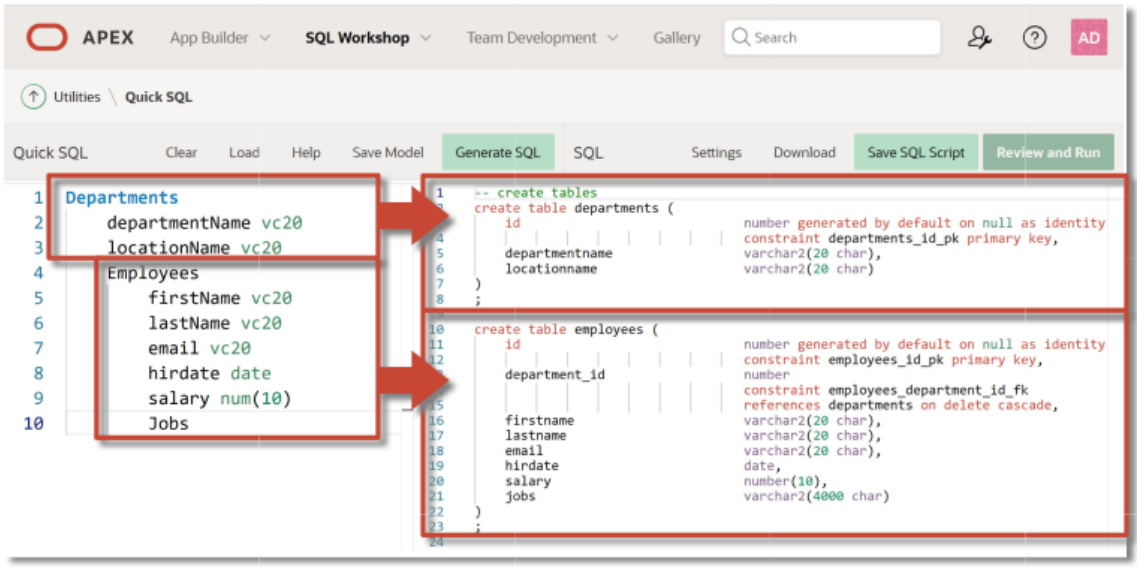
Slika 2.15: Dostop do Quick SQL prek SQL Workshop

Poslužujmo se preproste sintakse Quick SQL, ki v osnovi uporablja nekaj načel oblikovanja, da določimo zahteve za generiranje tabel:

- **Tabele Starši:** Imena nadrejenih tabel (starši) se vnesejo brez kakršne koli alineje. Imena tabel se oblikujejo samodejno, morebitni presledki se nadomestijo s podčrtaji.
- **Atributi:** Imena atributov se vnesejo z enotnim zamikom dveh ali več presledkov. Tako kot pri imenih tabel se tudi imena atributov samodejno oblikujejo, pri čemer se morebitni presledki nadomestijo s podčrtaji.
- **Razmerja med tabelami Starš-otrok:** Povezave med starši in otroki oblikujemo na način, da podrejene tabele (otroci) vstavimo pod nadrejene tabele (starši). Podrejene tabele morajo biti zamaknjene na isto raven kot stolpci v nadrejeni tabeli.
- **Podatkovni tipi:** Na podlagi angleškega besedila imena stolpca, če ni naveden podatkovni tip, SUBP samodejno določi podatkovne tipe. Če mora imeti stolpec določen podatkovni tip, so možni naslednji podatkovni tipi: `NUM`, `INT`, `VC` (za `VARCHAR2`) ali `DATE`. Podatkovni tip lahko določimo za imenom stolpca, ki ju ločimo s presledkom. Če želimo določiti dolžino niza `VC`, moramo vnesti `VCn`, kjer je `n` dolžina `VARCHAR2`. Sintaksa je prikazana v meniju "Help".

Slika 2.16 prikazuje primer zapisa Quick SQL, ki prikazuje na levi strani nadrejeno tabelo "Departments" in podrejeno tabelo "Employees", v podoknu SQL Output pa samodejno ustvarjeno

skripto SQL. SQL se ustvari po vsaki pritisku tipke Enter. Končni zapis SQL lahko shranite za nadaljnjo uporabo oz. morebitno prilagoditev in ga izvedete, da ustvarite prazne tabele.

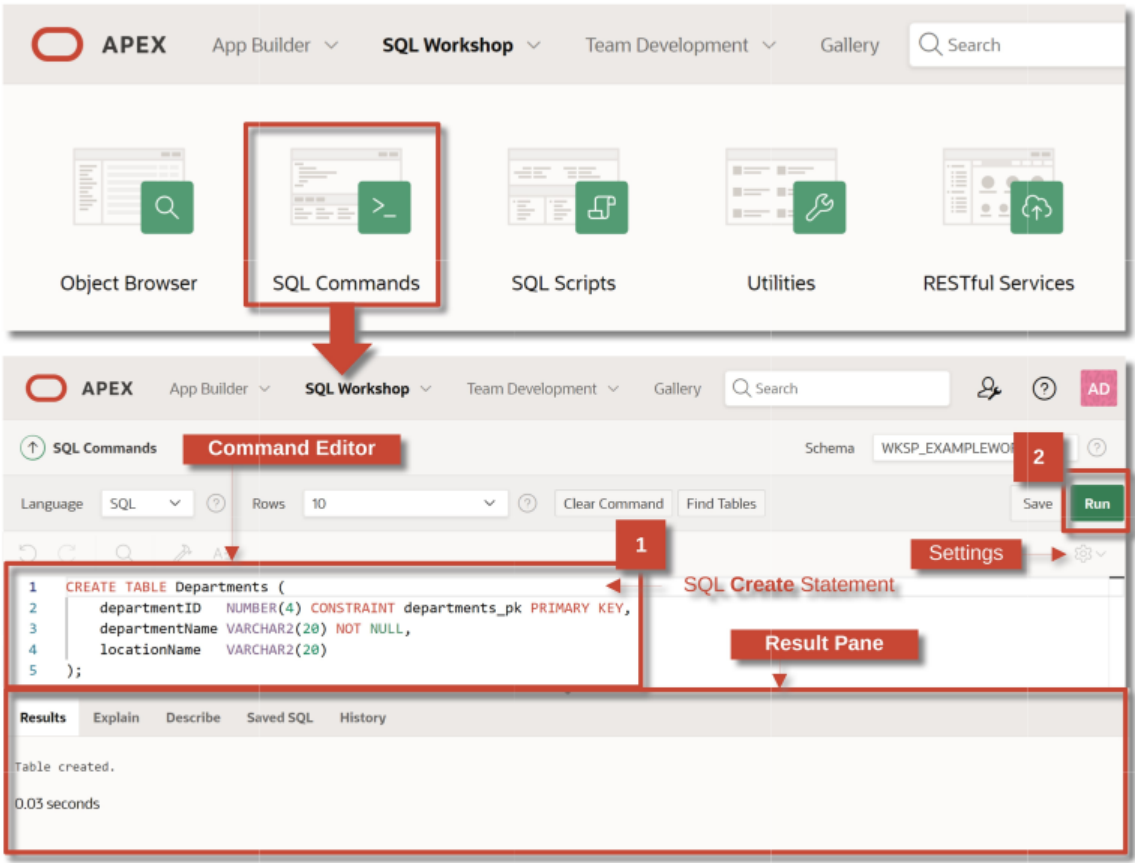


Slika 2.16: Ustvarjena skripta SQL, ki temelji na Quick SQL.

Quick SQL ima tudi sintakso za določanje omejitev (FK in dovoljene vrednosti) za določeno podatkovno polje ter za ustvarjanje testnih podatkov.

2.2.3 Ročno ustvarjanje tabel z uporabo SQL-DDL

Malokodno ustvarjanje tabel, ki je opisano v podpoglavjih 2.2.1 in 2.2.2, lahko v primeru poznavanja SQL nadomestite z ukazi SQL-DDL `CREATE TABLE` v urejevalniku SQL-Command-Editor. Ta se nahaja v *SQL-Workshop* (slika 2.17). Po končanem pisanju ukazov SQL z uporabo SQL-DDL (1) lahko shranite in izvedete nastalo skripto SQL (2). Tako ustvarite prazne tabele (slika 2.17).



Slika 2.17: SQL Command Editor.

Stavki SQL so bolj ali manj preprosti, izhajajo iz angleščine in imajo posebno sintakso. Stavek `CREATE TABLE` je sestavljen iz *specifičnih ključnih besed SQL*, kot je `CREATE`, ter *določanje omejitev* (podatkovni tipi, primarni in tuji ključ). Na sliki 2.18 so prikazane vse ključne besede in drugi sintaktični simboli, značilni za SQL (kot sta oklepaj in vejica), zapisani z velikimi črkami in obarvani modro. Ključne besede SQL ne razlikujejo velikih in malih črk, kar pomeni, da npr. `CREATE` pomeni enako kot `create`. Definicija atributa je podana v oklepaju, atributi so ločeni z vejicami. Najbolje je, če je vsak atribut opredeljen v svoji vrstici, čeprav so to le neformalne konvencije oblikovanja in niso del sintakse SQL. Omejitve je mogoče opredeliti z eksplicitnim imenom (npr. departments_pk), ki ni obvezno (prim. npr. `NOT NULL` za departmentName), ampak služi boljši berljivosti. Na koncu vsakega stavka SQL je podpičje.

```
CREATE TABLE Departments
(   departmentID   NUMBER(4), CONSTRAINT departments_pk PRIMARY KEY,
    departmentName VARCHAR2(20) NOT NULL,
    locationName   VARCHAR2(20)
);
```

Slika 2.18: Stavek SQL za ustvarjanje tabele `Departments`.

Poleg stavka `CREATE TABLE` obstaja tudi stavek za brisanje tabele `DROP TABLE` , ki deluje ne glede na to, ali je tabela prazna ali pa že vsebuje podatke, ki se skupaj s tabelo izbrišejo (glejte sliko 2.19).

```
DROP TABLE Departments;
```

Slika 2.19: Stavek SQL za brisanje tabele `Departments`.

Definicijo že obstoječih tabel lahko tudi spremenite, npr. z dodajanjem novih atributov ali s spreminjanjem podatkovnih tipov (glejte sliko 2.20):

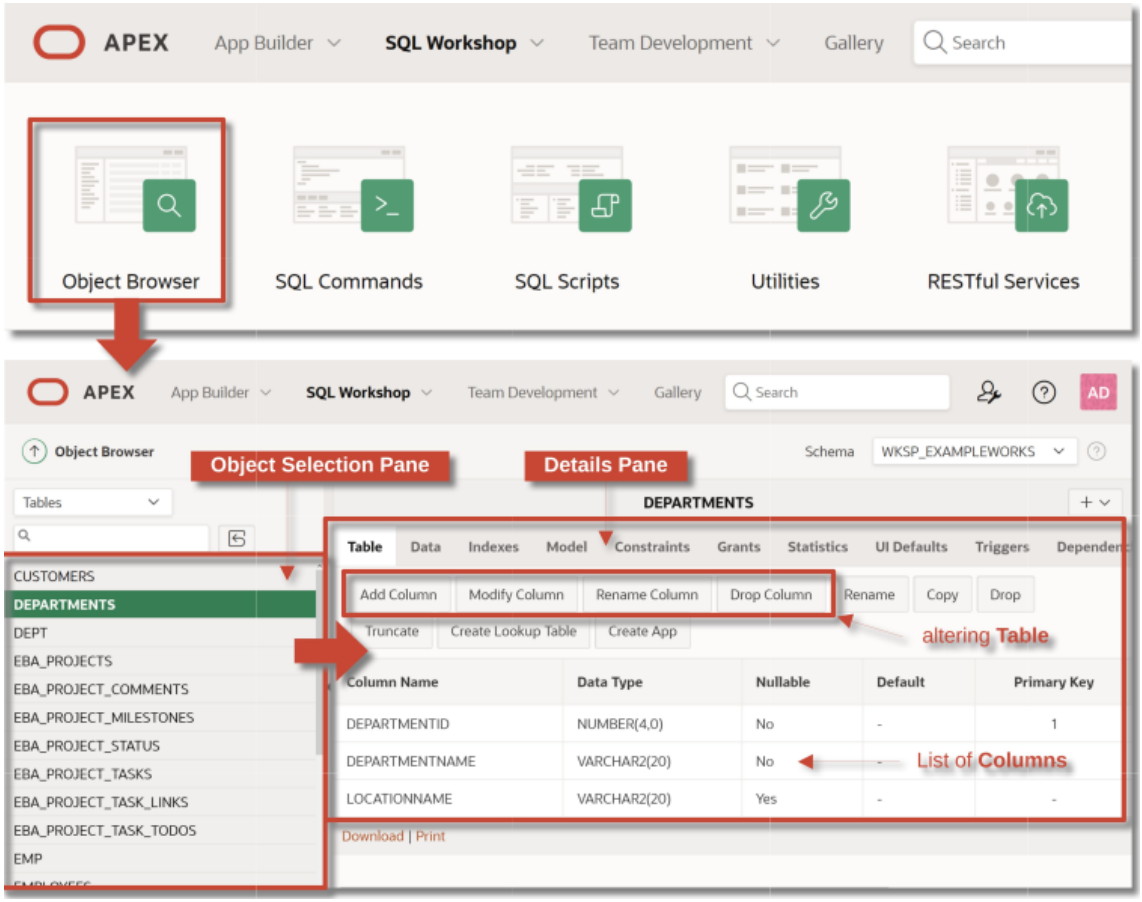
```
ALTER TABLE Departments
      ADD annualRevenue NUMBER(10);
```

Slika 2.20: Stavek SQL za spreminjanje tabele `Departments`

V primeru, ko želimo spremeniti strukturo tabele, ki že vsebuje podatke, so spremembe delno omejene, saj lahko npr. sprememba podatkovnega tipa povzroči nenamerne posledice za obstoječe podatke. Temu vprašanju se v tem poglavju ne posvečamo podrobneje.

2.2.4 Ročno upravljanje tabel z orodjem ORACLE Object Browser

Orodje Object Browser, ki je del ORACLE SQL Workshop omogoča poleg pregledovanja ustvarjenih tabel, tako imenovanih *"Objects"* v terminologiji podjetja ORACLE, skupaj z njihovimi ustreznimi atributi in omejitvami, tudi njihovo spreminjanje in ustvarjanje novih z uporabo preprostega čarovnika z grafičnim vmesnikom (glejte Sliko 2.21).



Slika 2.21: Upravljanje tabel z ORACLE Object Browser.

2.3 Kako upravljati podatke - SQL-DML

Prazne tabele, ustvarjene v prejšnjem podpoglavju, zdaj "čakajo" na zapolnitev s podatki. Podatke je mogoče tudi spreminjati oz. izbrisati. V ta namen uporabimo stavke SQL-DML `INSERT`, `UPDATE` in `DELETE`, ki jih bomo na kratko predstavili v naslednjem podpoglavju 2.3.1 na našem primeru. Druga možnost, ki ne zahteva programske kode, je ORACLE Object Browser ("OB"), ki omogoča ročen vnos podatkov v tabele in je primeren za uporabnike brez znanja SQL (glejte podpoglavje 2.3.2). Tretja alternativa za vstavljanje podatkov, ki je precej omejena, je Quick SQL (glejte podpoglavje 2.3.3). Naštete možnosti so prikazane na Sliki 2.45.

2.3.1 Upravljanje podatkov z uporabo SQL-DML

Za obdelavo podatkov z uporabo SQL-DML se uporablja SQL Command Editor (glejte sliko 2.17), ki smo ga uporabili za ustvarjanje tabel. V nadaljevanju je prikazanih nekaj primerov za ponazoritev sintakse in funkcionalnosti jezika SQL-DML.

`INSERT` se uporablja za *dodajanje novih vrstic oz. zapisov* v tabelo, kot je razvidno iz naslednjega primera (glejte sliko 2.22):

```
INSERT INTO Departments (departmentID, departmentName, locationName)
     VALUES (10. 'ACCOUNTING', 'NEW YORK');
```

Slika 2.22: Stavek SQL za vpisovanje novih podatkov v tabelo `Departments`.

Poleg te programske možnosti smo v podpoglavju 2.3.3 omenili tudi možnost uporabe Quick SQL za vstavljanje podatkov, ki pa je omejena le na vstavljanje naključnih podatkov.

`UPDATE` se uporablja za spreminjanje oz. posodabljanje ene ali več vrstic podatkov v tabeli. Naslednji primer je najpreprostejša oblika izjave `UPDATE`, ki plače vseh zaposlenih poveča za 10 % (glejte Sliko 2.23):

```
UPDATE Employees
     SET salary = salary * 1.1;
```

Slika 2.23: Stavek SQL za povečanje plače vseh zaposlenih za 10% v tabeli `Employees`.

Naslednji primer prikazuje nadgradnjo z uporabo "`WHERE`", da se poveča plačo zaposlenega s priimkom "Miller" (glejte Sliko 2.24):

```
UPDATE Employees
     SET salary = salary * 1.1
     WHERE lastName = 'Miller';
```

Slika 2.24: Stavek SQL za povečanje plače zaposlenega s priimkom "Miller".

Nenazadnje, lahko z enim stavkom posodobimo tudi več atributov tabele, kot je razvidno iz naslednjega primera (glejte sliko 2.25):

```
UPDATE Employees SET
     jobID  = '4711',
     salary = salary + 1000,
     departmentID = 120;
```

Slika 2.25: Stavek SQL hkrati spreminja več atributov v tabeli `Employees`.

DELETE se uporablja za brisanje ene ali več vrstic podatkov iz tabele. Naslednji primer je najpreprostejša oblika stavka DELETE, ki izbriše vse podatke v tabeli Employees (glejte Sliko 2.26).

```
DELETE FROM Employees;
```

Slika 2.26: Stavek SQL izbriše vse podatke iz tabele Employees.

Predhodni stavek lahko razširimo z uporabo WHERE. Omogoča selektivno brisanje zaposlenega z ID delovnega mesta “4711” (glejte sliko 2.27).

```
DELETE FROM Employees
     WHERE jobID  = '4711';
```

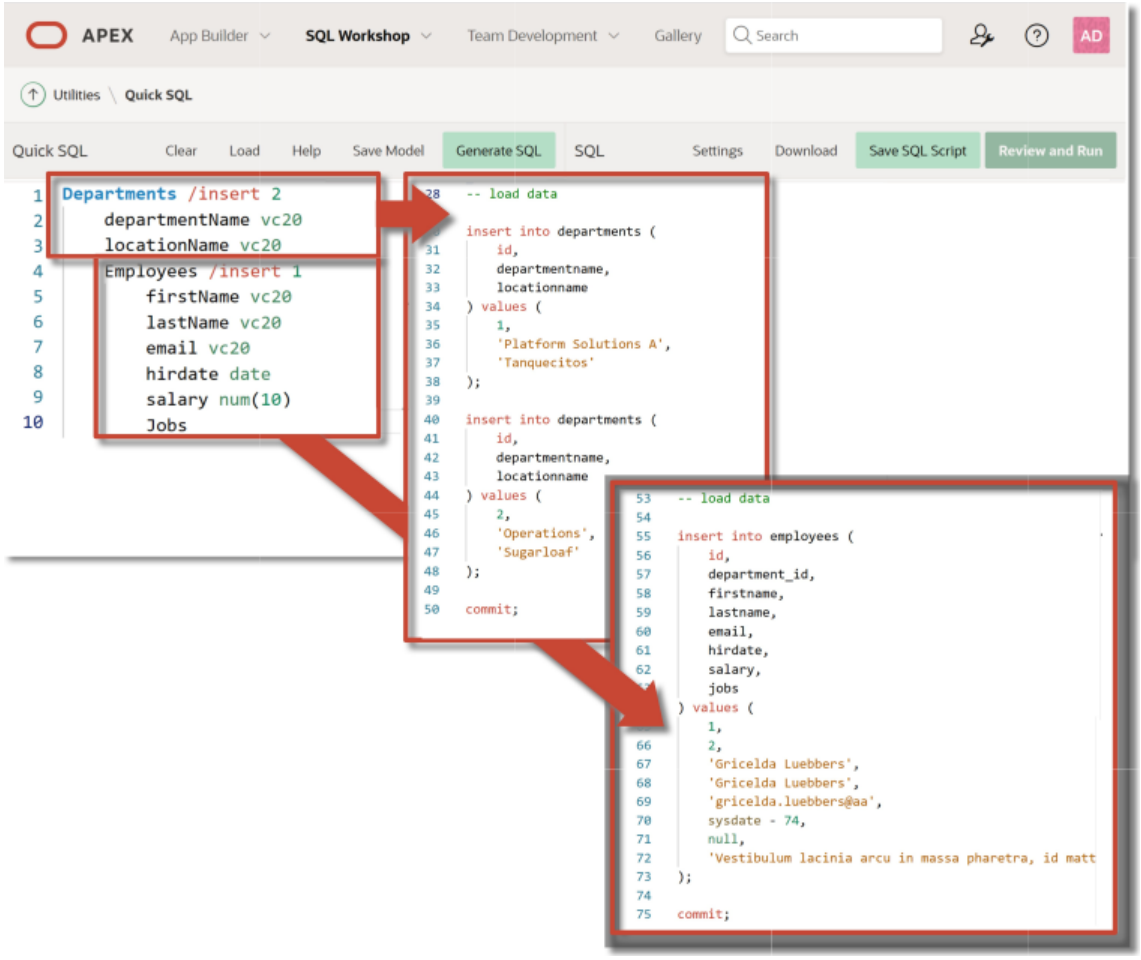
Slika 2.27: Stavek SQL izbriše le nekatere podatke iz tabele Employees.

2.3.2 Upravljanje podatkov z ORACLE Object Browser (OB)

Poleg navedenih možnosti za upravljanje podatkov jih lahko tudi ročno vnašamo, spreminjamo oz. brišemo z urejevalnikom *Object Browser*, glejte podpoglavje 2.2.4, slika 2.21, z uporabo podokna “Data”. Kadar uporabljamo OB za upravljanje podatkov, ORACLE samodejno prevede vse postopke v stavke SQL-DML.

2.3.3 Upravljanje podatkov z uporabo Quick SQL

Z uporabo Quick SQL je mogoče samodejno vstaviti nekatere podatke v tabele. Slika 2.28 prikazuje primer z uporabo stavkov `/insert 2` in `/insert 1`. Ta stavka določita število zapisov z naključnimi vrednostmi, ki se vstavijo v tabele.



Slika 2.28: Vstavljanje naključnih podatkov s Quick SQL.

2.4 Kako poizvedovati po podatkih - SQL-DQL

Do sedaj smo predstavili različne možnosti za ustvarjanje praznih tabel v podpoglavju 2.2 in načine upravljanja podatkov v podpoglavju 2.3. V nadaljevanju se osredotočamo v podpoglavju 2.4.1), na našem kadrovskem primeru, ki mu sledi kratek pregled orodja ORACLE Query Builder (“QB”), ki omogoča “nizkokodno” alternativo za poizvedovanje za uporabnike z malo ali nič znanja SQL (prim. podpoglavje 2.4.2). Možnosti so prikazane na sliki 2.45.

2.4.1 Poizvedovanje po podatkih z uporabo SQL-DQL

Jezik SQL-DQL omogoča izjemno izraznost. Osredotočili se bomo zgolj na nekatere izmed konceptov. Kljub temu želimo podati vtis o veliki izraznosti in široki uporabnosti SQL-DQL, in sicer tako, da bomo na primeru predstavili različne možnosti poizvedovanja, ki bodo v nadaljevanju razvrščene po naraščajoči zahtevnosti. Pri tem uporabljamo orodje SQL Command Editor (glejte Sliko 2.17), ki smo ga že uporabili za ustvarjanje tabel in upravljanje podatkov.

(1) Začnimo z najenostavnejšo poizvedbo – *Poišči vse zaposlene* (npr. Slika 2.29).

```
SELECT * FROM Employees;
```

Slika 2.29: Stavek SQL, ki prikliče vse podatke iz tabele `Employees`.

Tabela (`Employees`), na katero se nanaša poizvedba, je navedena v delu `FROM`, želeni rezultat

poizvedbe pa je določen v delu `SELECT` - v danem primeru z uporabo zvezdice kot nadomestnega znaka, ki pomeni, da želimo pridobiti vrednosti vseh stolpcev.

Slika 2.30 prikazuje nekaj vrstic rezultata te poizvedbe:

EMPLOYEEID	FIRSTNAME	LASTNAME	EMAIL	HIREDATE	SALARY	DEPARTMENTS_ DEPARTMENTID
100	Steven	King	SKING	06/17/2003	24000	90
101	Neena	Kochhar	NKOCHHAR	09/21/2005	17000	90
102	Lex	De Haan	LDEHAAN	01/13/2001	17000	90
103	Alexander	Hunold	AHUNOLD	01/03/2006	9000	60
104	Bruce	Ernst	BERNST	05/21/2007	6000	60
105	David	Austin	DAUSTIN	06/25/2005	4800	60
106	Valli	Pataballa	VPATABAL	02/05/2006	4800	60
107	Diana	Lorentz	DLORENTZ	02/07/2007	4200	60
108	Nancy	Greenberg	NGREENBE	08/17/2002	12008	100
109	Daniel	Faviet	DFAVIET	08/16/2002	9000	100
110	John	Chen	JCHEN	09/28/2005	8200	100
111	Ismael	Sciarra	ISCIARRA	09/30/2005	7700	100
112	Jose Manuel	Urman	JMURMAN	03/07/2006	7800	100
113	Luis	Popp	LPOPP	12/07/2007	6900	100
114	Den	Raphaely	DRAPHEAL	12/07/2002	11000	30

Slika 2.30: Del rezultata poizvedbe za priklic vseh podatkov iz tabele `EMPLOYEES`.

Če želimo izpis (rezultat poizvedbe) urediti, na primer glede na plače zaposlenih, dodano `ORDER BY` (glejte sliko 2.31):

```
SELECT * FROM Employees ORDER BY salary;
```

Slika 2.31: Stavek SQL za priklic vseh podatkov iz tabele `Employees`, urejenih po plačah.

Slika 2.32 prikazuje rezultat poizvedbe:

EMPLOYEEID	FIRSTNAME	LASTNAME	EMAIL	HIREDATE	SALARY	DEPARTMENTS_ DEPARTMENTID
100	Steven	King	SKING	06/17/2003	24000	90
101	Neena	Kochhar	NKOCHHAR	09/21/2005	17000	90
102	Lex	De Haan	LDEHAAN	01/13/2001	17000	90
108	Nancy	Greenberg	NGREENBE	08/17/2002	12008	100
114	Den	Raphaely	DRAPHEAL	12/07/2002	11000	30
103	Alexander	Hunold	AHUNOLD	01/03/2006	9000	60
109	Daniel	Faviet	DFAVIET	08/16/2002	9000	100
110	John	Chen	JCHEN	09/28/2005	8200	100
112	Jose Manuel	Urman	JMURMAN	03/07/2006	7800	100
111	Ismael	Sciarra	ISCIARRA	09/30/2005	7700	100
113	Luis	Popp	LPOPP	12/07/2007	6900	100
104	Bruce	Ernst	BERNST	05/21/2007	6000	60
105	David	Austin	DAUSTIN	06/25/2005	4800	60
106	Valli	Pataballa	VPATABAL	02/05/2006	4800	60
107	Diana	Lorentz	DLORENTZ	02/07/2007	4200	60

Slika 2.32: Del rezultata poizvedbe za priklic vseh podatkov iz tabele `Employees`, urejenih po plačah.

(2) Kako poizvedujemo po določenih stolpcih oz. določenih vrsticah - *"Poišči ime, naslov in plačo zaposlenih, ki imajo plačo višjo od 5,000."*

Da bi rezultat poizvedbe vrnil vrednosti le nekaterih stolpcev, namesto zvezdice navedemo želene atribute, ki jih ločimo z vejicami. Da bi rezultat zajemal le izbrane zapise oz. vrstice, uporabimo `WHERE`, da določimo logični pogoj za ustrezni atribut, v našem primeru za atribut plačo. Takih pogojev je lahko več in se lahko nanašajo na različne atribute, ločimo pa jih s veznikoma `AND` oz. `OR` (glejte sliko 2.33). Za določitev vrstnega reda lahko uporabimo oklepaje kot pri izrazih pri matematiki.

```
SELECT lastName, hireDate, salary FROM Employees
WHERE salary > 5000;
```

Slika 2.33: Stavek SQL za priklic izbranih podatkov iz tabele `Employees`.

Rezultat te poizvedbe je prikazan na sliki 2.34:

LASTNAME	HIREDATE	SALARY
King	06/17/2003	24000
Kochhar	09/21/2005	17000
De Haan	01/13/2001	17000
Hunold	01/03/2006	9000
Ernst	05/21/2007	6000
Greenberg	08/17/2002	12008
Faviet	08/16/2002	9000
Chen	09/28/2005	8200
Sciarra	09/30/2005	7700
Urman	03/07/2006	7800
Popp	12/07/2007	6900
Raphaely	12/07/2002	11000

Slika 2.34: Del rezultata poizvedbe za priklic izbranih podatkov iz tabele `Employees`.

Če želimo izpis vrednosti atributa `hireDate` v drugačni obliki, lahko uporabimo funkcijo `TO_CHAR()`, ki je del jezika ORACLE SQL. Omenjena funkcija ima dva parametra. Prvi je vrednost `DATE`, drugi pa želena oblika izpisa datuma. V nadaljevanju je prikazan primer uporabe te funkcije (glejte sliko 2.35), pri čemer je za funkcijo `TO_CHAR()` uporabljen atribut `hireDate`. Nabor vseh možnosti oblikovanja datuma je na voljo v dokumentaciji ORACLE.

```
SELECT lastName, TO_CHAR(hireDate, 'Month, DD, YYYY') hireDate, salary
FROM Employees
WHERE salary > 1000;
```

Slika 2.35: Stavek SQL za priklic izbranih podatkov iz tabele `Employees` z uporabo funkcije `TO_CHAR()`.

Rezultat te poizvedbe je prikazan na Sliki 2.36:

LASTNAME	HIREDATE	SALARY
King	June, 17, 2003	24000
Kochhar	September, 21, 2005	17000
De Haan	January, 13, 2001	17000
Hunold	January, 03, 2006	9000
Ernst	May, 21, 2007	6000
Austin	June, 25, 2005	4800
Pataballa	February, 05, 2006	4800
Lorentz	February, 07, 2007	4200
Greenberg	August, 17, 2002	12008
Faviet	August, 16, 2002	9000
Chen	September, 28, 2005	8200
Sciarra	September, 30, 2005	7700
Urman	March, 07, 2006	7800
Popp	December, 07, 2007	6900
Raphaely	December, 07, 2002	11000

Slika 2.36: Del rezultata poizvedbe za priklic izbranih podatkov iz tabele `Employees` z uporabo funkcije `TO_CHAR()`.

(3) Osredotočimo se na poizvedovanje po več tabelah - *Poiščimo vse zaposlene in njihove oddelke*. Pridobivanje podatkov iz dveh ali več tabel znotraj enega stavka `SELECT` zahteva poznavanje koncepta *Join*. Join združi vrstice iz dveh ali več tabel, ki jih navedemo v delu `FROM`. Poleg tega potrebujemo `WHERE`, ki ima vlogo *povezovanja* dveh tabel - saj moramo opredeliti, da mora SUBP za vsakega zaposlenega v tabeli `Employees` vzeti njegovo vrednost Departments_ departmentID (ki je FK) in jo primerjati z vrednostjo PK v tabeli `Departments` ter vrniti ujemajoče se podatke o zaposlenih in njihovih oddelkih. Bolj formalno, ujemanje določimo s tako imenovanim *spojitvenim pogojem*, kot je razvidno iz naslednjega primera (glejte Sliko 2.37):

```
SELECT lastName, salary, departmentName
FROM Employees, Departments
WHERE Employees.Departments_departmentID = Departments.departmentID;
```

Slika 2.37: Stavek SQL za združevanje podatkov iz tabel `Employees` in `Departments`.

Rezultat te poizvedbe je prikazan na sliki 2.38:

LASTNAME	SALARY	DEPARTMENTNAME
King	24000	Executive
Kochhar	17000	Executive
De Haan	17000	Executive
Hunold	9000	IT
Ernst	6000	IT
Austin	4800	IT
Pataballa	4800	IT
Lorentz	4200	IT
Greenberg	12008	Finance
Faviet	9000	Finance
Chen	8200	Finance
Sciarra	7700	Finance
Urman	7800	Finance
Popp	6900	Finance
Raphaely	11000	Purchasing

Slika 2.38: Del rezultata poizvedbe za združevanje podatkov iz tabel `Employees` in `Departments`.

Ustrezni atributi spajanja so *opredeljeni* z uporabo imen tabel in atributov, ločenih s piko. Tako želimo izboljšati berljivost in, kadar je ime FK enako imenu PK, zagotoviti unikatno identifikacijo posameznega atributa.

(4) Oglejmo si, kako lahko združujemo vrednosti - *Izpiši število zaposlenih in vsoto njihovih plač*

Za izpis števila vseh zaposlenih v tabeli `Zaposleni` lahko uporabimo SQL funkcijo `COUNT`, ki prešteje različne `EmployeeID`. Za izračun vsote vseh vrednosti lahko uporabimo funkcijo `SUM`, ki sešteje vrednosti v stolpcu. V našem primeru želimo izračunati vsoto vseh plač, zato uporabimo atribut `salary` (glejte Sliko 2.39):

```
SELECT COUNT(employeeID), SUM(salary)
FROM Employees;
```

Slika 2.39: Stavek SQL, ki prešteje vse zaposlene in izračuna vsoto njihovih plač iz tabele `Employees`.

Rezultat poizvedbe je prikazan na sliki 2.40:

COUNT(EMPLOYEEID)	SUM(SALARY)
15	149408

Slika 2.40: Izpis števila zaposlenih in vsote plač iz tabele `Employees`.

(5) Kako združujemo določene vrstice v skupine - *Izpiši število zaposlenih, razvrščenih po njihovih plačah v padajočem vrstnem redu*

Za združevanje istih podatkov v stavku `SELECT` uporabimo `GROUP BY`. V kombinaciji s funkcijo `COUNT`, ki prešteje različne `employeeID` lahko izračunamo število vrstic v vsaki skupini. Za razvrščanje rezultatov uporabimo `ORDER BY`, ki določa, kateri atribut naj se uporabi za razvrščanje. Pri tem `ASC` določa naraščajoči vrstni red, `DESC` pa padajoči vrstni red (glejte Sliko 2.41):

```
SELECT COUNT(employeeID), SUM(salary)
FROM Employees
GROUP BY salary
ORDER BY COUNT(employeeID) DESC;
```

Slika 2.41: Stavek SQL za grupiranje podatkov po plačah in razvrščanje rezultatov po številu zaposlenih z določeno plačo, na podlagi podatkov iz tabele `Employees`.

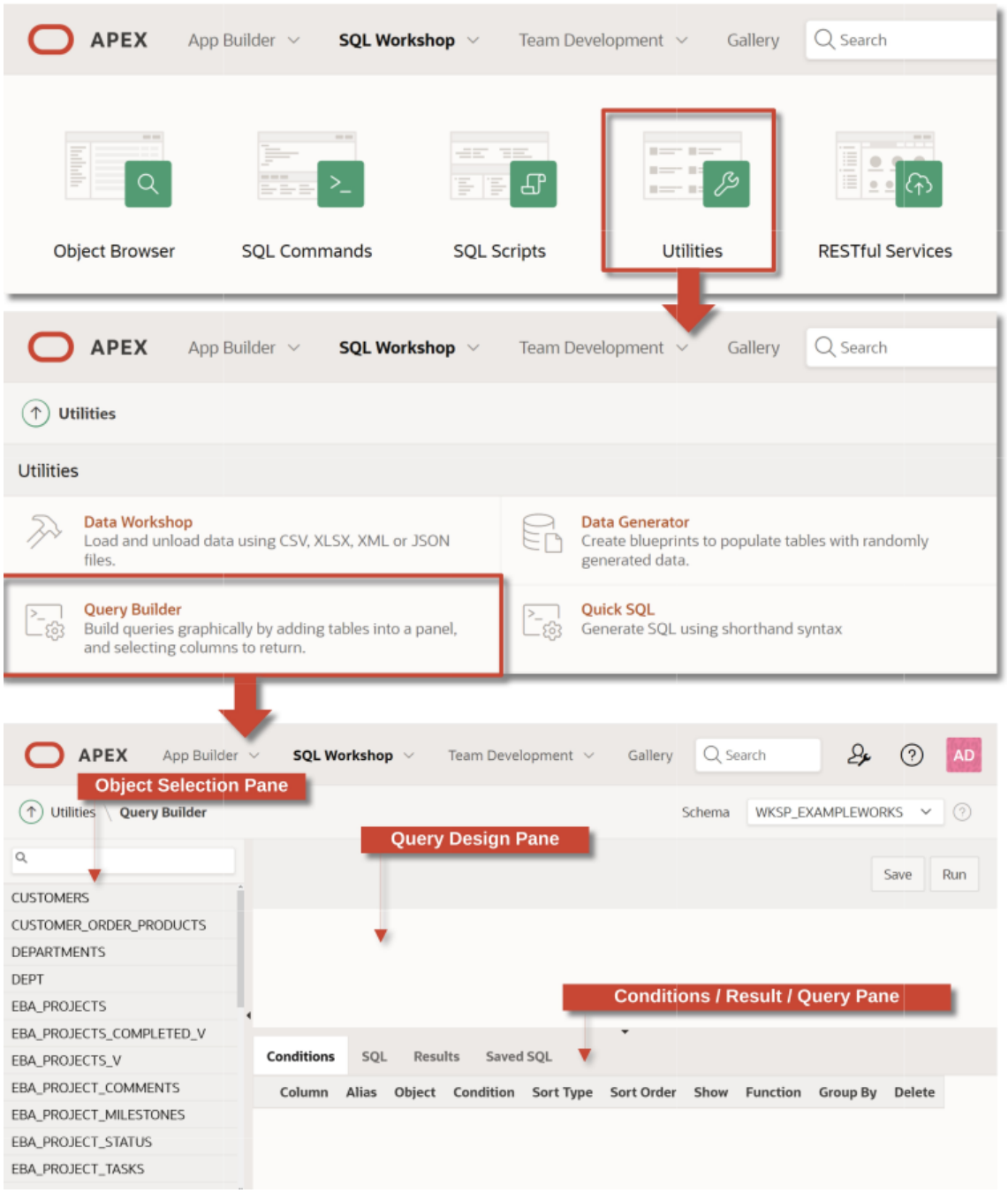
Rezultat poizvedbe je prikazan na Sliki 2.42:

COUNT(EMPLOYEEID)	SUM(SALARY)
2	9600
2	34000
2	18000
1	6900
1	7700
1	12008
1	11000
1	24000
1	6000
1	7800
1	4200
1	8200

Slika 2.42: Rezultat grupiranja podatkov po plačah in razvrščanja rezultatov po številu zaposlenih z določeno plačo, na podlagi podatkov iz tabele `Employees`.

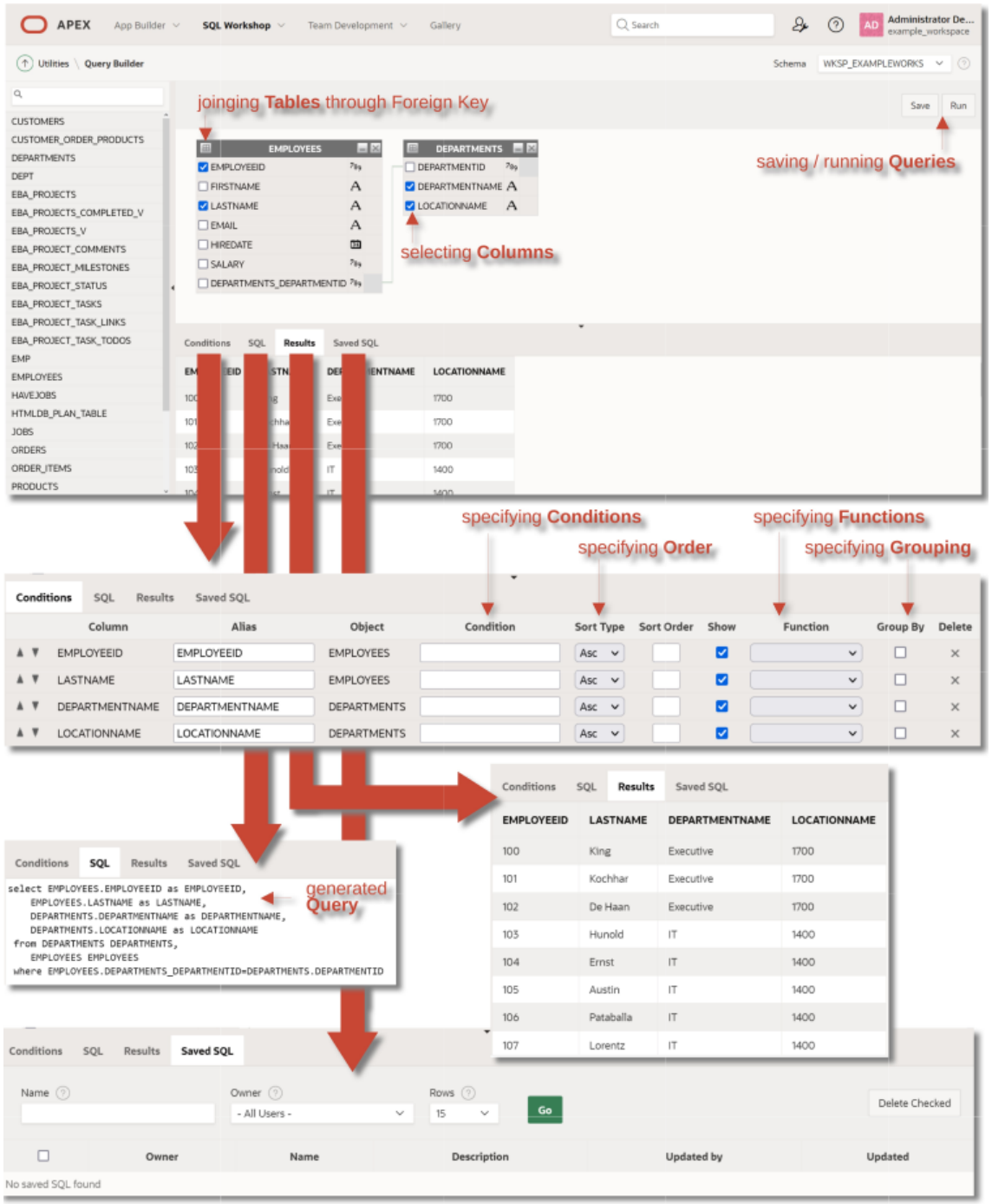
2.4.2 Poizvedovanje po podatkih z uporabo orodja ORACLE Query Builder (QB)

ORACLE Utilities v obliki Query Builder (“QB”) omogoča uporabnikom z malo ali celo brez znanja SQL, da poizvedujejo po tabelah v BP in poizvedbe shranijo za nadaljnjo uporabo. Do BP lahko dostopate na naslednji način (glejte Sliko 2.43):



Slika 2.43: Prikaz programa ORACLE Query Builder ("QB").

Koraki, potrebni za oblikovanje poizvedb, so prikazani na Sliki 2.44 z uporabo “JOIN”, ki je prikazan na Sliki 2.37 in na kratko opisan v nadaljevanju:



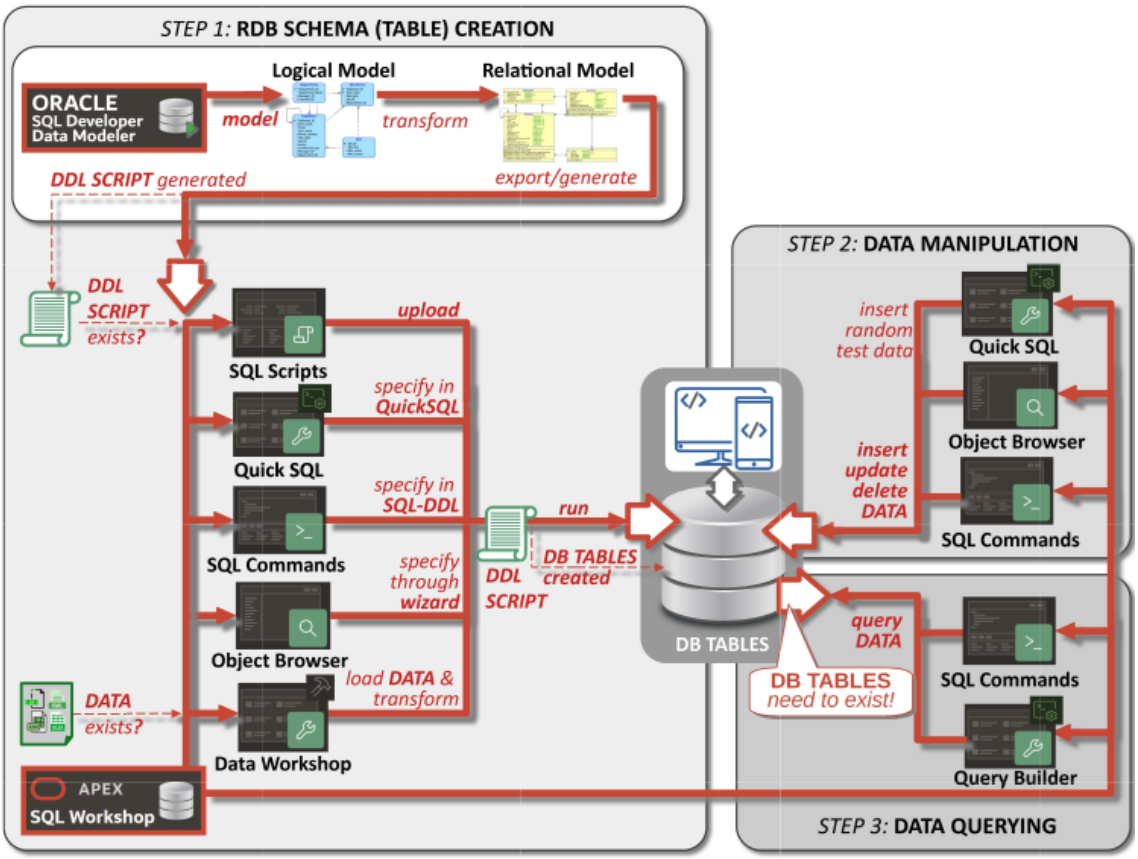
Slika 2.44: Uporaba programa Query Builder za oblikovanje poizvedb z uporabo Join.

- Izberite *ciljne tabele poizvedbe* v smislu ene ali več tabel v BP(v QB se imenujejo "objekti") v podoknu "Object Selection pane" (glejte Sliko 2.43). To predstavlja `FROM` del stavka SQL-DQL.
- Dodajte izbrane predmete v podokno "Design pane" in izberite želene stolpce. To določa `SELECT` del stavka v jeziku SQL-DQL.
- Neobvezno: Dodajte pogoje poizvedbe, tj. del `WHERE` v SQL-DQL.
- Izvedite poizvedbo in si oglejte rezultate (kliknite gumb "Run").
- Neobvezno: Oglejte si kodo SQL-DQL, ki se samodejno ustvari (kliknite zavihek "SQL").
- Neobvezno: Poizvedbo shranite za nadaljnjo uporabo (kliknite gumb "Shrani").

Poizvedbe, ki jih ustvarimo z uporabo QB, se samodejno prevedejo v ustrezne SQL-DQL stavke.

2.5 Izgradnja podatkovne plasti - širša slika

Ob branju tega poglavja ste morda zmedeni nad vsemi različnimi možnostmi, ki jih ponuja platforma orodij ORACLE za upravljanje podatkovnega nivoja spletnih aplikacij. Zato je na sliki 2.45 prikazan zbirni pregled vseh možnosti in orodij, ki so na voljo za upravljanje podatkovne baze za spletno aplikacijo.



Slika 2.45: Pregled možnosti za upravljanje podatkovne plasti.

Na levi strani sta prikazani dve različni orodji, ODM in APEX SQL Workshop. Končni cilj, tj. ustvarjanje sheme R-BP v obliki tabel (korak 1), je mogoče doseči po različnih poteh. Vse pa vodijo do ustvarjanja skript SQL, ki se na koncu izvedejo in ustvarijo prazne tabele. Od tu naprej je mogoče v tabele vstavljati podatke, jih posodabljati, brisati (korak 2) ali poizvedovati po podatkih (korak 3) z uporabo orodij delovnega orodja SQL Workshop, kot je opisano v tem poglavju.

Spletna aplikacija se lahko ustvari na praznih tabelah. Ta postopek je podrobno obravnavan v poglavju 3.

2.6 Vprašanja

1. Kakšna je razlika med logičnim in relacijskim modelom? Kateri elementi sestavljajo oba modela?
2. Kaj je normalizacija sheme R-BP?
3. Opišite možnosti za ustvarjanje tabel z uporabo nabora orodij Oracle.

2.7 Odgovori

1. Logični model je abstraktnejši. Elementi logičnega modela so entitete, atributi in povezave. Relacijski model predstavlja bolj konkreten oz. specifičen nivo baze podatkov. Elementi relacijskega modela so tabele, stolpci tabel (podatkovna polja) in povezave.
2. Normalizacija sheme R-BP je sistematičen postopek v treh korakih, imenovanimi “prva normalna oblika”, “druga normalna oblika” in “tretja normalna oblika”, ki vodijo k dobri shemi R-BP.
3. Obstaja vsaj pet načinov ustvarjanja tabel: samodejno generiranje tabel s programom Oracle Data Modeler, samodejno generiranje tabel s programom Quick SQL v Oracle APEX, ročno generiranje tabel z uporabo SQL-DDL, ročno ustvarjanje tabel s programom Oracle Object Browser in samodejno ustvarjanje tabel s programom Oracle Data Workshop.



3. Kako navigirate v APEX-u?

E. KAPSAMMER, W. RETSCHITZEGGER, W. SCHWINGER; PREVOD U. RAJKOVIČ

3.1 Uvod

Kaj je Oracle APEX? Oracle Application Express (Oracle APEX) je malokodno razvojno okolje za spletne aplikacije za razvoj spletnih aplikacij, ki temeljijo na bazah podatkov ORACLE. Do razvojnega okolja dostopamo preko spletnih brskalnikov. "Low-code" pomeni, da je za izgradnjo obsežne spletne aplikacije potrebnih le nekaj programerskih znanj - za prvi, enostavni prototip spletne aplikacije programerska znanja sploh niso potrebna.

Za katere namene se lahko uporablja APEX? Na splošno APEX ne omogoča le izgradnje podatkovnega nivoja spletnih aplikacij z uporabo APEX "SQL Workshop", kot je bilo obravnavano v predhodnem poglavju, temveč tudi izgradnjo namiznih ali mobilnih spletnih aplikacij z uporabo APEX "App Builder". APEX uporabljajo tako velika kot majhna podjetja na številnih področjih aplikacij, ki se spopadajo s širokim spektrom poslovnih potreb. Te poslovne potrebe segajo od preproste pretvorbe preglednice v spletno aplikacijo do izvedbe popolne spletne aplikacije (glejte primer, predstavljen v predhodnem poglavju), ki omogoča shranjevanje, pridobivanje, vizualizacijo in nadaljnjo obdelavo vsake posamezne prodaje v vsaki od 10.500 Walmartovih trgovin skupaj z odgovornimi oddelki, zaposlenimi in informacijami o njihovem delu.

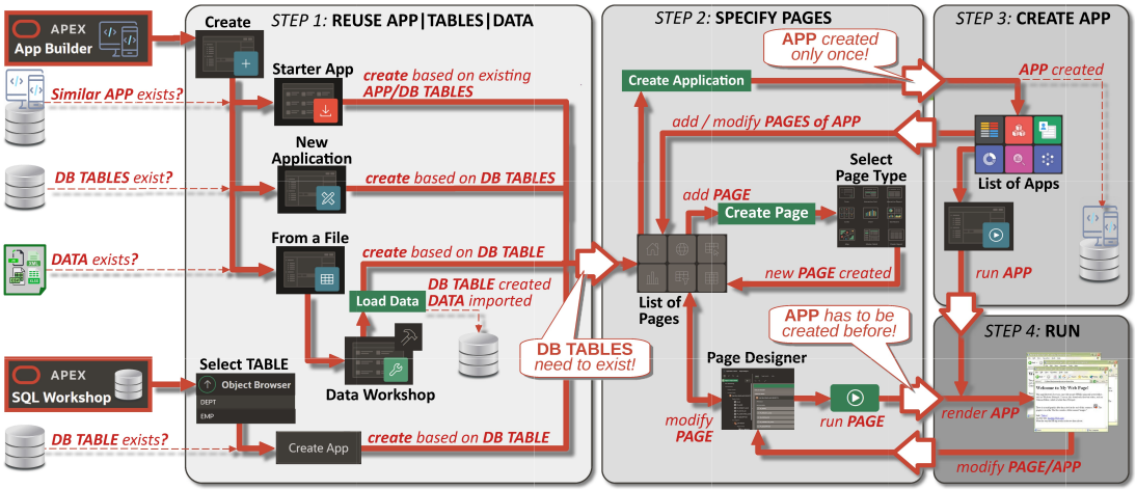
Čemu je namenjeno to poglavje? Namen tega poglavja je obravnavati celoten postopek razvoja spletnih aplikacij, ki mu sledi program Oracle APEX "App Builder". Ta proces je osnova za navigiranje med različnimi orodji programa App Builder, imenovanimi "razvojne komponente", da bi zgradili spletno aplikacijo. Poleg tega podaja pregled različnih razvojnih komponent na precej visoki ravni, podrobnosti katerih so obravnavane v nadaljnjih poglavjih.

3.2 Proces razvoja spletne aplikacije z APEX

Preden se lotimo podrobnosti o različnih razvojnih komponentah programa APEX "App Builder", si oglejmo postopek razvoja spletnih aplikacij, ki ga uporablja program APEX.

APEX sledi večstopenjskemu in cikličnemu procesu razvoja. Celoten razvojni proces, ki ga podpira APEX in je prikazan na sliki 3.1, je organiziran *večstopenjsko in delno ciklično*, kar omogoča *inkrementalni razvoj*, tj. postopno izpopolnjevanje spletne aplikacije in njene podatkovne plasti. Vsak korak razvoja je podprt z ustreznimi čarovniki razvojnih komponent. To pomeni, da se z možnostmi, ki jih ponujajo meniji na osnovni ravni, poglobimo v funkcionalnost programa *App Builder*, da bi določili vsako podrobnost spletne aplikacije in se po potrebi vračamo in premikamo

naprej, da aplikacijo postopoma izpopolnimo.



Slika 3.1: Splošni razvojni proces v APEX-u.

APEX sledi razvojnemu procesu "začnimo s podatkovnim nivojem". Pomembno je, da vedno pričnemo z razvojem podatkovnega nivoja, saj se APEX osredotoča na razvoj spletnih aplikacij, ki temeljijo na podatkovni bazi (glejte sliko 3.1 "STEP 1: REUSE APP | TABLES | DATA"). To lahko storite, kot je opisano v predhodnem poglavju, ali neposredno v programu App Builder, če npr. že obstajajo podatkovne datoteke, ki jih lahko ponovno uporabite za ustvarjanje tabel BP, npr. podatki v Excelovih preglednicah (glejte sliko 3.1 "From file"). Če tabele v BP že obstajajo, se lahko postopek razvoja začne (i) znotraj programa App Builder, s čimer se ponovno uporabi (in prilagodi) obstoječa aplikacija skupaj z njenimi tabelami BP (glejte sliko 3.1 "Starter App") ali se ustvari nova aplikacija od začetka, zopet na podlagi obstoječih tabel BP (glejte sliko 3.1 "Starter App"). Slika 3.1 "New application") ali (ii) iz SQL Workshop s preprosto izbiro tabele BP v Object Browser-ju (prim. Slika 3.1 "Select TABLE") in pričetkom procesa razvoja aplikacije na podlagi izbrane tabele.

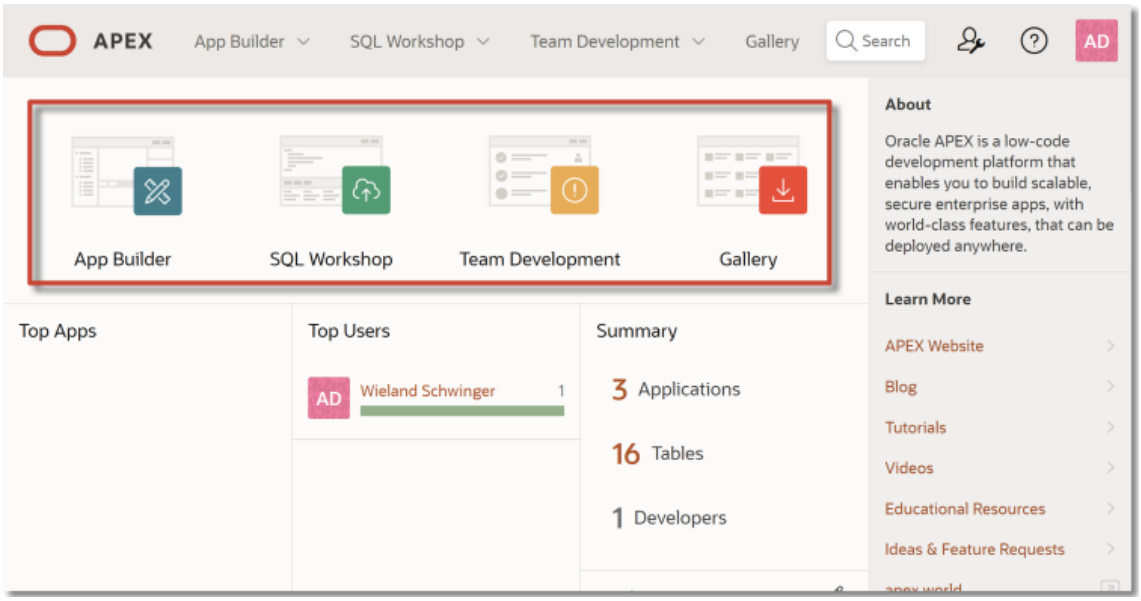
APEX sledi razvojnemu procesu, ki temelji na straneh. Takoj, ko obstajajo tabele BP za shranjevanje podatkov spletne aplikacije, jih je mogoče uporabiti za ustvarjanje nove spletne aplikacije po postopku, ki temelji na strani (glejte sliko 3.1 "STEP 2: SPECIFY PAGES"). To pomeni, da lahko določite eno ali več spletnih strani, ki sestavljajo strani HTML končne spletne aplikacije. Vsaka od teh strani lahko temelji na eni ali več tabelah BP in omogoča vizualizacijo podatkov v obliki različnih tipov strani, ki vključujejo npr. interaktivna poročila, obrazce, sezname, grafe ali koledarje, pa tudi interaktivno "igranje" s podatki (npr. povečevanje podrobnosti) in celo njihovo obdelavo ter shranjevanje sprememb nazaj v BP. Te *strani je mogoče povezati med seboj* z uporabo navigacijskih menijev, zavihkov, gumbov ali hipertekstnih povezav.

Takoj ko je ustvarjena ena ali več strani, je potrebno ustvariti celotno aplikacijo (glejte Sliko 3.1 "STEP 3: CREATE APP"), pri čemer lahko dodatno določite nekatere splošne lastnosti celotne spletne aplikacije (npr. videz aplikacije) ali pa uporabite privzete nastavitve. Nazadnje je treba aplikacijo oz. vsako stran izrisati, da se ustvarijo končne strani HTML (slika 3.1 "STEP: 4 RUN"), ki jih je mogoče nadalje interaktivno preizkusiti in se občasno vrniti, da izboljšamo spletno aplikacijo.

3.3 Začetni meni programa APEX

Ob prijavi v Oracle APEX se prikaže *"Workspace Homepage"*. Domača stran delovnega prostora omogoča dostop do štirih različnih razvojnih komponent (glejte sliko 3.2):

- *"App Builder"* se večinoma uporablja za izdelavo spletnih strani in predstavlja osrednjo temo tega poglavja.
- *"SQL Workshop"* se uporablja za izgradnjo podatkovnega nivoja in je že opisan v predhodnem poglavju.
- *"Team Development"* se uporablja za npr. interakcijo z drugimi razvijalci prek kartičnega sistema (ang. ticketing system) za določanje faz razvoja itd.
- *"Gallery"* se uporablja za namestitev vnaprej pripravljenih demo aplikacij, ki jih je mogoče spremeniti in prilagoditi lastnim potrebam z uporabo "App Builder".



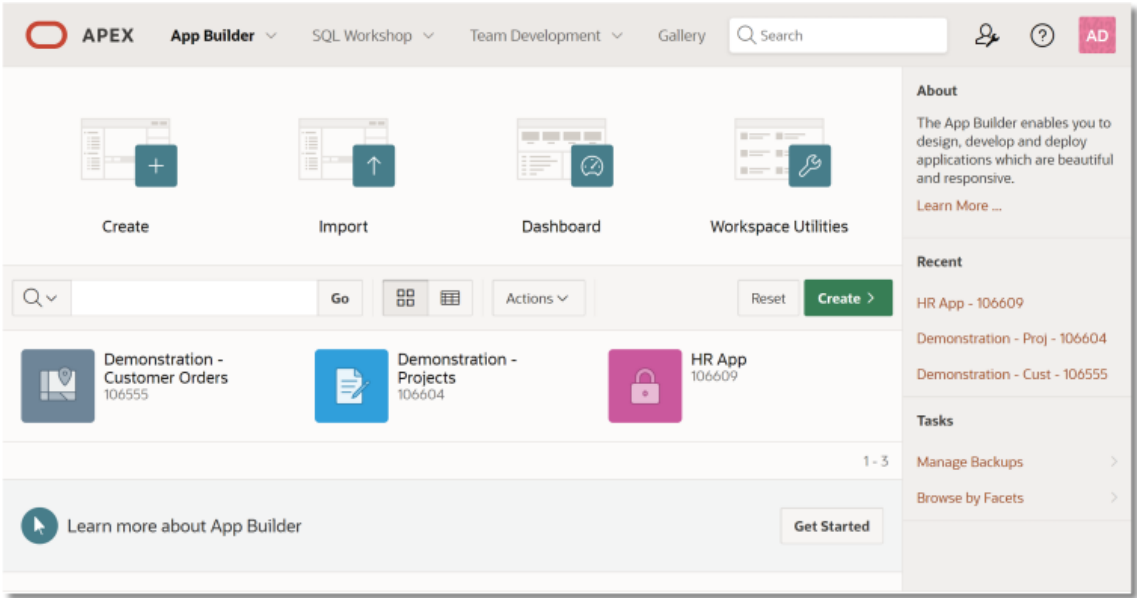
Slika 3.2: Domača stran delovnega okolja Oracle APEX.

Na dnu domače strani delovnega prostora (glejte sliko 3.2) področja "Top Apps", "Top Users" in "Summary" ponujajo informacije v realnem času o razvojnih dejavnostih v trenutnem delovnem prostoru. Na desni strani so na voljo dodatni izobraževalni viri.

3.4 Pregled App Builderja - ustvarjanje in upravljanje aplikacij

Ko se pomaknete na domačo stran programa App Builder, imate na voljo naslednje možnosti (glejte sliko 3.3):

- *Create* - ustvari novo spletno aplikacijo.
- *Import* - uvoz predhodno izvoženih spletnih aplikacij.
- *View the "Dashboard"*, ki prikaže statistične podatke o že razvitih spletnih aplikacijah.
- *Access "Workspace Utilities"*, ki zagotavljajo široko paleto storitev, kot sta oddaljeni dostop do podatkov in funkcija varnostnega kopiranja.



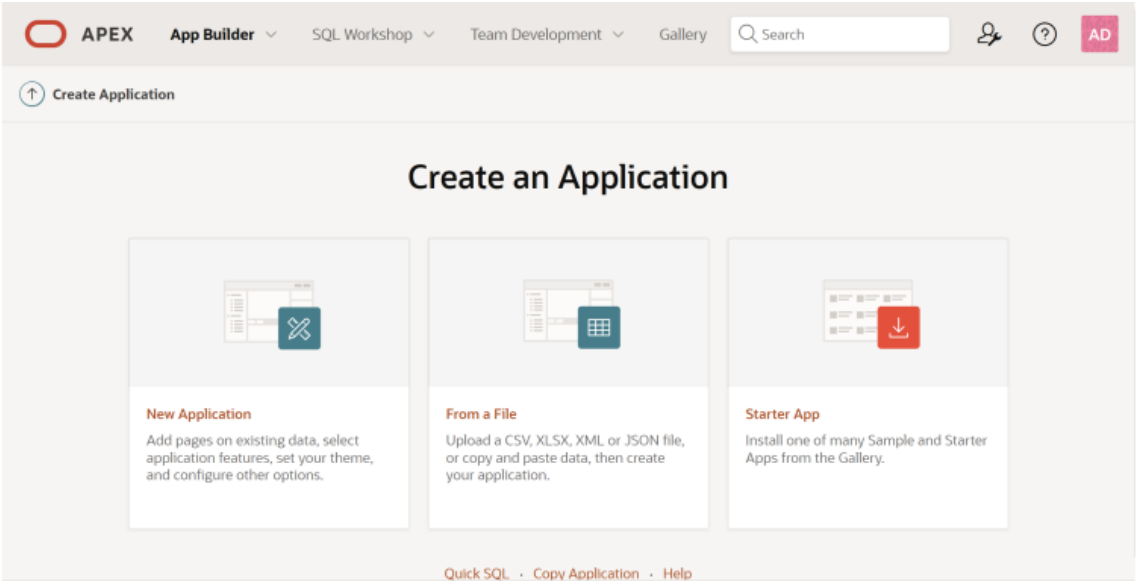
Slika 3.3: Pregled v App Builder.

V spodnji polovici okna (glejte sliko 3.3) je seznam obstoječih spletnih aplikacij, ki jih lahko izberete za nadaljnje urejanje ali posodabljanje (glejte naslednja poglavja).

3.5 Izdelava aplikacije - trije primeri uporabe

Ko v programu App Builder izberete “Create”, se odpre “Create Application Wizard”. Prikaže tri dodatne možnosti (primer na sliki 3.4), ki podpirajo resnične primere uporabe in omogočajo ponovno uporabo podatkov in same aplikacije:

- *“New Application”* izberemo, če podatkovni nivo (ali njegov del) že obstaja ali če je potrebno podatkovni nivo razviti sproti, med razvojem aplikacije.
- *“From a File”* izberemo, če želimo BP ustvariti na osnovi zunanjih obstoječih datotek, npr. datoteke EXCEL.
- *“Starter App”* omogoča ponovno uporabo (in nadaljnjo spreminjanje) obstoječe aplikacije skupaj z njeno BP iz “Galerije” aplikacij.



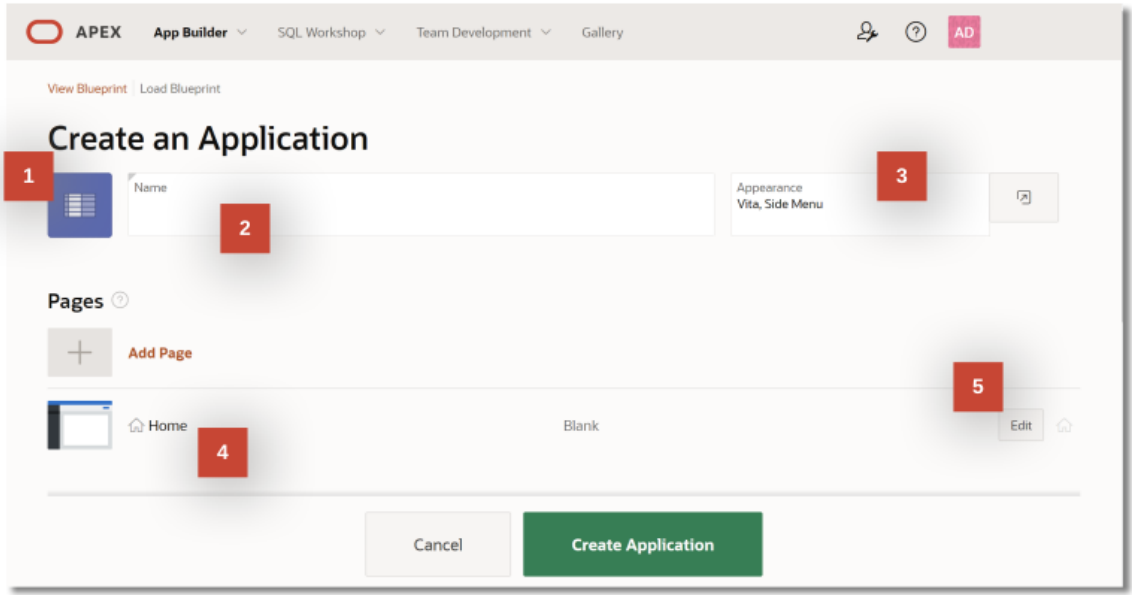
Slika 3.4: Čarovnik za ustvarjanje aplikacije: Trije načini ustvarjanja aplikacije.

V nadaljevanju se bomo osredotočili na prvo možnost *"New Application"*, druga možnost *"From a File"* bo podrobneje opisana v poglavju 2, tretja možnost *"Starter App"* pa ne bo podrobneje obravnavana.

3.6 Izdelava aplikacije - lastnosti, strani, funkcije in nastavitve

Pri ustvarjanju nove spletne aplikacije je potrebno opredeliti nekatere *splošne lastnosti*, ustvariti *strani* in izbrati nekatere dodatne *funkcije* in *nastavitve*. V nadaljevanju so le-te podrobneje obravnavane, pri čemer je na Sliki 3.5 prikazan zgornji del grafičnega vmesnika, na Sliki 3.6 pa njegov spodnji del.

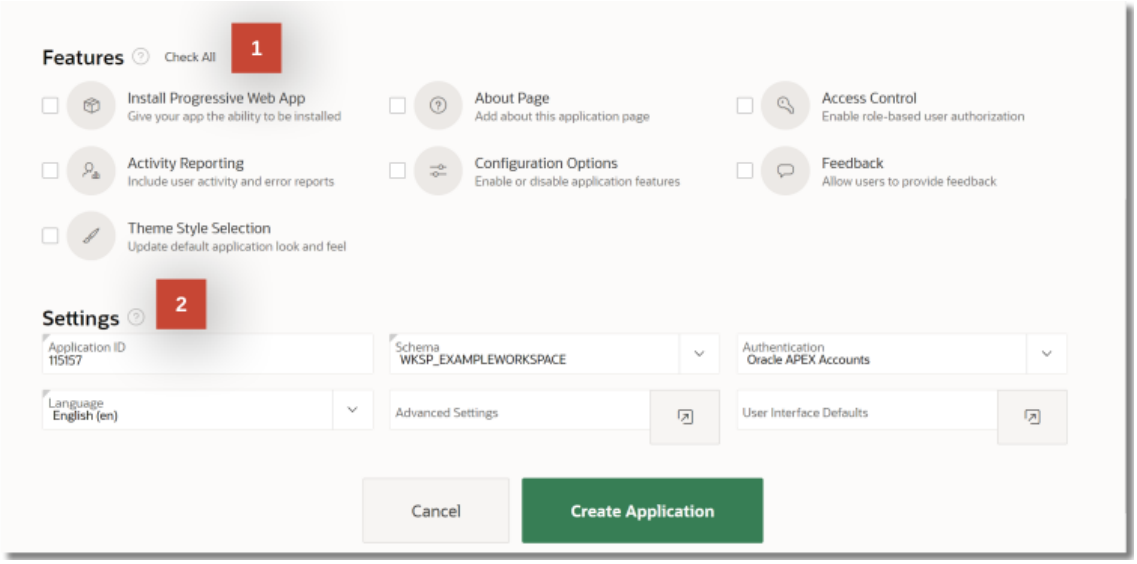
Skupne lastnosti - ikona, ime in videz aplikacije. Pri skupnih lastnostih (glejte Sliko 3.5) (1) lahko izberete ali naložite poljubno *ikono*, ki se uporablja kot Favicon in ikona za PWA, Apple touch ali App Builder, (2) vnesete *ime* aplikacije in (3) *izgled* spletne aplikacije. Pri tem lahko izbirate med vnaprej določenimi možnostmi in določate "videz" aplikacije z vidika sloga teme (npr. barve, velikosti pisave itd.) in *vrste navigacijskega menija* (npr. stranski ali zgornji meni), pri čemer je na voljo tudi privzeti videz.



Slika 3.5: Lastnosti in strani.

Ustvarjanje strani aplikacije. Glavna naloga je postopna izgradnja spletne aplikacije z zaporednim *dodajanjem novih strani* z uporabo čarovnika “Create Page Wizard” (zažene se z *“Add page”*, glejte (4) na sliki 3.5. Ko je stran ustvarjena, lahko kadarkoli med razvojem aplikacije *uredite vsebino* strani, *spremenite* vrstni red strani ali stran *izbrišete* (glejte (5) na sliki 3.5). Na spodnjem delu slike 3.5 je prikazana že ustvarjena stran “Home”. Nadaljnje podrobnosti o “*čarovniku za ustvarjanje strani*” so podane v podpoglavju 3.7.

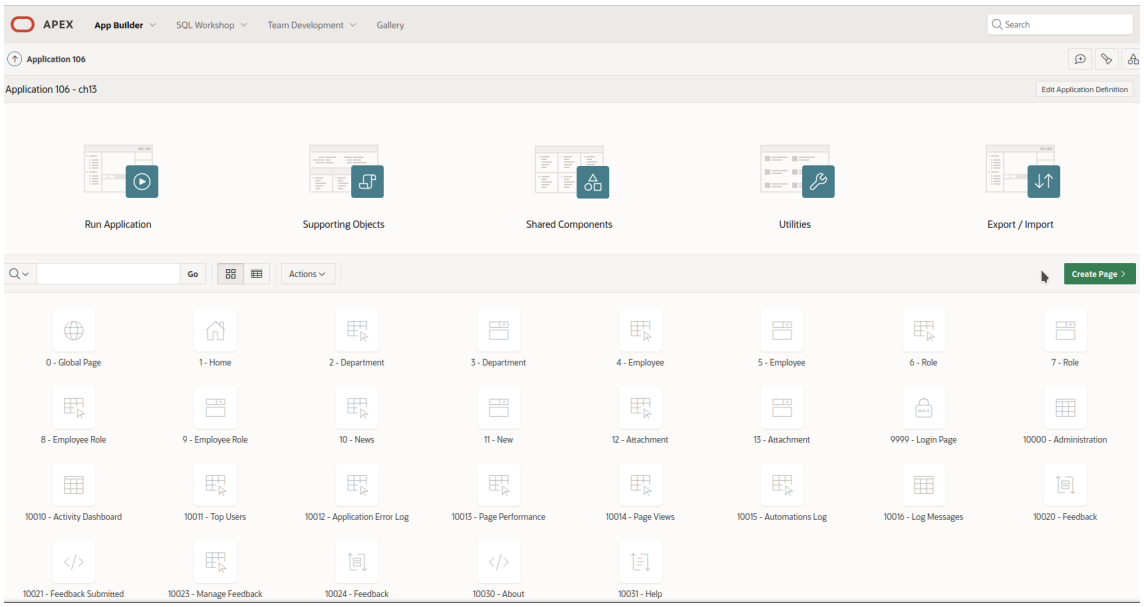
Lastnosti in nastavitve aplikacije. Nenazadnje je mogoče aplikacijo razširiti z nekaterimi zelo uporabnimi *funkcijami* (prim. (1) na Sliki 3.6), kot so "About Page", stran za nadzor dostopa “Access Control” ali stran za povratne informacije uporabnikov “User Feedback” , pa tudi z nekaterimi običajnimi *nastavitvami*, npr. ime sheme, ki vsebuje vse tabele v BP in tvori podatkovno plast aplikacije ali uporabljeni jezik (glejte (2) na Sliki 3.6).



Slika 3.6: Lastnosti in nastavitve.

Po kliku na gumb “Create Application” se prikaže “App Builder Homepage”, ki prikazuje novo

ustvarjeno aplikacijo s stranmi (glejte sliko 3.7).

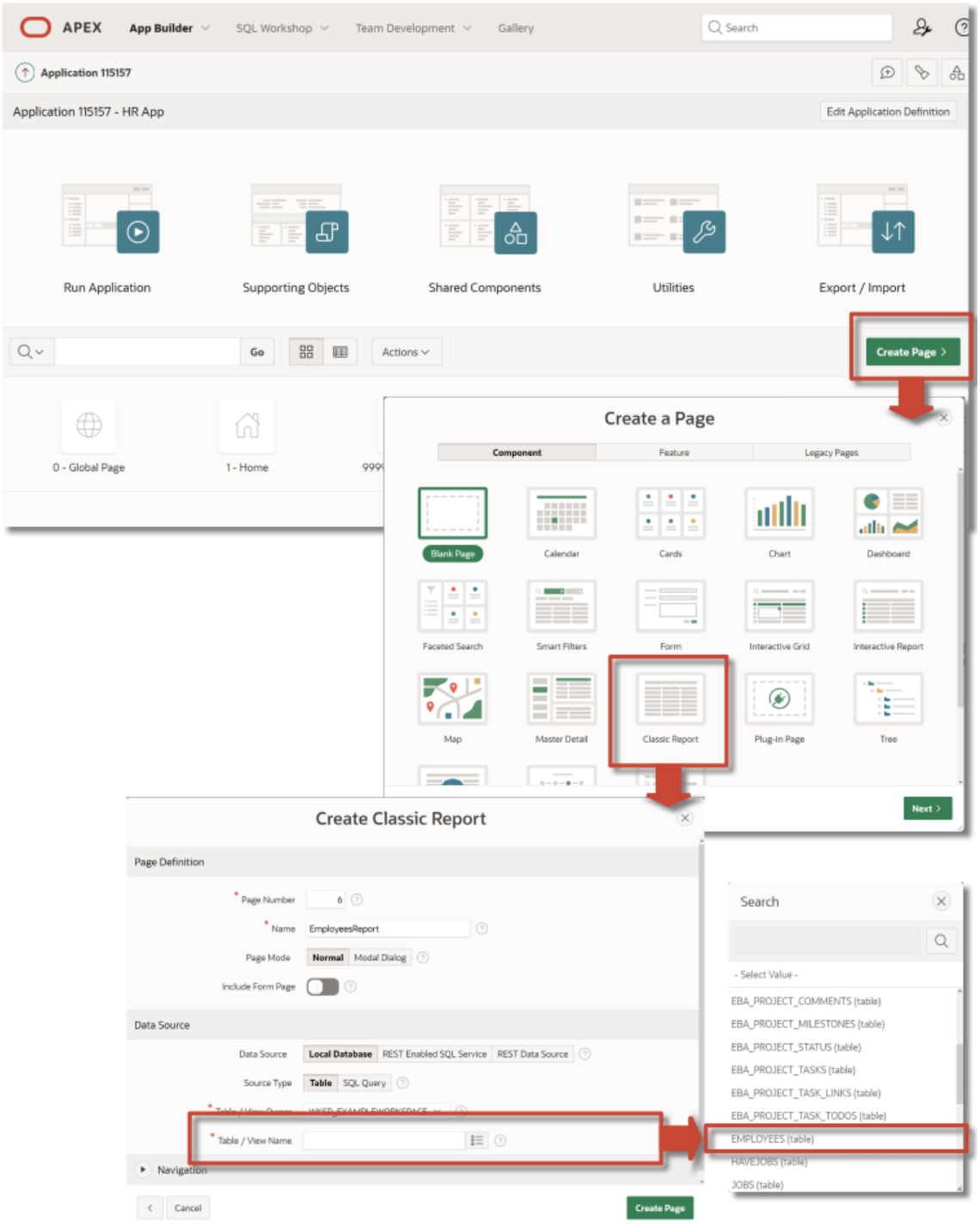


Slika 3.7: Domača stran aplikacije - pogled razvijalca.

3.7 Izbera vrste strani

Kaj sestavlja stran? Kot smo že omenili, je stran osnovni gradnik spletne aplikacije, tj. vsaka aplikacija je sestavljena iz ene ali več strani. Vsaka stran ima lahko gumbe in polja (imenovana "items") , ki so združeni v okvirje, imenovane "regions". Strani lahko vključujejo tudi aplikacijsko logiko, npr. izvajajo izračune (ang. computations) in preverjanje, kot so preverjanja pravilnosti med obdelavo podatkov.

Izbirate med 18 različnimi vrstami strani. Stran lahko prikazuje podatke iz tabel BP v različnih oblikah, ki jih določajo tako imenovane "page types", ki vključujejo na primer interaktivna poročila, sezname, koledarje ali grafikone. Tip strani izberemo vedno, ko se ustvarimo novo stran, bodisi z uporabo gumba "Add page" v pogovornem oknu "Create Application" (glejte Sliko 3.5) bodisi z uporabo gumba "Create Page" na domači strani aplikacije v pogledu razvijalca (glejte Sliko 3.8). Skupaj je na voljo 18 različnih vrst strani, kot je prikazano na sliki 3.8. Določena vrsta strani določa *vsebine*, *sestavo* in *razporeditev elementov* strani, kar lahko prepoznamo, če si ogledamo ikone vrst strani na sliki 3.8. Podrobnosti o vsaki vrsti strani so navedene v naslednjih poglavjih.



Slika 3.8: Ustvarjanje strani.

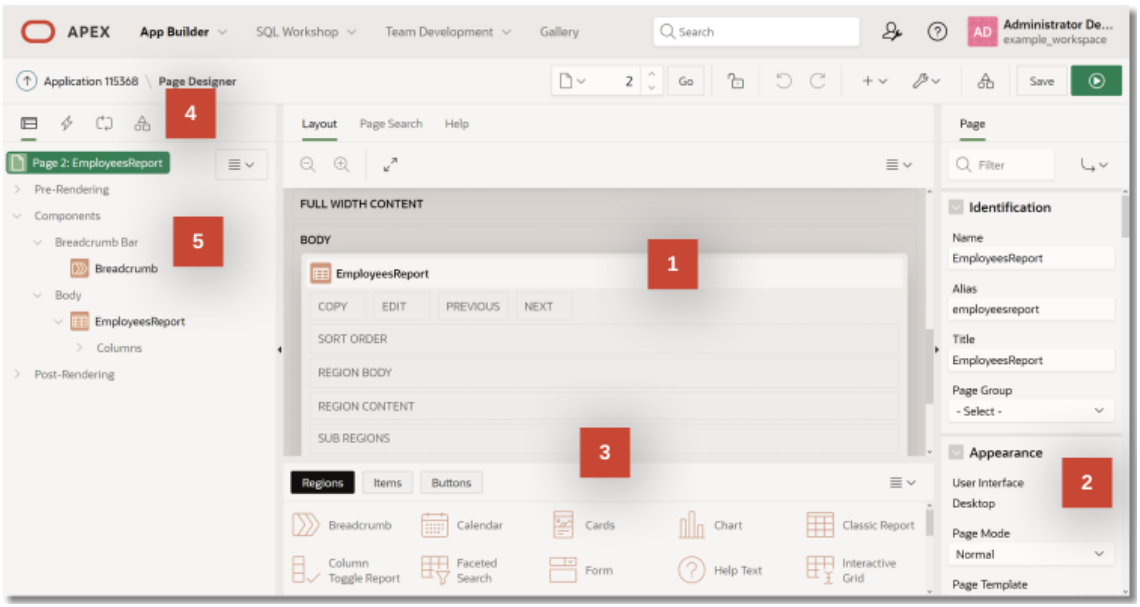
Dialog za specifikacijo strani - preslikava tabele na stran. Ko se odločamo za določeno vrsto strani, je vsebina pogovornega okna za opredelitev strani, ki ga prikaže čarovnik za ustvarjanje strani, seveda odvisna od izbrane vrste strani. Ker pa skoraj vse razpoložljive vrste strani temeljijo na tabelah BP, je glavna naloga običajno preslikava ustreznih tabel BP na stran, ki zagotavlja vir za vizualizacijo podatkov in cilj za upravljanje podatkov. Slika 3.8 prikazuje primer pogovornega okna za pogosto vrsto strani *Klasično poročilo*. Videti je, da je poleg nekaterih osnovnih informacij, kot je "Ime", potrebno določiti tudi preslikavo na podatkovni nivo z izbiro “Vira podatkov” - tabele BP, ki naj bo vir za poročilo (na sliki 3.8 izberemo tabelo “Employees” v našem uporabljenem

primera). APEX za prikaz podatkov iz osnovne tabele BP v ozadju uporablja SQL-DQL, takoj zatem, ko se prikaže stran HTML.

Ko je dodana nova stran, se komponenta APEX-a “Page Designer” (glejte podpoglavje 3.8) samodejno odpre, kar omogoča ogled in urejanje podrobnosti o strani. Hkrati lahko stran tudi takoj in samodejno prikažete v HTML-ju, npr. za namene testiranja in postopnega razvoja, z uporabo gumba "Run"(glejte podpoglavje 3.9).

3.8 Vzdrževanje in spreminjanje strani - Page Designer

Ustvarjene strani lahko vzdržujete in nadgrajujete z orodjem *Page Designer*, ki predstavlja *Integrirano razvojno okolje (IDE)*, ki vključuje orodno vrstico in več podoken (glejte Sliko 3.9). Page Designer ponuja veliko funkcionalnosti, med drugim omogoča spreminjanje sestave strani in razporeditve elementov na strani, njenih sestavnih delov (glejte (1), (2) in (5)), vključno z elementi, gumbi in okvirji (glejte (3)), pa tudi spreminjanje prikazovanja strani in logike aplikacije v smislu izračunov in preverjanj (glejte (4)).



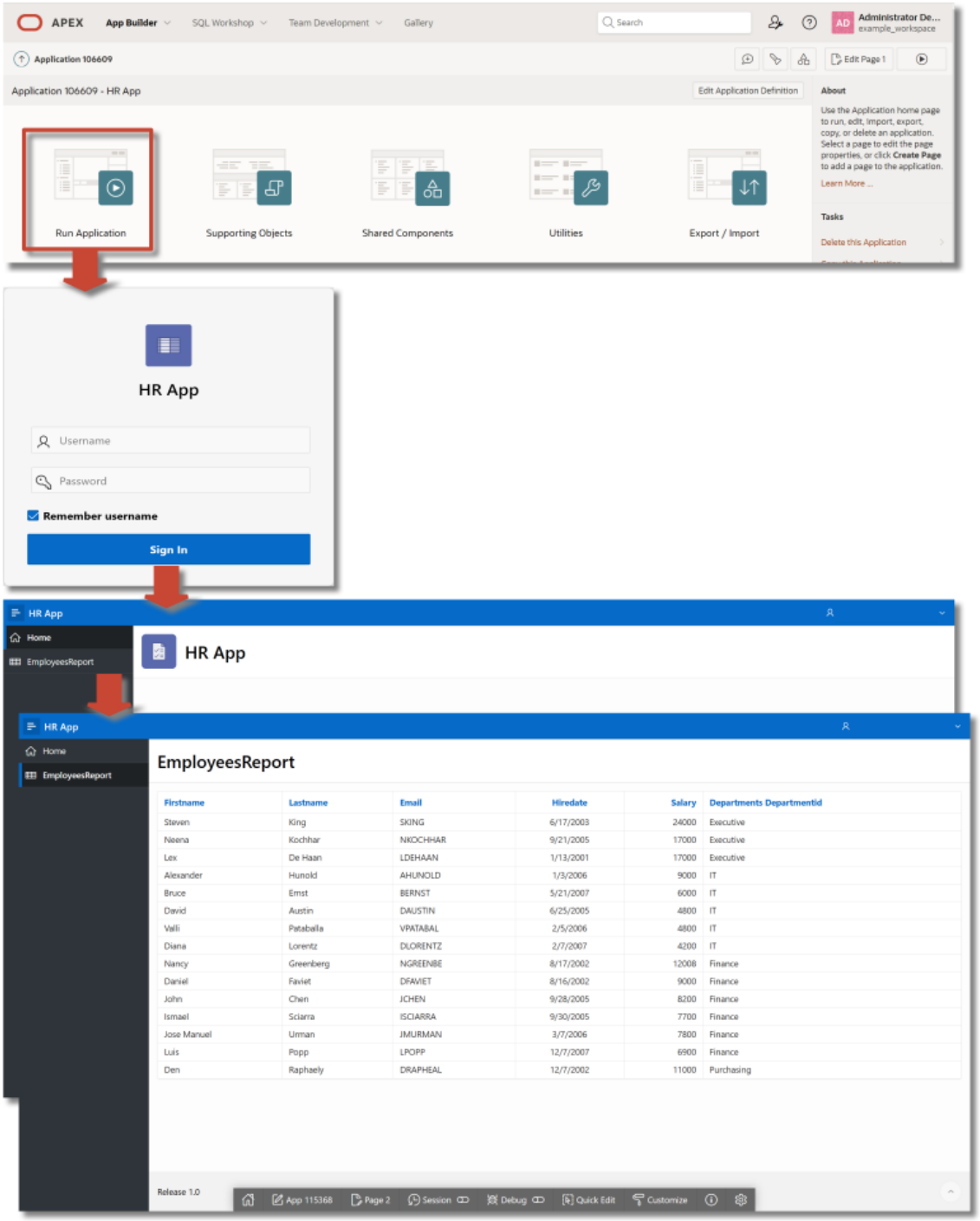
Slika 3.9: Page Designer.

Podrobnosti o funkcionalnostih, ki jih zagotavlja Page Designer, najdete v naslednjih poglavjih.

3.9 Zagon aplikacije ali strani

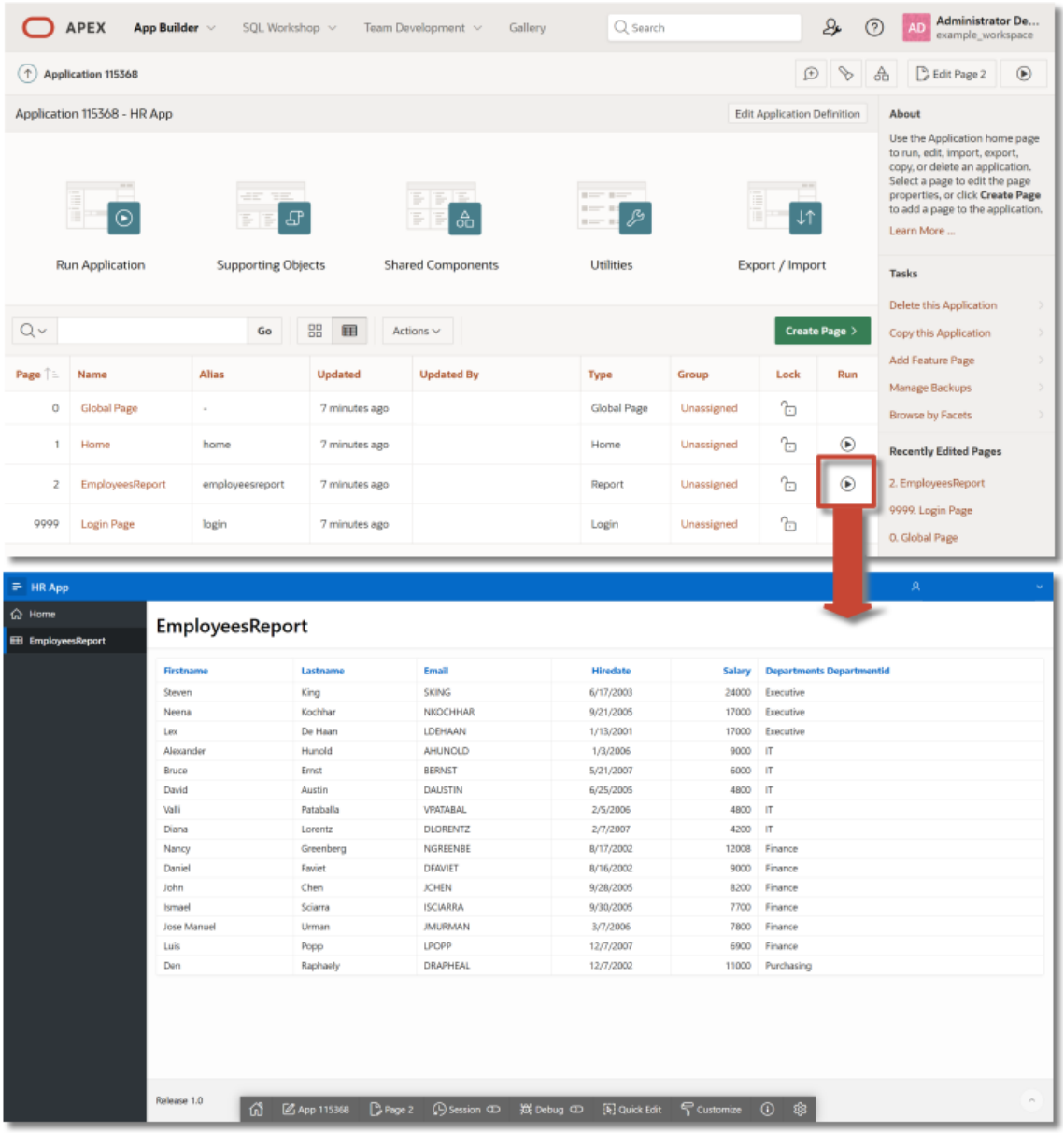
Če si želite ogledati izrisano različico aplikacije ali posamezne strani, jo je treba predložiti tako imenovanemu *Oracle APEX Engine* s klikom na gumb “Run” na *začetni strani aplikacije* (npr.. Sliko 3.10 in Sliko 3.11). APEX Engine dinamično izrisuje in procesira strani v vidne *strani HTML* na podlagi podatkov o aplikaciji, ki so shranjeni v “notranjih” tabelah BP. Na voljo sta dve različni možnosti na *Domači strani aplikacije*, odvisno od tega, ali želite zagnati celotno aplikacijo, tj. vse strani, ali samo posamezno stran, npr. za namene testiranja.

- **Zaženi celotno aplikacijo**, tj. vse strani: Ta možnost je na voljo na domači strani aplikacije, kot je prikazano na sliki 3.10.



Slika 3.10: Zaganjanje celotne aplikacije.

- **Zaženi posamezno stran**. Ko ustvarjate nove strani, lahko zaženete vsako stran posebej, da dobite takojšen vtis o videzu in delovanju dejanske strani. Gumb “Zaženi stran” je podoben majhni črni ikoni za igranje in je prikazan na desni strani vsake strani vaše aplikacije, kot je prikazano na sliki 3.11. Četudi zaženete samo posamezno stran, so na voljo tudi vse druge strani aplikacije.



Slika 3.11: Zaganjanje posamezne strani.

Druge možnosti, ki so poleg “Run Application” na razpolago na domači strani aplikacije (npr. Supporting Objects), so obravnavane v naslednjih poglavjih.

3.10 Vprašanja

1. Katere možnosti ponuja čarovnik za ustvarjanje aplikacije in za kaj se uporabljajo?
2. Katere funkcije omogoča čarovnik za izdelavo nove aplikacije (New Application)?
3. Katere vrste strani lahko ustvarimo v programu APEX?

3.11 Odgovori

1. Obstajajo tri možnosti: a) “New Application” omogoča razvijalcu dodajanje strani za obstoječe podatke, izbrati funkcije aplikacije, temo in nastavitve, b) “From a file” omogoča razvijalcu, da naložiti datoteke v formatih, kot so CSV, XLSX, XML in JSON, ali da kopira/prilepi podatke in sproži ustvarjanje aplikacije, c) “Starter App” omogoča razvijalcu, da

namesti eno ali več vzorčnih aplikacij iz galerije.

2. Na voljo je več uporabnih funkcij, kot so samodejno ustvarjanje strani o aplikaciji “About Page”, funkcija za skrbnika aplikacije - stran za dodeljevanje privilegijev uporabnikom v “Access Control”, samodejno ustvarjanje strani “User Feedback”, poročanje o dejavnostih, izbira sloga teme, možnosti konfiguracije in možnost namestitve progresivne spletne aplikacije.
3. APEX lahko ustvari široko paleto strani: obrazec, poročilo (interaktivna mreža, interaktivno poročilo, klasično poročilo, glavno podrobno poročilo), prazna stran, koledar, kartice, grafikon, nadzorna plošča, fasetno iskanje, pametni filter, zemljevid, iskalna stran, stran vtičnika, drevo, nalaganje podatkov, čarovnik in poenoten seznam opravil.

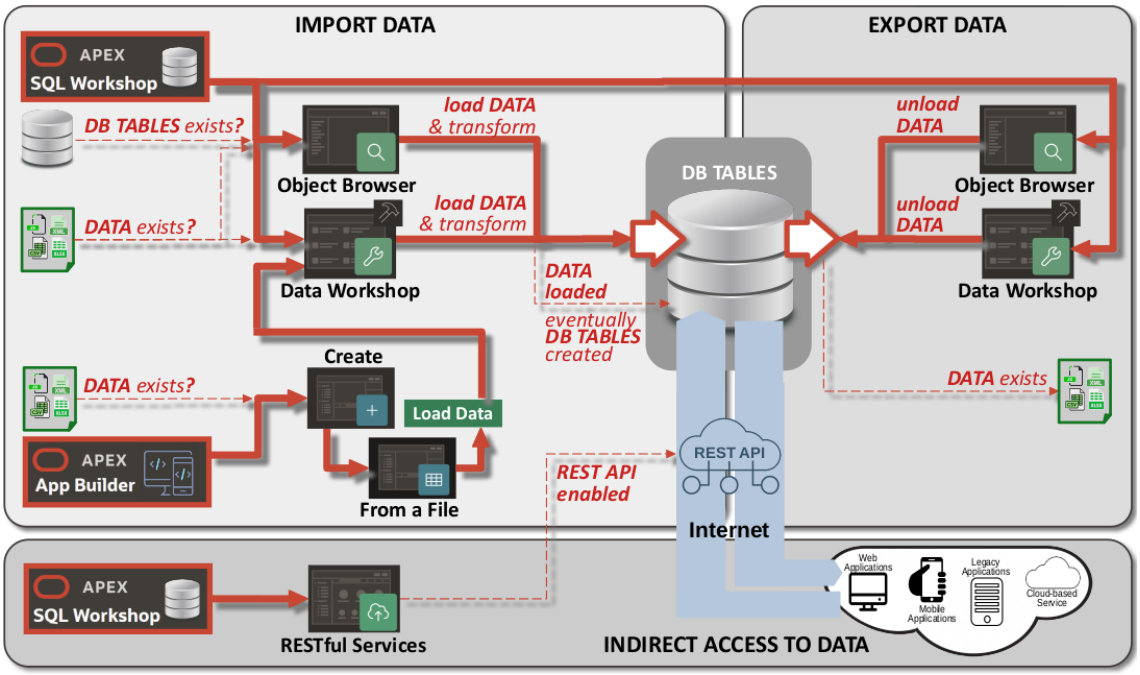


4. Kako izmenjujete podatke v APEX-u?

E. KAPSAMMER, W. RETSCHITZEGGER, W. SCHWINGER; PREVOD A. BAGGIA

Ko pripravljamo bazo podatkov za spletno aplikacijo, je glavna naloga priprava tabel v bazi podatkov in uvoz primernih podatkov (glejte poglavje 2). To pa ni nujno, da izvajamo ročno, tudi če podatki še ne obstajajo. V primeru, da imamo na razpolago podatke v zunanjih datotekah, jih lahko enostavno uporabimo in uvozimo (angl. *import*) v bazo podatkov Oracle. Poleg tega moramo občasno podatke tudi izvoziti (angl. *export*) v datoteko, da jih lahko uporabimo v kakšnem drugem okolju. V primeru, da uporabljamo zunanje odjemalce za podatke (npr. druge spletne/mobilne/opuščene aplikacije ali oblačne storitve), je smiselno, da namesto prenosa datotek s podatki, poskrbimo za neposredno povezavo z našo bazo podatkov (angl. *online access*).

Ti trije primeri izmenjave podatkov so prikazani na sliki 4.1, na kateri je predstavljen povzetek različnih možnosti, ki jih Oracle APEX ponuja za uvoz, izvoz in neposreden dostop do podatkov. V tem poglavju bomo vse tri možnosti podrobneje predstavili.



Slika 4.1: Izmenjava podatkov v Oracle APEX.

4.1 Uvoz in izvoz podatkov z “Data Workshop”

Kdaj uporabimo Data Workshop za izmenjavo podatkov? Data Workshop je enostavno orodje, ki s pomočjo čarovnika poenostavi proces uvažanja / izvažanja podatkov iz / v zunanje datoteke. Data Workshop je primeren v primeru, da razpolagamo z zmerno količino podatkov (manj kot 10 tabel v bazi podatkov) z enostavnimi podatkovnimi tipi (npr. brez polj v več vrednostmi ali vgnezdenih struktur). Za ponovno uporabo ali uvažanje velikih, kompleksnih nizov podatkov, Oracle zagotavlja bolj primerno orodje, to je tako imenovana “*SQL*Loader Utility*”. V nadaljevanju bomo v podpoglavjih 4.1.1 in 4.1.2 prikazali kako uvažamo podatke, v podpoglavju 4.1.3 pa kako potek izvoz podatkov.

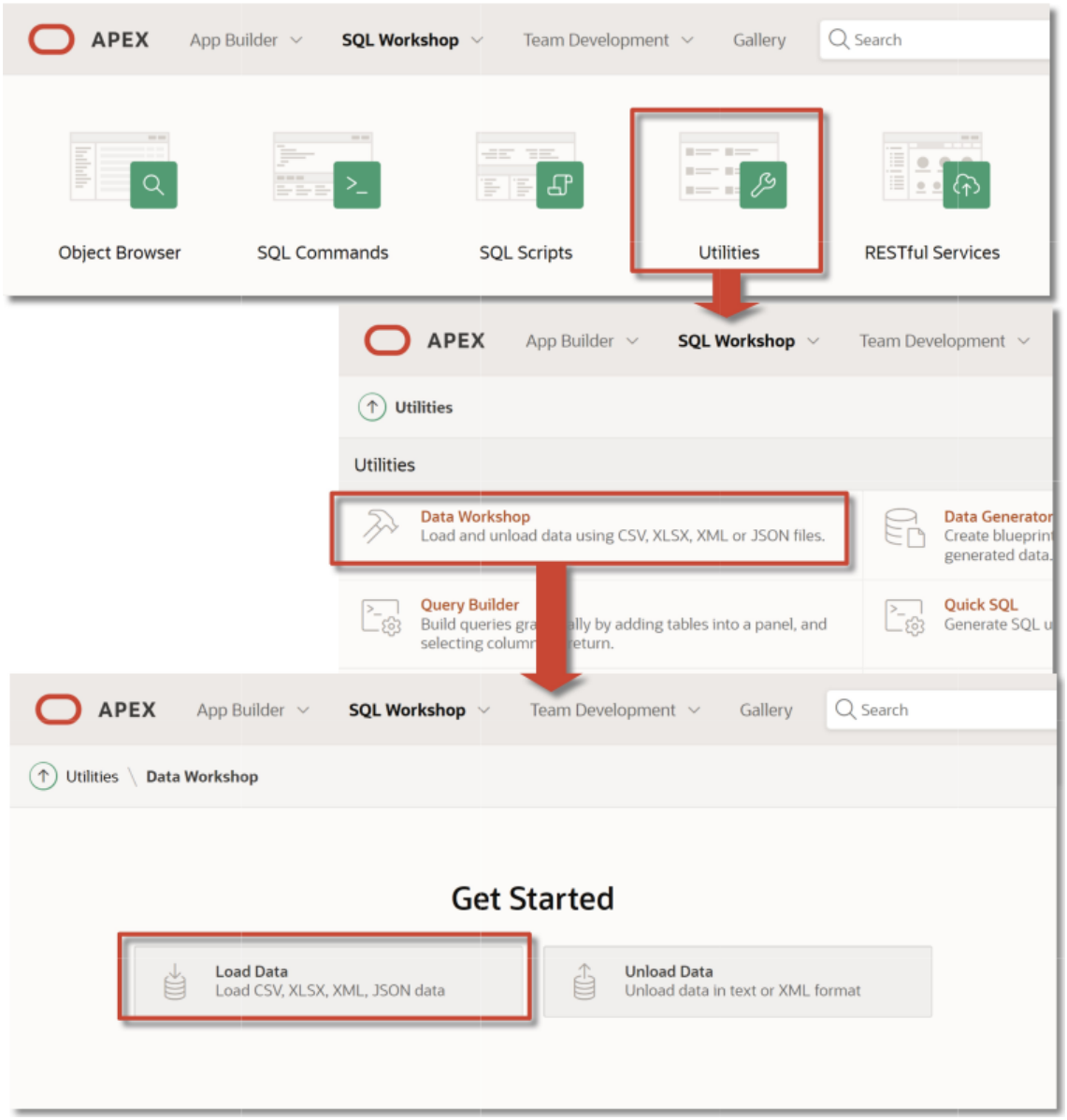
4.1.1 Uvoz podatkov

Vrste datotek. Data Workshop omogoča uvoz podatkov iz naslednjih vrst zunanjih datotek:

1. Katerikoli standardni *format z ločilom*, ki je lahko *tabluator* ali *vejica* (“*CSV*” – *Comma Separated Values*).
2. *XLSX datoteke* (na primer *Excelov delovni zvezek*). V primeru da XLSX datoteka vsebuje več delovnih listov, je privzeto uvožen samo prvi delovni list. Za nalaganje podatkov iz drugih listov, lahko izberemo list z opcijo “Select Sheet”.
3. *JSON datoteke* (*Java Script Object Notation*). Podprta je samo ena raven gnezdenja.
4. *XML datoteke* (*eXtended Markup Language*). Enako kot pri JSON datotekah je tudi tu podprta samo ena raven gnezdenja.

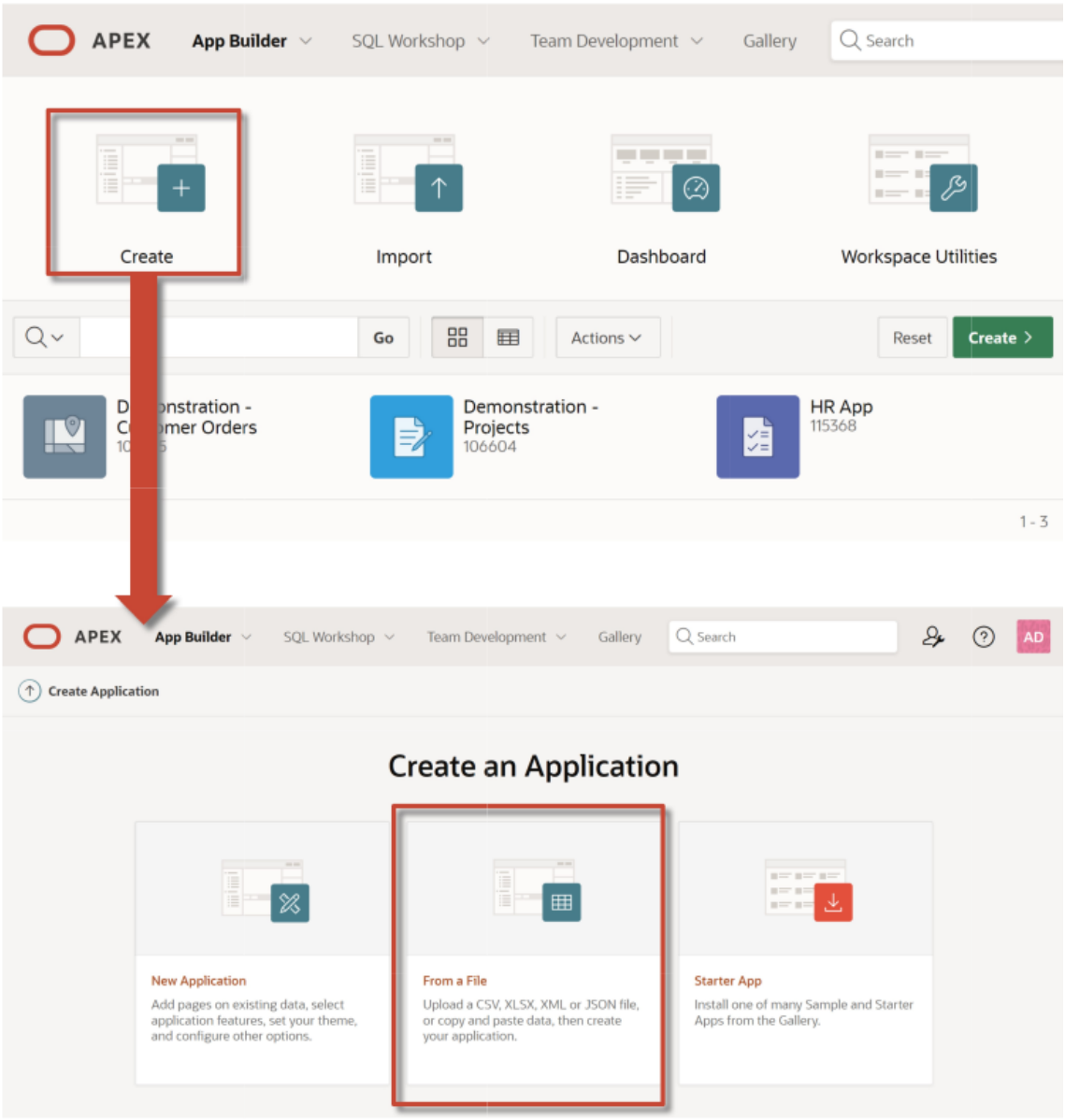
Dostop do Data Workshop. Do Data Workshop lahko dostopamo na dva načina:

1. Zaženemo Data Workshop iz SQL Workshop, kot prikazuje slika 4.2.



Slika 4.2: Uporaba Data Workshop za uvoz podatkov - dostop preko SQL Workshop.

2. Do Data Workshop lahko dostopamo tudi iz APEX App Builder, ko kreiramo aplikacijo, kot smo že omenli v poglavju 3 (glej sliko 4.3), preko "From a File" opcije (glej sliko 4.1).

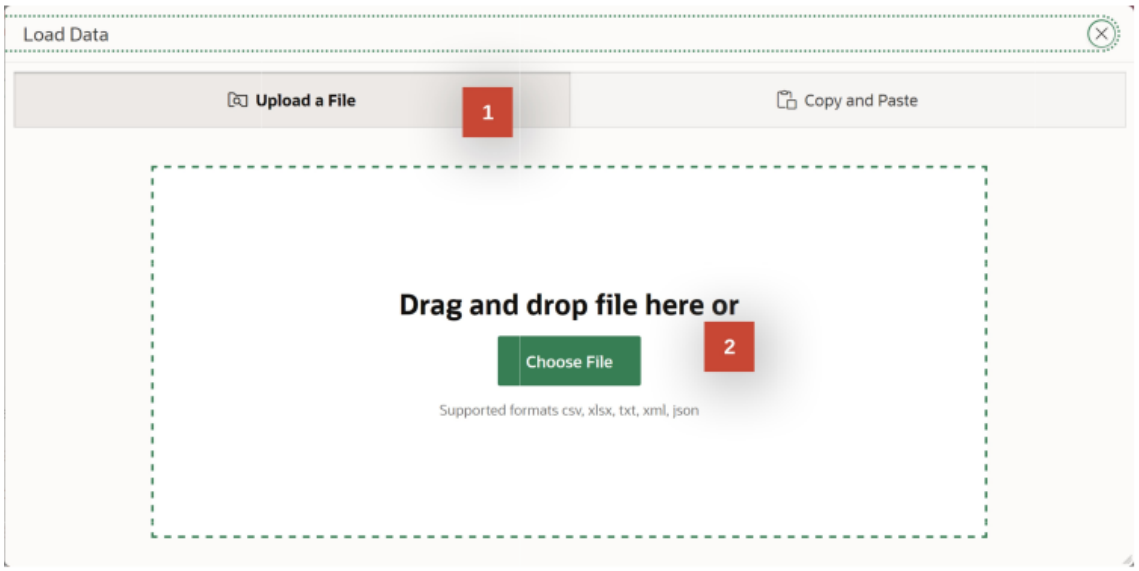


Slika 4.3: Uporaba Data Workshop za uvoz podatkov - dostop preko App Builder.

4.1.2 Koraki uvoza

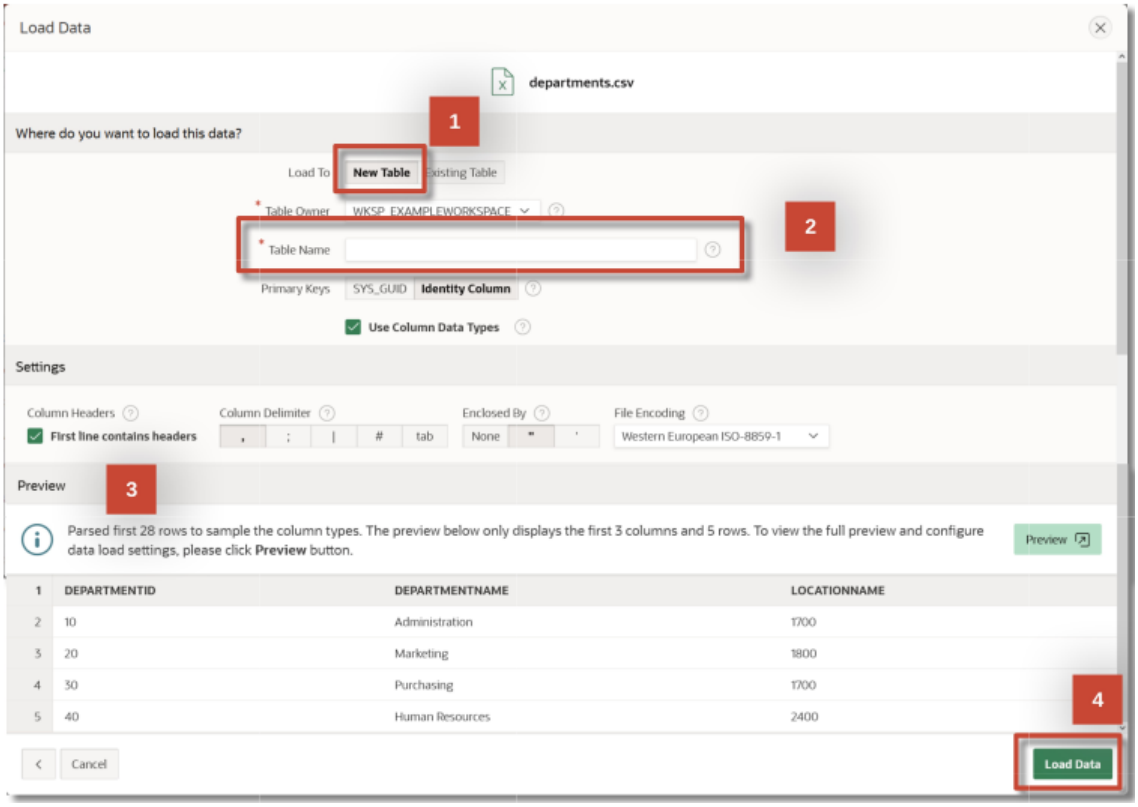
V nadaljevanju kratko predstavljamo korake, ki so potrebni za uvoz podatkov, hkrati pa podajamo nekaj pomembnih informacij za uporabo in razumevanje delovanja čarovnika za uvoz "Load Wizard":

Zagotavljanje vira podatkov. V prvem koraku uvoza moramo zagotoviti datoteko s podatki. To lahko storimo tako, da naložimo (vleci in spusti ali izbira datoteke) CSV, XLSX, JSON ali XML datoteko, ali pa da enostavno kopiramo podatke (glej sliko 4.4). Pogovorno okno se odpre, ko kliknemo gumb *"Load Data"* (slika 4.2) ali gumb *"From a File"* (slika 4.12).



Slika 4.4: Zagotavljanje vira podatkov.

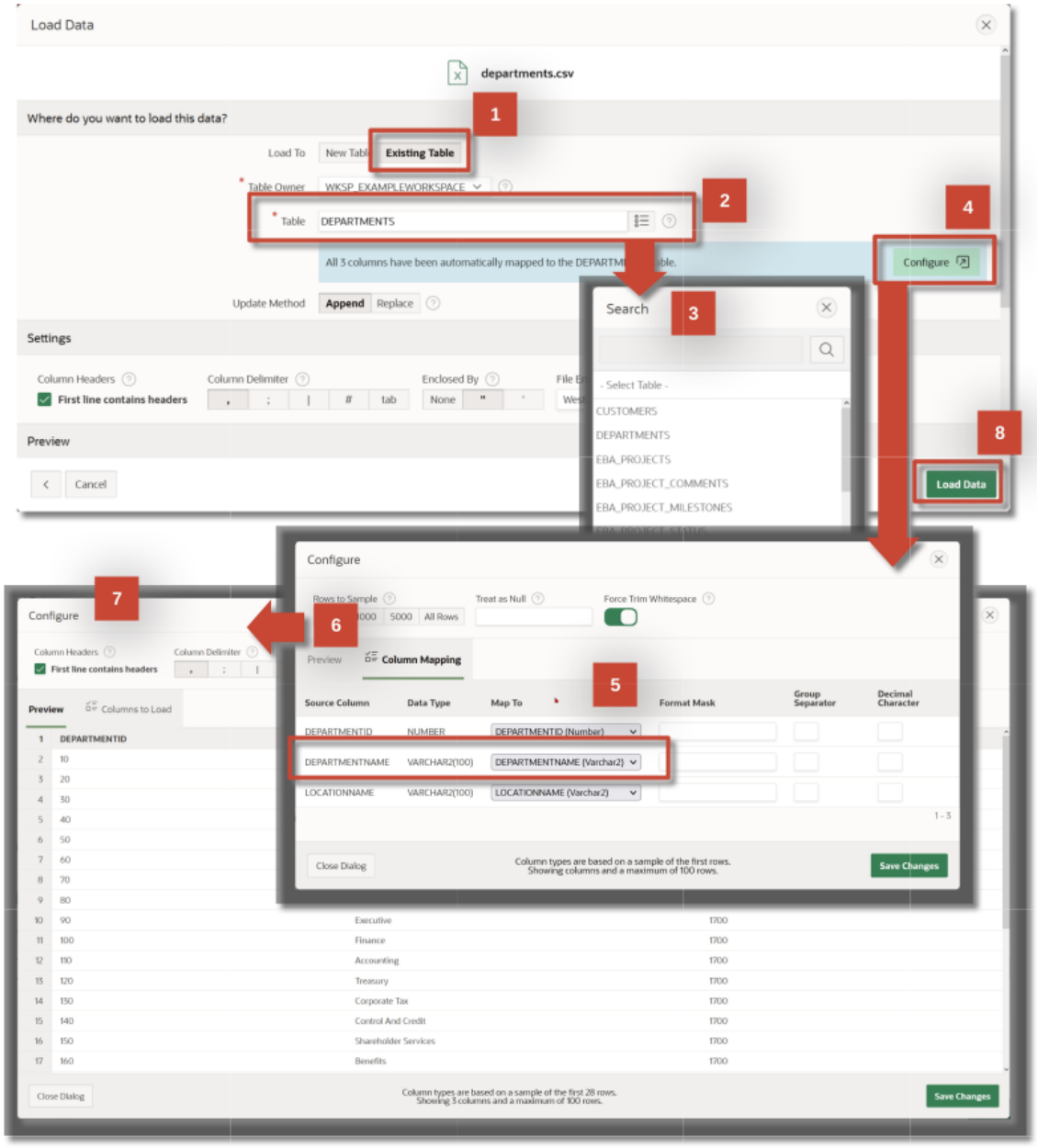
Nastavitve prenosa iz podatkov v tabelo. Ključni korak uvoza z Load Wizard je nastavitev prenosa podatkov iz datoteke v tabelo kot prikazujeta sliki 4.5 in 4.6, kjer so najpomembnejši deli oštevilčeni z (1)-(4) in (1)-(8), ter opisani v nadaljevanju.



Slika 4.5: Prenos podatkov v novo tabelo.

1. **Prenos podatkov v novo ali v obstoječo tabelo.** Najprej se moramo odločiti, ali naj se podatki uvozijo v že obstoječo tabelo v bazi podatkov, ali pa bomo uporabili novo tabelo, kot prikazujeta sliki 4.5 (1) in 4.6 (1).

2. **Določitev lastnika in imena tabele.** V obeh primerih moramo določiti ime lastnika tabele in ime tabele (slika 4.5 (2) in slika 4.6 (2)).
3. **Nastavitve za datoteke ločene z ločili.** V razdelku *setting* (slika 4.5 (3)) lahko nastavimo tri pomembne lastnosti CSV datotek. Pri lastnosti "*Column Headers*" določimo ali prva vrstica podatkov vsebuje imena stolpcev. Pri lastnosti "*Column Delimiter*" določimo ločilo, ki ga uporabljamo v podatkovni datoteki, kot prikazuje slika 4.5 (3). Na ta način omogočimo pridobivanje informacij o strukturi podatkov iz zunanje datoteke - proces, ki ga imenujemo tudi "*parsing*", kar je predpogoj, da bomo lahko prenesli tudi podatke. Pri lastnosti "*Enclosed by*", lahko označimo začetno in končno mejo podatkovnih vrednosti. Če določimo ločilo, Data Workshop ne upošteva presledkov, ki bi se morda pojavili pred ali po začetni meji vrednosti podatkov. To možnost lahko uporabimo tud, če želimo podatkovno vrednost označiti z določenim ločilom.
4. **Vzorčni pregled podatkov za prenos.** Oracle avtomatsko izvede prenos podatkov, kot smo že omenili, da lahko pridobi strukturo podatkov ter tako ugotovi podatkovne tipe posameznih stolpcev. V spodnjem delu slike 4.5 (4), vidimo del (do 10 stolpcev in 5 vrstic) rezultatov procesiranja podatkov.
5. **Razširjen predogled in podrobne nastavitve.** S klikom na gumb "*Preview*", izpišemo do 100 stolpcev in 100 vrstic, nato pa lahko nastavimo še druge lastnosti, recimo ročno spremenimo predlagan podatkovni tip, izberemo kateri stolpci se sploh prenesejo ipd.
   V primeru, da se v 1. koraku odločite, da boste uvozili podatke v obstoječo tabelo (1), je potrebno tabelo izbrati iz seznama tabel v bazi podatkov (slika 4.6 (2) in (3). S klikom na gumb "*Configure*" (slika 4.6 (4), lahko *izberete stolpce* ter na ta način določite, kateri od avtomatsko zaznanih stolpcev, ki se prenašajo iz datoteke, se preslika v katerega od stolpcev v obstoječi bazi podatkov. Kot prikazuje slika 4.6 (5) in (6), ima stolpec `DEPARTMENTNAME` podatkovni tip `VARCHAR2`, ki je bil avtomatično določen pri sintaktični analizi podatkov v datoteki, in se bo prenesel v tabelo v bazi podatkov, v stolpec z imenom "DEPARTMENTNAME", ki ima enak podatkovni tip.

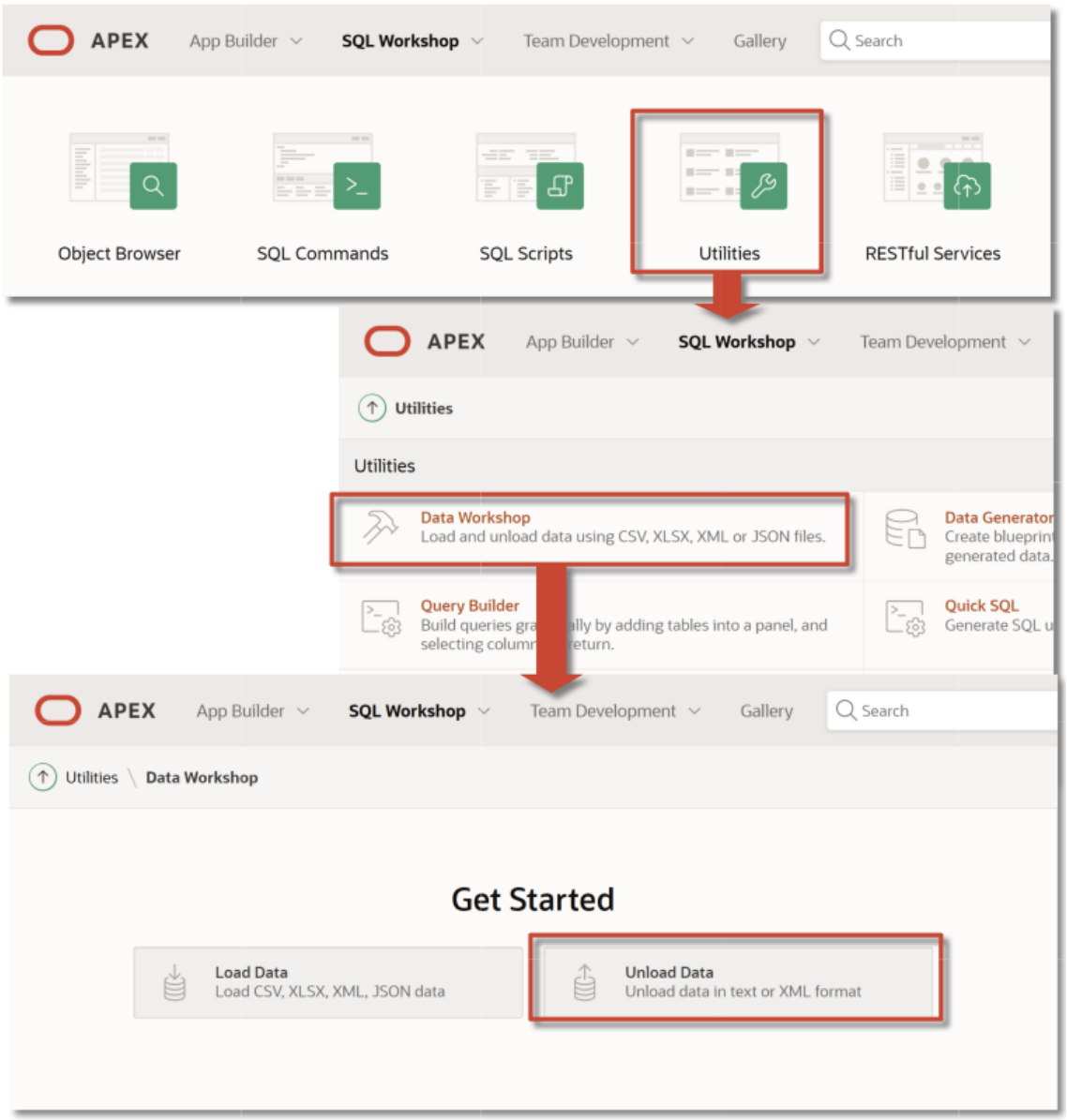


Slika 4.6: Prenos podatkov v obstoječo tabelo.

6. **Nalaganje podatkov.** S potrditvijo možnosti *"Load Data"* (slika 4.6 (8)), se generira nova tabela v bazi podatkov (v primeru da ste v prvem koraku izbrali to opcijo (1)), podatki se naložijo v novo ali obstoječo tabelo, pogovorno okno o prenosu podatkov pa nas na koncu postopka obvesti, koliko vrstic je bilo naloženih. Proces nalaganja podatkov se izvaja v zaledju, kar je posebej dobrodošlo pri večjih datotekah, saj lahko pogovorno okno zapremo in nadaljujemo z delom. V procesu prenosa podatkov lahko pride pri nekaterih vrsticah podatkov tudi do *napak*. Teh vrstic ni mogoče vnesti v ciljno tabelo, saj je lahko vrednost neskladna s podatkovnim tipom ali pride do kakšne druge napake. V tem primeru se vrstice z napakami avtomatsko shranijo v tabelo napak, od koder jih potem lahko ročno prenesemo. O številu vrstic z napako smo obveščeni tudi preko pogovornega okna. Tabelo z s prenesenimi podatki (in morda tudi tabelo z napakami) si lahko ogledamo v "Object Browser", ali pa takoj nadaljujemo z razvojem spletne aplikacije.

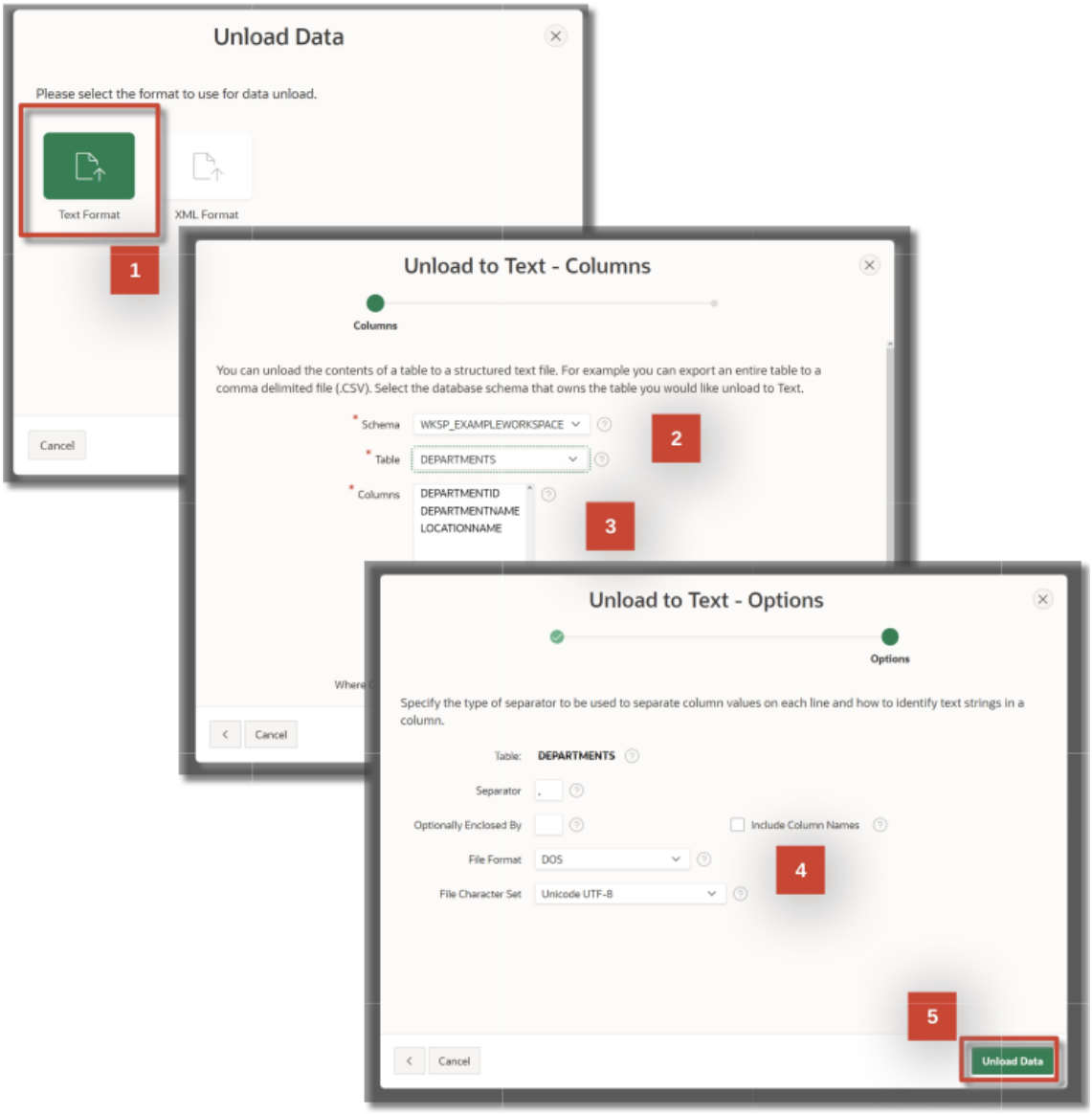
4.1.3 Izvoz podatkov

Do čarovnika za izvoz podatkov v Data Workshop dostopamo na enak način kot je opisano v podpoglavju 4.1.1, preko "Load Wizard" orodja (slika 4.7).



Slika 4.7: Izvoz (Odlaganje) podatkov.

V prvem koraku izberemo vrsto datoteke, pri čemer Data Workshop omogoča izvoz podatkov v zunanje datoteke v obliki CSV ali XML (slika 4.8 (1)). V naslednjem koraku izberemo tabelo in stolpce iz katerih bomo prenašali podatke (slika 4.8 (2) in (3). Če smo izbrali izvoz v obliki XML datoteke, lahko začnemo z izvozom. Če smo izbrali CSV pa moramo vpisati še nekaj parametrov, ki določajo ločila med vrsticami, ali naj se prenesejo tudi imena vrstic, ali naj bo podatkovni format prilagojen za DOS ali UNIX operacijski sistem (slika 4.8 (4)), preden dejansko aktiviramo "Unload Data", ki bo shranil izvoženo datoteko (slika 4.8 (5)).

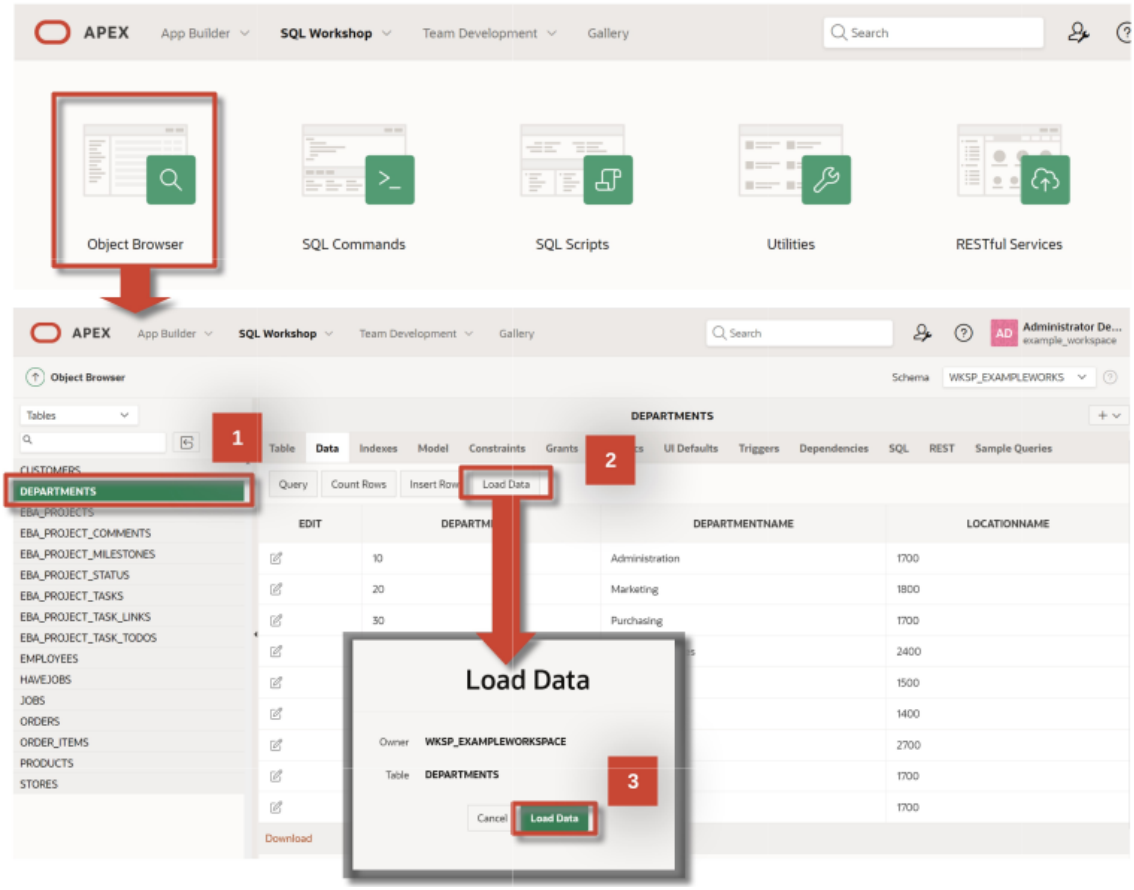


Slika 4.8: Čarovnik za izvoz podatkov.

Pri prenosu datoteke za operacijski sistem DOS ali UNIX obstaja manjša, a pomembna razlika, saj imajo DOS datoteke drugače zaključene vrstice kot datoteke, ki so namenjene okolju UNIX/Linux. DOS uporablja dva ukaza kot oznako za prehod v novo vrstico (''Carriage Return'' in ''Line Feed'': \r\n), UNIX pa samo enega (''Line Feed'': \n). Zato moramo biti pozorni pri prenosu datotek med Windows in Unix operacijskim sistemom, ter poskrbeti da uporabljamo prave zapise za zaključek vrstice.

4.2 Uvoz in izvoz podatkov z "Object Browser"

Za uvoz in izvoz podatkov lahko uporabimo tudi drugo možnost in sicer "Object Browser", ki ga najdemo v SQL Workshop. Tukaj lahko prenose izvajamo na posamezni tabeli. Torej moramo najprej izbrati tabelo, kot prikazuje slika 4.9 (1)).

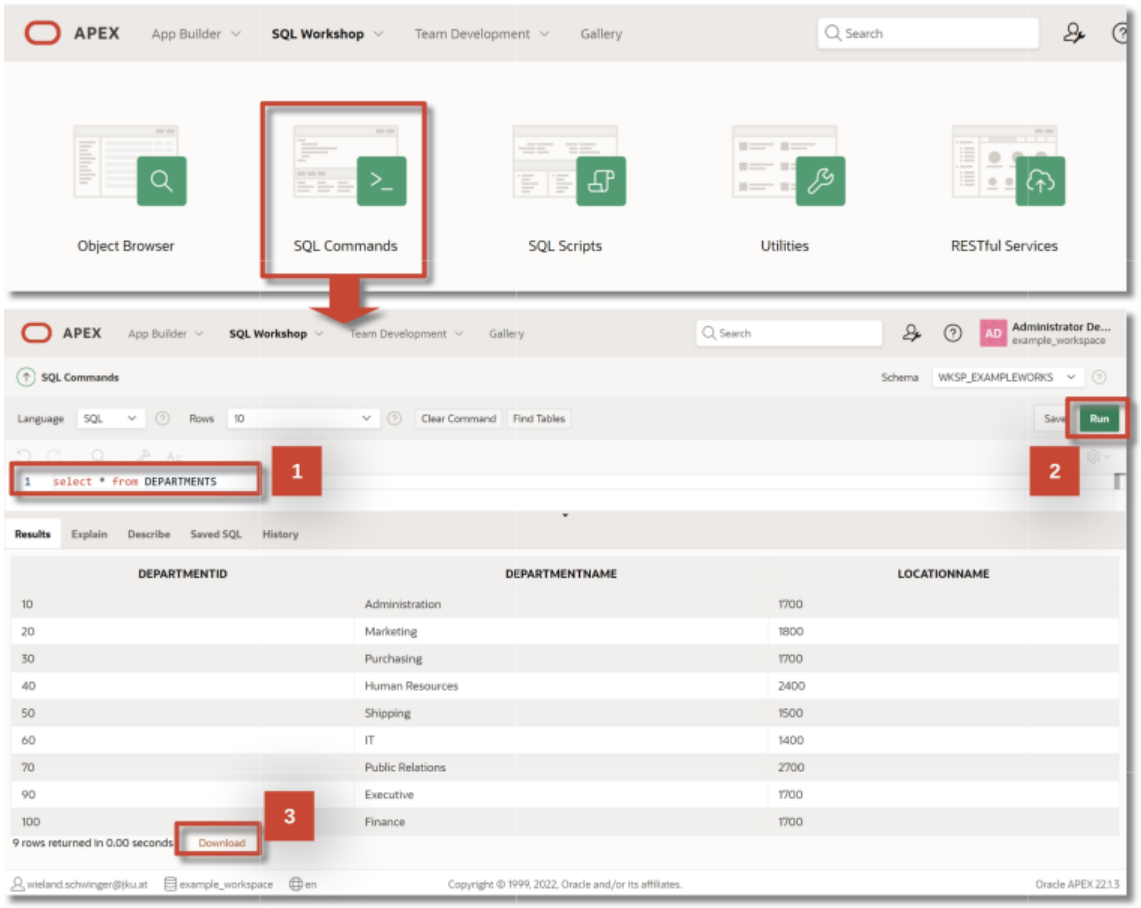


Slika 4.9: Uvoz / izvoz podatkov iz tabele s pomočjo Object Browser.

Če želimo uvoziti podatke, izberemo “Load Data”, ki odpre pogovorno okno “Load Data” (slika 4.9 (2), (3)), kot smo že opisali v podpoglavju 4.1.1. Za izvoz podatkov pa enostavno izberemo možnost *“Download-button”* na dnu strani. Z ukazom zaženemo postopek generiranja CSV datoteke s podatki, ki so zbrani v tabeli.

4.3 Izvoz rezultata SQL poizvedbe

Podobno kot lahko izvozimo vse podatke shranjene v tabeli, lahko izvozimo tudi samo del teh podatkov, ali pa podatke pridobljene iz več tabel. Izvozimo lahko rezultat katerekoli delujoče SQL poizvedbe (slika 4.10 (1)).

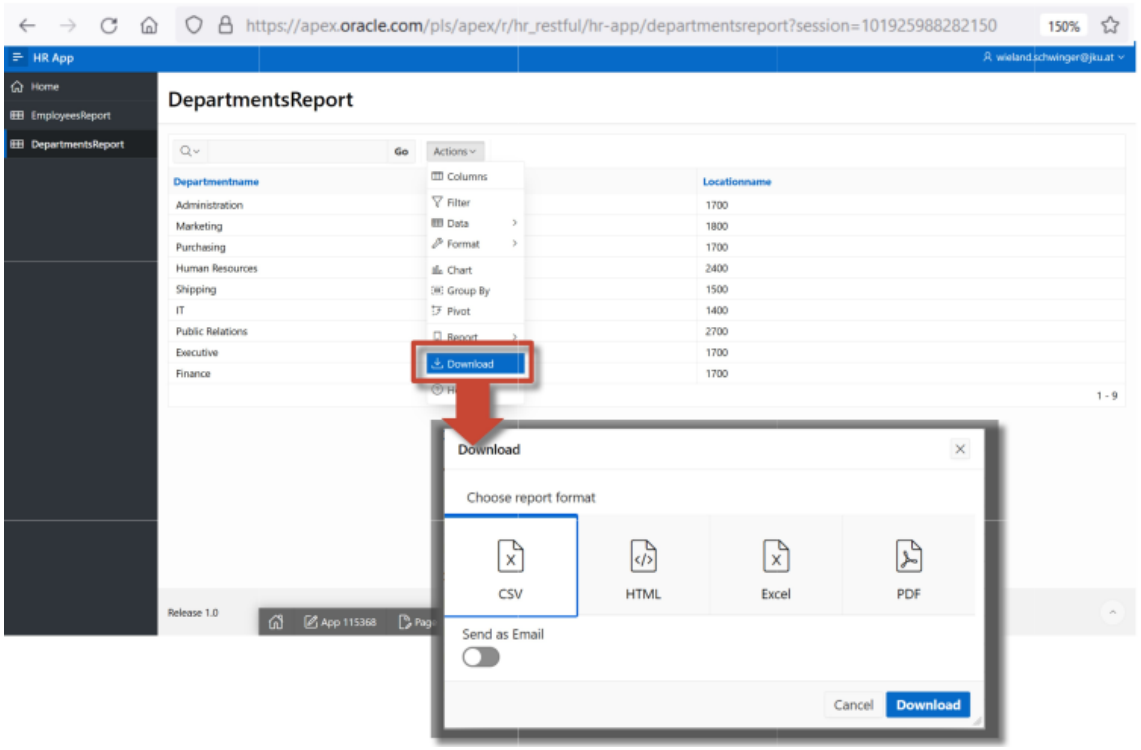


Slika 4.10: Izvoz podatkov, ki so rezultat SQL poizvedbe.

V prikazanem primeru je po tem, ko se izvede SQL ukaz in se prikaže rezultat poizvedbe (slika 4.10 (2)), ta rezultat lahko prenesemo s klikom na gumb “Download” (slika 4.10 (3)).

4.4 Izvoz podatkov iz poročila

Kot smo opisali v poglavju 3, moramo pri razvoju spletne aplikacije z APEX App Builder za vsako stran, ki jo dodamo v aplikacijo, izbrati vrsto strani. Če je stran v obliki interaktivnega poročila, lahko med pregledom podatkov le-te tudi izvozimo. Na sliki 4.11 je prikazan primer interaktivnega poročila “DepartmentsReport”.

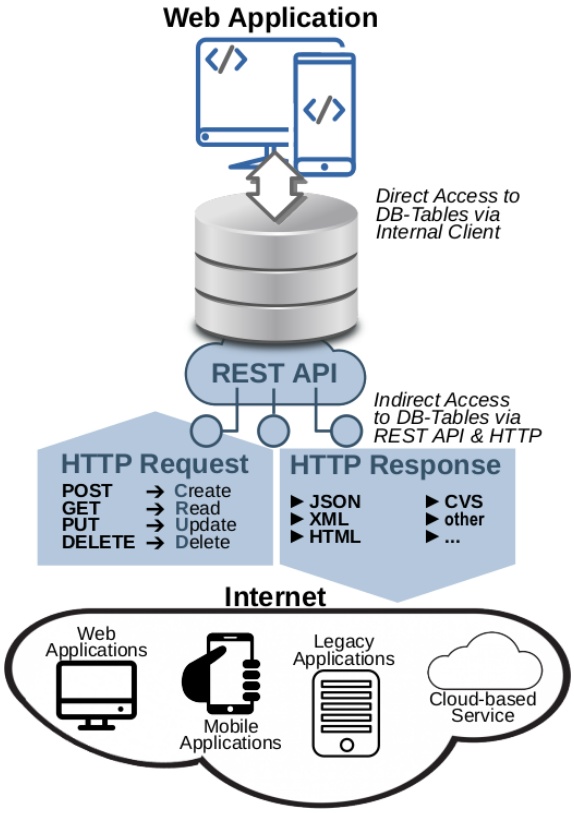


Slika 4.11: Izvoz podatkov iz poročila.

V primeru izvoza podatkov iz poročila, lahko izbiramo med različnimi vrstami datotek, CSV, HTML, PDF in Excel, lahko pa izberemo tudi možnost pošiljanja datoteke preko e-pošte.

4.5 Izmenjava podatkov z RESTful Services

Izmenjava podatkov iz baze podatkov APEX spletne aplikacije z drugimi zunanjimi odjemalci, kot so spletne in mobilne aplikacije, podedovane (angl. legacy) aplikacije ali oblačne storitve, se lahko enostavno izvede preko tako imenovanih *RESTful services*. *REST (Representational State Transfer)* je arhitekturni vzorec, ki sta ga leta 2000 predlagala Roy Fielding in Richard Taylor [2] in omogoča interoperabilnost poljubnih sistemov prek Interneta. Omogoča poizvedbe in delo s podatki brez neposrednega dostopa do povezanih tabel v bazi podatkov. Na sliki 4.12 je prikazan pregled osnovnega arhitekturnega vzorca REST.



Slika 4.12: RESTful Services za izmenjavo podatkov – Osnovna arhitektura.

Glede na REST arhitekturni vzorec, je potrebno za zagotovitev neposrednega doptopa kreirati RESTful service kot niz programskih vmesnikov (API, angl. Application Programming Interfaces) na tabelah v bazi podatkov. Brez neposredne povezave z bazo podatkov, ti API-ji omogočajo zunanjim sistemom varno interakcijo s podatki, od poizvedb, vnosov, spreminajnja in brisanja podatkov. Definicije RESTful service v Oracle APEX se shranjujejo v *Oracle REST Data Services (ORDS)* repozitoriju , in jih imenujemo *ORDS-based REST Services*.

4.5.1 REST Arhitekturni vzorec

Kot smo že omenili, storitev imenujemo RESTful kadar je skladna z načeli protokola REST. Čeprav je definicija REST izven obsega tega poglavja, pa omenimo njene značilnosti (splošne značilnosti REST kot so opisane v [2], smo delno prilagodili za naš kontektst baze podatkov):

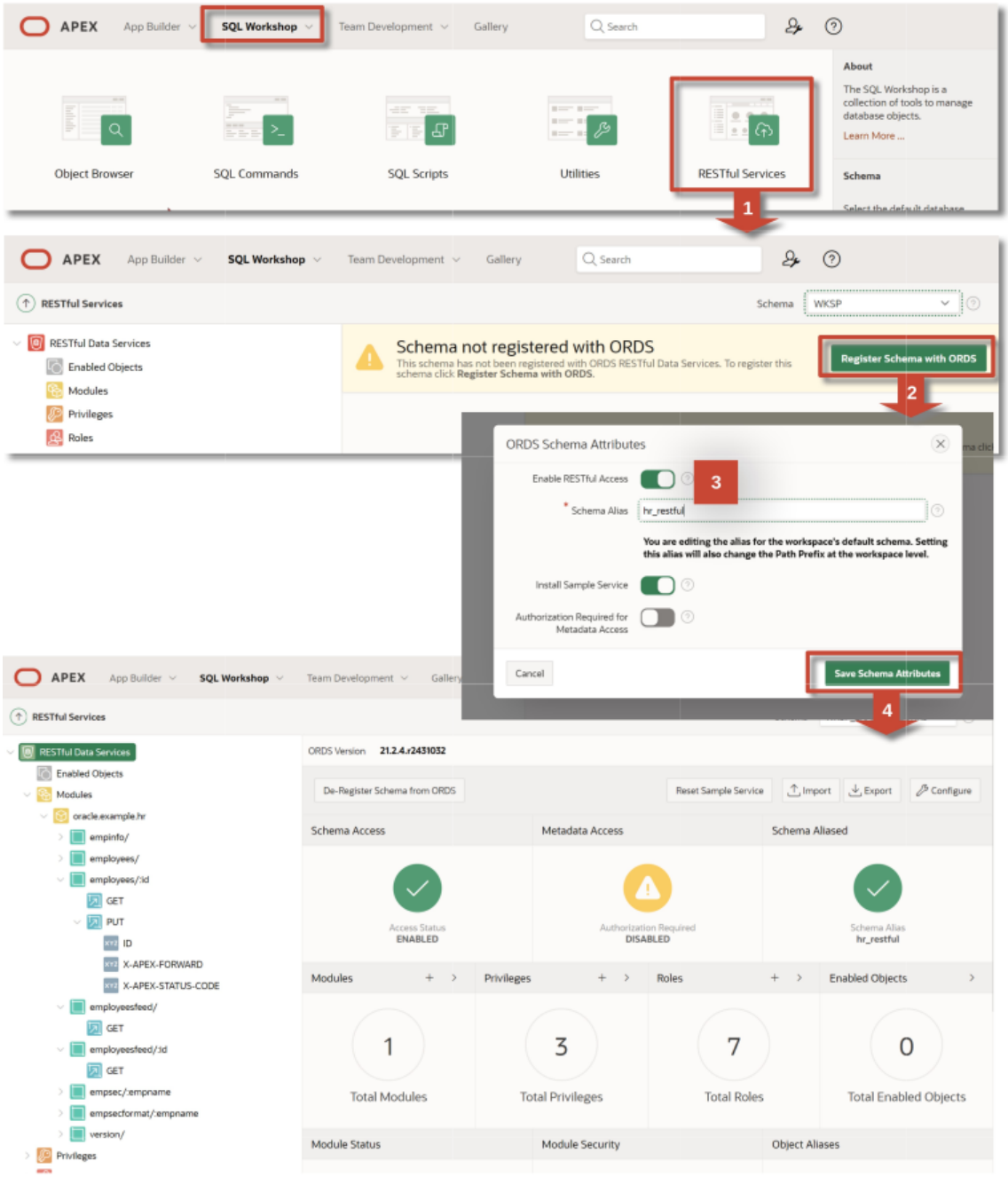
1. **Tabele v bazi podatkov kot vir storitve**.RESTful Service je storitev, ki jo sestavlja niz virov, ki so v našem primeru tabele v bazi podatkov. Te vire idenntificiramo z URL naslovom in do njih dostopamo z HTTP ali HTTPS spletnim protokolom.
2. **Operacije poslane preko HTTP zahtev**. Za ravnanje z viri se uporablja majhen nabor operacij v obliki zahtevkov HTTP, ki vključujejo `POST`, `GET`, `PUT`, `DELETE`, kar deluje podobno kot operacije CRUD (Create, Read, Update, Delete).
3. **Storitve ne shranjujejo stanja**. Storitve RESTful ne shranjujejo stanja (angl. stateless), tj. kontekst odjemalca ni shranjen med zahtevki - stanje seje se v celoti hrani na odjemalcu. To pomeni, da mora vsaka zahteva odjemalca do strežnika vsebovati vse informacije, ki jih strežnik potrebuje za razumevanje zahteve, in ne more uporabiti nobene vsebine shranjene na strežniku. Glavna prednost tega pristopa je, da lahko ustvarimo ponovno uporabne in skalabilne storitve, ki jih je mogoče upravljati in posodabljati brez vpliva na sistem, tudi med izvajanjem. Slabost je, da lahko zmanjša zmogljivost omrežja zaradi povečanega števila ponavljajočih se podatkov poslanih v seriji zahtevkov, saj teh podatkov ni mogoče pustiti v

strežniku med deljenimi vsebinami.

4. **HTTP odzivi**. Zahteve za storitev RESTful vedno zahtevajo odgovor. Ta odgovor je v obliki XML, JSON, HTML ali kakšni drugi določeni obliki. Odgovori vsebujejo podrobnosti o spremembi osnovnih podatkov, sporočila o napakah in hipertekstne povezave do drugih storitev. povezane vire, odvisno od operacije.

4.5.2 Nastavitve sheme baze podatkov za RESTful dostop

Če želite omogočiti shemo DB za dostop RESTful, jo je treba najprej registrirati v sistemu ORDS . Ta lahko preprosto naredimo, kot je prikazano na sliki 4.13 tako da uporabimo RESTful services orodje v SQL Workshop (1), nato izberemo “Register Schema with ORDS” (2), vnesemo naziv, ki se uporablja pri oblikovanju URL, ki se sklicuje na katerokoli storitev RESTful v shemi (3) in na koncu potrdimo z *“Save Schema Attributes”* (4), da se prikaže nadzorna plošča ORDS RESTful Services, ki jo uporabljamo za upravljanje RESTful storitev.



Slika 4.13: Shemi na bazi podaktov omogočimo RESTful dostop.

V primeru, da v 3. koraku (3) omogočimo možnost “Install Sample Service”, se namesti tudi vzorčni modul `oracle.example.hr` v katerem je na voljo primer *predloge* in *upravljavca* (glej podpoglavje 4.5.3), ki izvajajo več različnih operacij za pridobivanje in prikaz informacij o zaposlenih iz tabele Employees, kot je prikazano na sliki 4.15.

4.5.3 Moduli | Predloge | Upravljavci virov

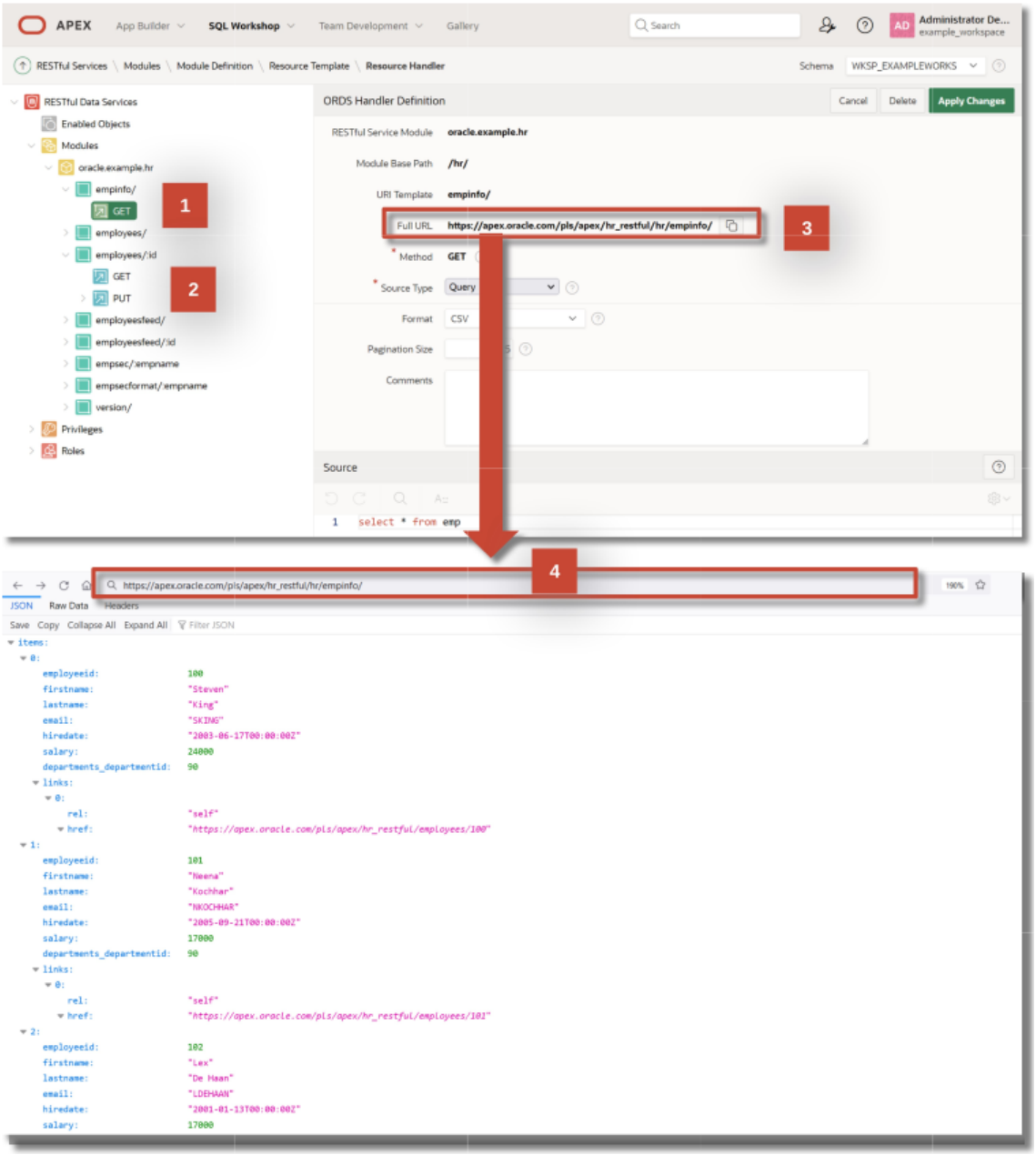
V sistemu ORDS je mogoče storitve RESTful ustvariti z uporabo treh osnovnih komponent, ki jih sestavljajo moduli virov (angl. Resource modules), predloge virov (angl. Resource Templates) in upravljavci virov (angl. Resource Handlers) (nekaj primerov prikazanih na sliki 4.15). Povezave med temi osnovnimi gradniki so prikazana z ER-diagramom na sliki 4.14.



Slika 4.14: Povezave med različnimi gradniki ORDS RESTful storitev.

- **Moduli virov.** Modul virov je vsebnik, ki združuje niz povezanih storitev RESTful iz določene sheme baze podatkov, ki ima omogočen REST. Shema baze podatkov lahko vsebuje več modulov virov. Modul virov ne zagotavlja le načina za enolično identifikacijo skupine, ampak določa tudi enolično osnovno pot do URI (dostop do niza storitev), ki je določen za vsak modul, Na primer, modul virov, ki omogoča dostop do podatkov o zaposlenih se imenuje `oracle.example.hr` in pot do storitve je `/hr/`, kot prikazuje slika 4.15.
- **Predloge virov.**. Predloga vira določa posamezno storitev, ki jo lahko pokličemo, na primer `employees/:id`, kot prikazuje slika 4.15. Predloge virov se nahajajo v predlogah modulih virov. Vsaka predloga vira določa *URI vzorec*, kjer je dostopen, npr., "empinfo/" (slika 4.15) in vsebuje vsaj en upravljavec virov (glej spodaj).
- **Upravljavci virov.** Vsak upravljavec vira implementira eno (in samo eno) od štirih različnih HTTP operacij, ki ustrezajo eni od standardnih CRUD-operacij (Create, Read, Update, Delete) .
  - `POST` kreira nov vir ali doda vir v zbirko, podobno kot SQL `INSERT` stavek.
  - `GET` pridobi predstavitev vira, podobno kot SQL `SELECT` stavek.
  - `PUT` spremeni vrednosti obstoječega vira, podobno kot SQL `UPDATE` stavek.
  - `DELETE` izbriše obstoječ vir, podobno kot SQL `DELETE` stavek.

Za vsako operacijo, ki je povezana z isto predlogo vira, moramo določiti upravljavec vira. Na primer, če imamo operacijo ki vrne podatke in operacijo, ki shrani podatke, moramo definirati upravljavca vira za vsako od operacij (slika 4.15 – za predlogo vira `employees/:id`. Definiramo dve operaciji `GET` (1) in `PUT` (2)).



Slika 4.15: Primer RESTful Services za tabelo Employees.

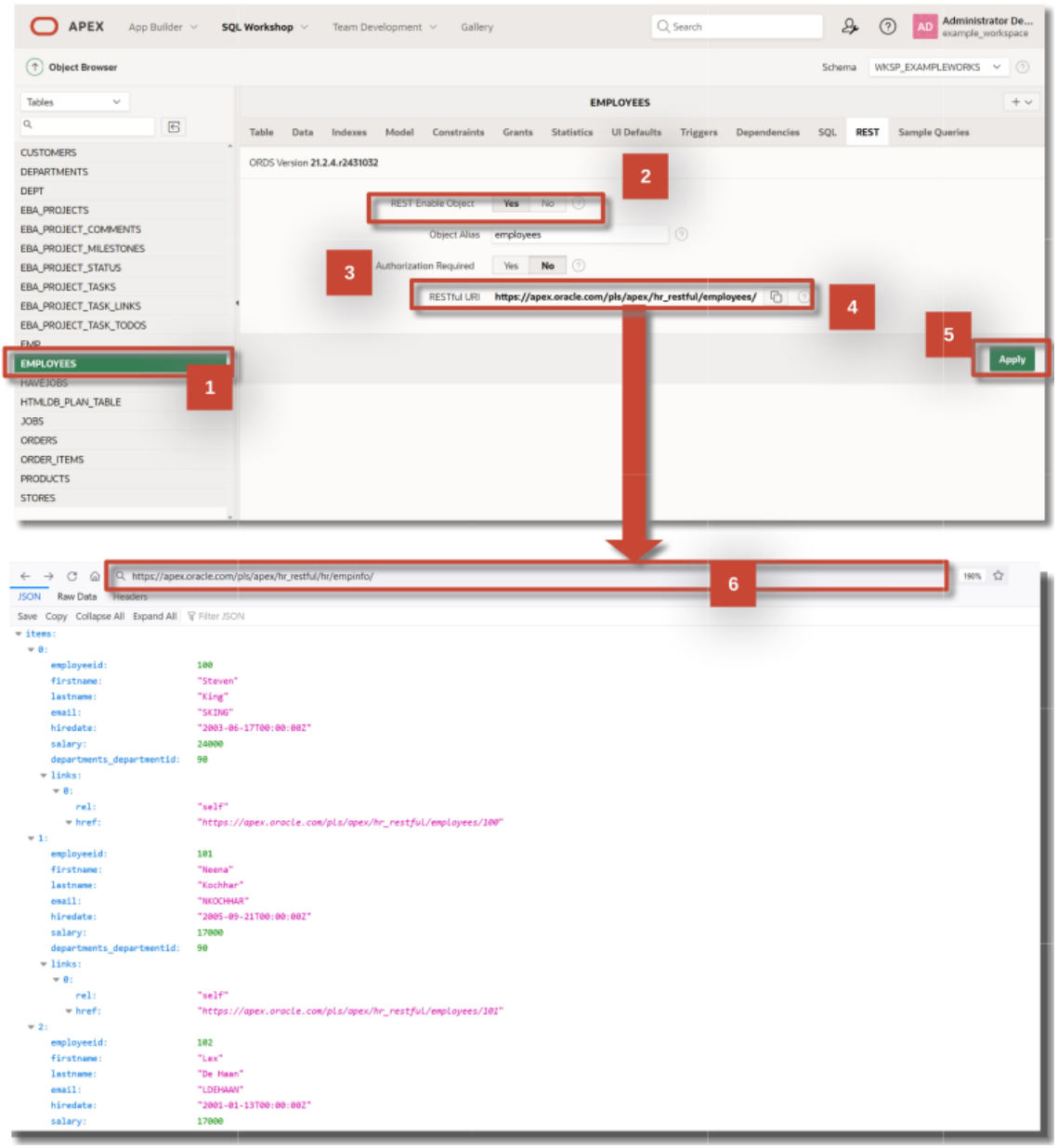
Slika 4.15 podaja primer različnih gradnikov RESTful storitev, vključno z obliko želenega formata odziva (CSV) ter SQL poizvedbo, ki določa tip operacije upravljavca virov. Ko kreiramo RESTful storitev, jo lahko testiramo tako, da enostavno vnesemo generiran naslov za dostop v spletni brskalnik (slika 4.15 (3) in (4)).

4.5.4 Uporaba "AutoREST"

Kot je bilo prikazano v predhodnem besedilu, je potrebno za vsako vrsto operacije na bazi podatkov, ki jo kot RESTful storitev ponujamo zunanjim odjemalcem, izvesti nekaj korakov. Za najbolj enostavne poizvedbe, kot je pregled vseh podatkov, pa APEX ponuja tudi posebno bližnjico, tako imenovano AutoREST funkcionalnost. AutoREST je enostaven in hter način predstavljanja tabele v bazi podatkov kot REST vira. Pri tem sicer nismo najbolj fleksibilni in ne moremo prilagajati poizvedbe, vendar pa se v veliki meri izognemo času in trudu, ki bi ga sicer porabili za ročno

nastavitev RESTful storitve. Kot smo že omenili, AutoREST omogoča samo poizvedbo cele tabele, in ima samo eno možno obliko izhodnega zapisa, to je JSON format.

Autorest lahko enostavno uporabimo na naslednji način: Po tem, ko smo shemo omogočili za RESTful dostop (glej 4.5.2), lahko vsako tabelo posamezno določimo kot vir z uporabo Object Browser v SQL Workshop kot prikazuje slika 4.16.



Slika 4.16: Omogočanje REST storitve v Object Browser.

Ko izberemo ustrezno tabelo v bazi podatkov (1), se pomaknemo na zavihek REST (2), izberemo zahtevani način avtorizacije (3) in potrdimo nastavitve (5), je za to tabelo REST aktiviran in avtomatsko generiran URL za dostop se pojavi v prikaznem polju (4). Tudi v tem primeru lahko REST storitev testiramo tako, da URL naslov vpišemo v spletni brskalnik (6).

4.6 Vprašanja

1. Opišite postopek uvoza podatkov preko SQL Workshop ali preko App Builder?

2. Na kakšne načine lahko izvozimo podatke?
3. Zakaj uporabljamo REST? Kakšen je namen in katere so prednosti uporabe REST?

4.7 Odgovori

1. Data Workshop omogoča uvoz podatkov iz zunanjih datotek s pomočjo čarovnika ali preko Object Browser. Uvoz podatkov iz Data Workshop lahko zaženemo tudi iz App Builder.
2. Izvoz podatkov v datoteko iz ene tabele lahko izvajamo preko SQL Workshop, Object Browser. Data Workshop omogoča delo s čarovnikom za izvoz, ki omogoča da izberemo vrsto datoteke in tabele kot tudi stolpce, katerih podatke želimo izvoziti.
3. RESTful storitve zagotavljajo interoperabilnost med različnimi sistemi na Internetu, saj omogočajo poizvedbe in manipulacije podatkov.



5. Kako izdelate prvi osnutek aplikacije?

A. ANGEIOPLASTIS, G. MYLLIS, A. TSIMPIRIS, D. VARSAMIS; PREVOD A. BAGGIA

5.1 Zakaj potrebujemo aplikacije?

V današnjem hitro razvijajočem se poslovnem okolju so aplikacije postale ključno orodje, s katerim podjetja ohranjajo konkurenčnost in izpolnjujejo nenehno spreminjajoče se zahteve svojih strank. Izdelava aplikacij podjetjem omogoča racionalizacijo poslovanja, povečanje učinkovitosti in izboljšanje izkušenj strank, kar lahko na koncu privede do večje dobičkonosnosti.

Eden od glavnih razlogov, zakaj podjetja gradijo aplikacije, je avtomatizacija procesov, ki se trenutno izvajajo ročno. Podjetje za e-trgovino lahko na primer razvije aplikacijo, ki avtomatizira postopke upravljanja zalog in izpolnjevanja naročil. S tem lahko zmanjša verjetnost človeških napak ter izboljša hitrost in natančnost svojega poslovanja.

Drugi ključni razlog, zakaj podjetja izdelujejo aplikacije, je izboljšanje izkušnje strank. Aplikacije se lahko razvijejo tako, da strankam omogočajo samopostrežne možnosti, na primer možnost spremljanja naročil ali posodabljanja podatkov o računu. To lahko pomaga skrajšati čas čakanja strank in povečati njihovo zadovoljstvo.

Poleg tega lahko aplikacije podjetjem pomagajo zbirati dragocene podatke o njihovih strankah, ki jih lahko uporabijo za izboljšanje svojih tržnih strategij in razvoj ciljno usmerjenih oglaševalskih kampanj. Trgovec na drobno lahko na primer razvije aplikacijo, ki spremlja nakupne vzorce strank, kar mu omogoča, da svojim strankam pošilja prilagojena priporočila in promocije.

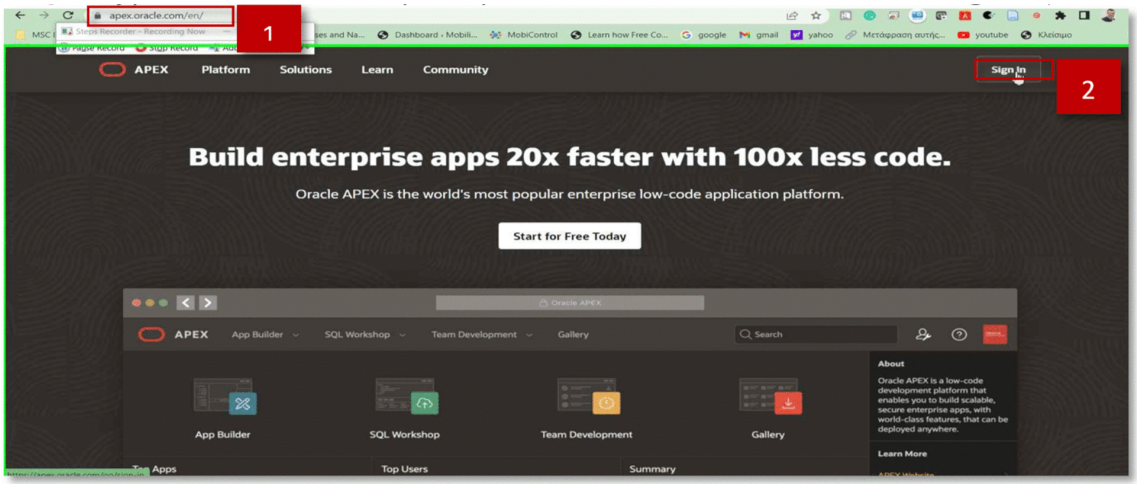
Na splošno je izdelava aplikacij postala močna poslovna potreba, saj podjetjem omogoča učinkovitejše poslovanje, boljšo izkušnjo za stranke in vpogled v vedenje njihovih strank. Ker poslovni svet postaja vse bolj digitalen, se bo pomen gradnje aplikacij le še povečeval.

5.2 Nastavitev okolja v ORACLE APEX

V predstavljenem primeru vas bomo seznaniliz z okoljem Oracle APEX. Vodili vas bomo preko nastavitev okolja v APEX, na primeru baze podatkov o zaposlenih, oddelkih in projektih prikazali delo z SQL Workshop ter predstavili kako v Oracle APEX razviti spletno aplikacijo na osnovi modela podatkov.

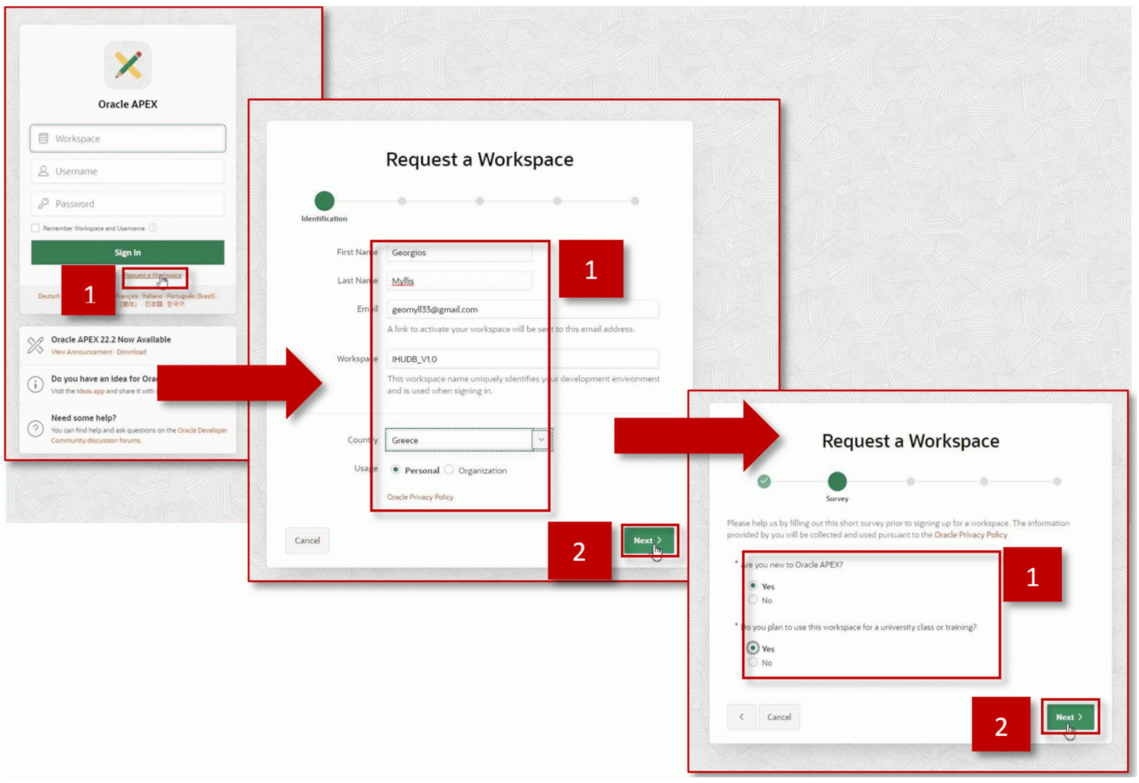
Za delo na primeru bomo sledili naslednjim korakom:

1. korak: Odprite spletni brskalnik in vpišite naslov “https://apex.oracle.com/en/”. Kliknite na gumb **Sign In** kot prikazuje slika 5.1.



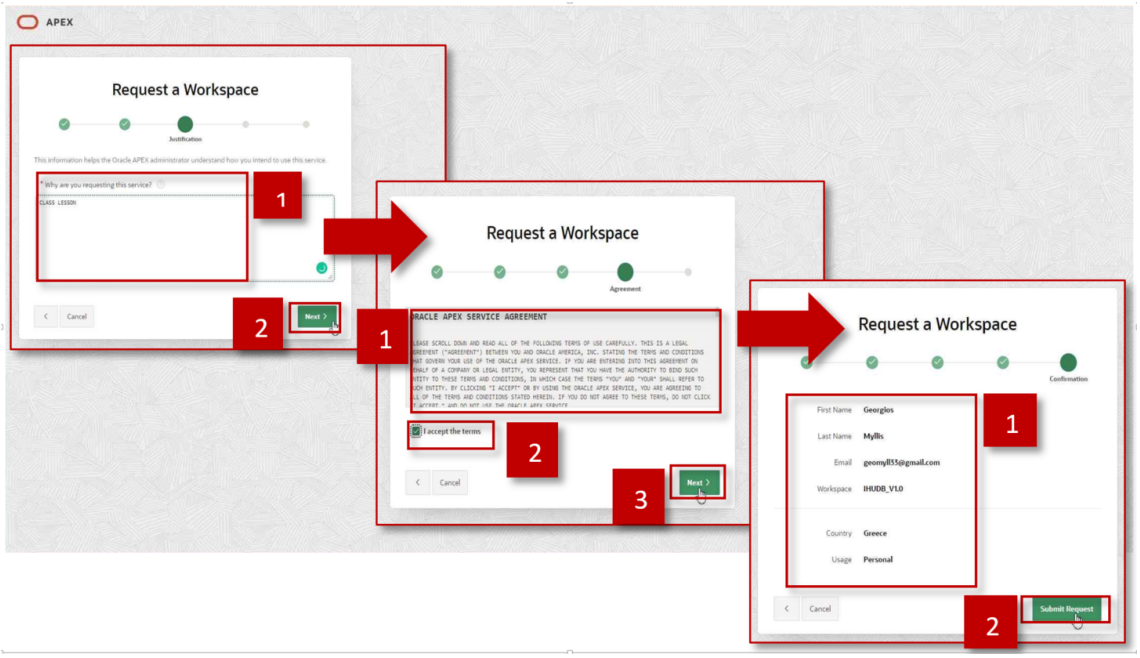
Slika 5.1: Odprite spletno stran in izberite možnost prijave.

2. korak: Kliknite na gumb **Request a Workspace**. V oknu, ki se odpre, vpišite zahtevane podatke za identifikacijo in potrdite izbiro s klikom na gumb **Next**. Nato odgovorite na vprašanja kot prikazuje slika 5.2 in nadaljujte s klikom na gumb **Next**.



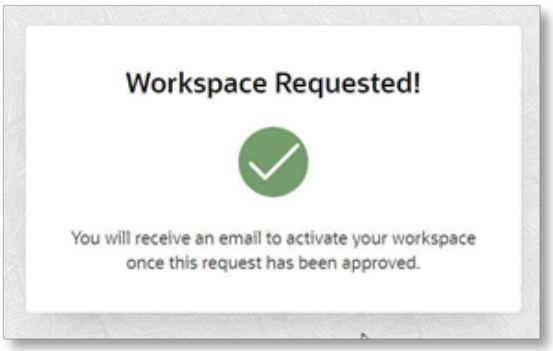
Slika 5.2: Začetni koraki pridobivanja delovnega prostora v Oracle APEX.

3. korak: Odgovorite na vprašanje o tem, zakaj boste uporabili Oracle APEX ter kliknite na gumb **Next**. V naslednjem oknu preberite pogoje, potrdite da se z njimi strinjate in nadaljujte s klikom na gumb **Next**. Preverite ali ste vnesli pravilne podatke, nato pa potrdite zahtevo s klikom na gumb **Submit request** kot prikazuje slika 5.3.



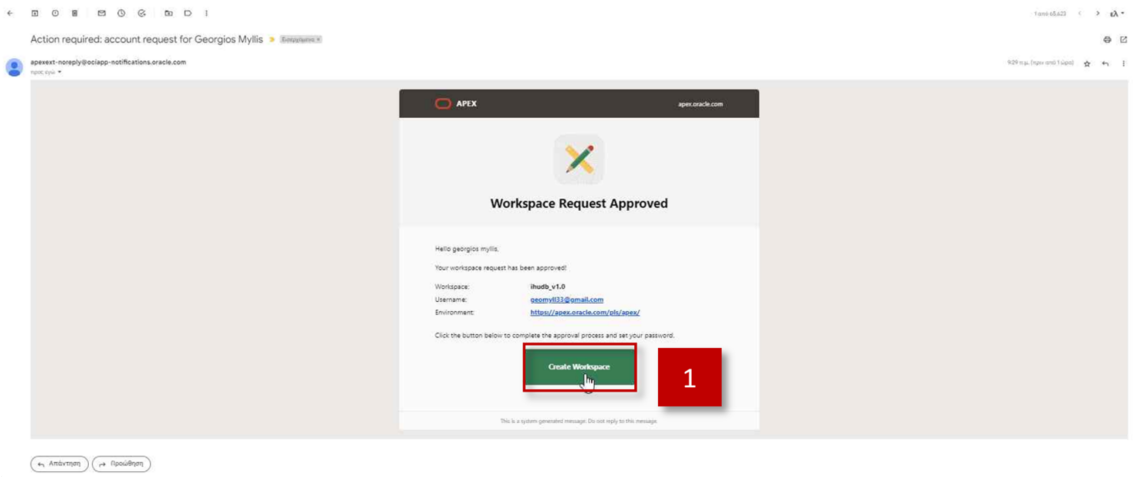
Slika 5.3: Zaključni koraki pri zahtevi za dodelitev brezplačnega delovnega prostora v Oracle APEX.

4. korak: Ko ste oddali zahtevo, se pojavi obvestilo, da ste zahtevo uspešno oddali in da boste v kratkem prejeli elektronsko sporočilo na navedeni spletni naslov, kot prikazuje slika 5.4.



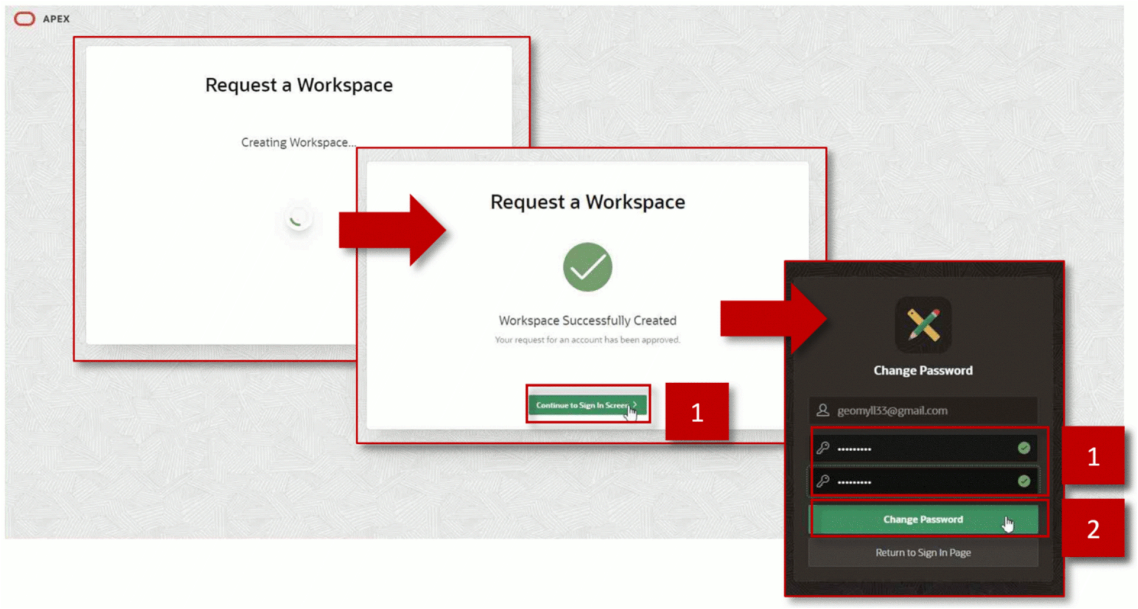
Slika 5.4: Potrditev zaključka procesa.

5. korak: Preverite, ali ste v svoj elektronski predal že prejeli sporočilo vezano na Oracle APEX. Pred nadaljevanjem dela morate namreč potrditi zahtevo preko povezave, ki jo prejmete v elektronskem sporočilu. Ko je zahteva potrjena, izberite gumb **Create Workspace** kot prikazuje slika 5.5.



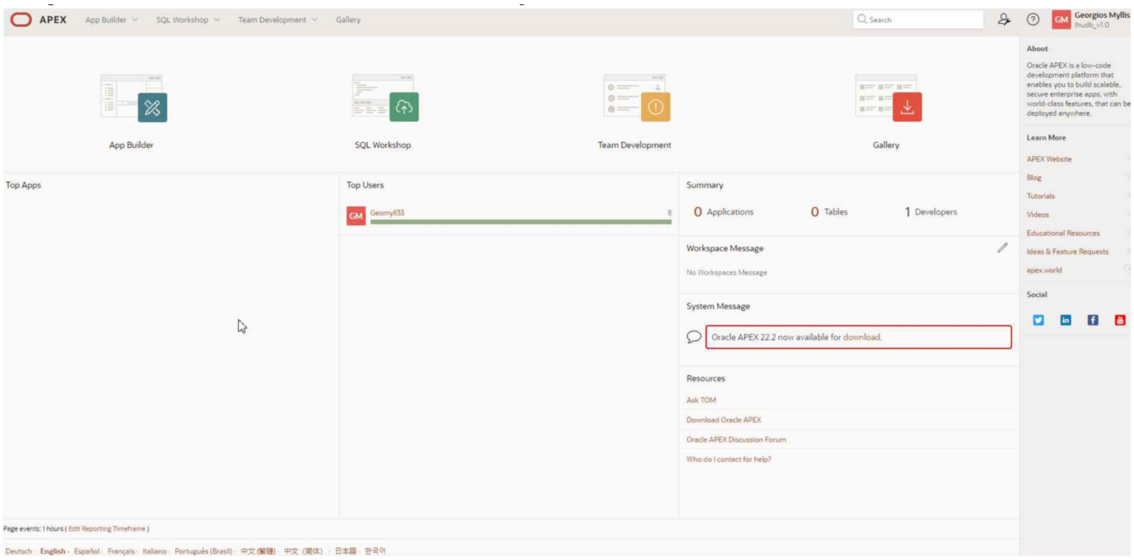
Slika 5.5: Elektronsko sporočilo za aktivacijo računa.

6. korak: Počakajte, da se zaključi proces priprave delovnega prostora. Ko bo proces zaključen, se bo odprlo pogovorno okno, ki vas bo o tem obvestilo. Kliknite **Continue to Sign-in Screen**. Odpre se novo pogovorno okno, ker boste pred prijavo nastavili še geslo. Nastavite geslo, ga potrdite ter nato kliknite na **Change Password**, kot prikazuje slika 5.6.



Slika 5.6: Nastavitev novega gesla za delovni prostor v Oracle APEX.

7. korak: Kot je prikazano na sliki 5.7, je zdaj okolje Oracle APEX pripravljeno za uporabo.



Slika 5.7: Delovni prostor Oracle APEX.

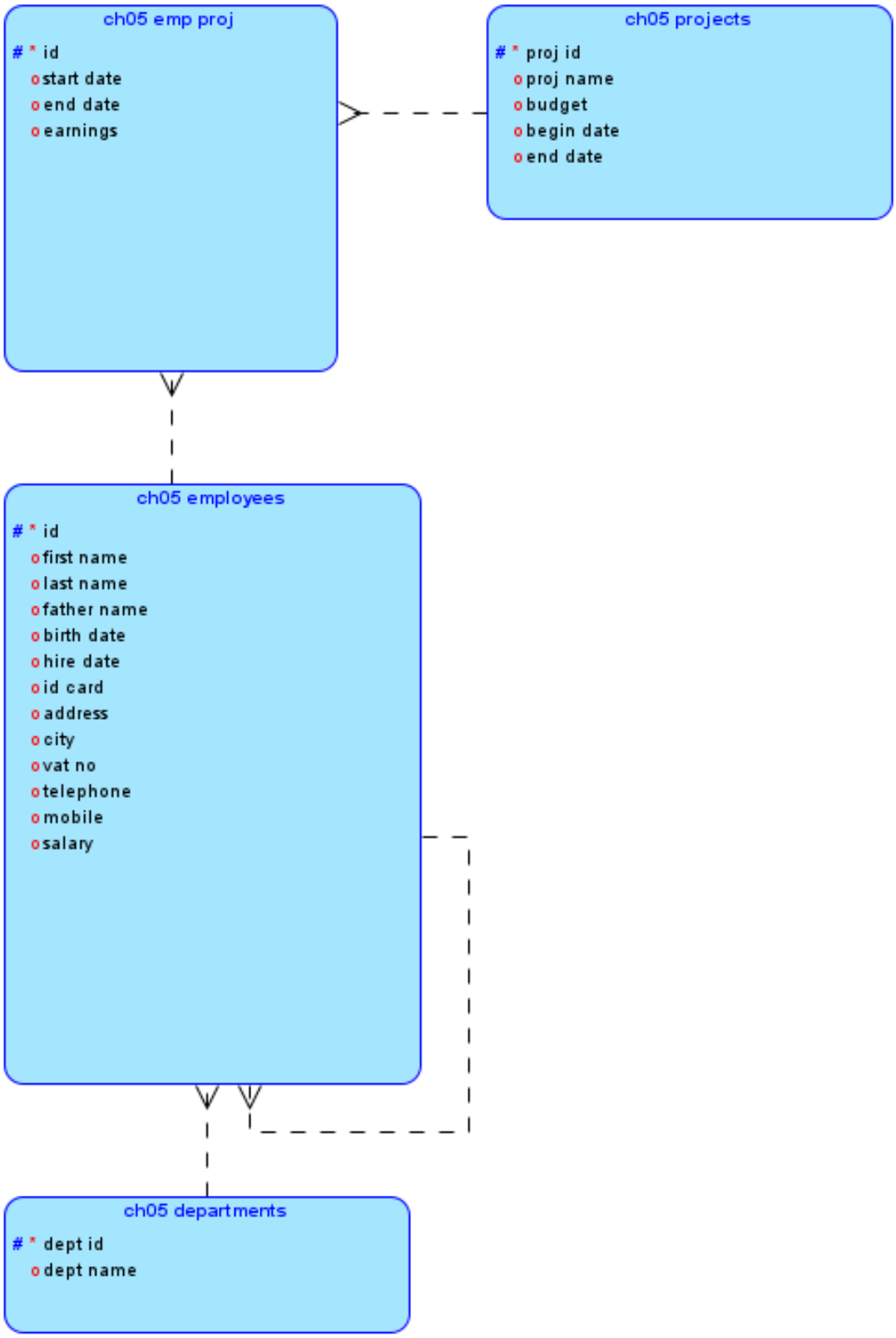
5.3 Priprava primera v SQL Workshop

Obstaja več načinov priprave baze podatkov v APEX okolju. Ena možnost je uporaba Object Browser, grafičnega orodja, ki uporabnikom omogoča ustvarjanje tabel in pol v tabelah preko vnaprej določenih predlog.

Druga možnost je uporaba SQL ukazov, ki omogočajo uporabniku kreiranje tabel in ostalih objektov na bazi podatkov z izvajanjem SQL ukazov. Ta metoda omogoča večjo prilagodljivost in nadzor nad strukturami v bazi podatkov, po drugi strani pa zahteva nekaj več znanja.

Po drugi strani pa lahko uporabimo tudi Quick SQL orodje, ki omogoča hitro generiranje SQL skript za kreiranje tabel v bazi podatkov, omejitev, ter tudi vzorčnih podatkov. Quick SQL poenostavi postopek kreiranja baznih objekotv tako da ni potrebno pisati zapletenih SQL skript.

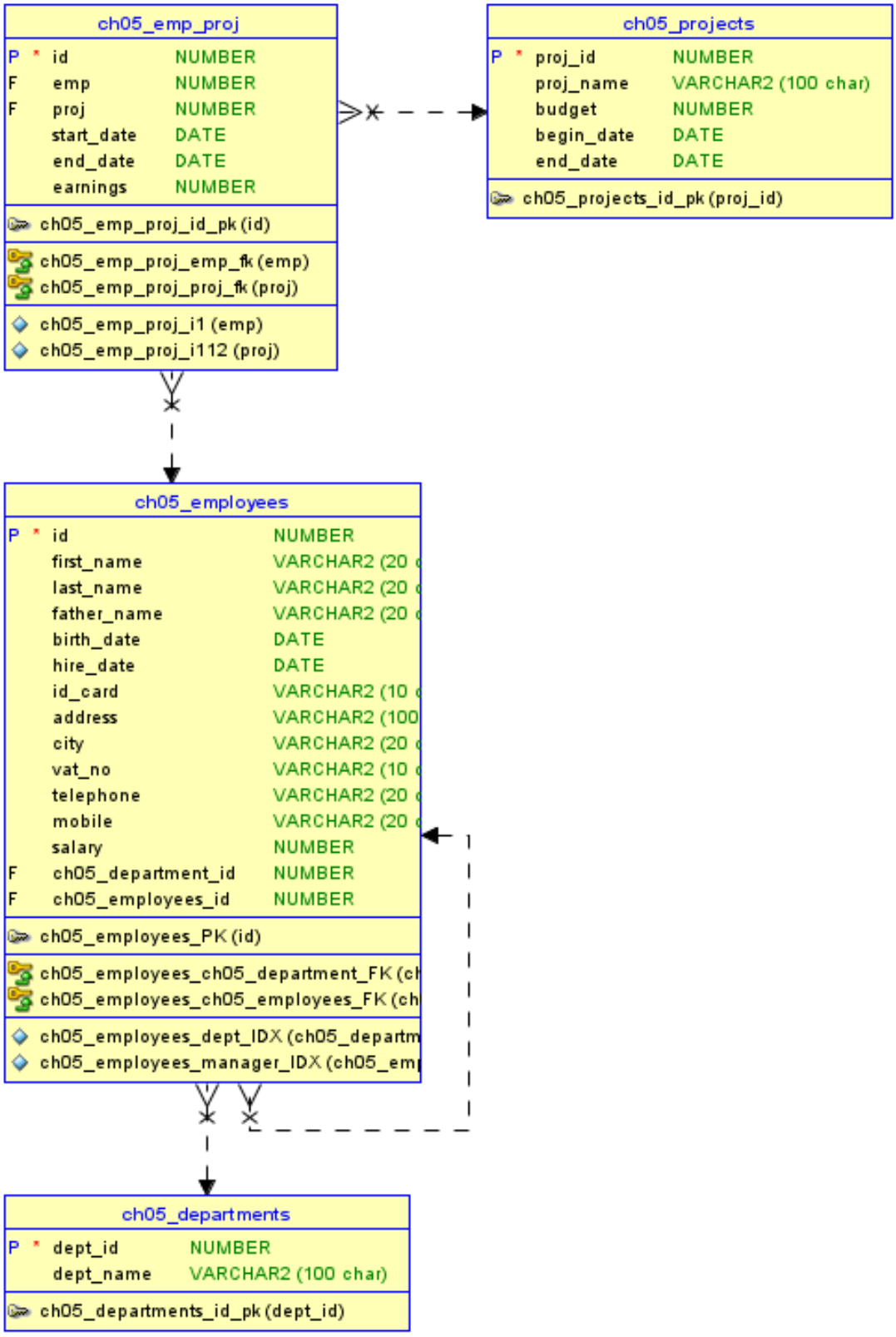
V predstavljenem primeru bomo uporabili metodo Quick SQL. Skripto smo generirali na osnovi logičnega modela, ki je predstavljen na sliki 5.8.



Slika 5.8: Logični model baze podatkov za primer Running HR.

Glede na relacijski podatkovni model, ki smo ga z Oracle Data Modeler (ODM) generirali na

osnovi logičnega modela in je predstavljen na sliki 5.9, bomo pripravili strukturo baze podatkov.



Slika 5.9: Relacijski podatkovni model za primer Running HR.

5.3.1 Quick SQL

Kodo za Quick SQL sestavlja niz navodil za generiranje vzorčnih podatkov. Vsak razdelek se začne z imenom tabele, ki mu sledi ključna beseda "/insert" in število, ki določa koliko vrstic testnih podatkov želimo imeti v tabeli.

```
ch05_departments /insert 5
    dept_name vc100

ch05_projects /insert 15
    proj_name vc100
    budget num
    begin_date date
    end_date date

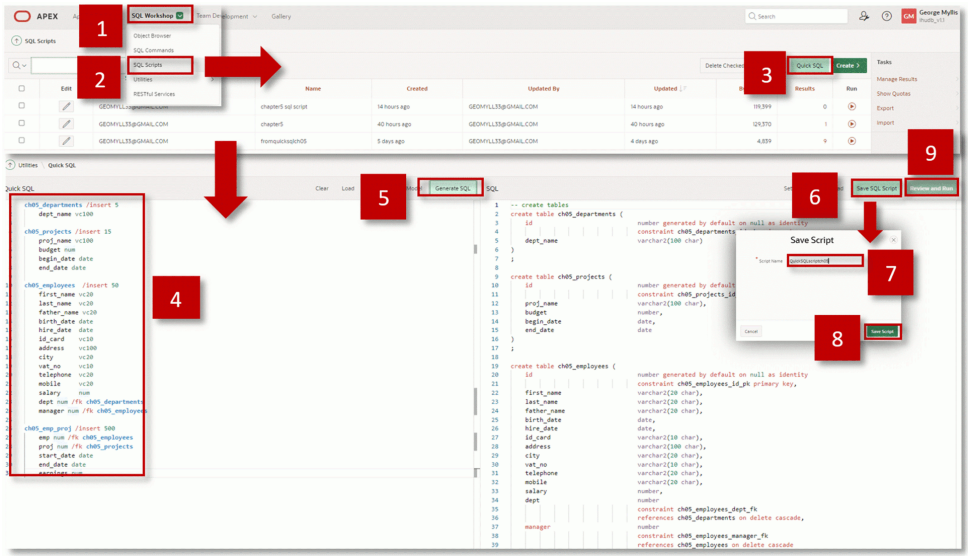
ch05_employees  /insert 50
    first_name vc20
    last_name  vc20
    father_name vc20
    birth_date date
    hire_date  date
    id_card    vc10
    address    vc100
    city       vc20
    vat_no     vc10
    telephone  vc20
    mobile     vc20
    salary     num
    dept num /fk ch05_departments
    manager num /fk ch05_employees

ch05_emp_proj /insert 500
    emp num /fk ch05_employees
    proj num /fk ch05_projects
    start_date date
    end_date date
    earnings num
```

S temi ukazi bomo pripravili shemo in testne podatke za bazo podatkov, ki bo vključevala oddelke, projekte, zaposlene in povezave med njimi.

1. korak: Sledite spodnjim korakom, ki so prikazani na sliki 5.10:

1. Premaknite se na **SQL Workshop**.
2. Izberite **SQL Scripts** izmed ponujenih možnosti.
3. Kliknite na **Quick SQL**.
4. Vstavite zgoraj prikazano kodo v vmesnik.
5. Kliknite na **Generate SQL**. Generirani SQL bo prikazan v Oracle SQL Output podoknu.
6. Kliknite na **Save SQL Script**.
7. Vpišite ime datoteke (skripte).
8. Kliknite na **Save script**.
9. Nadaljujte tako da kliknete na gumb **Review and Run**.



Slika 5.10: Vpišite Quick SQL kodo.

5.3.2 SQL Script

Kot smo že omenili, Quick SQL iz podanih kratkih ukazov generira SQL skript za kreiranje testne sheme s primerom sistema za vodenje kadrovske evidence. Pripravljeni so ukazi za kreiranje osnovnih tabel (zaposlen ch05_employees, oddelki ch05_departments, projekti ch05_projects) ter dodatne tabele ch05_emp_proj, ki omogoča povezovanje zaposlenih s projekti. Dodatno so vključeni tudi ukazi za kreiranje indeksov ter ostalih omejitev, ki bodo zagotovili konsistentnost podatkov in omogočili povezovanje.

```
-- create tables
create table ch05_departments (
    id            number generated by default on null as identity
                  constraint ch05_departments_id_pk primary key,
    dept_name     varchar2(100 char)
)
;

create table ch05_projects (
    id            number generated by default on null as identity
                  constraint ch05_projects_id_pk primary key,
    proj_name     varchar2(100 char),
    budget        number,
    begin_date    date,
    end_date      date
)
;

create table ch05_employees (
    id            number generated by default on null as identity
                  constraint ch05_employees_id_pk primary key,
    first_name    varchar2(20 char),
    last_name     varchar2(20 char),
    father_name   varchar2(20 char),
```

```
    birth_date   date,
    hire_date    date,
    id_card      varchar2(10 char),
    address      varchar2(100 char),
    city         varchar2(20 char),
    vat_no       varchar2(10 char),
    telephone    varchar2(20 char),
    mobile       varchar2(20 char),
    salary       number,
    dept         number
                 constraint ch05_employees_dept_fk
                 references ch05_departments on delete cascade,
    manager      number
                 constraint ch05_employees_manager_fk
                 references ch05_employees on delete cascade
)
;

-- table index
create index ch05_employees_i1 on ch05_employees (dept);
create index ch05_employees_i62 on ch05_employees (manager);

create table ch05_emp_proj (
    id           number generated by default on null as identity
                 constraint ch05_emp_proj_id_pk primary key,
    emp          number
                 constraint ch05_emp_proj_emp_fk
                 references ch05_employees on delete cascade,
    proj         number
                 constraint ch05_emp_proj_proj_fk
                 references ch05_projects on delete cascade,
    start_date   date,
    end_date     date,
    earnings     number
)
;

-- table index
create index ch05_emp_proj_i1 on ch05_emp_proj (emp);
create index ch05_emp_proj_i112 on ch05_emp_proj (proj);
```

V tabeli ch05_departments shranjujemo podatke o oddelkih, vključno z enoličnim označevalcem oddelka (id) in imenom (dept_name). V tabeli ch05_employees shranjujemo podatke o zaposlenih, vključno z enoličnim označevalcem (id), imenom (first_name), priimkom (last_name), imenom očeta (father_name), rojstnim datumom (birth_date), datumom zaposlitve (hire_date), številko kartice (ID_card), naslovom (address), mestom (city), davčno številko (VAT_number), telefonsko številko (telephone), številko mobitela (mobile), plačo (salary), ter identifikacijskimi številkami oddelka (dept) in nadrejenega (manager). V tabeli ch05_projects shranjujemo podatke o projektih, vključno z enoličnim označtevalcem (id), imenom (name), proračunom (budget), datumom začetka (start_date), in datumom zaključka (end_ date). Podatke o povezavah med zaposlenimi in projekti

shranjujemo v tabeli ch05_emp_proj. V tej tabeli imamo enolični označevalec (id), indentifikacijo zaposlenega (emp) in projekta (proj), datum začetka (start_date), datum konca (end_date), in zaslužek (earnings).

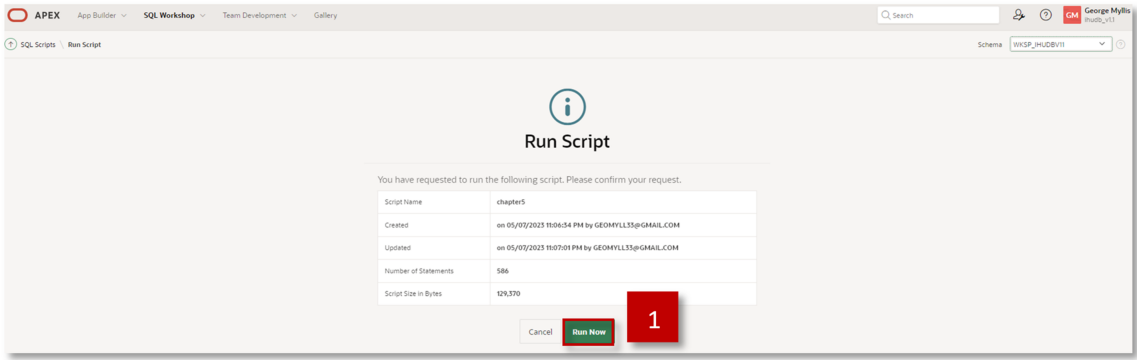
Na stolpcih v tabeli ch05_emp_proj so kreirani indeksi za stolpca zaposleni in projekt, da zagotovimo boljše performanse pri izvajanju poizvedb. Prav tako se indeksi kreirajo na tabeli ch05_employee, na stolpcih ki določata managerja in oddelek.

Na koncu dodamo še omejitve za zagotavljanje konsistentnosti podatkov in referenčne integritete med tabelami. Tuja ključa ch05_emp_proj_emp_fk in ch05_emp_proj_proj_fk zagotavljata, da se vrednosti v stolpcih o zaposlenih in projektih v tabeli ch05_emp_proj ujemajo z obstoječimi vrednostmi v tabelah ch05_employees in ch05_projects.

Ključa ch05_employees_dept_fk in ch05_employee_manager_fk pa zagotavljata, da sta podatka o oddelku in nadrejenemu v tabeli ch05_employee vezana na obostoječe podatke v tabelah ch05_departments in ch05_employees.

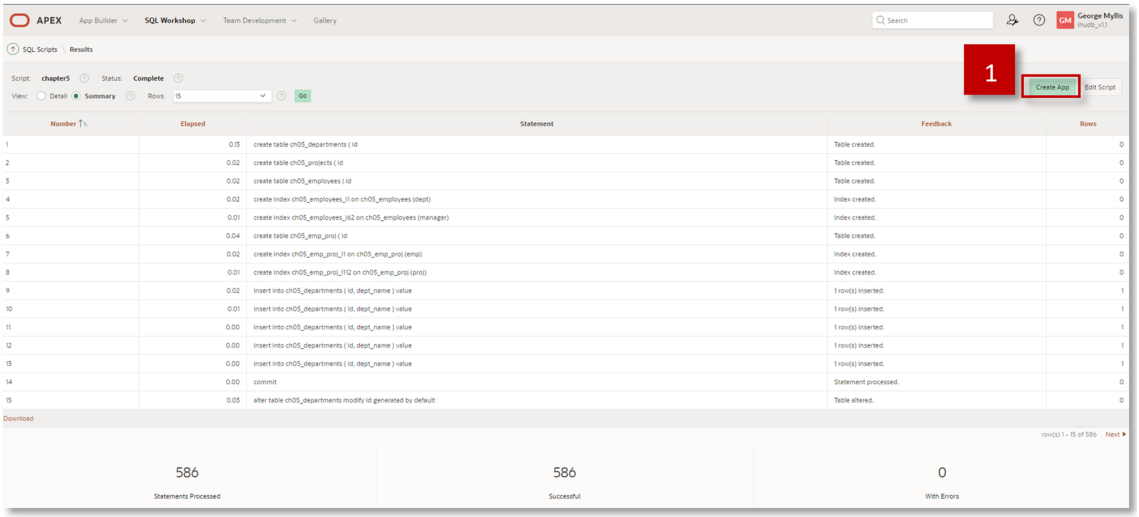
Poleg tega stavek ON DELETE CASCADE določa, da se, v primeru, ko pobrišemo vrednost iz tabele, vse povezane vrednosti v ostalih tabelah prav tako izbrišejo in tako zagotavljajo referenčno integriteto.

2. korak: Skript zaženemo s klikom na gumb **Run Now**, kot prikazuje slika 5.11.



Slika 5.11: Zagon SQL skript.

3. korak: Ko se vsi ukazi uspešno izvedejo, nadaljujemo s klikom na gumb **Create App**, kot prikazuje slika 5.12.



Slika 5.12: Začetek procesa kreiranja aplikacije.

5.4 Podatki v aplikaciji

Z razvito aplikacijo lahko izkoristimo moč podatkov ter pripravimo različne vpoglede ali omogočamo raznolike interakcije uporabnika z vmesnikom. Na ta način lahko s podatki poenostavimo proces odločanja. V tem podpoglavju se bomo osredotočili na podatke, kako omogočimo uporabnikom raziskovanje, analizo in interakcijo s podatkov na dinamičen in čim bolj intuitiven način.

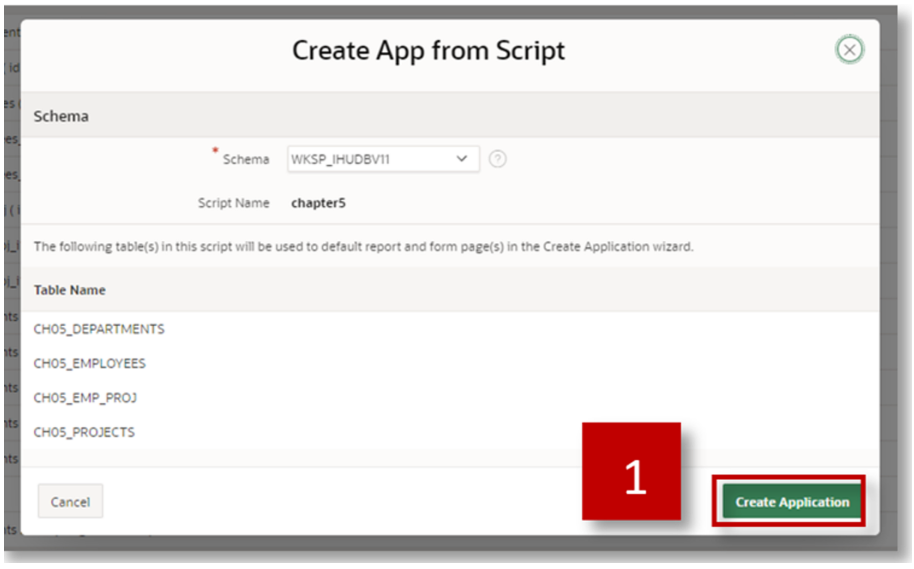
Glavni cilj podpoglavja je zagotoviti celovit pregled nad podatki, kar uporabniku omogoča pridobivanje dragocenih vpogledov in posledično sprejemanje premišljenih odločitev. Z različnimi tehnikami vizualizacije podatkov, kot so preglednice, grafi in interaktivne nadzorne plošče, lahko uporabnik zlahka razume zapletene podatkovne nize ter prepozna vzorce, trende in odstopanja.

Poleg tega v poglavju o podatkih predstavljamo zanesljive načine upravljanja s podatki, kot je na primeru omogočanje filtriranja, razvrščanja in iskanja različnih podatkov ali segmentov podatkov. Napredne funkcije kot je združevanje podatkov, poglobitev (angl. drill down) in funkcionalnost izvoza podatkov dodatno izboljšajo uporabniško izkušnjo ter omogočajo poglobljeno raziskovanje in analizo.

V tem podpoglavju poudarjamo tudi pomen integritete in kakovosti podatkov. Predstavljamo mehanizme za validacijo podatkov, upravljanje z napakami, ter čiščenje podatkov za zagotavljanje točnih, zanesljivih in konsistentnih podatkov.

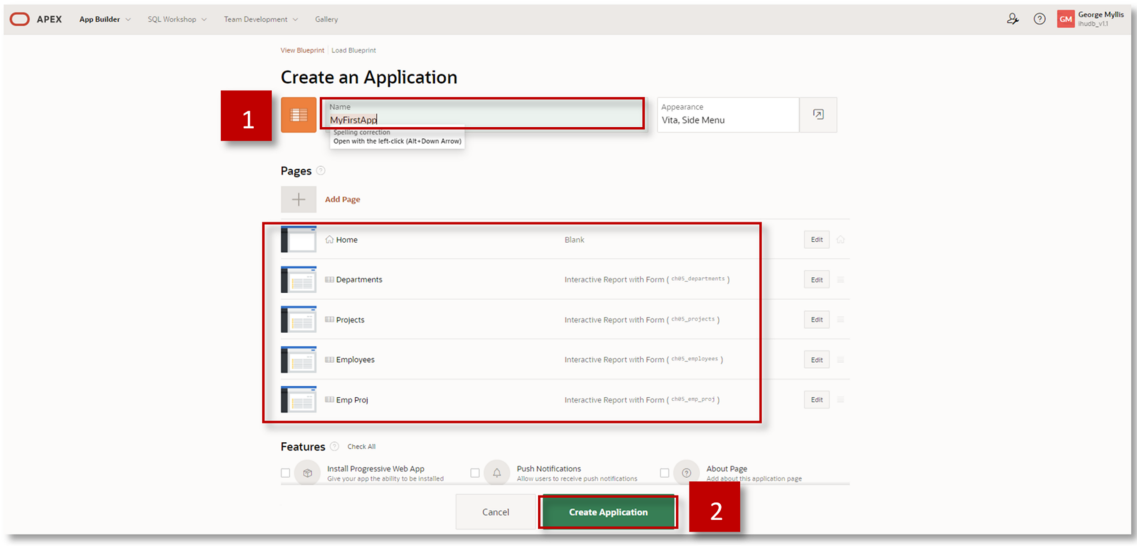
Predstavljamo, kako lahko uporabniki raziskujejo podatke z različnih zornih kotov, kako prilagajajo poglede glede na svoje preference, ter uporabljajo podatke za pridobivanje koristnih informacij. Tovrstni vpogledi služijo kot močno orodje za analizo podatkov, poročanje in podporo odločanju znotraj aplikacije, na osnovi česar uporabniki lahko sprejemajo pomembne odločitve na osnovi zbranih podatkov ter dosegajo uspešne rezultate.

1. korak: Ko se pojavi pogovorno okno **Create App from the Script**, kot prikazuje slika 5.13, preverite ali so bile kreirane vse tabele in ali bodo vse tabele uporabljene v naši novi aplikaciji, nato pa izbiro potrdite s klikom na gumb **Create Application**.



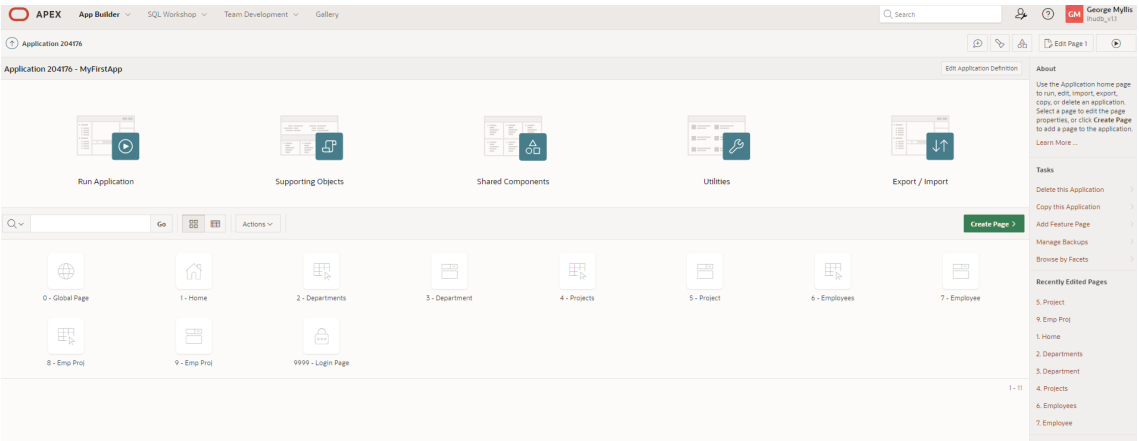
Slika 5.13: Pogovorno okno Create App from Script.

2. korak: Vpišite ime aplikacije, preverite ali so bili dodani vsi obrazci (angl. forms) in poročila (angl. reports) in ali jih bo aplikacija uporabljala, nato pa kliknite na gumb **Create Application**, kot prikazuje slika 5.14



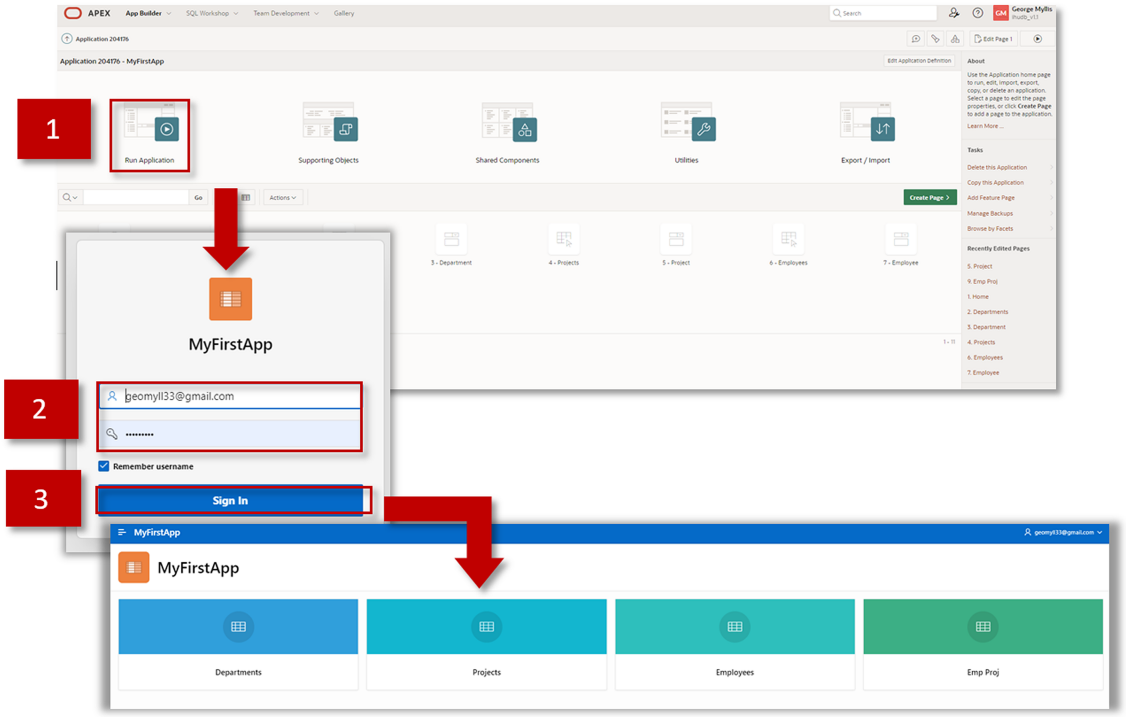
Slika 5.14: Spletna aplikacija je kreirana.

3. korak: Za vašo aplikacijo je bilo pripravljeno okolje v App builder vmesniku, kot prikazuje slika 5.15. Bodite pozorni, da znotraj tega okolja lahko korenito spremenite aplikacijo. Lahko dodajate, odstranjujete ali spreminjate polja na obrazcu, prilagajate izgled, določate validacijska pravila, ter izvajate veliko drugih sprememb, da bi aplikacijo prilagodili vašim potrebam.



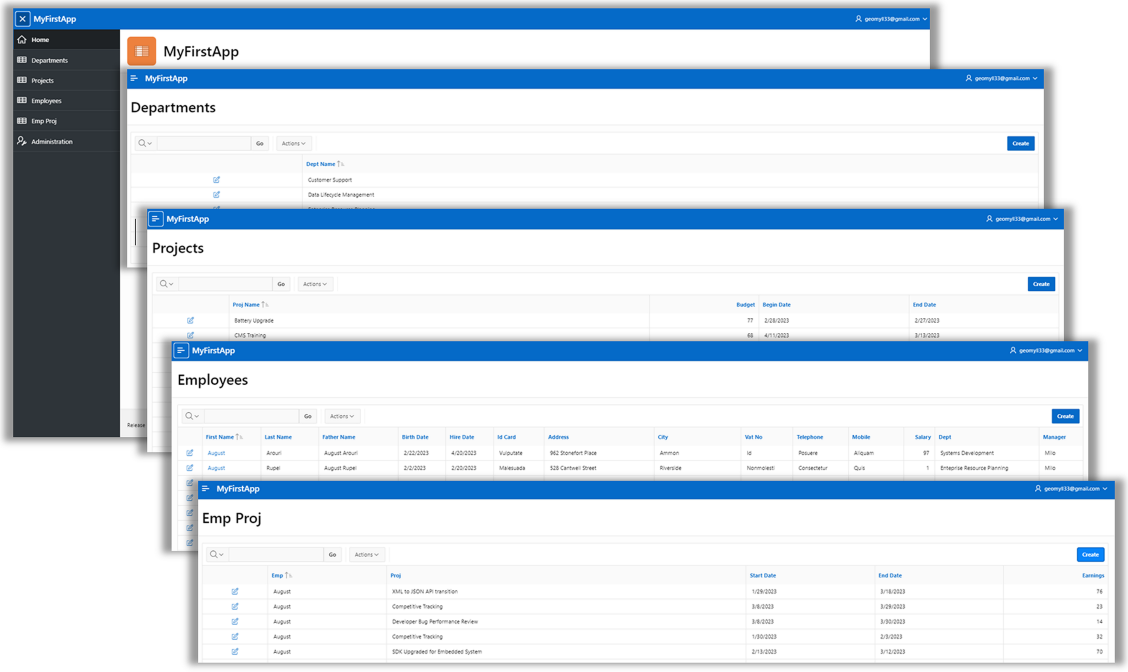
Slika 5.15: Okolje App builder.

4. korak: Kliknite na ikono **Run Application** v okolju App builder ter zaženite aplikacijo kot prikazuje slika 5.16. Odprla se bo nova stran v spletnem brskalniku s specifično povezavo do vaše APEX aplikacije. Vnesite svoje uporabniško ime in geslo, ki sta enaka kot ste jih določili za dostop do delovnega prostora na začetku, nato pa kliknite na gumb **Sign in**. Preusmerjeni boste na domačo stran vaše aplikacije.



Slika 5.16: Stran za prijavo v spletno aplikacijo.

5. korak: Po aplikaciji se enostavno lahko premikate z uporabo stranskega menija, kot prikazuje slika 5.17. Aplikacija je uporabniku prijazno okolje, ki omogoča dodajanje, spreminjanje ter predstavitev podatkov glede na vaše potrebe in želje.



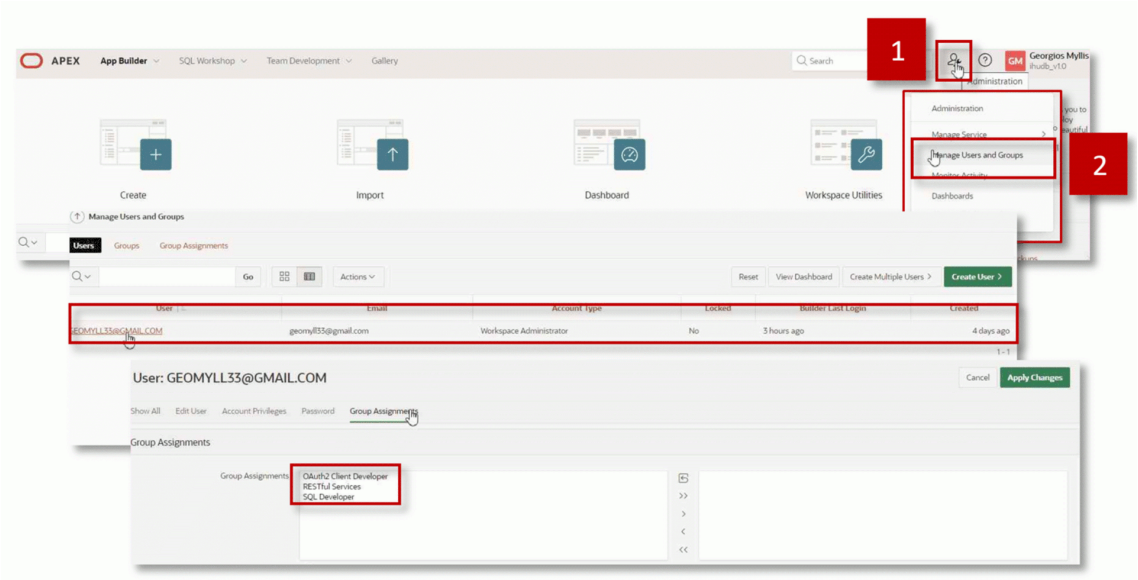
Slika 5.17: Okolje razvite aplikacije.

5.5 Upravljanje z aplikacijo

Oracle APEX ima vgrajeno metodo avtentikacije, ki omogoča ustvarjanje in upravljanje uporabniških računov v aplikaciji. Ta metoda je še posebej uporabna, če aplikacija uporablja Application Express Accounts.

V delovnem prostoru aplikacije izberite **Administration** in nato **Manage Users and Groups**. Osnovni uporabniški račun, ki ste ga kreirali na začetku, lahko uporabite kot administratorski uporabniški račun za celotno aplikacijo. S tem uporabniškim računom pa lahko določite tudi druge uporabniške račune, ter opredelite, kakšne pravice dostopa do aplikacije imajo.

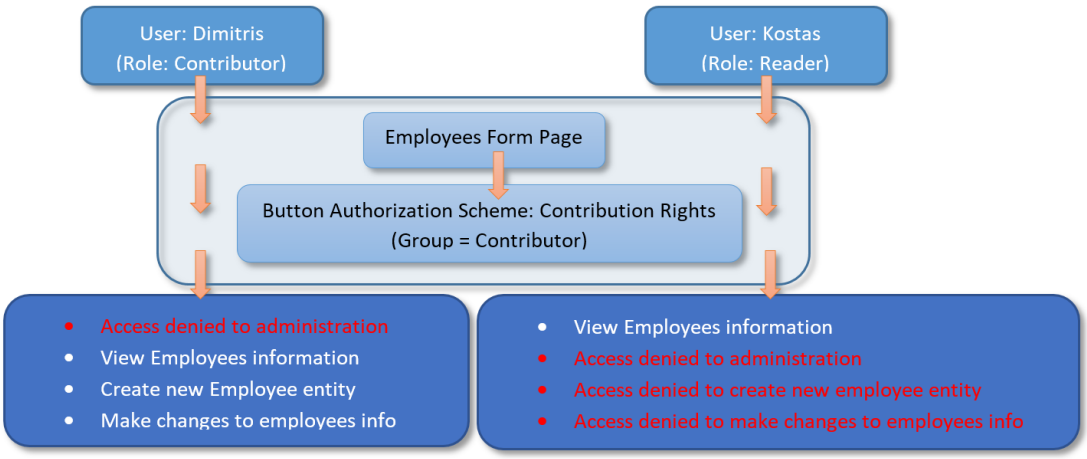
Administrator delovnega prostora ima pravico kreirati in spreminjati aplikacije in objekte v bazi podatkov, prav tako pa lahko upravlja z uporabniki, skupinami in razvojnimi storitvami. Razvijalci lahko kreirajo in spreminjajo aplikacije in objekte v bazi podatkov, medtem ko končni uporabniki nimajo pravic za razvoj aplikacij. Končni uporabniki lahko dostopajo do aplikacij, ki ne uporabljajo zunanje avtentikacijske sheme, kot je prikazano na sliki 5.18.



Slika 5.18: Okolje za upravljanje z uporabniki.

5.6 Nadzor dostopa

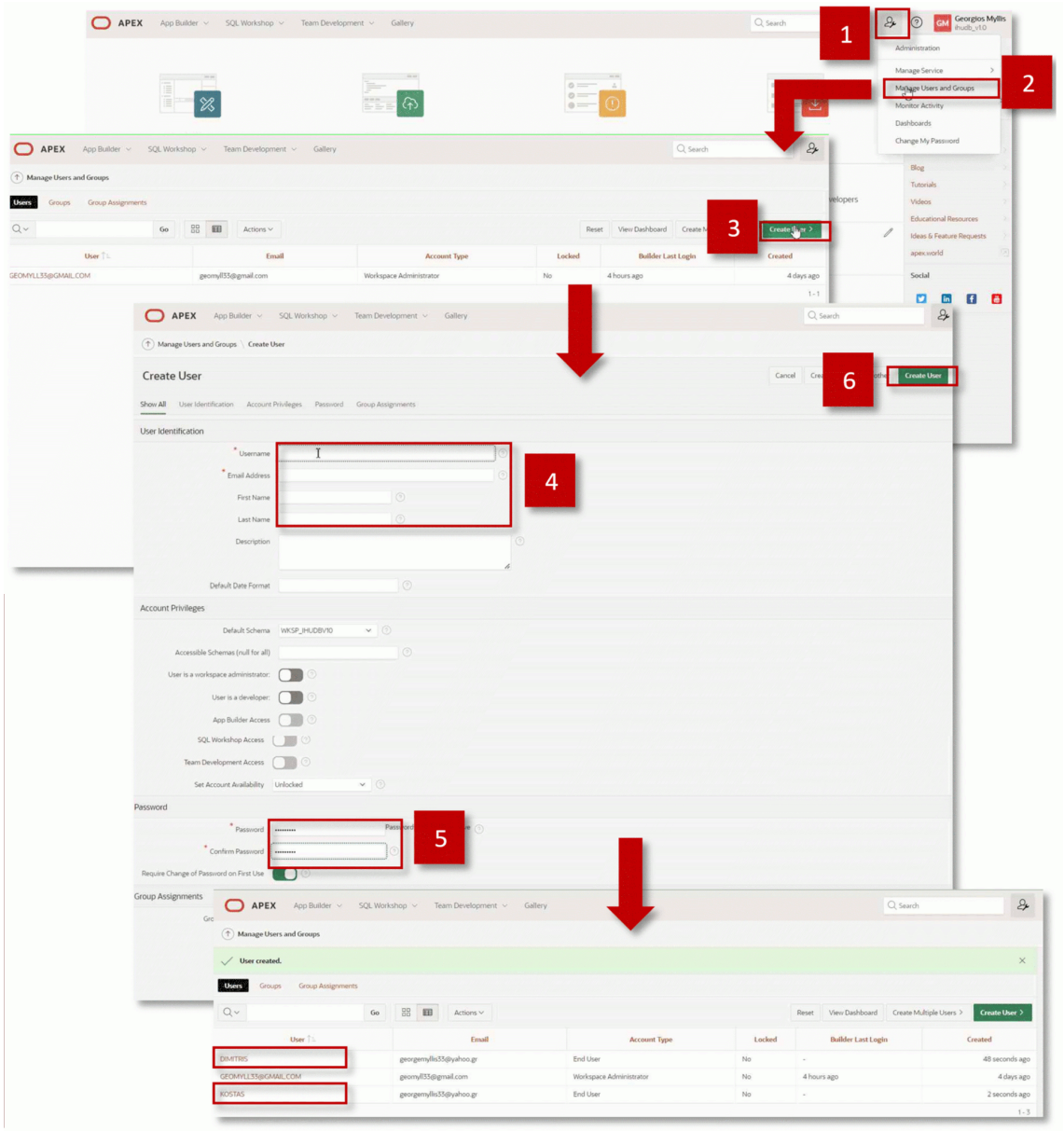
V nadaljevanju predstavljamo primer nadzora dostopa za uporabnike aplikacije glede na logiko, ki je predstavljena na sliki 5.19.



Slika 5.19: Primer nadzora dostopa.

Oracle APEX ponuja tri že vgrajene privilegije, ki omogočajo nadzor dostopa do aplikacije ali njenih komponent. Ti privilegiji so administracija, urejanje in vpogled, za vsakega od njih pa je določena posebna vloga. Administrcijo omogoča vloga “Administrator role”, urejanje vloga “Contributor role”, ter vpoglede “Reader role”.

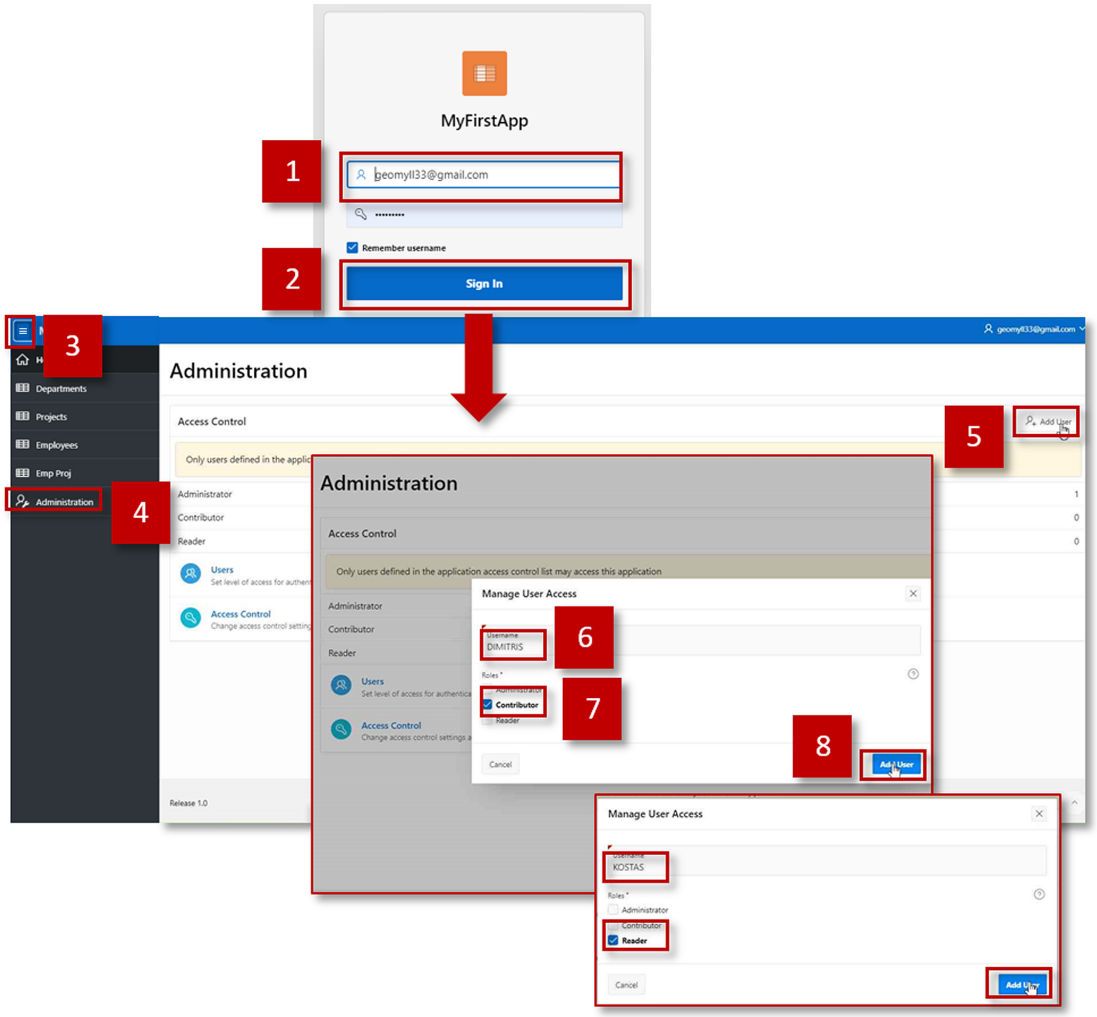
Za dodajanje uporabnikov in določanje njihovih vlog se premaknemo na **Manage Users and Groups** v osnovnem oknu delovnega prostora. Izberemo možnost **Administration**, ter nato **Manage Users and Groups**. V nadaljevanju izberemo **Create User**, vpišemo uporabniško ime in geslo, ter potrdimo z **Create User**, kot prikazuje slika 5.20.



Slika 5.20: Dodajanje novih uporabnikov.

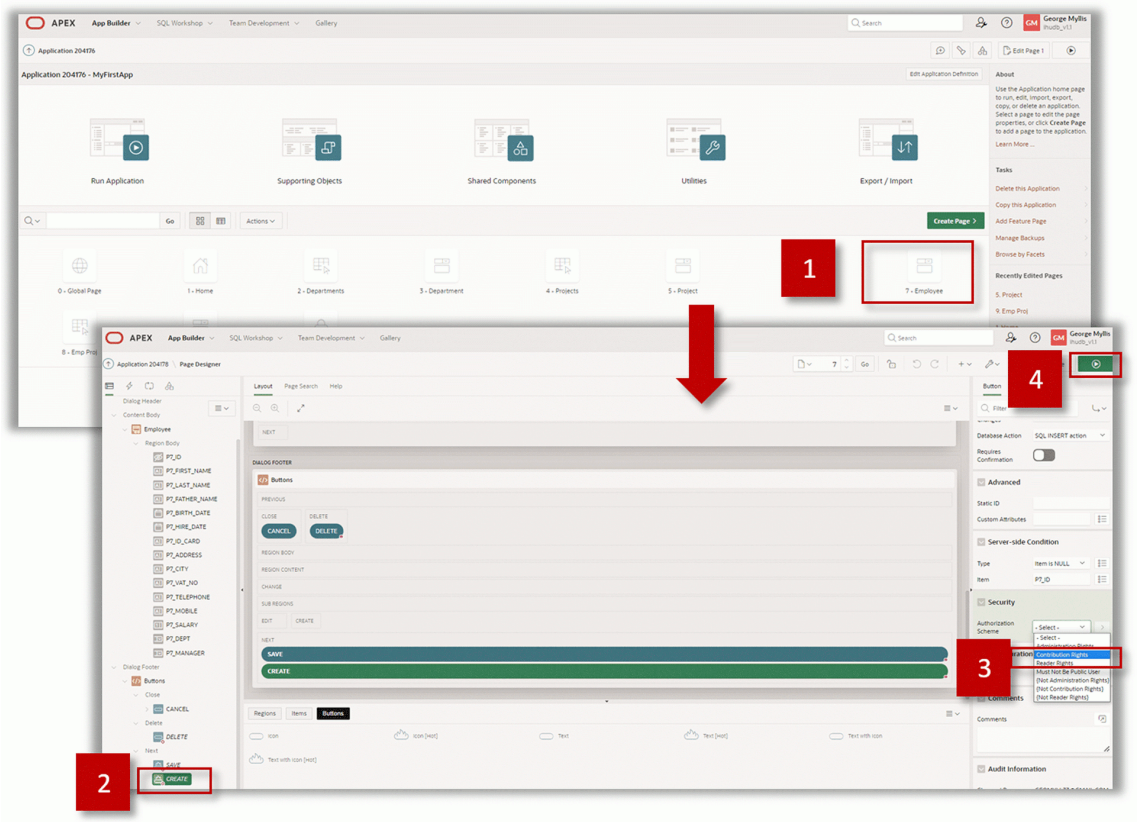
V nadaljevanju določimo vloge za uporabnike. V nadaljevanju prikazujemo, kako to lahko storimo neposredno v aplikaciji.

1. korak: Prijavimo se v aplikacijo s privilegiji administratorja, na to pa v stranskem meniju izberemo možnost **Administration**, ter **Add User**, izberemo uporabnika in vlogo ter potrdimo z **Add User** kot prikazuje slika 5.21.



Slika 5.21: Dodajanje vlog uporabnikom.

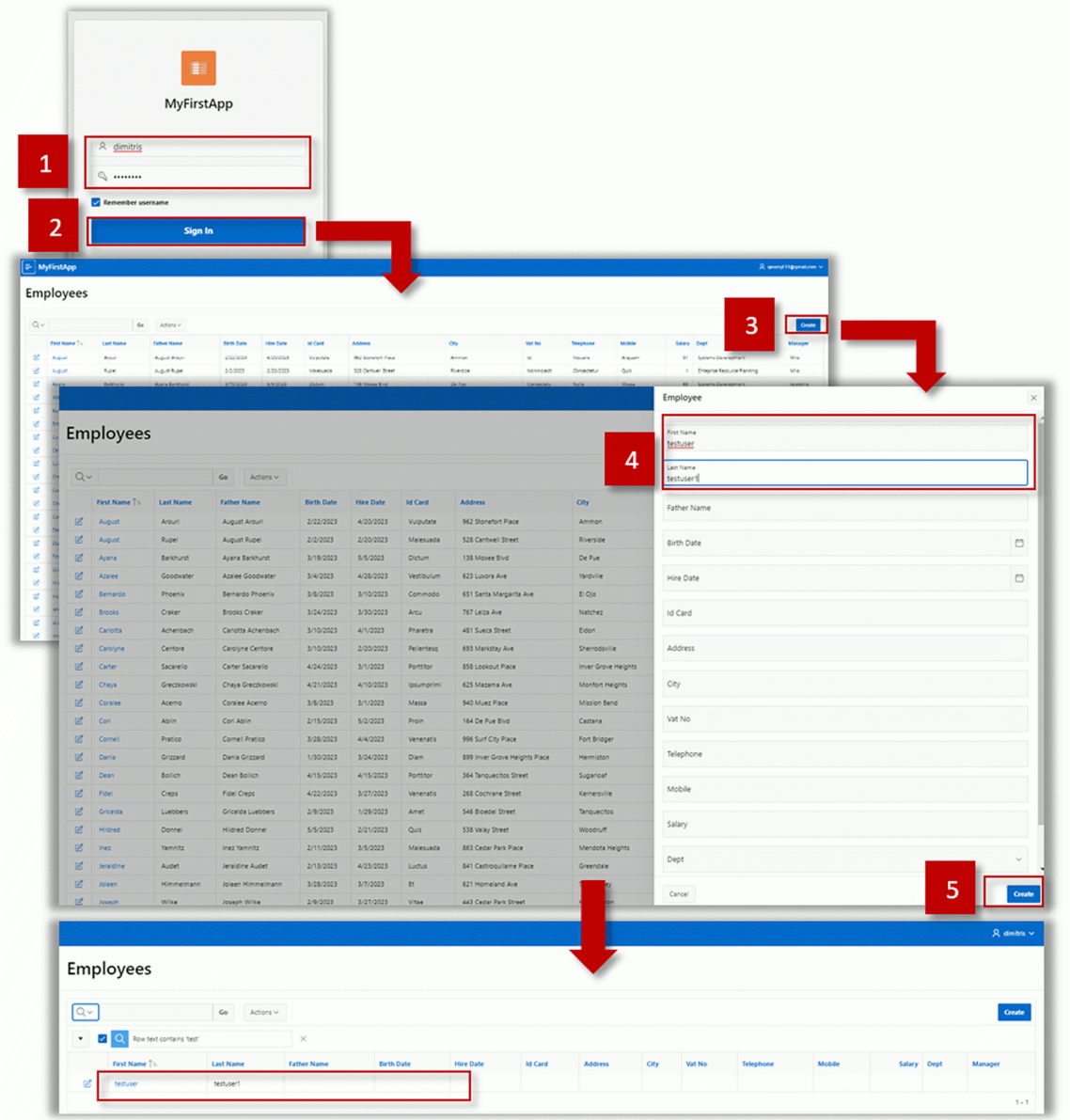
2. korak: Odpremo delovni prostor in izberemo **employees-Form**, kot prikazuje slika 5.22. V levem podoknu za oblikovanje obrazca izberemo **Create** gumb, nato pa v razdelku lastnosti **Security**, izberemo avtorizacijsko temo **Contribution Rights**. Potrdimo s klikom na gumb **Save**. Zdaj bodo lahko samo uporabniki, ki imajo vlogo **Contributor** lahko kreirali in dodajali nove zaposlene.



Slika 5.22: Nastavitve dostopnih pravic do gumba Create.

3. korak: V tem koraku preverjamo konfiguracijo nadzora dostopa za vlogo “Contributor role”.

- Prijavimo se v aplikacijo s pravicami uporabnika Dimitris, ki ima dodeljeno vlogo Contributor.
- Odpremo meni na levi in se prepričamo, da uporabnik nima možnosti odpreti obrazca za administracijo (opcija Administration).
- Poskusimo kreirati in dodati novega zaposlenega, kot prikazuje slika 5.23.
- Preverimo, ali lahko pregledujemo podatke o obstoječih zaposlenih in spreminjamo posamezne vrednosti.
- Preverimo, ali lahko kreiramo novega zaposlenega.

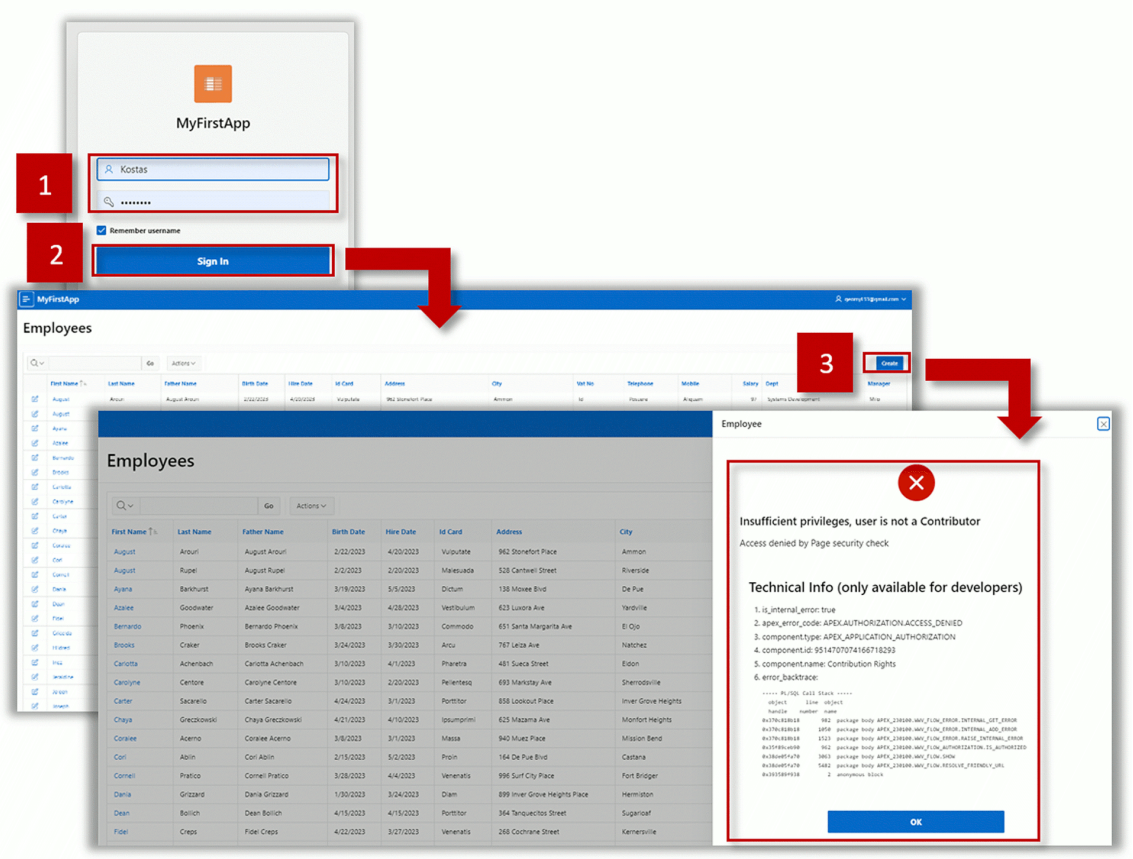


Slika 5.23: Dostop in pravice vloge Contributor.

4. korak: V tem koraku preverjamo konfiguracijo nadzora dostopa za vlogo “Reader role”.

- Prijavimo se v aplikacijo s pravicami uporabnika Kostas, ki ima dodeljeno vlogo Reader.
- Odpremo meni na levi in se prepričamo, da uporabnik nima možnosti odpreti obrazca za administracijo (opcija Administration).
- Na meniju izberemo in odpremo odpremo Employees-Form.
- Poskušamo dodati novega zaposlenega. Prejeli bomo sporočilo “access denied”, kot prikazuje slika 5.24.
- Preverimo, ali lahko pregledujemo podatke o obstoječih zaposlenih, pri čemer lahko opazimo, da ne moremo spreminjati podatkov.

S temi koraki smo določili pravice dostopa in preverili, ali nadzor dostopa deluje.



Slika 5.24: Dostop in pravice vloge Reader.

5.7 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija
- video vodiči

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter20 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

5.7.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

5.7.2 Video vodiči

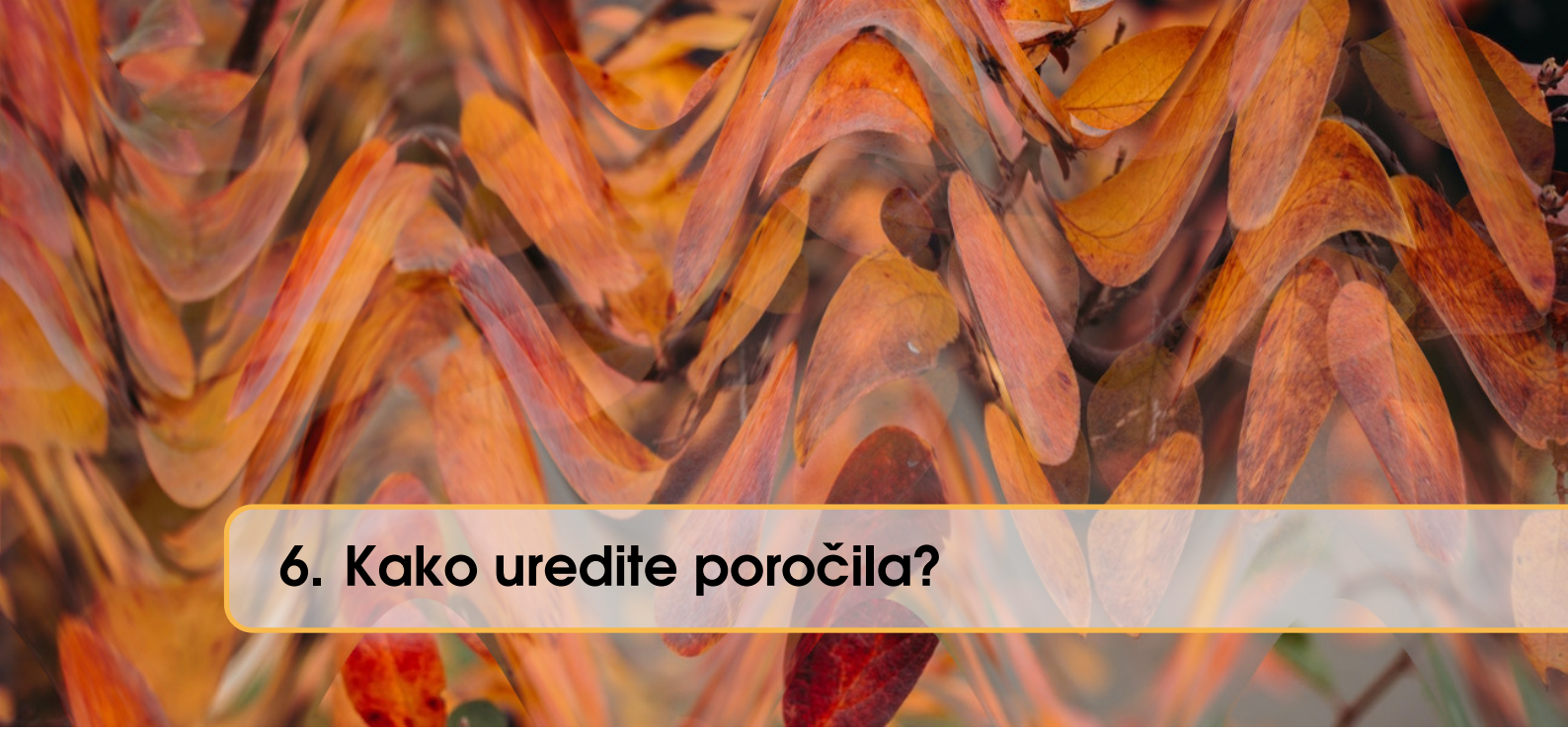
Video vodiči natančno prikazujejo razvoj aplikacije.

5.8 Vprašanja

1. Na koliko različnih načinov lahko v Oracle APEX Application Builder izdelamo aplikacijo?
2. Kako lahko aplikaciji dodajamo uporabnike?
3. Koliko vgrajenih uporabniških privilegijev ponuja Oracle APEX za nadzor dostopa do aplikacije ali njenih komponent?

5.9 Odgovori

1. Obstajajo 3 načini
   - New Application
   - From a File
   - Starter app
2. V delovnem okolju aplikacije izberemo Manage Users and Group
3. Na voljo so 3 različne vloge
   - Administrator
   - Contributor
   - Reader



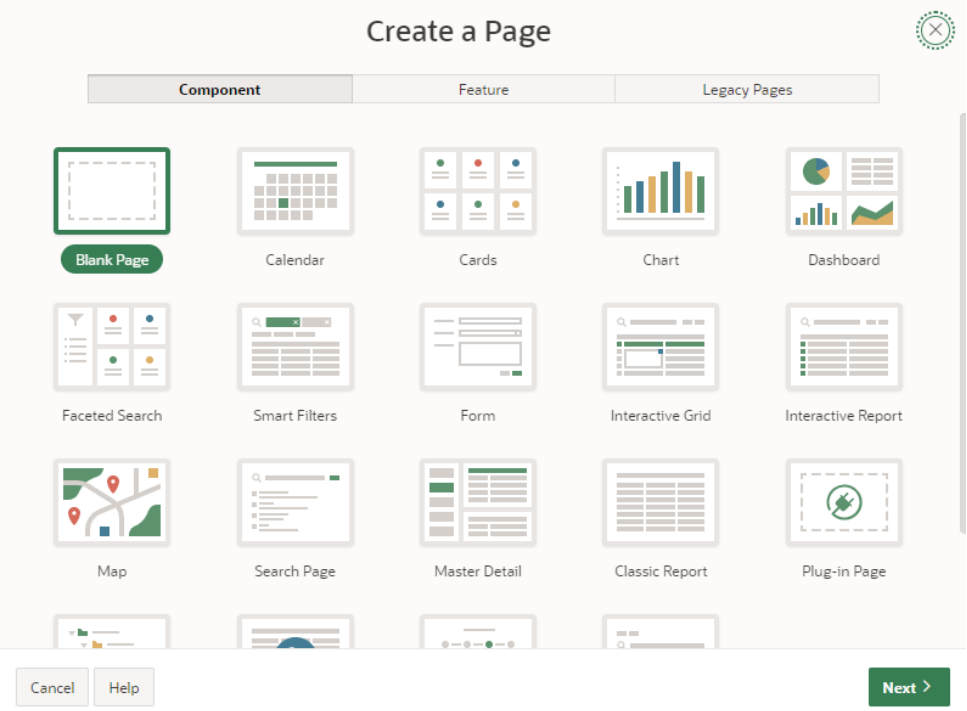
6. Kako uredite poročila?

ZUZANA ŽILLOVÁ, ERIK MALINA, MATEJ GROCHAL, ANDREJ STANÍK, ANDREA MELEKOVÁ, MICHAL KVET, MIROSLAV POTOČÁR; PREVOD A. BAGGIA

Podaki, ki jih zbiramo v bazah podatkov, so pogosto brez pomena, če jih ne predstavimo na pravi način oziroma tako, da iz njih pridobimo uporabne informacije. Za predstavitev, pregled in iskanje po podatkih uporabljamo poročila, ki jih podrobneje predstavljamo v tem poglavju. Spoznali boste, kako enostavno kreiramo poročilo v APEX okolju in kako poročilo urejamo. Predstavili bomo različne tipe poročil, kot je na primer interaktivno poročilo, klasično poročilo, poročilo s preklopom stolpcev ali poročilo v obliki grafične predstavitve.

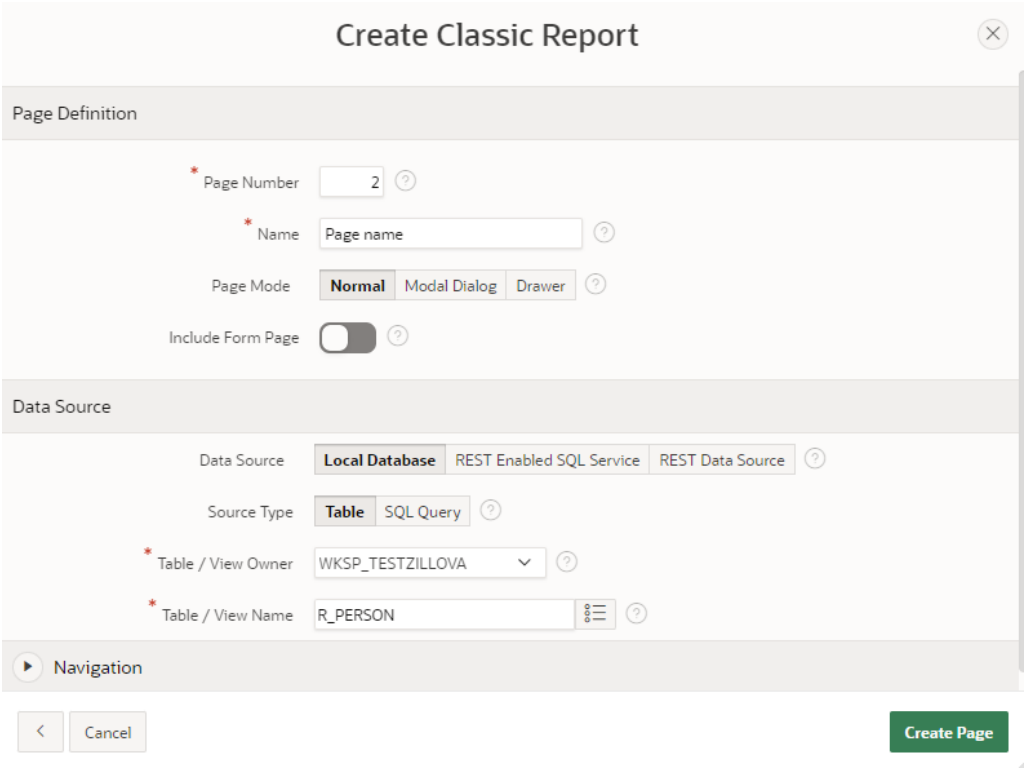
6.1 Poročilo

Poročilo (angl. Report) je *regija* oziroma del spletne strani, na katerem so podatki prikazani na zaslonu v vrsticah. Poročilo je lahko ena enota, ali pa je predstavljeno na več straneh, pri čemer so podatki razdeljeni v logične enote. Poročilo je lahko statično, ima prikazan samo naslov in je oblikovan na uporabniku prijazen način, nima pa dodanih funkcionalnosti. Druga vrsta poročil pa so dinamična poročila, ki imajo specifične funkcionalnosti kot na primer *koledar, poročilo* ali *seznam*. Tovrstne regije imajo vnaprej programirane nekatere osnovne funkcionalnosti, ki pa jih uporabniki lahko tudi prilagodijo.



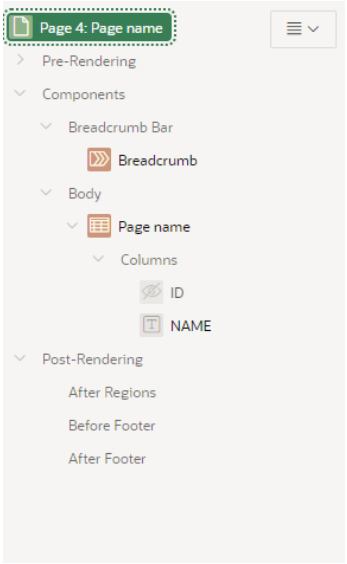
Slika 6.1: Izbira vrste strani.

Kot je prikazano na sliki 6.1, se lahko uporabniki pri kreiranju strani odločajo med različnimi vnaprej definiranimi tipi strani, ki že vsebujejo poročila, torej ni nujno na na stran poročilo vstavimo kot regijo. Na sliki 6.2 je prikazano pogovorno okno čarovnika za kreiranje klasičnega poročila. Pri kreiranju strani, ki vsebuje poročilo, moramo obvezno vpisati ime strani in vir podatkov, ki jih bo stran prikazovala. Podatki se lahko naložijo iz *lokalne baze podatkov, SQL poizvedbe z REST* ali *REST* vira. Kadar izberemo lokalno bazo podatkov, je nujno, da tudi natančno opredelimo, kaj bo vir podatkov. Pri tem lahko izbiramo med tabelami v bazi podatkov, vpogledi ali pa lastnimi SQL poizvedbami.



Slika 6.2: Kreiranje strani s poročilom.

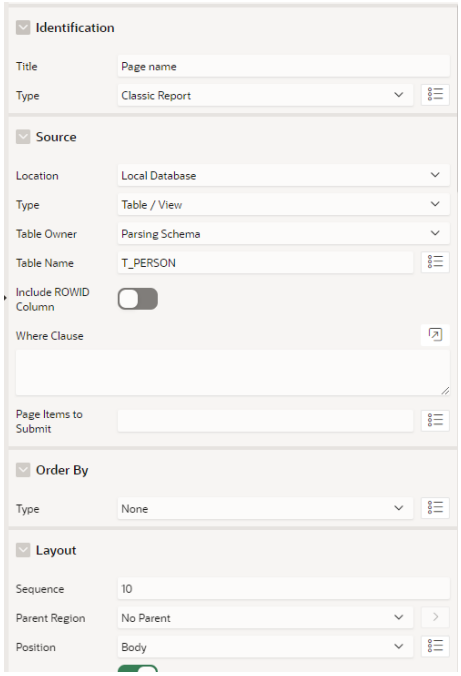
Ko je stran pripravljena, se na levi strani pojavi razdelek za urejanje elementov na spletni strani, kot prikazuje slika 6.3. Na vrhu je zapisano ime strani, spodaj pa so v drevesni strukturi predstavljeni vsi elementi strani. S klikom označimo posamezen enement, odprejo se tudi podrejeni elementi, ki so lahko podrejene regije ali pa posamezni elementi. Z desnim klikom na element lahko kreiramo nov element, na primer regijo.



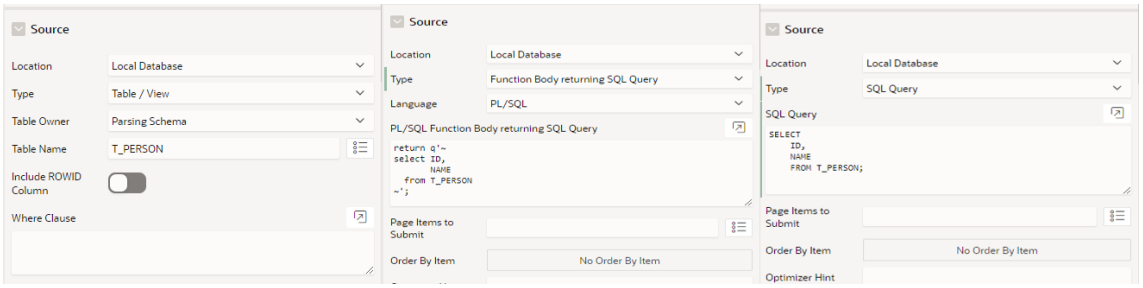
Slika 6.3: Prikazovanje.

Na sliki 6.4 so predstavljene lastnosti regije, ki jih sicer vidimo na desni strani zaslona, ko izberemo posamezen element spletne strani. V razdelku **Identification** je določen naziv strani

in vrsta poročila (npr. klasično poročilo). Slika 6.5 prikazuje različne primere razdelka o virih poročila, pa tudi podatke o izvoru podatkov (kot smo že omenili: *tabela, vpogled, funkcija* ali *SQL poizvedba*). **Where Clause** določa pogoje, na osnovi katerih se bodo filtrirali podatki.



Slika 6.4: Lastnosti regije.



Slika 6.5: Različni viri podatkov.

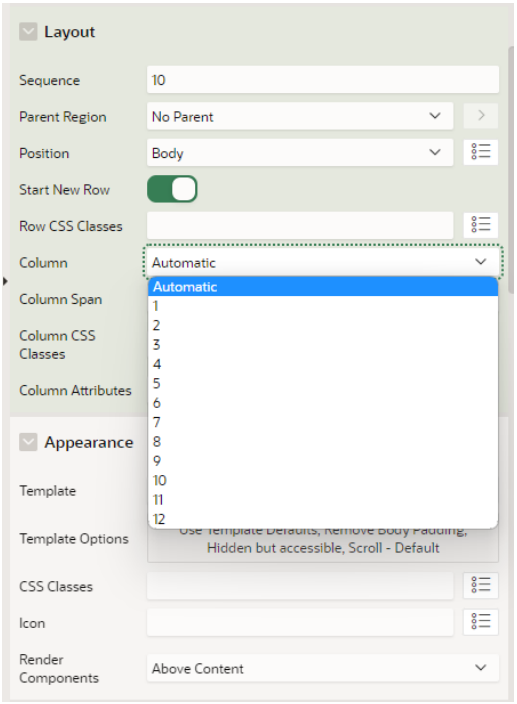
Na sliki 6.6 vidimo razdelek **Layout**, kjer lahko določimo lokacijo regije na strani. **Parent Region** je regija, v katero bo vključeno naše poročilo. To lahko nastavimo tudi preko lastnosti **Position**. Pozicije so vnaprej določene glede na uporabljeno predlogo. Ko stran kreiramo, se zgradi z določeno temo in z izgledom, ki je odvisen od izbrane predloge. Privzeta predloga za temo **Universal** se imenuje **Standard**.

Izgled regije določa mreža (ang. *grid*) - vsaka regija ima svojo lokacijo, določeno z vrstico in stolpcem. Te regije so organizirane po zaporedni številki (ang. **sequence number**). Prva regija je v prvi vrstici in v prvem stolpcu mreže. Vsaka naslednja regija je nato organizirana po enem od teh vzorcev: v isti vrstici in stolpcu kot prejšnja regija in je prikazana pod prejšnjo regijo. Atributa **Start New Row** in **New Column** se uporabljata za določanje načina organizacije za te regije.

Size in *location* se izračunata po dodelitvi vrstice in stolpca. To se doseže z izračunom zasedenosti in širine za vsak del vrstice z vrednostmi parametrov **Column** and **Column Span**.

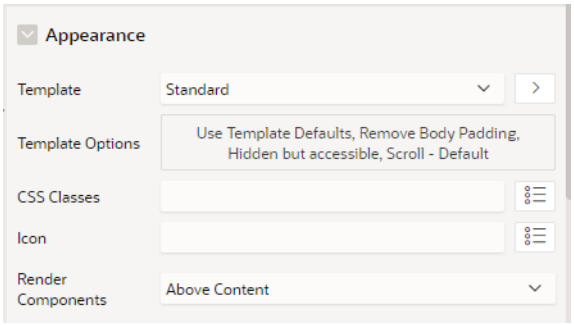
Zamik področja v vrsti je določen z atributom **Column**. Vrednost tega stolpca je v območju **1 - 12** in se nanaša na **1** od **12 točk** mreže, ki delijo širino. Vrednost **1** pomeni najbolj *levi del*, **7** pa

sredino strani.

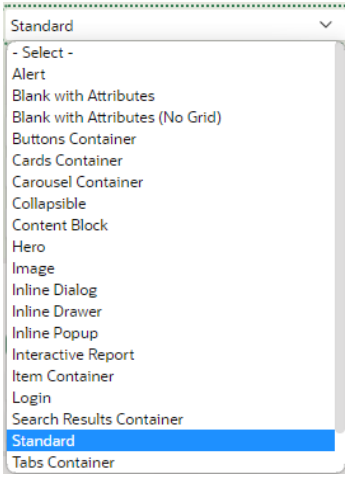


Slika 6.6: Izgled (ang. Layout).

Slika 6.7 prikazuje razdelek **Appearance** v katerem lahko nastavimo lastnosti in urejamo predlogo posamezne regije. Slika 6.8 prikazuje vnaprej določeno predlogo, ki jo uporabnik lahko izbere. Vsako predlogo lahko prilagodimo tako da izberemo drugačne opcije predloge (angl. Template Options) ali pa tako, da zapišemo svoje **CSS** ukaze. Pri opcijah predloge lahko določimo različna pravila, kot na primer višina besedila, višina glave strani, ali pa celo razmik med dvema elementoma.



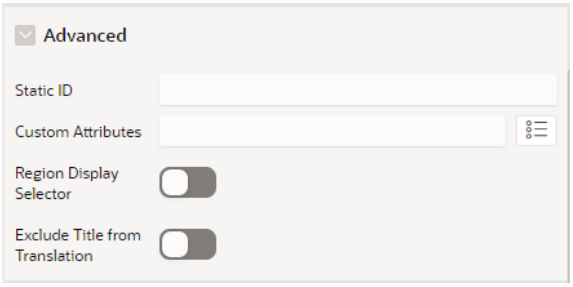
Slika 6.7: Razdelek Appearance.



Slika 6.8: Razdelek Templates.

Slika 6.9 prikazuje razdelek **Advanced**, v katerem lahko določimo štiri lastnosti. **Static ID** določa enolični označevalec poročila. Atribut **Static ID** se uporabi kot identifikacija **HTML** elementa na strani kadar želi uporabnik nastaviti izgled strani s pomočjo **CSS**. V **Custom Attributes** lahko določimo še druge lastnosti **HTML** elementa. Te lastnosti se shranjujejo v obliki ime/vrednost, na primer: *name* = "*MY_REPORT*"

Region Display Selector je del regije, na katerem najdemo navigacijske gumbe za prehod na druge regije znotraj iste strani. **Exclude Title from Translation** pa omogoča, da v primeru, ko ima stran definiran tudi prevod, ne upošteva naslova strani.



Slika 6.9: Razdelek Advanced.

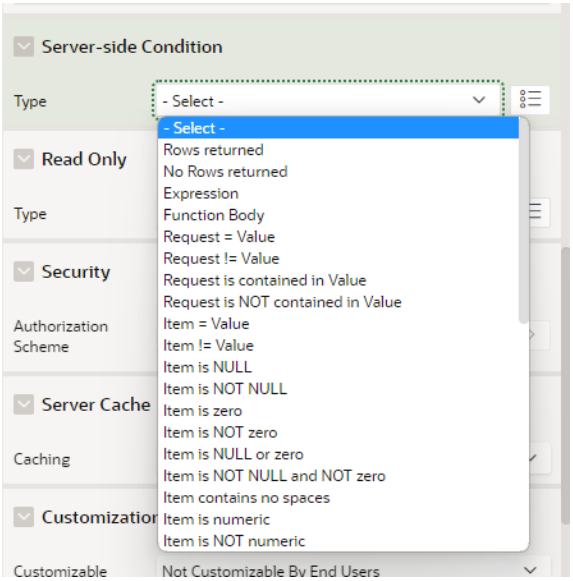
Header and Footer razdelek, ki je prikazan na sliki 6.10 uporabimo za urejanje vsebine glave in noge poročila. Pomembno je vedeti, da gre za glavo in nogo poročila in ne cele strani.



Slika 6.10: Razdelek Header and Footer.

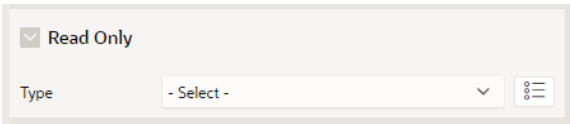
Na sliki 6.11 so prikazane različne oblike pogojev strežnika **angl. Server-side Conditions** med katerimi lahko izbiramo. Pri aplikacijah, ki jih razvijemo v okolju Oracle APEX, se **Server-side**

Conditions upoštevajo v času postavitve strani. Poročilo pa se ne prikaže, v kolikor niso izpolnjeni navedeni pogoji. Določimo lahko veliko različnih pogojev. Primerjamo lahko vrednosti elementov ali pa preverjamo pogoje za zahteve.



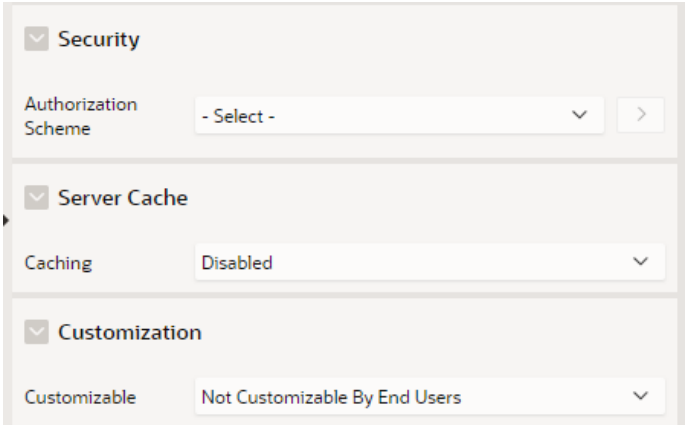
Slika 6.11: Razdelek Server-side Condition.

Read Only lastnost, ki jo prikazuje slika 6.12 ima enake možnosti izbire kot **Server-side Condition**, ki ga prikazuje slika 6.11. Vpogleda na določeno regijo ne moremo spreminjati medtem ko je nastavljen kot read-only. V primeru osnovnega poročila pa ne moremo urejati vrstnega reda podatkov izpisanih v stolpcih.



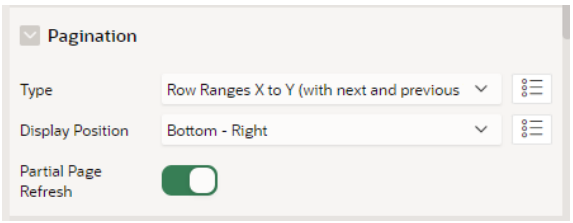
Slika 6.12: Določanje lastnosti Read Only.

Na sliki 6.13 so predstavljeni zadnji trije razdelki **Security, Server Cache**, in **Customization**. V razdelku **Security** nastavljamo možnosti omejitev prikaza spletne strani - npr. *prosto dostopna spletna stran, spletna stran za avtorizirane uporabnike*. V razdelku **Server Cache** pa določamo nastavitve upravljanja predpomnilnika. V razdelku **Customization** lahko opredelimo dostop do sprememb poročila.



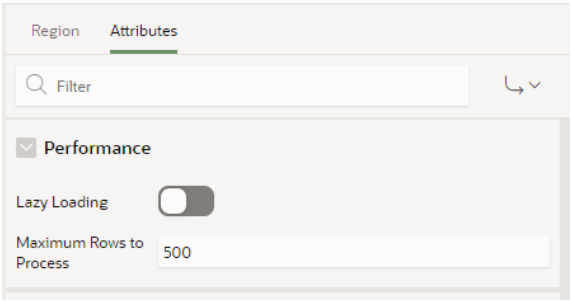
Slika 6.13: Razdelek Security, Server Cache and Customization.

V drugem zaviku nastavljamo lastnosti, **angl. Attributes**. Tukaj lahko nastavimo lastnosti kot so *število prikazanih vrstic*, možnost *priprave seznama iz rezultatov poročila* in *na kakšen način se bomo premikali* po poročilu.

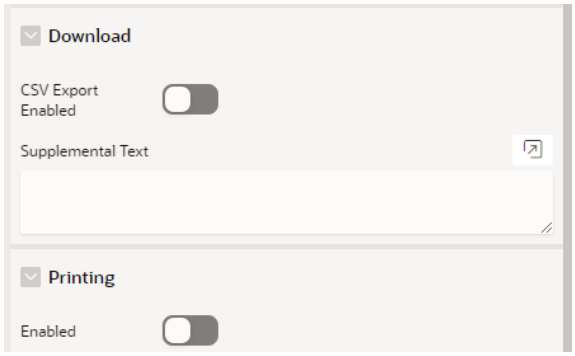


Slika 6.14: Razdelek Pagination.

Zanimiva lastnost je tudi nastavitev minimalnega števila prikazanih vrstic. Določimo lahko tudi kaj naj se izpiše, kadar poizvedba ne najde nobenih rezultatov angl. no data found. Poročilo lahko prenesemo kot **.csv** datoteko ali ga natisnemo.

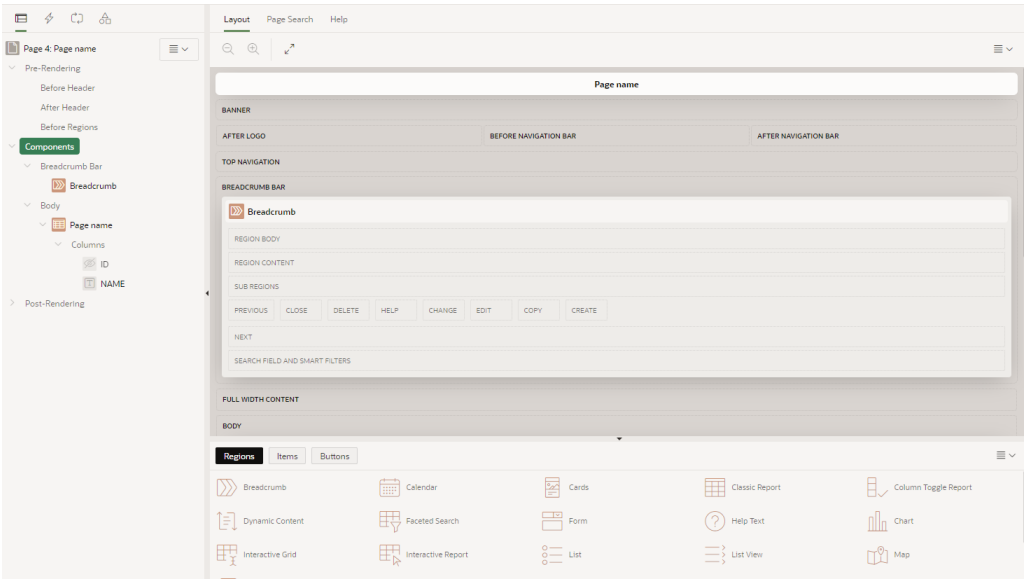


Slika 6.15: Določanje števila naloženih prikazanih vrstic.



Slika 6.16: Nastavitev možnosti prenosa in tiskanja.

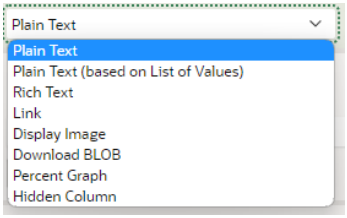
Zelo pomemben je tudi centralni del zaslona za urejanje strani. Na sliki 6.17 vidimo možnost prikaza (angl. **Layout**), na katerem vidimo kako so komponente dejansko postavljene na stran. Celotna stran je strukturirana kot neke vrste mreža, da si lahko lažje predstavljamo kam dejansko bomo komponente postavili. Posamezen element ali regijo lahko premikamo po strani tako, da jih enostavno premaknemo s pomočjo miške. Za vsako regijo obstaja vnaprej določena predloga. Vse nastavitve so že privzeto nastavljene, ko regijo premaknemo.



Slika 6.17: Osrednji del okna za oblikovanje strani.

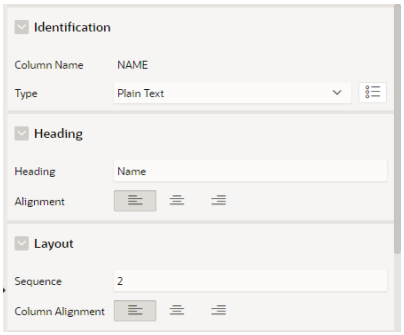
Na levi strani slike 6.17 lahko vidimo, da APEX določene komponente privzeto vstavi na poročilo. Vsaka komponenta predstavlja en stolpec, ki ga lahko spreminjamo podobno kot smo spreminjali celotno poročilo.

Slika 6.19 prikazuje lastnosti stolpca. V razdelku **Identification** nastavimo dve lastnosti - **Type** in **Name of Column**. Spreminjanje imena stolpca (angl. **Name of Column**) je onemogočeno, ker to vrednost pridobimo iz *SQL poizvedbe*. Vrsta stolpca (angl. **Type of Column**), ki je prikazana na sliki 6.18, določa način prikaza stolpca. Če naštejemo samo nekaj teh možnosti: *Hidden* (stolpec se ne prikaže), *Plain Text* (običajno besedilo), *Display Image* (prikaže sliko), *Percent Graph* (prikaže podatke v obliki grafa) itd. Privzeto so vsi stolpci v obliki *Plain Text*.



Slika 6.18: Različne oblike stolpcev.

Razdelek **Heading** omogoča spreminjanje dveh lasnosti glave strani. Za razliko od imena stolpca, ime glave strani lahko spremenimo. Glava strani lahko vsebuje tudi **HTML** kodo. Možnost **Alignment** določa način poravnave glave strani. V skupini **Layout** nastavimo zaporedno številko (angl. **Sequence**), ki določa vrstni red prikaza stolpcev kot tudi možnost poravnave.



Slika 6.19: Nastavitve stolpca.

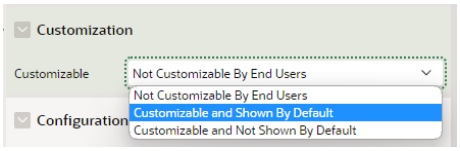
Zadnji del so lastnosti vezane na izgled (angl. **Appearance**), ki ga določamo, kadar želimo na primer številke prikazovati v kakšni posebni obliki. Kot prikazuje slika 6.20, s klikom na **Format Mask** lahko izbiramo med veliko vnaprej določenimi oblikami. Poleg oblik, ki jih predlaga APEX pa lahko dodajamo tudi svoje.



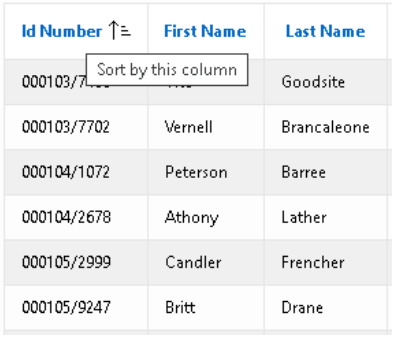
Slika 6.20: Lastnot prikaza Format Mask.

6.2 Klasično poročilo

Klasično poročilo (angl. **Classic Report**) je vrsta poročila, ki omogoča osnoven prikaz podatkov, brez dodatnih možnosti spreminjanja prikaza, ki bi jih določal končni uporabnik. Nastavimo lahko samo to, da uporabnik lahko razvršča podatke po posameznih stolpcih.

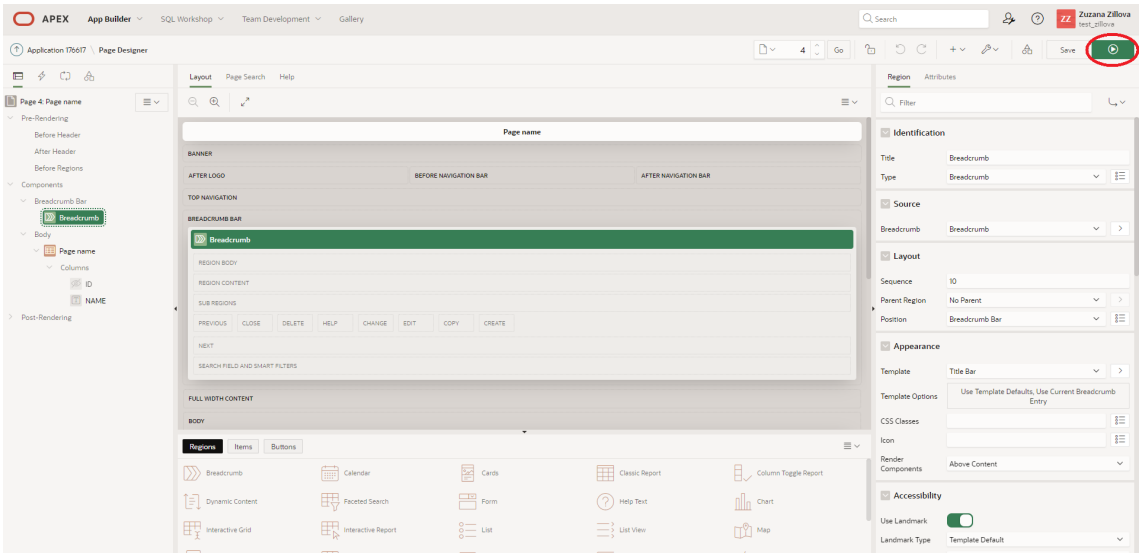


Slika 6.21: Prilagoditev možnosti prikaza končnemu uporabniku.

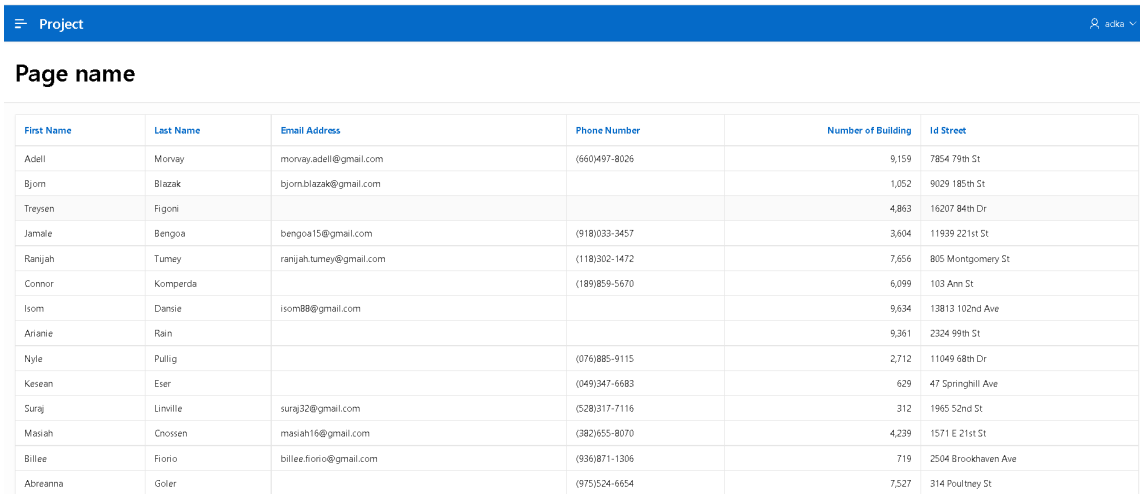


Slika 6.22: Možnost urejanja stolpca.

Za predogled generirane strani kliknemo na gumb za zagon, na zeleno puščico v krogu. Kot prikazuje slika 6.23, je gumb v zgornjem desnem kotu. Na sliki 6.24 je prikazan primer klasičnega poročila. Ima stolpce z imeni in podatke, razen razvrščanja končni uporabnik nima možnosti nastavljanja izgleda poročila.



Slika 6.23: Gumb za prikaz spletne strani.

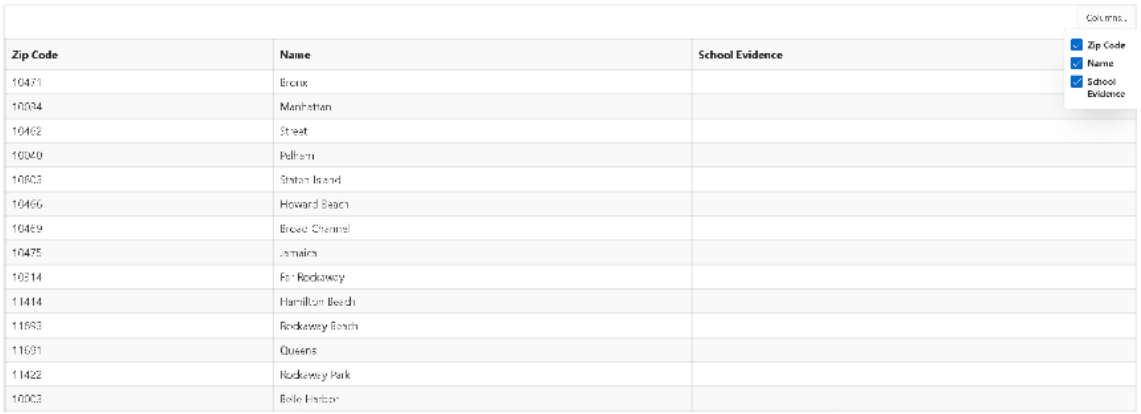


First Name	Last Name	Email Address	Phone Number	Number of Building	Id Street
Adell	Morvay	morvay.adell@gmail.com	(660)497-8026	9,159	7854 79th St
Bjorn	Blazak	bjorn.blazak@gmail.com		1,052	9029 185th St
Treysen	Figoni			4,863	16207 84th Dr
Jamale	Bengoa	bengoa15@gmail.com	(918)033-3457	3,604	11939 221st St
Ranijah	Tumey	ranijah.tumey@gmail.com	(118)302-1472	7,656	805 Montgomery St
Connor	Komperda		(189)859-5670	6,099	103 Ann St
Isom	Dansie	isom88@gmail.com		9,634	13813 102nd Ave
Arianie	Rain			9,361	2324 99th St
Nyle	Pullig		(076)885-9115	2,712	11049 68th Dr
Kesean	Eser		(049)347-6683	629	47 Springhill Ave
Suraj	Linville	suraj32@gmail.com	(528)317-7116	312	1965 52nd St
Masiah	Cnossen	masiah16@gmail.com	(382)655-8070	4,239	1571 E 21st St
Billee	Fiorio	billee.fiorio@gmail.com	(936)871-1306	719	2504 Brookhaven Ave
Abreanna	Goler		(975)524-6654	7,527	314 Poultney St

Slika 6.24: Primer klasičnega poročila.

6.3 Poročilo s preklopom stolpcev

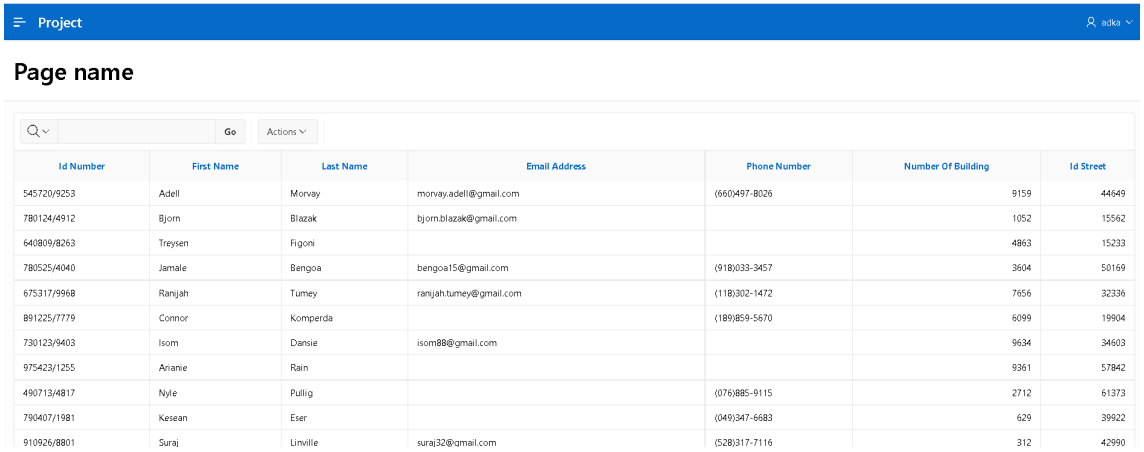
Poročilo s preklopom stolpcev (angl. **Column Toggle Report**) je zelo podobno klasičnemu poročilu, le da lahko pri tem poročilu določite kateri stolpci so prikazani in kateri ne. Možnost nastavljanja vidimo v zgornjem desnem kotu na sliki 6.25.



Slika 6.25: Možnost spreminjanja vidnosti stolpca.

6.4 Interaktivno poročilo

Interaktivno poročilo (angl. **Interactive Report**) je prikazano na sliki 6.26. To poročilo uporabnikom omogoča določene prilagoditve poročila medtem ko pregledujejo podatke, brez da bi morali pri tem dostopati do ogrodja aplikacije. Interaktivno poročilo pripravimo podobno kot ostala poročila in se na prvi pogled ne razlikuje veliko od klasičnega poročila. Če pogledamo posamezne funkcionalnosti in možnosti prilagoditve, pa lahko vidimo, da je razlik kar nekaj.

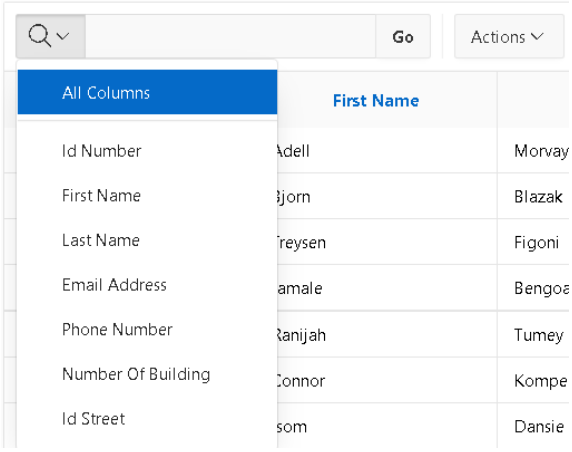


Slika 6.26: Interaktivno poročilo.

Iskalnik, ki je prikazan na sliki 6.27 sestavljajo štirje elementi:

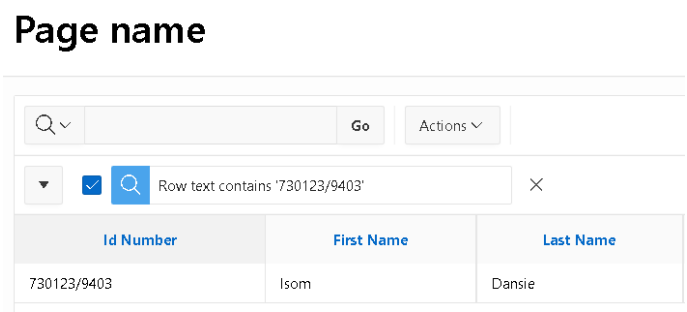
- Povečevalno steklo (angl. **Magnifying Glass,**)
- Besedilno polje (angl. **Text Box,**)
- Gumb za zagon (angl. **Go Button,**)
- Gumb Actions (angl. **Actions button**).

Prvi trije elementi uporabniku omogočajo filtriranje prikazanih podatkov. V besedilno polje vpišemo iskano vrednost, izberemo povečevalno steklo, kjer določimo kateri stolpci naj se prikazujejo, nato pa potrdimo izbiro s klikom na gumb **Go**. Prikazali se bodo samo zapisi, ki vsebujejo iskani niz.



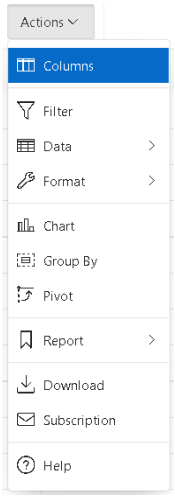
Slika 6.27: Iskalna plošča.

Primer iskanja, ko želimo poiskati zapise, ki vsebujejo številko **"730123/9403"** v kateremkoli stolpcu, je prikazan na sliki 6.28. Filter lahko ugasnemo s klikom na gumb **X**. Istočasno imamo lahko aktivnih več filtrov na različnih stolpcih.



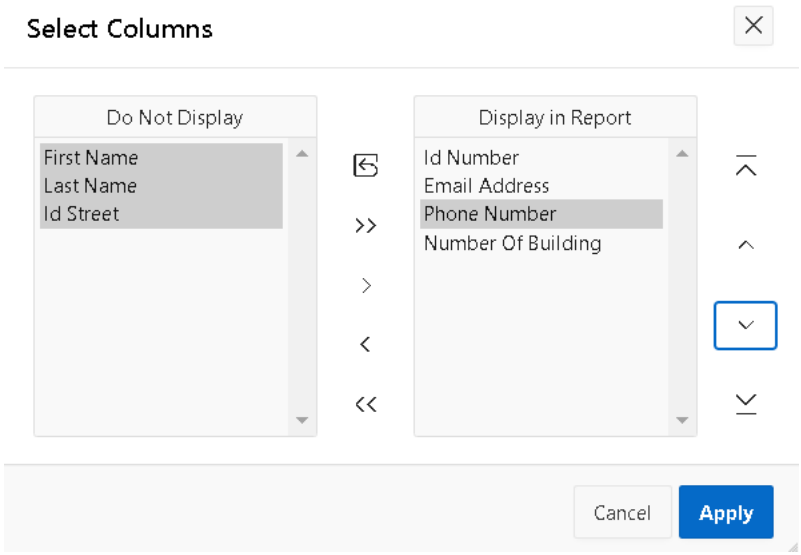
Slika 6.28: Iskanje.

Gumb **Actions** omogoča dodatne prilagoditve prikazanega poročila. Ko kliknemo na gumb, se odpre spustni seznam z vrsto možnosti, kot prikazuje slika 6.29.



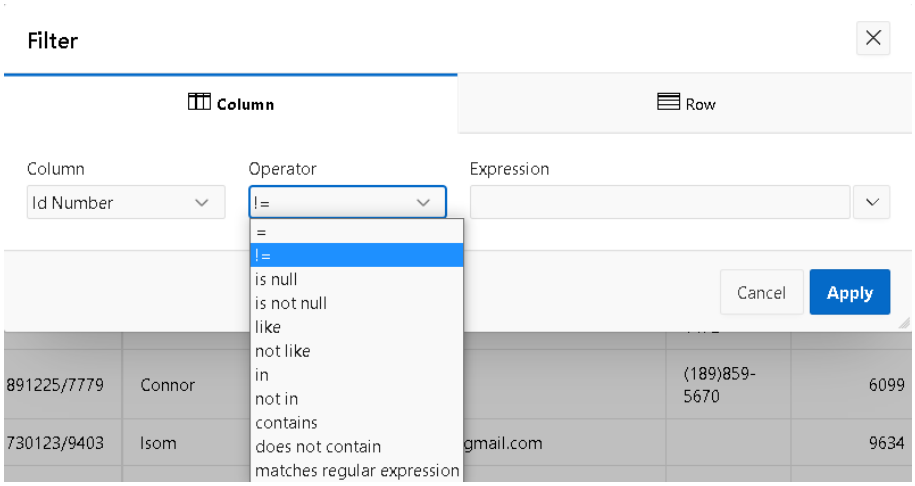
Slika 6.29: Gumb Actions.

Izbira **Columns** odpre pogovorno okno, ki je prikazano na sliki 6.30. V tem primeru lahko uporabnik izbere, kateri stolpci se bodo prikazali v poročilu, ter vrstni red stolpcev. Stolpce lahko premikamo med sekcijama **Do not Display** in **Display in Report** ter tako določamo, kaj se bo v poročilu prikazovalo. Za premikanje lahko uporabimo tudi puščice, ki omogočajo premik enega ali več elementov.



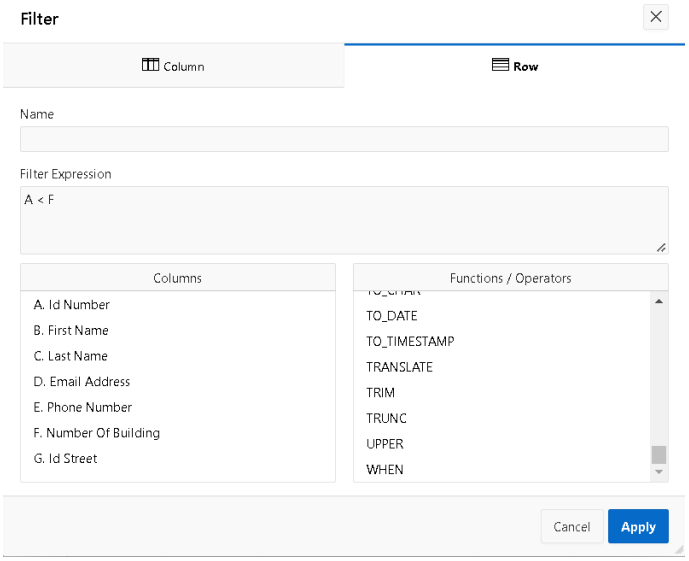
Slika 6.30: Izbira stolpcev za prikaz.

Druga možnost je uporaba **Filtra** kot prikazuje slika 6.31. Za filtriranje vrednosti lahko vpišemo izraze, kot na primer *like, not equal, is null*, in podobno. Izbiramo lahko med vnaprej določenimi izrazi, ali pa vpišemo svoje.



Slika 6.31: Filtriranje po stolpcih.

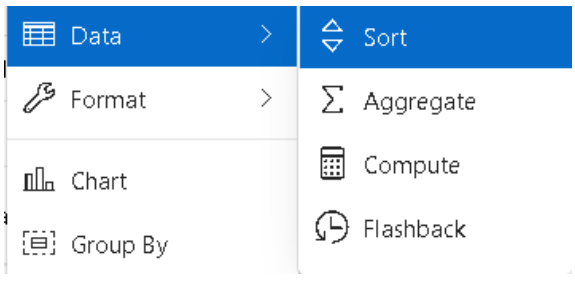
V primeru filtriranja po vrsticah, imamo na voljo številne operacije in izraze, ki jih lahko sestavljamo sami. Ta filter je možno nastaviti v oknu **Filter Expression**, ki ga prikazuje slika 6.32.



Slika 6.32: Filtriranje po vrsticah.

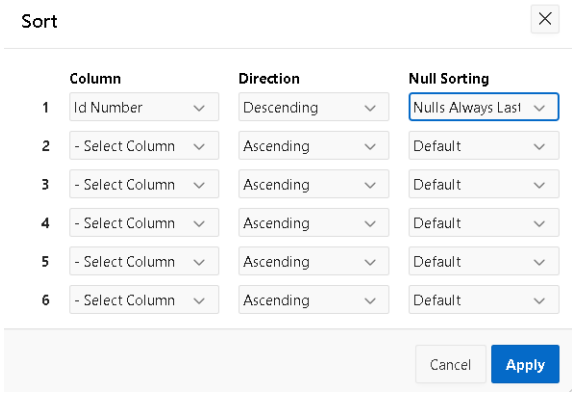
Z izbiro možnosti **Data** odpremo meni, ki ga prikazuje slika 6.33. V meniju lahko izbiramo med štirimi možnostmi:

- sortiranjem (angl. sort),
- združevanjem (angl. aggregate),
- preračunavanjem (angl. compute), ter
- ter vrnitvijo na začetno stanje (angl. flashback).



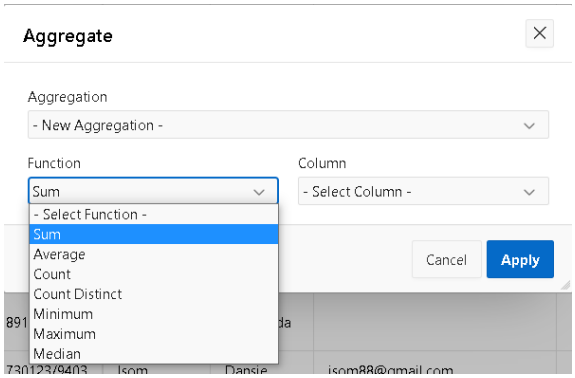
Slika 6.33: Možnosti dela s podatki.

Na sliki 6.34 je prikazana možnost **Sort**, ki jo lahko uporabimo na kateremkoli stolpcu, uporabnik pa lahko določi tudi kako naj se prikazujejo prazne vrednosti (angl. *null values*).



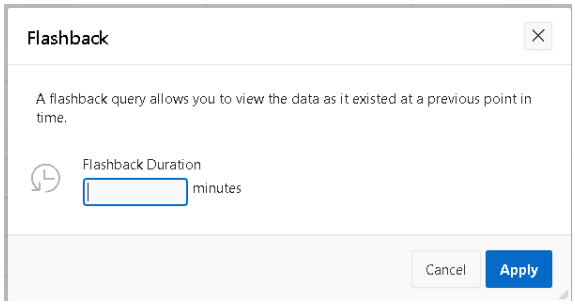
Slika 6.34: Sortiranje podatkov.

Z uporabo funkcije **Aggregate** lahko na različne načine združimo podatke v enem stolpcu. Agregatne funkcije, ki jih lahko uporabimo, prikazuje slika 6.35.



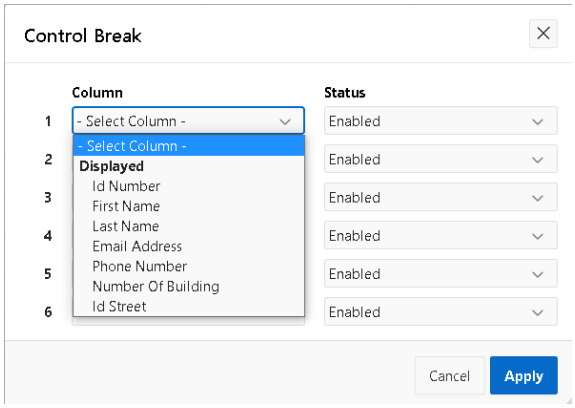
Slika 6.35: Funkcija Aggregate.

Funkcija **Flashback**, ki je prikazana na sliki 6.36 omogoča, da se vrnemo na poročilo v obliki, ki smo jo prikazali pred zadnjimi spremembami.



Slika 6.36: Funkcija Flashback.

Možnost **Control Break**, ki je prikazana na sliki 6.37 omogoča, da izpisane podatke na poročilu razdelimo v več manjših delov, kot bi prikazovali ločena poročila. Deli poročila so ločeni glede na vrednosti v stolpcih. Podatki v vsakem delu imajo enako vrednost v izbranem stolpcu (npr. vsi zaposleni v enem oddelku).



Slika 6.37: Funkcija Control Break.

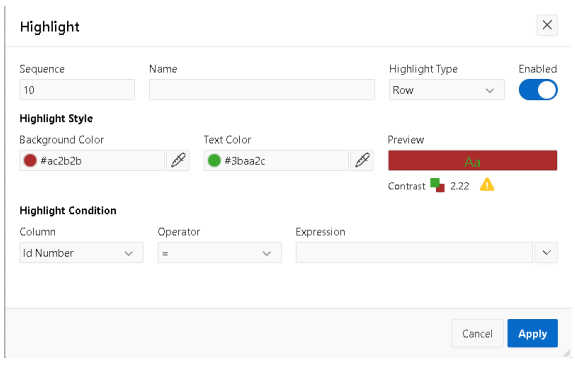
Na sliki 6.38 je prikazana uporaba Control Break funkcije na stolpcu *First Name*.

First Name: Aaban

Id Number	Last Name	Email Address	Phone Number	Number Of Building	Id Street
650724/0703	Holmstedt	aaban4@gmail.com	(435)936-9323	4881	61479
640517/1114	Sartor	aaban.sartor@gmail.com		8368	6494
921024/6657	Kovari	aaban.kovari@gmail.com	(268)217-2739	1935	7728
740410/0666	Dantico	aaban.dantico@gmail.com		3115	40604
870703/0560	Labreck	aaban.labreck@gmail.com	(888)203-2292	2639	6159

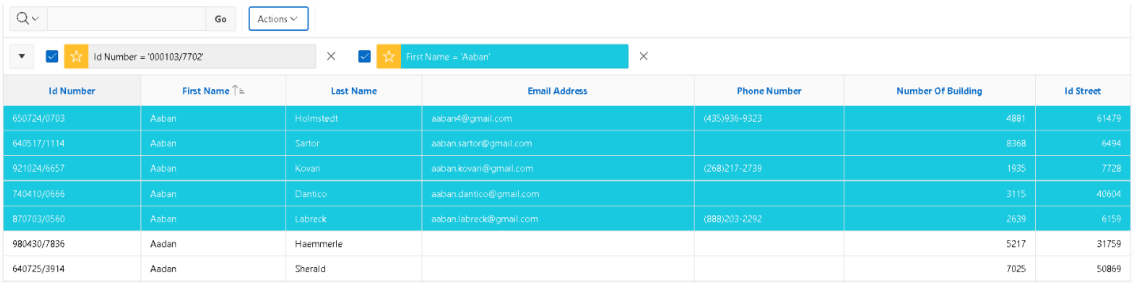
Slika 6.38: Rezultat Control Break funkcije.

Na sliki 6.39 je prikazana uporaba funkcije **Highlight**, s katero lahko označimo določene skupine podatkov. Z izbranimi barvami lahko označimo celico ali celotno vrstico. Primerjava vrednosti v stolpcih deluje na enak način kot **Filter**, le da imamo v tem primeru še vedno prikazane tudi podatke, ki ne ustrezajo pogojem.



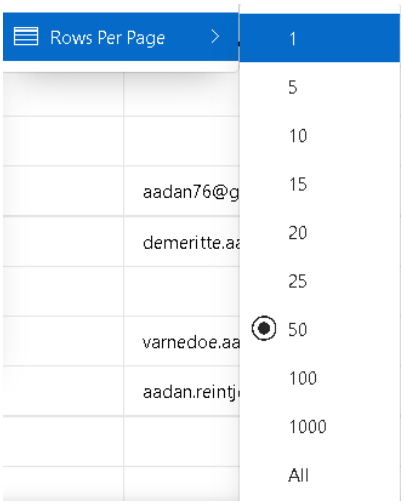
Slika 6.39: Funkcija označevanja.

Na sliki 6.40 je prikazano poročilo z označenimi vrsticami, ki ustrezajo pogoju, da je ime enako **"Aaban"**.



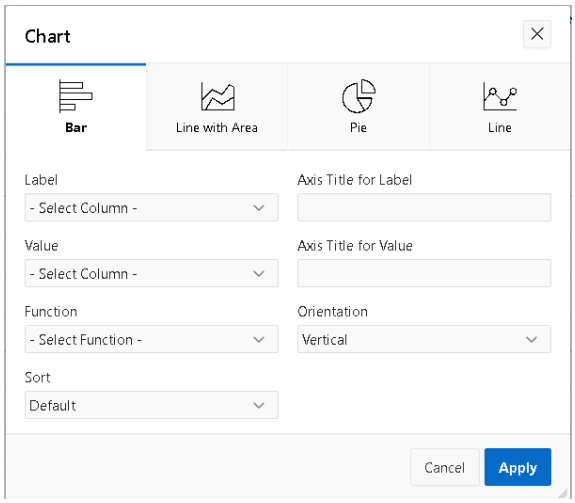
Slika 6.40: Rezultat označevanja.

Slika 6.41 prikazuje omejevanje števila prikazanih vrstic (angl. **Rows Per Page**) na strani. Deluje podobno kot lastnost **Pagination**.



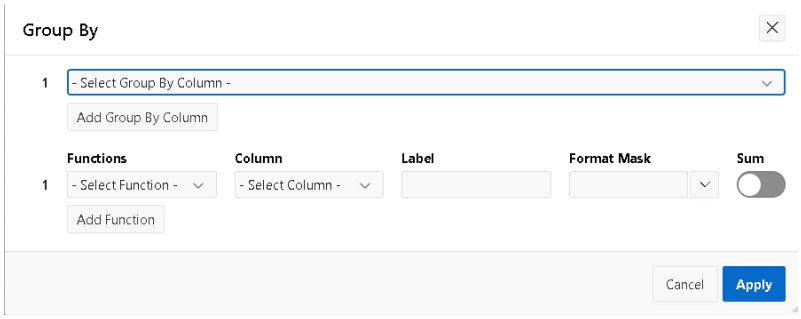
Slika 6.41: Omejevanje števila prikazanih vrstic na strani.

Če nam prikazovanje podatkov v preglednici ne zadostuje, lahko podatke predstavimo tudi v grafični obliki, preko opcije **Chart**, kot prikazuje slika 6.42. Izbiramo lahko med štirimi osnovnimi oblikami grafičnega prikaza. Določimo lahko tudi funkcijo združevanja prikazanih podatkov.



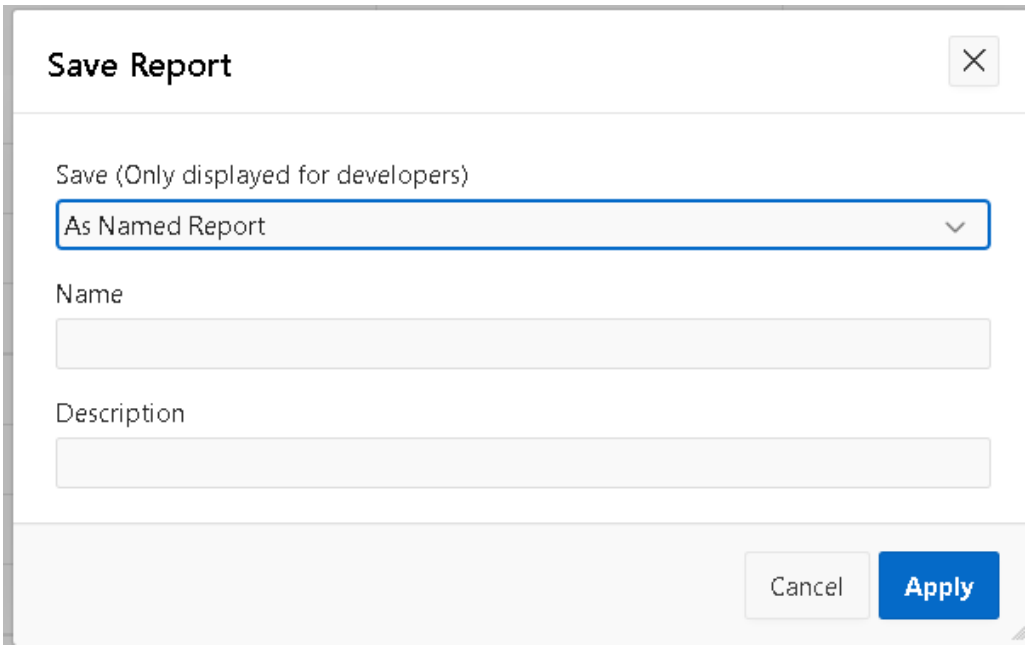
Slika 6.42: Graf v interaktivnem poročilu.

Zelo pomembna funkcionalnost interaktivnega poročila je tudi funkcija **Group By**, ki je prikazana na sliki 6.43. Deluje na enak način kot v SQL poizvedbi.



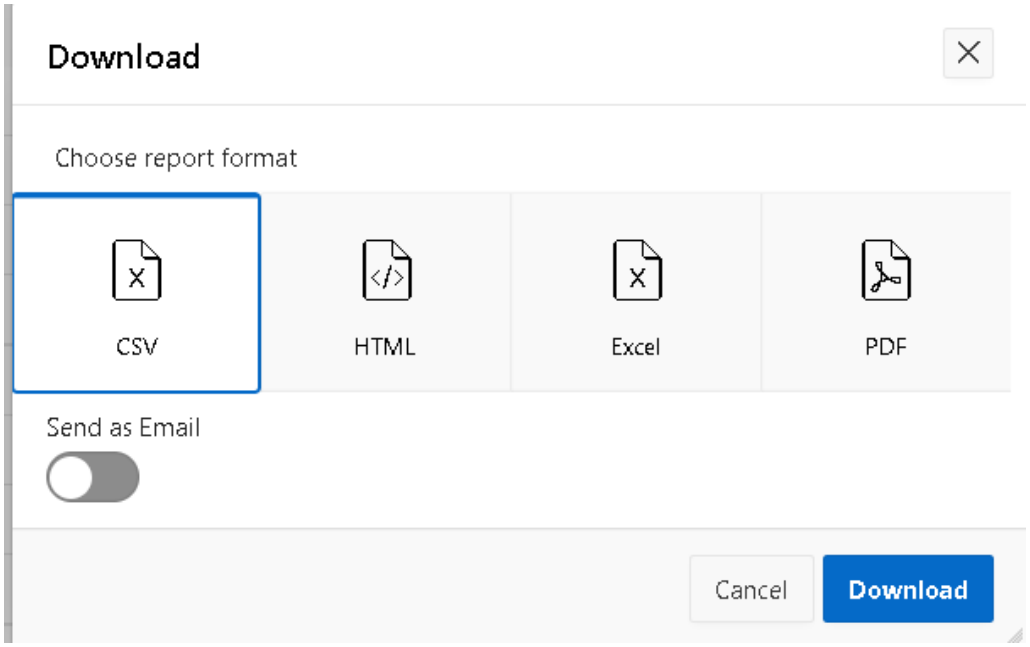
Slika 6.43: Funkcija Group By.

Možnost **Save Report**, ki je prikazana na sliki 6.44 omogoča shranjevanje poročila v trenutni obliki. Vendar pa to shranjeno poročilo lahko vidi samo razvijalec, ko se prijavi v sistem, ne pa tudi končni uporabnik.



Slika 6.44: Shranjevanje poročila.

Poročilo lahko povrnemo v osnovno stanje ali pa trenutno verzijo poročila prenesemo. Ta možnost je prikazana na sliki 6.45.



Slika 6.45: Pogovorno okno za prenos poročila.

6.5 Vprašanja

1. V katero kategorijo spada poročilo? Je to regija, element ali gumb?
2. Kaj lahko uporabimo kot vir podatkov pri poročilu?
3. Katere vrste virov lahko uporabimo pri klasičnem poročilu?
4. Kaj je glavna značilnost klasičnega poročila?
5. Katera funkcionalnost pri interaktivnem poročilu omogoča, da poročilo povrnemo v predhodno stanje?

6.6 Odgovori

1. Poročilo je regija, ki omogoča pregled in iskanje po podatkih.
2. Viri podatkov so lahko lokalna baza podatkov, SQL z omogočenim REST ali REST vir.
3. Uporabimo lahko tabelo ali SQL poizvedbo.
4. Glavna značilnost klasičnega poročila je ta, da podatke lahko samo pregledujemo in ne moremo spreminjati načina prikaza podatkov ali podatkov samih.
5. Funkcija Flashback omogoča, da poročilo povrnemo v stanje, kot je bilo pred določenim časom.

7. Kako uredite obrazce?

VERONIKA ŠALGOVÁ, MICHAL KVET, MIROSLAV POTOČÁR; PREVOD A. BAGGIA

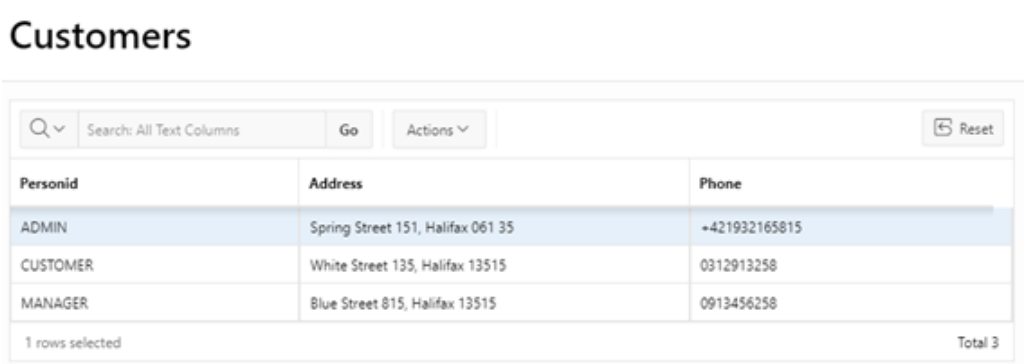
7.1 Vrste obrazcev?

V Oracle APEX lahko uporabljamo tri osnovne tipe obrazcev:

- interaktivno mrežo, ki jo lahko urejamo (angl. interactive grid),
- obrazec na osnovi tabele (angl. form),
- glavni obrazec s podrobnostmi (angl. master detail form).

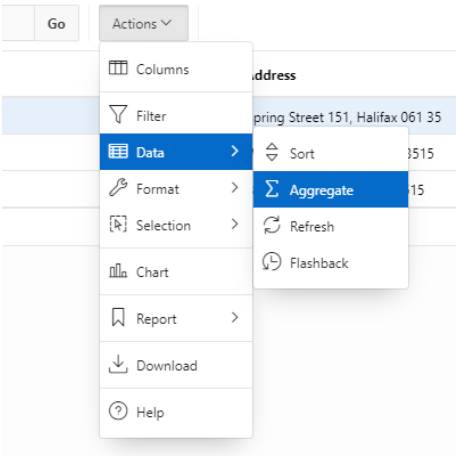
7.2 Interaktivna mreža

V interaktivni mreži, ki je dejansko vrsta poročila, lahko dodajamo, spreminjamo, brišemo podatke ali pa izvajamo iskanje po podatkih. Ta komponenta ima veliko uporabnih lastnosti ter ponuja širok nabor možnosti spreminjanja neposredno na strani. Poleg tega tovrstno poročilo lahko prilagajamo in preurejamo interaktivno, s pomočjo miške, z neposrednim spreminjanjem strukture in vsebine na mreži. Interaktivno mrežo lahko kreiramo tako, da pretvorimo poročilo, ki smo ga izdelali kot interaktivno mrežo, namenjeno samo za branje. To storimo tako, da pri kreiranju strani izberemo *Report* in nato *Interactive Grid*. Druga možnost je, da kreiramo *Report and Form*, nato pa pri lastnosti Report type določimo *Interactive Grid*. S to možnostjo prav tako kreiramo verzijo, ki omogoča samo pregled podatkov, vendar jo kasneje lahko spremenimo v obrazec, ki ga lahko urejamo.



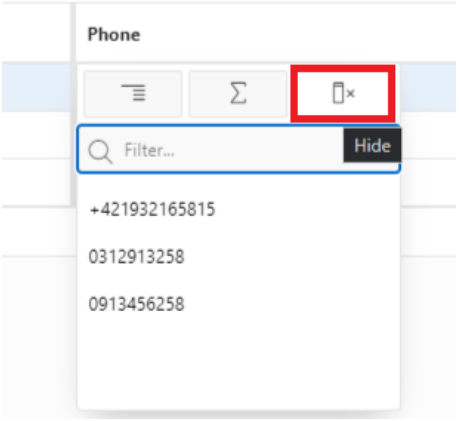
Slika 7.1: Poročilo Interactive Grid s prikazom podatkov o kupcih.

Iskanje po podatkih lahko izvajamo na osnovi besedila, ki ga vpišemo v iskalno polje na vrhu strani. Ostale možnosti, ki jih ponuja interaktivna mreža pa so prikazane na sliki 7.2.



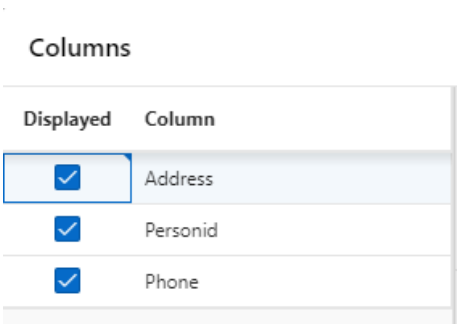
Slika 7.2: Možnosti urejanja prikaza na interaktivni mreži.

Stolpcu na mreži lahko spreminjamo širino, tako da s klikom miške primemo in vlečemo rob glave stolpca. Stolpec lahko tudi skrijemo, tako da označimo glavo stolpca, nato pa z desnim klikom izberemo možnost *Hide*, kot prikazuje slika 7.3.



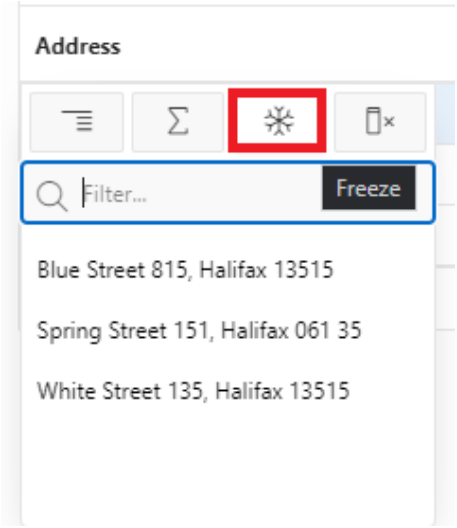
Slika 7.3: Gumb za skrivanje stolpca.

Ko stolpec skrijemo, se na mreži ne prikazuje več. Če ga želimo ponovno prikazati, izberemo *Actions*, nato *Columns* ter označimo stolpce, ki jih želimo prikazati.



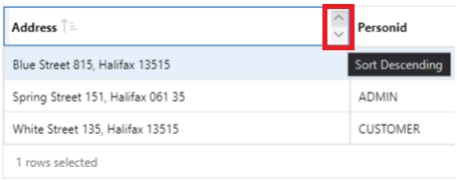
Slika 7.4: Stolpci za prikaz.

Stolpec lahko zamrznemo z izbiro opcije *Freeze*, kot prikazuje slika 7.5. Če stolpec zamrznemo, se s premiki drsnika njegova vsebina ne bo spreminjala.



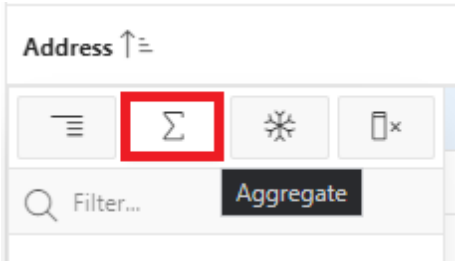
Slika 7.5: Možnost zamrznitve stolpca.

Podatke v stolpcu lahko sortiramo po vrstnem redu. Naraščajoče ali *Sort Ascending* oziroma padajoče ali *Sort Descending*, kot prikazuje slika 7.6.



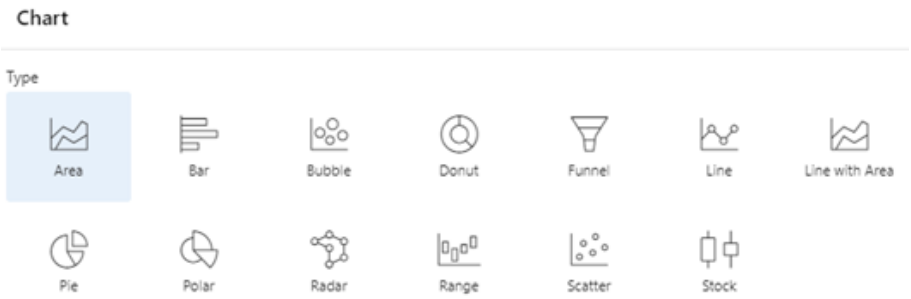
Slika 7.6: Sortiranje podatkov v stolpcu.

Podatke lahko združujemo na različne načine, z uporabo možnosti *Aggregate* na posameznem stolpcu, kot prikazuje slika 7.7. Izbiramo lahko med štetjem števila zapisov (angl. Count), štetjem števila različnih zapisov (angl. Count Distinct), najmanjšo (angl. Minimum) in največjo (angl. Maximum) vrednostjo v izbranem stolpcu.



Slika 7.7: Možnost združevanja podatkov Aggregate.

Podatke v mreži lahko uporabimo za kreiranje grafov tako, da v meniju *Actions* izberemo možnost *Charts*. Na sliki 7.8 so prikazane različne vrste grafov. Za grafični prikaz lahko uporabimo različne agregatne funkcije.



Slika 7.8: Vrste grafov.

7.3 Obrazec na osnovi tabele

Oracle APEX omogoča izgradnjo enostavnih obrazcev za vstavljanje ene vrstice v tabelo, kot prikazuje slika 7.9. Po začetku procesa kreiranja strani (angl. *Create a Page*), izberemo možnost *Form*. Izbrati moramo tabelo, ki bo povezana z obrazcem ali pa napisati SQL poizvedbo s primarnim ključem tabele.



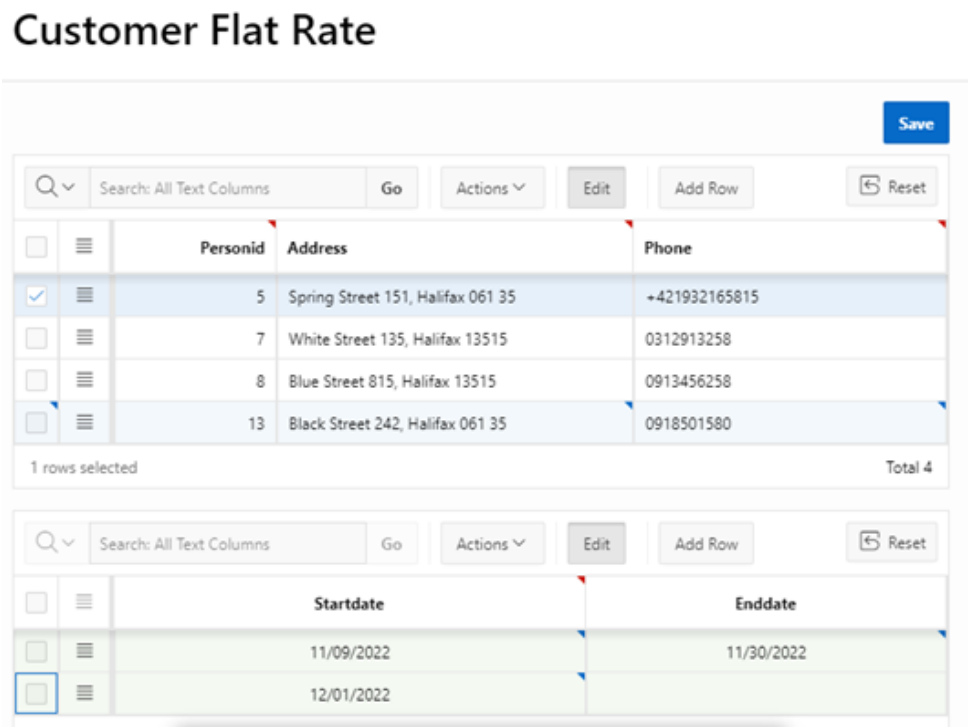
Slika 7.9: Obrazec za vnos podatkov o kupcu

7.4 Glavni obrazec s podrobnostmi

Možnost glavnega obrazca s podrobnostmi (angl. Master Detail Form) omogoča poizvedovanje, vnos, spremembe in brisanje podatkov iz dveh povezanih tabel ali vpogledov. V tem primeru moramo v procesu kreiranja strani izbrati možnost *Master Detail*. V naslednjem oknu potem izberemo način prikaza nadrejenega in podrejenega obrazca. Izbiram lahko med tremi različnimi načini prikaza, zloženim (angl. *Stacked*), ob strani (angl. *Side by Side*), ali z odpiranjem (angl. *Drill Down*). Glavni obrazec s podrobnostmi običajno prikazuje glavno vrstico in več povezanih podrejenih vrstic. To pomeni, da moramo pri kreiranju izbrati tabelo ali vpogled, ki je vir podatkov za glavni obrazec, ter tabelo ali vpogled, ki je vir podatkov za podrejeni obrazec. Izberemo tudi primarne ključe v vsaki tabeli ter tuj ključ, ki povezuje obe izbrani tabeli.

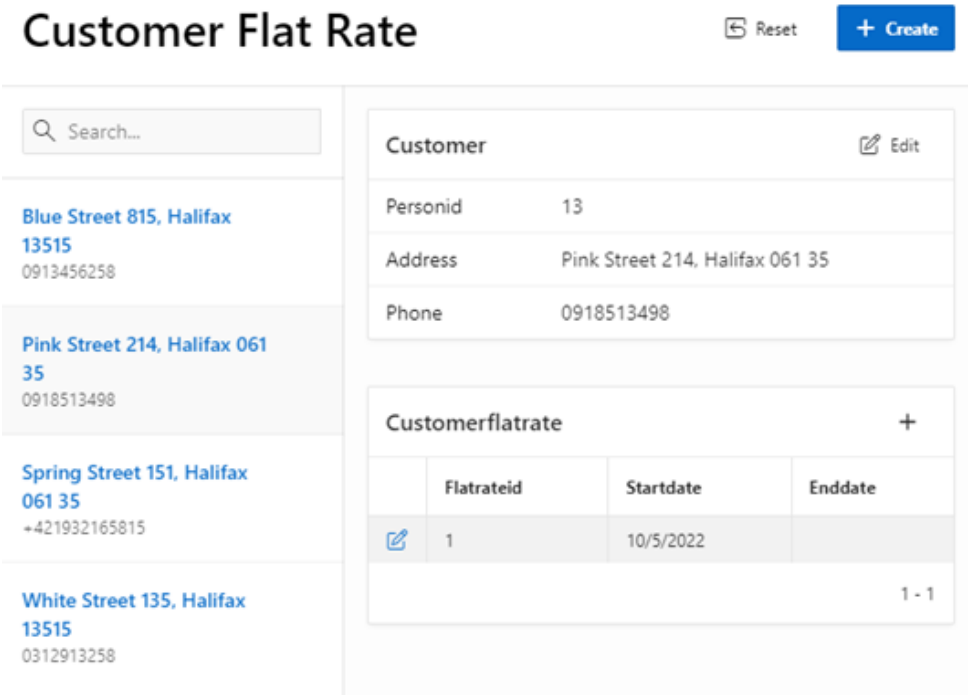
Slika 7.10 prikazuje glavni obrazec s podobrazcem v zloženi obliki (*Stacked*). Obrazec je izdelan na osnovi interktivne mreže, ki jo lahko spreminjamo. Z izbiro vrstice v glavnem obrazcu

se ustrezno spremeni prikaz povezanih vrstic v podrejenem obrazcu.



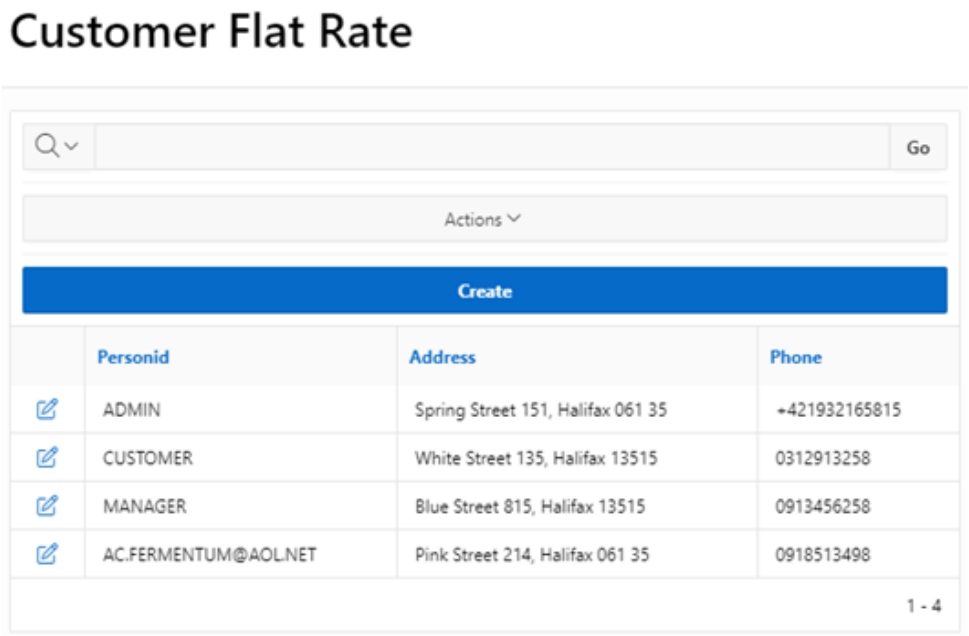
Slika 7.10: Primer obrazca s podobrazcem v zloženem načinu.

Na sliki 7.11 je prikazan obrazec s podobrazcem ob strani (angl. *Side by Side*). Pri tej obliki so podrobnosti glavnega obrazca predstavljene na eni strani, na drugi strani pa imamo na vojo modalna okna za urejanje tako podatkov na glavnem obrazcu, kot tudi na podrejenem obrazcu. Na levi strani imamo tudi možnost izbire podatkov v glavnem obrazcu, s katerimi se premikamo po zapisih. Desna stran pa vsebuje glavni zapis, s povezanimi podrejenimi zapisi.

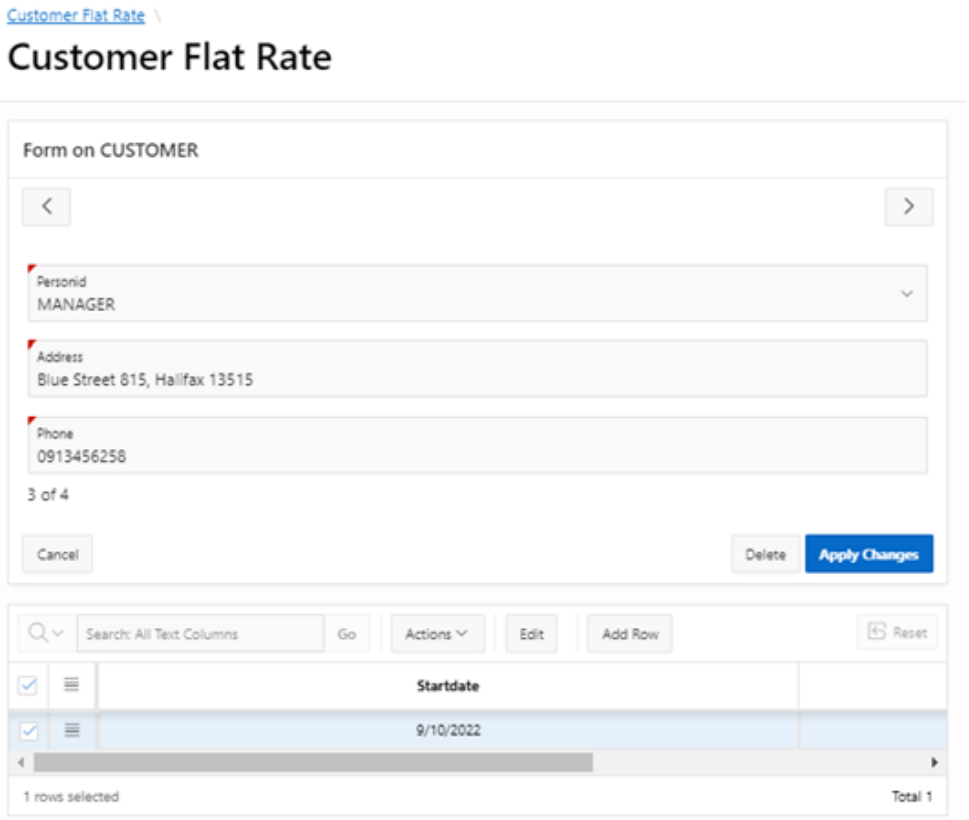


Slika 7.11: Oblika obrazca z podpobrazcem - Side by Side.

Glavni obrazec z podobrazcem, ki ga prikažemo z odpiranjem (angl. *Drill Down*) je prikazan na dveh straneh. Na prvi strani je interaktivno poročilo o zapisih v glavni tabeli, kot prikazuje slika 7.12. Na drugi strani pa je standardni obrazec za glavni zapis in interaktivna mreža z možnostjo urejanja za podrejeni zapis, kot prikazuje slika 7.13.



Slika 7.12: Oblika obrazca s podobrazcem - prva stran, Drill Down .



Slika 7.13: Oblika obrazca s podobrazcem - druga stran, Drill Down.

7.5 Vprašanja

1. Ali lahko na interaktivni mreži spreminjamo prikaz podatkov?
2. Katere sestavni del tabele moramo izbrati, ko kreiramo obrazec?
3. Katera lastnost izključi premikanje vsebine stolpca z drsnikom?

7.6 Odgovori

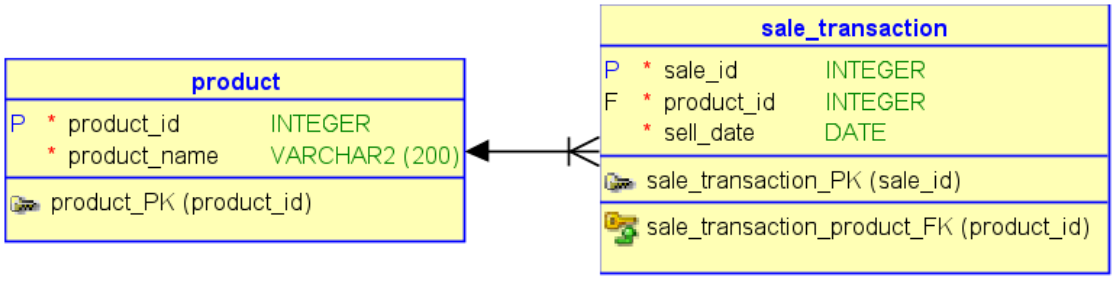
1. Da, poročilo lahko interaktivno prilagodimo in preuredimo z uporabo miške, ter na ta način spremenimo strukturo in vsebino interaktivne mreže.
2. Določen mora biti stolpec, ki je primarni ključ tabele.
3. Stolpec lahko zamrznemo z izbiro opcije Freeze. Na ta način se vsebina stolpca ne spreminja, če se po strani premikamo z drsnikom.

8. Kako poročila spremenite v grafikone?

IVAN PASTIERIK, MICHAL KVET, MIROSLAV POTOČÁR; PREVOD A. BAGGIA

8.1 Grafikon

Regije z grafikoni v Oracle APEX-u so zmogljive komponente, ki omogočajo razvijalcem in uporabnikom globlji vpogled v podatke preko vizualne predstavitve. Dinamični grafikoni, ki vključujejo različne prikaze, od stolpičnih in črtnih grafov do krožnih grafov in še več, pomagajo do boljše uporabniške izkušnje in poenostavijo odločanje na podlagi podatkov. V tem poglavju bomo prikazali, kako ustvariti preprost graf s **filtriranjem** in **razvrščanjem**. **Filtriranje** in **razvrščanje** izboljšata interaktivnost in berljivost grafov. V ta namen smo ustvarili preprost podatkovni model, ki je prikazan na sliki 8.1. Podatkovni model sestavljata dve entiteti: **"product"** in **"sale_transaction"**. Entiteta **"product"** predstavlja posamezen izdelek, pri čemer je primarni ključ atribut **"product_id"**, ime izdelka pa je shranjeno v atributu **"product_name"**. Entiteta **"sale_transaction"** hrani informacije o vseh prodanih izdelkih, pri čemer je primarni ključ **"sale_id"**. Atribut **"product_id"** je tuji ključ, ki se nanaša na atribut **"product_id"** v entiteti **"product"**. Atribut **"sell_date"** vsebuje datum prodaje izdelka.

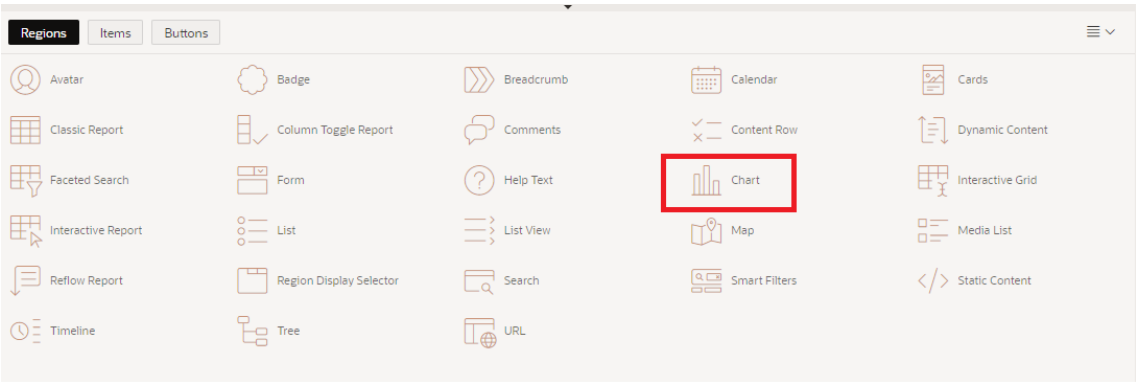


Slika 8.1: Podatkovni model za prikazani primer razvoja aplikacije.

Na začetku ustvarimo novo aplikacijo z nazivom **"Chart Showcase"**. Ta aplikacija bo vsebovala globalno stran **"Global Page"**, domačo stran **"Home"** in prijavno stran **"Login Page"**. Grafikone bomo ustvarjali na domači strani. Podrobna navodila za ustvarjanje aplikacij so opisana v poglavju 5. Po kreiranju aplikacije vstopimo preko urejavalnika strani (**Page Designer**) v stran z nazivom **"Home"**.

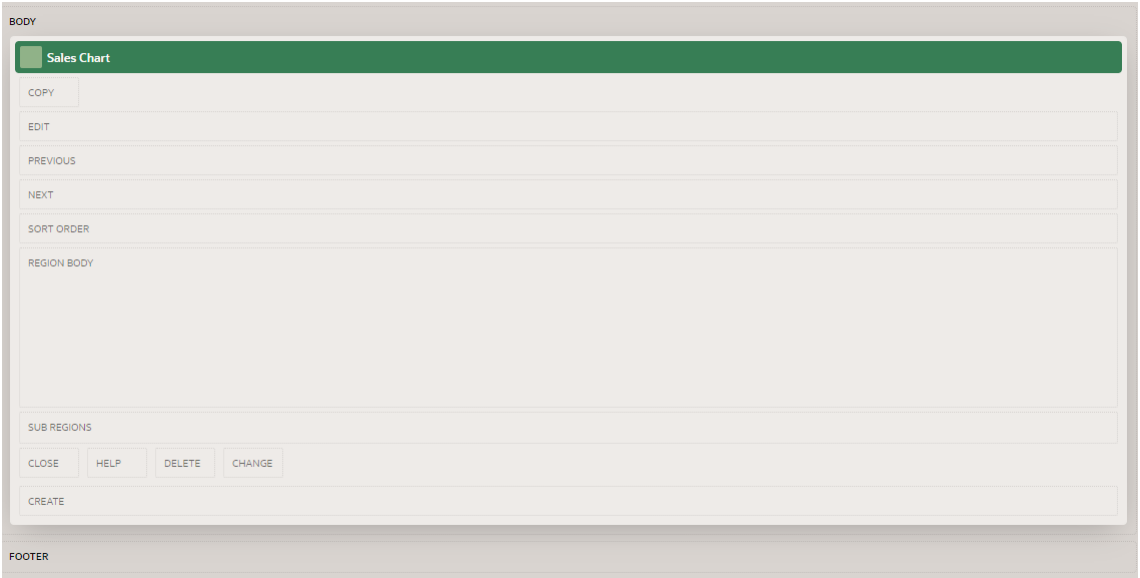
8.2 Izdelava stolpičnega grafikona

Za vstavljanje regije za grafikone v aplikaciji, moramo regijo z miško povleči v razdelek vpogleda (ang. Layout section) v urejevalniku strani (**Page Designer**). Primer je prikazan na sliki 8.2.



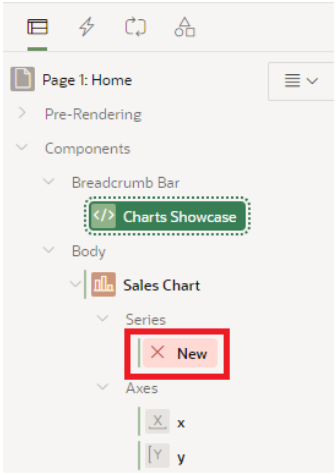
Slika 8.2: Vstavljanje regije za grafikone.

Regijo za grafikone bomo vstavili v razdelek **body** saj želimo, da je grafikon glavni element naše strani. Novo regijo bomo poimenovali **"Sales Chart"**. Na sliki 8.3 vidimo, kako bo sekcija **body** izgledala po dodajanju in imenovanju.



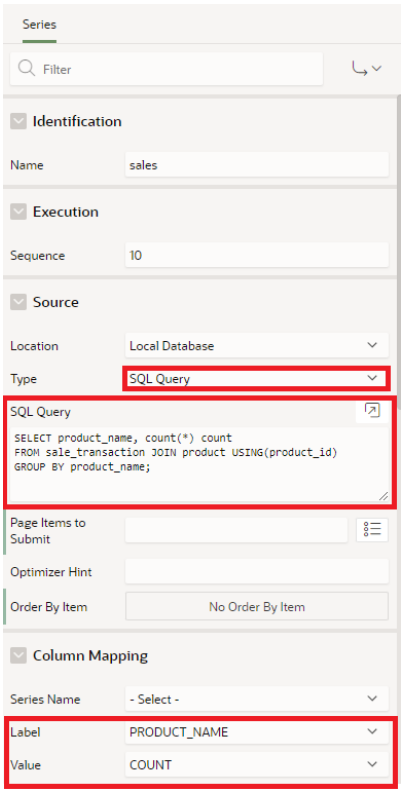
Slika 8.3: Izgled strani po vstavljanju regije za grafikone.

V levem delu urejevalnika strani (**Page Designer**), vidimo drevesni pogled vseh elementov. Če pogledamo komponento **Sales Chart** podrobneje, vidimo, da se znotraj vrste (**"Series"**) prikaže napaka. Napaka, ki je prikazana na sliki 8.4, se izpiše, ker še nismo določili vira podatkov za prikaz na grafikonu.



Slika 8.4: Drevesni pregled, ki kaže napako v elementu grafikona.

S klikom na vrsto, označeno z **"New"** v drevesni strukturi, se odprejo nastavitve na desni strani urejevalnika (**Page Designer**). V nastavitvah bomo spremenili naziv vrste **"sales"**, saj želimo prikazati podatke o prodaji izdelkov na grafikonu **Sales Chart**. Nato bomo določili vir vrste **ang. source**. Spremenili bomo tip izvora v **"SQL Query"**, v SQL poizvedbo pa bomo vpisali **SELECT** stavek (glejte sliko 8.5). Ukaz **SELECT** mora vrniti skupne seštevke prodajnih količin, združe po imenu izdelka. Ko smo določili vir podatkov za grafikon, moramo določiti stolpce v ukazu **SELECT**: za os X **Label (X)** in os Y **Value (Y)**. Končna nastavitev naše vrste je prikazana na sliki 8.5.



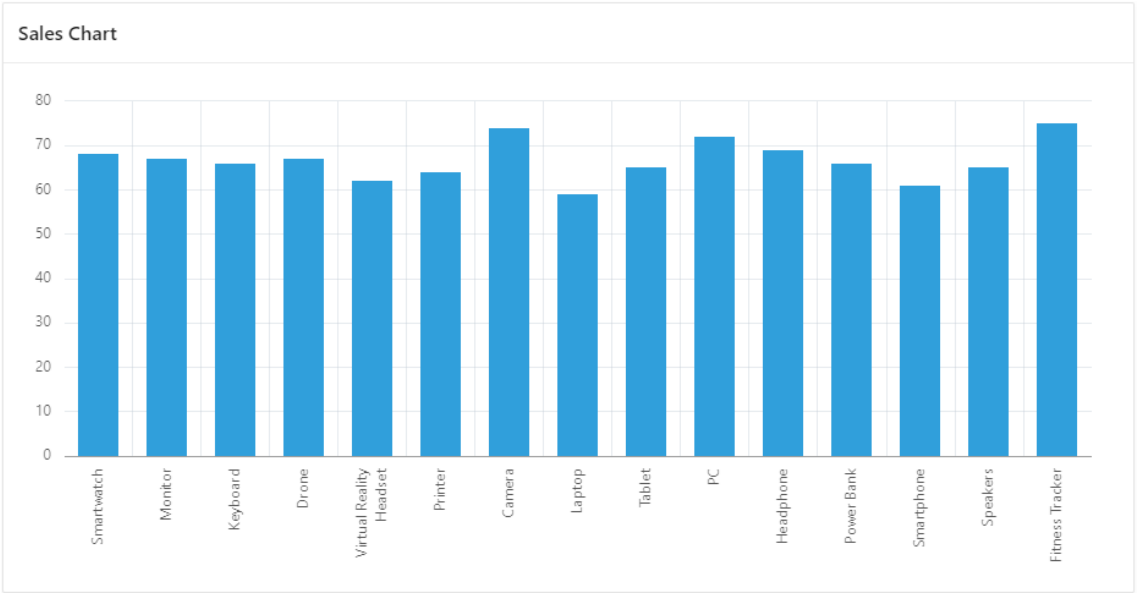
Slika 8.5: Nastavitev vrste o prodaji.

Če uporabimo poizvedbo SQL, ki je prikazana na sliki 8.5, dobimo grafikon na sliki 8.6.

PRODUCT_NAME	COUNT
Smartwatch	68
Monitor	67
Keyboard	66
Drone	67
Virtual Reality Headset	62
Printer	64
Camera	74
Laptop	59
Tablet	65
PC	72

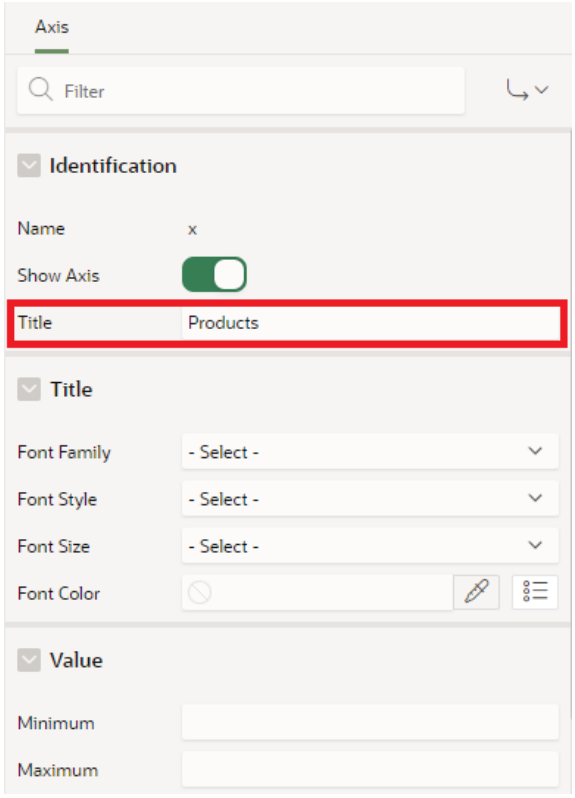
Slika 8.6: Rezultat poizvedbe SQL.

Zdaj lahko shranite stran in jo zaženete. Ne pozabite nastaviti **avtentikacije** strani na **"Page Is Public"**, da ne bo potrebna prijava. Ko zaženete stran, se bodo prikazali rezultati, kot prikazuje slika 8.7. Opazimo lahko, da grafikon nima označenih **X** in **Y** osi.



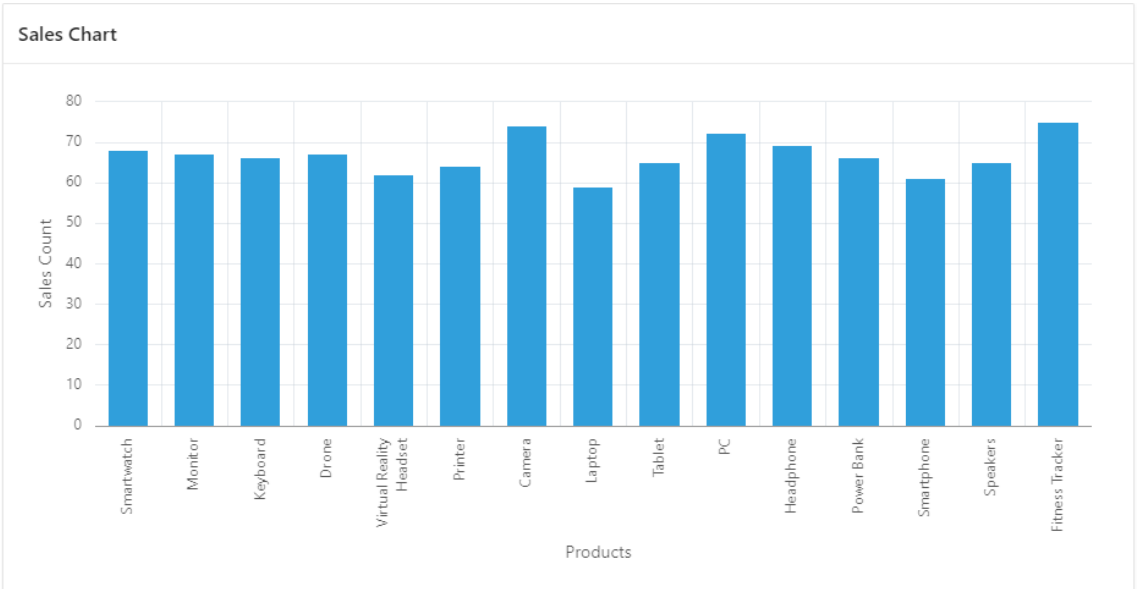
Slika 8.7: Grafikon, ki se prikaže ob zagonu aplikacije.

Osi poimenujemo podobno kot smo poimenovali vrste. Izberemo posamezno os na drevesnem prikazu na levi ter nato spremenimo naslov **ang. Title** (glejte sliko 8.8). Os **X** poimenujte **"Products"**, os **Y** pa **"Sales Count"**.



Slika 8.8: Spremenjen naziv x osi.

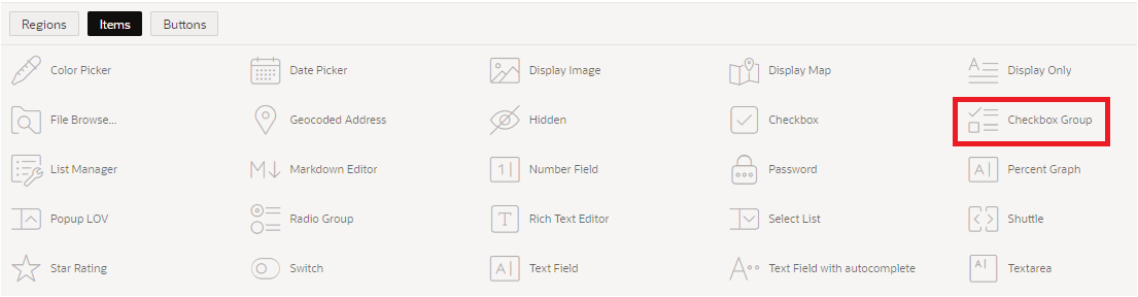
Ko preimenujete obe osi, bo grafikon izgledal tako, kot je prikazano na sliki 8.9.



Slika 8.9: Pripravljen grafikon prodaje.

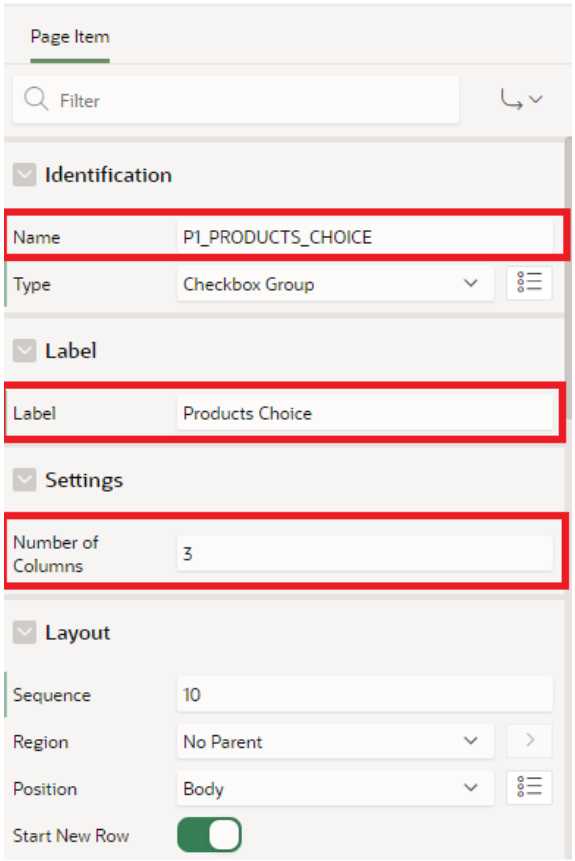
8.3 Dodajanje filtrov na stolpični grafikon

Dodajte **filtriranje** izdelkov v grafikonu. S filtrom lahko interaktivno izbiramo, katere izdelke bomo prikazali. Dodali boste element **Checkbox Group** iz spodnje orodne vrstice urejevalnika (**Page Designer**) ter ga postavili nad regijo **Sales Chart**.



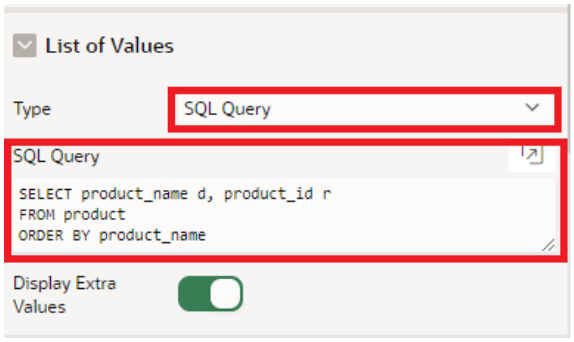
Slika 8.10: Dodajanje elementa Checkbox Group.

Element **Checkbox Group** imenujte **"P1_PRODUCTS_CHOICE"** in nastavite oznako elementa na **"Products Choice"**. Poleg tega lahko nastavite tudi prikaz v treh stolpcih, kot je prikazano na sliki 8.11.



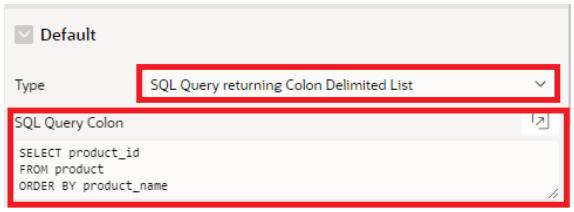
Slika 8.11: Nastavitev osnovnih lastnosti elementa Checkbox Group.

Da bomo napolnili element **Checkbox Group** s podatki o izdelkih, morate najprej določiti seznam vrednosti. V razdelku **"List of Values"** med nastavitvami **Page Item**, spremenite lastnost **"Type"** v **"SQL Query"**. Vpišite poizvedbo z ukazom **SELECT** (glejte sliko 8.12). Ta ukaz vrne dva stolpca s podatki: v prvem stolpcu so vrednosti, ki jih prikazujemo in je poimenovan kot (**"d"**), v drugem stolpcu pa so vrednosti, ki jih shranjujemo in je poimenovan kot (**"r"**).



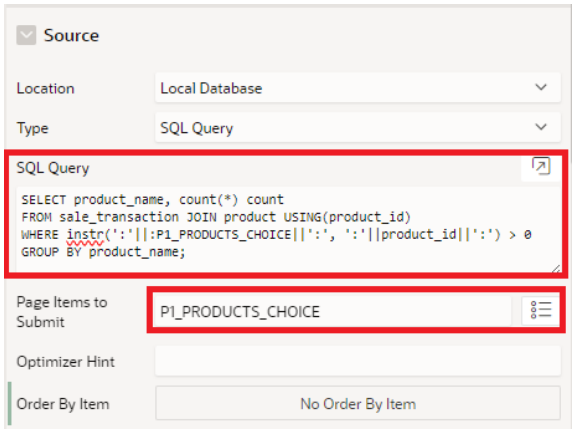
Slika 8.12: Nastavitve seznama vrednosti za element Checkbox Group.

Privzeto ima element **Checkbox Group** ob zagonu aplikacije vse oznake prazne. Za zagotovitev boljše uporabniške izkušnje pri prikazu grafikonov je bolje, da uporabnik že takoj vidi vse izdelke. Zato v sekciji **"Default"**, v nastavitvah elementa **Checkbox Group** spremenite lastnost **"Type"** v **"SQL Query returning Colon Delimited List"**. Ta vrsta SQL poizvedbe omogoča določitev privzetih vrednosti. Običajne poizvedbe SQL ne moremo uporabiti pri elementu **Checkbox Group**, ker določa samo eno privzeto vrednost. V okno **"SQL Query Colon"** boste napisali ukaz **SELECT**, kot je prikazano na sliki 8.13. Ukaz **SELECT** vrne natanko eno vrednost, ki se potem pretvori v **Colon Delimited List** in lahko izgleda takole: **"1:5:7:2:3"**.



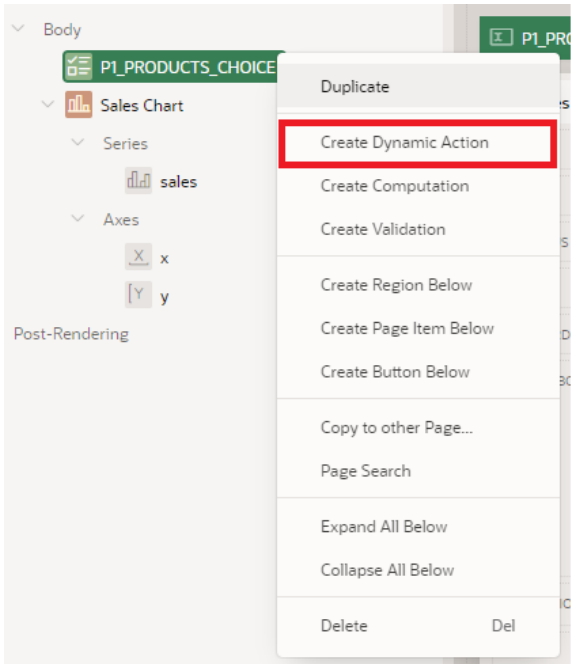
Slika 8.13: Nastavljanje privzetih vrednosti za element Checkbox Group.

Ko ste elementu **Checkbox Group** nastavili vse potrebne lastnosti, ga lahko povežete z regijo **Sales Chart**. Ime **Checkbox Group** boste dodali v polje **"Page Items to Submit"** kot **"Source"** sekcije vrste **sales** in sicer v regiji **Sales Chart**. **Obvezno** morate dodati vsak element, ki ga uporablja poizvedba SQL v **"Page Items to Submit"**. Vrednost, ki jo vrne **"P1_PRODUCTS_CHOICE"**, je seznam **Colon Delimited List**, na katerem so podatki o izbranih izdelkih. Uporabite funkcijo **INSTR**, ki vrne indeks prvega pojava delnega niza (drugi parameter funkcije) v celotnem nizu (prvi parameter funkcije). Funkcija **INSTR** vrne **0**, če v nizu ni najden delni niz. Funkcijo **INSTR** boste uporabili za preverjanje ali je identifikacija izdelka **"(product_id)"** na izbranem seznamu identifikacijskih številk v **"P1_PRODUCTS_CHOICE"**. Nato funkcija**INSTR** poišče **':'||product_id||':'** v nizu **':'||:P1_PRODUCTS_CHOICE||':'**. Če najde zadetek, to pomeni, da je izdelek med izbranimi izdelki. Združiti morate **':'** tako **"product_id"** kot **":P1_PRODUCTS_CHOICE"**, ker želimo, da se ID izdelkov popolnoma ujema. Tako boste zagotovili, da se na primer **":1: "** ujema z **":1:2:3:8:9:11: "**. Na ta način iščete popolna ujemanja in zagotovite, da je vsak ID izdelka obravnavan neodvisno. Če želite, da se **"1"** ujema z **"1:2:3:8:9:11"**, bi lahko izpisali tudi izdelke z obema ID-jema **"1"** in **"11"**, prav tako pa vse ostale izdelke, katerih ID vsebuje **"1"**. V tem primeru bi dobili napačne rezultate in predstavitev podatkov ne bi bila točna. Ker dodamo **':'** na obeh straneh ID-ja, določimo konsistentno ločilo ter tako zagotovimo, da je primerjava ID-jev izvedena natančno in pri preverjanju ujemanja ne pride do napak. Tako bo grafikon prikazaval natanko tiste podatke, ki jih je izbral uporabnik.



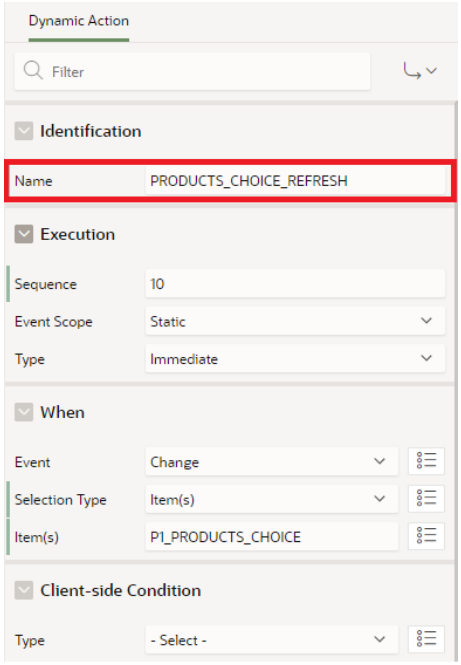
Slika 8.14: Povezovanje regije Sales Chart z elementom Checkbox Group.

Zadnji korak pri nastavitvah je določitev osveževanja regije **Sales Chart** glede na spremembe v elementu **Checkbox Group**. Za vzpostavitev osveževanja določite dinamično aktivnost **(ang. Dynamic Action)** na elementu Checkbox Group **"P1_PRODUCTS_CHOICE"**, kot prikazuje slika 8.15.



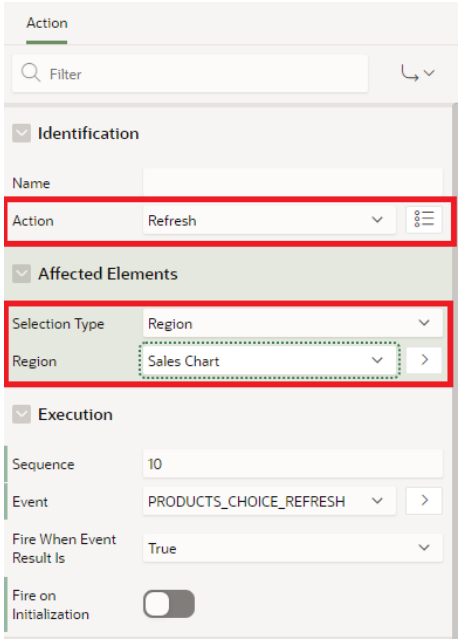
Slika 8.15: Določanje dinamične aktivnosti za element P1_PRODUCTS_CHOICE.

Lastnost **"Name"**nastavite na **"PRODUCTS_CHOICE_REFRESH"**, kot prikazuje slika 8.16. Lastnost **"Event"** v sekciji **"When"** nastavite na **"Change"** ter lastnost **"Selection Type"** na **"Item(s)"**. Pri **"Item(s)"** vpišite ime trenutnega elementa, da se **dinamična aktivnost** sproži, ko se spremeni vrednost **"P1_PRODUCTS_CHOICE"**.



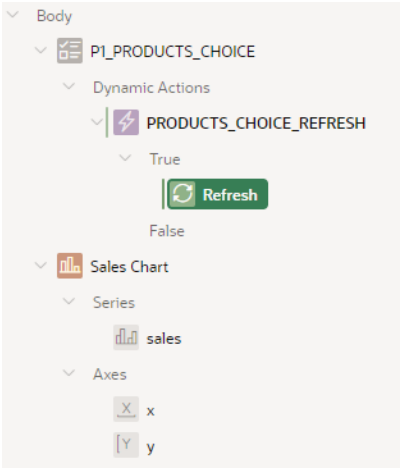
Slika 8.16: Nastavitev osnovnih lastnosti dinamične aktivnosti ob spreminjanju.

V drevesnem pogledu na levi strani urejevalnika (**Page Designer**), izberemo akcijo **Dynamic Action**, ki se izvede ob izpolnitvi pogoja **(true)**. To se zgodi ob spremembi vrednosti elementa **Checkbox Group**. Ko kreirate novo akcijo (**Dynamic Action**), lahko pri lastnosti **"Show"** vidite napako. Nastavite lastnost **"Identification"** na **"Refresh"**. V sekciji **"Affected Elements"** spremenite **"Selection Type"** v **"Region"** ter nastavite **"Region"** na **"Sales Chart"**, kot prikazuje slika 8.17.



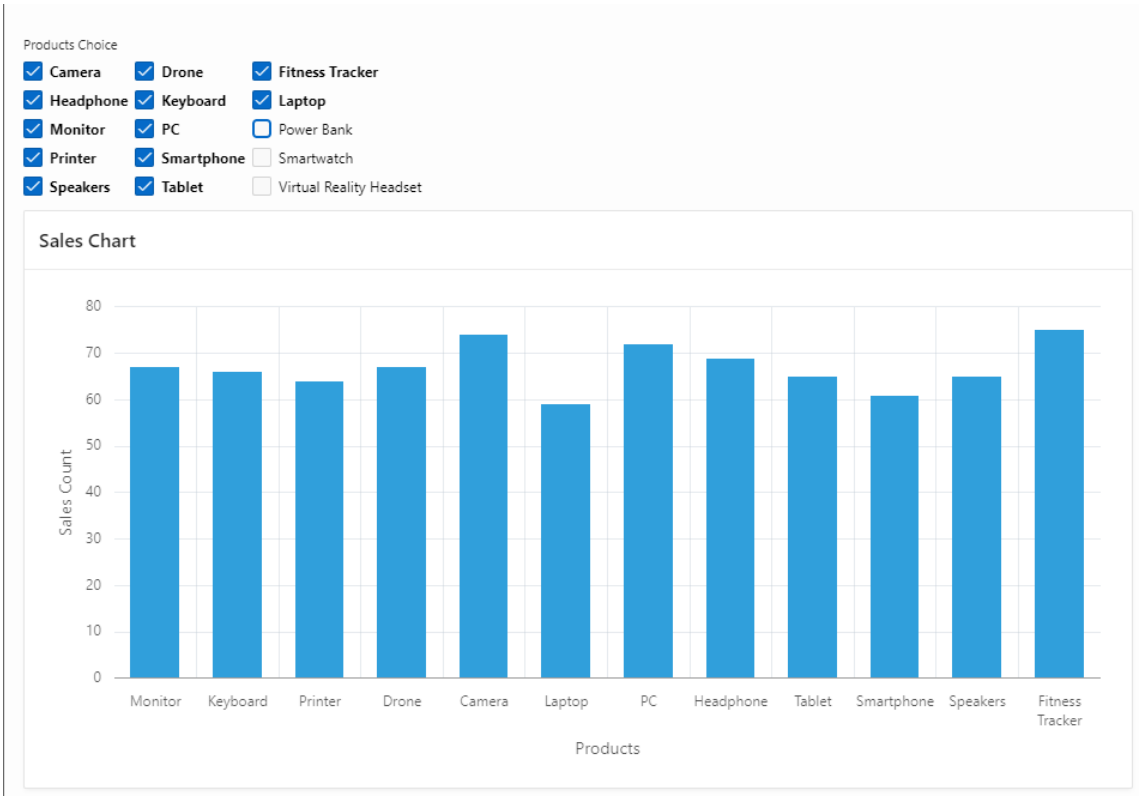
Slika 8.17: Nastavitev lastnosti ob osveževanju.

Slika 8.18 prikazuje kako bodo izgledale nastavitve v urejevalniku (**Page Designer**) na levi strani, če ste sledili vsem zgornjim korakom.



Slika 8.18: Dinamična aktivnost za Sales Chart v drevesnem pogledu.

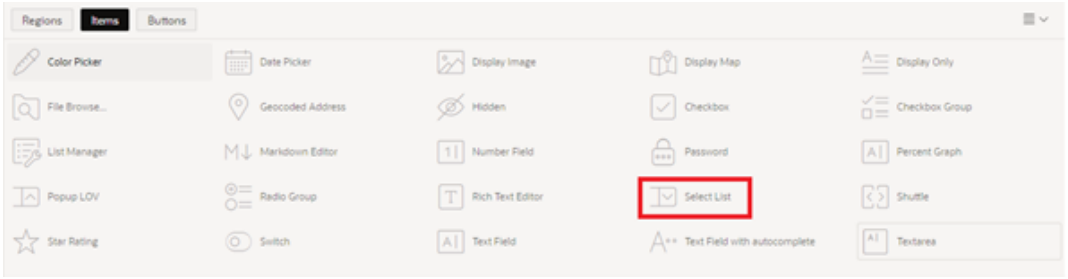
Ko zaženete aplikacijo, bo **filtriranje** in **osveževanje** grafikona delovalo. Izgled aplikacije po vseh zgoraj navedenih spremembah je prikazan na sliki 8.19.



Slika 8.19: Aplikacija z vključenim filtrom za izbiro izdelkov.

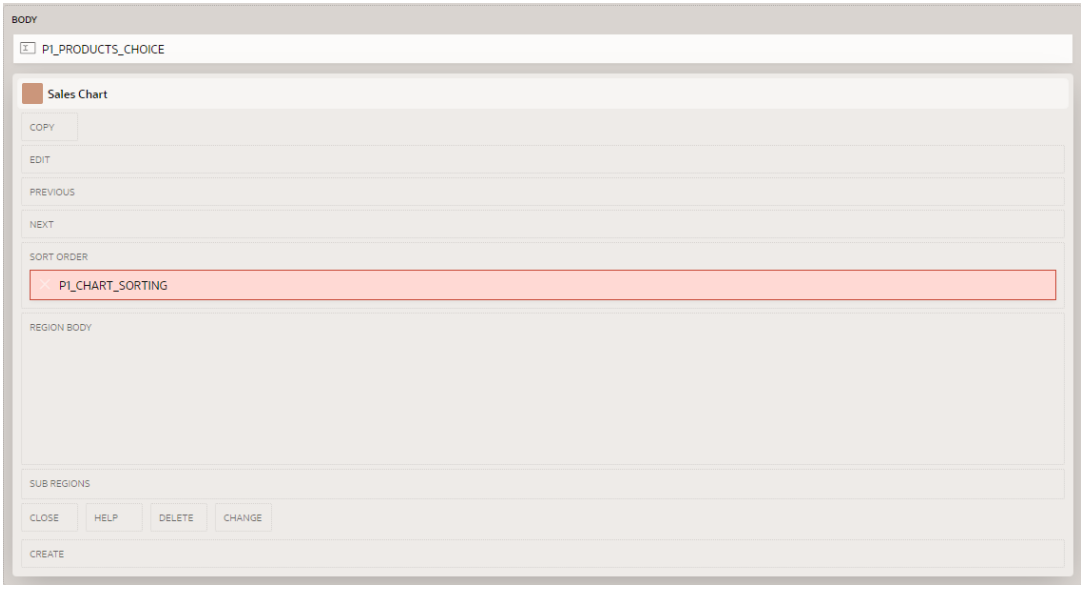
8.4 Razvrščanje v stolpičnemu grafikonu

Uporabniško izkušnjo lahko še nadgradimo, če uporabnikom omogočimo, da razvrščajo vrednosti, ki so prikazane na grafikonu, po različnih kriterijih. Za izbiro kriterija razvrščanja bomo izbrali element **Select List**, kot prikazuje slika 8.20.



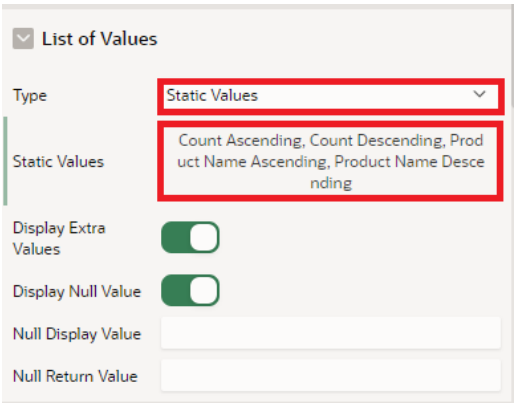
Slika 8.20: Izbira in postavitev elementa Select List.

Element **Select List** bomo postavili v regijo **Sales Chart** pod sekcijo **"Sort Order"** in ga poimenovali **"P1_CHART_SORTING"** (glejte sliko 8.21).



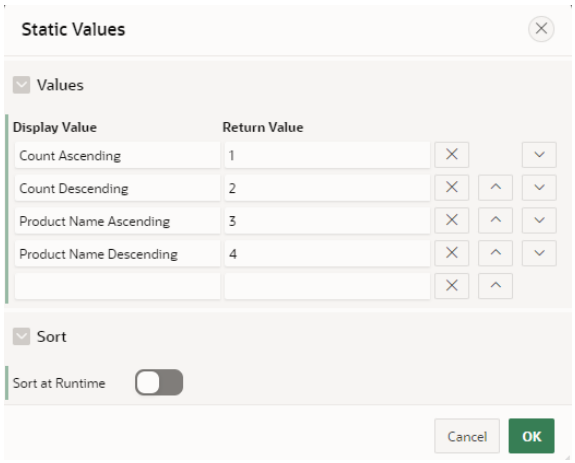
Slika 8.21: Regija Sales Chart z dodanim elementom Select List.

V nadaljevanju moramo določiti seznam vrednosti, ki ga bo prikazoval element Select List. V nastavitvah elementa **"P1_CHART_SORTING"** spremenite lastnost **"Type"** pri sekciji **"List of Values"** na **"Static Values"**. Nato izberete gumb, ki prikazuje privzete vrednosti **"Display1, Display2"** poleg lastnosti **"Static Values"**.



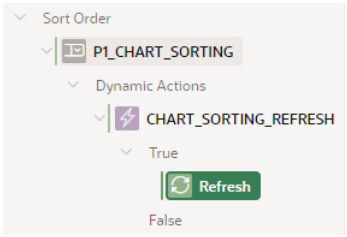
Slika 8.22: Nastavitev seznama vrednosti za element Select List.

Odpre se novo okno z imenom **"Static Values."** V tem oknu določimo vrednost, ki se prikazuje in vrednost, ki se shrani. Pri vrednosti **display** zapišemo opise načinov razvrščanja, pri vrednosti **return**, pa vpišemo števila po vrsti od **1** naprej, kot prikazuje slika 8.23. Če želite prikazati vrednosti v istem vrstnem redu, kot smo jih vpisali, od vrha navzdol, odstranimo oznako pri **"Sort at Runtime"**.



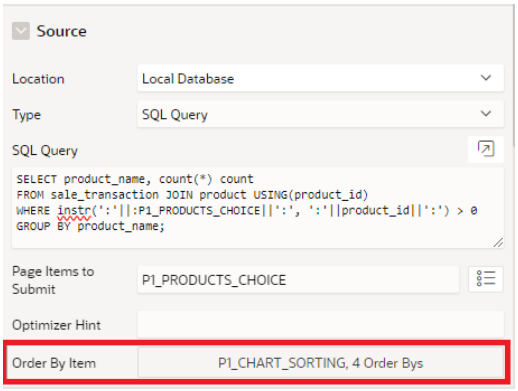
Slika 8.23: Nastavitev statičnih vrednosti pri elementu Select List.

Dodajte dinamično aktivnost **Sales Chart** za osveževanje na enak način, kot ste jo nastavili pri elementu **"P1_PRODUCTS_CHOICE"**. Rezultat je prikazan na sliki 8.24.



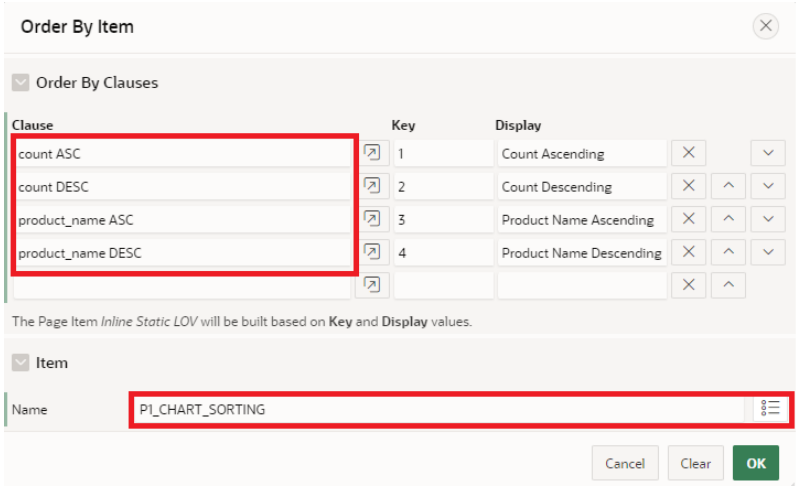
Slika 8.24: Dodajanje dinamične aktivnosti za osveževanje pri elementu Select List.

Ko ste določili vse lastnosti elementa **Select List**, ga morate povezati z regijo **Sales Chart**. To lahko storite tako, da izberete gumb pri lastnosti **"Order By Item"**, ki je privzeto nastavljena na **"No Order By Item"** (glejte sliko 8.25).



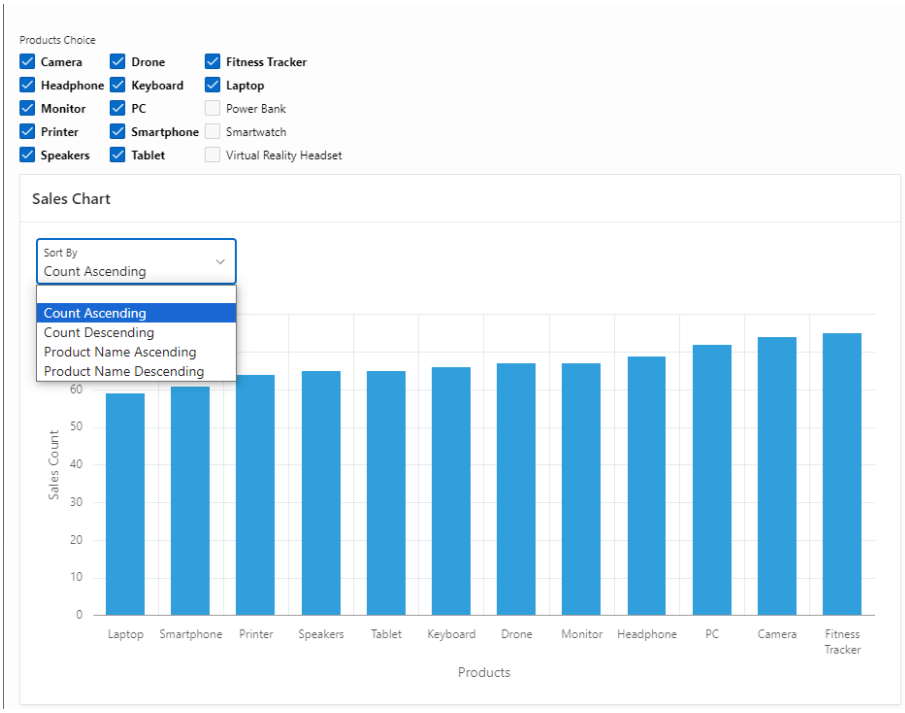
Slika 8.25: Nastavitev elementa Order By za regijo Sales Chart.

Ob kliku na gumb se odpre novo okno z imenom **“Order By Item.”** . V tem oknu najprej nastavite ime na **“P1_CHART_SORTING”**. Nato pa napolnite stolpec “Clause”, kot prikazuje slika 8.26. To so dejansko ukazi za razvrščanje **“ORDER BY”**, ki so ekvivalentni ukazu **SELECT**, vendar brez ključne besede **“ORDER BY”** .



Slika 8.26: Nastavitev ukazov Order By za razvrščanje pri elementu Select List.

Če ste pozorno sledili vsem opisanim korakom, bo grafikon omogočal tako **razvrščanje** kot **osveževanje**. Shranite stran in preskusite aplikacijo.

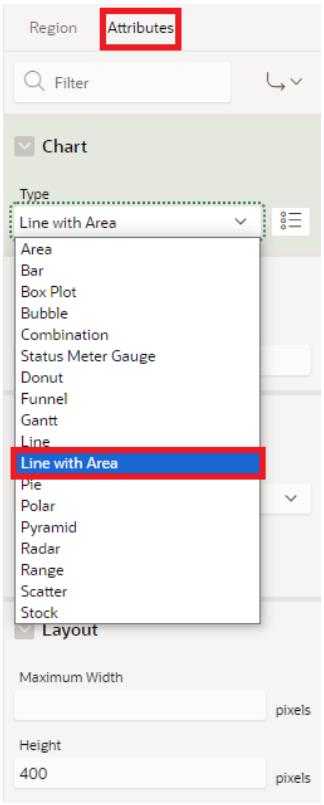


Slika 8.27: Aplikacija z vključenim razvrščanjem.

8.5 Izdelava različnih grafikonov

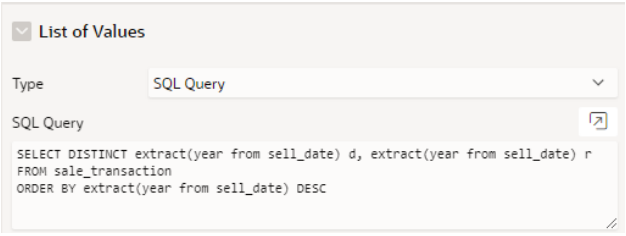
Če želite uporabiti drugo vrsto grafikona, v zavihku **“Attributes”** v desnem delu urejevalnika (**Page Designer**) spremenite lastnosti **“Type”** . Prikazali bomo izdelavo črtnega grafikona s področjem

(**ang. Line with Area**). Grafikon bo prikazal mesečno vrednost prodaje in bo omogočal filtriranje po letih. Zato ga bomo poimenovali **"Monthly Sales Chart"**.



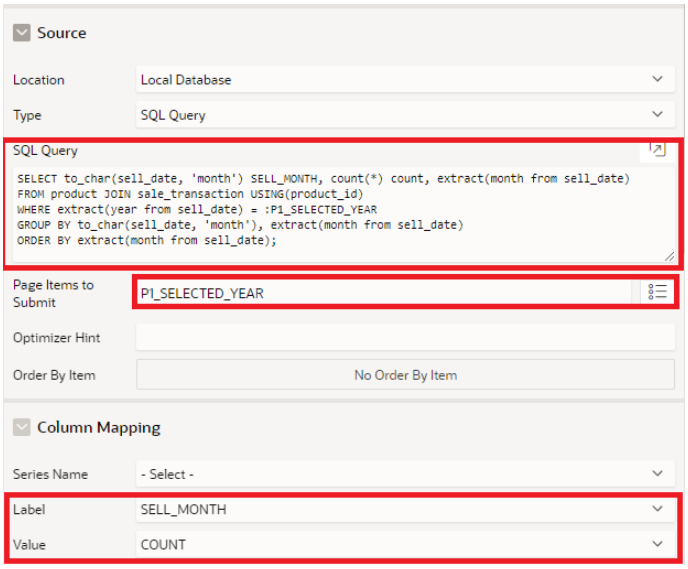
Slika 8.28: Spreminjanje vrste grafikona v obliko Line with Area.

V naslednjem koraku dodamo element **Select List**, ki ga bomo uporabili za filtriranje prikazanih podatkov po letih. Element bomo vstavili v regijo **"Monthly Sales Chart"**, pod sekcijo **"Edit"**. Poizvedba SQL za izpis vrednosti je prikazana na sliki 8.29.



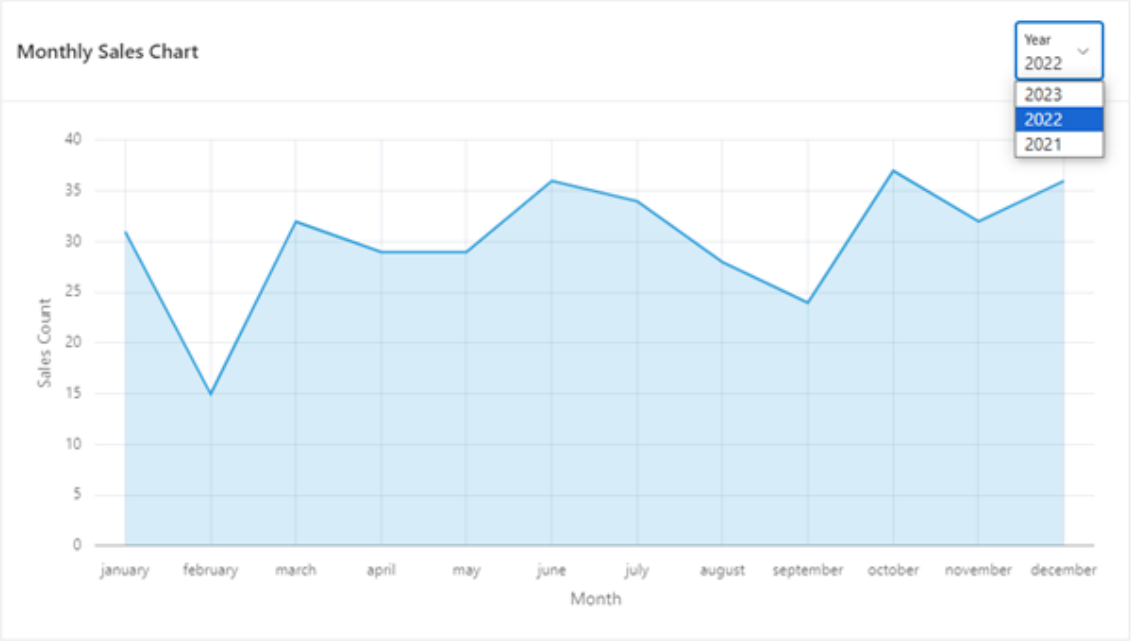
Slika 8.29: Seznam vrednosti za filtriranje izpisa prodaje po letih pri elementu Select List.

Ko smo pripravili element **Select List** za filtriranje, lahko napišemo poizvedbo SQL za vrste v regiji **"Monthly Sales Chart"**. Poizvedba je prikazana na sliki 8.30. Oznako **"Label"** bomo povezali z **"SELL_MONTH"** ter **"Value"** z **"COUNT"**. Vrednosti razvrstimo po mesecu naraščajoče in prikažemo celo ime meseca na osi **X**. Nastavimo oznako osi **X** na **"Month"** ter osi **Y** na **"Sales Count"**.



Slika 8.30: Nastavitev poizvedbe SQL ter povezovanje stolpcev v regiji Monthly Sales Chart.

Če ste uspešno sledili predstavljenim korakom, bo grafikon izgledal kot na sliki 8.31.



Slika 8.31: Stran z grafičnim prikazom Monthly Sales Chart.

8.6 Vprašanja

1. Kaj je glavni namen regij za grafikone v Oracle APEX?
2. Katere tipe grafikonov lahko kreiramo z regijami v Oracle APEX?
3. Kako lahko osvežimo regijo za grafikon, če uporabnik zahteva filtriranje ali sortiranje podatkov?
4. Kaj naredi funkcija INSTR in za kaj se uporablja v regiji za grafikone v Oracle APEX?

8.7 Odgovori

1. Glavni namen regij za grafikone v Oracle APEX je grafična predstavitev podatkov. To uporabnikom omogoča hitrejši vpogled v podatke ter odločanje na podlagi podatkov.
2. V regijah za grafikone v Oracle APEX so omogočeni različni tipi kot so stolpčni, črtni, tortni in podobni.
3. Za osveževanje regije z grafikonom lahko določimo dinamično akcijo (Dynamic Action), če uporabnik zahteva filtriranje ali sortiranje podatkov.
4. Funkcija INSTR vrne pozicijo začetka prvega primera iskanega niza v besedilnem nizu. V regiji za grafikone v Oracle APEX funkcijo uporabimo za preverjanje ali je določena vrednost najdena v podatkih (npr. del kateregakoli stolpca v poročilu).

9. Kako urejate menije?

VERONIKA ŠALGOVÁ, MICHAL KVET, MIROSLAV POTOČÁR; PREVOD A. BAGGIA

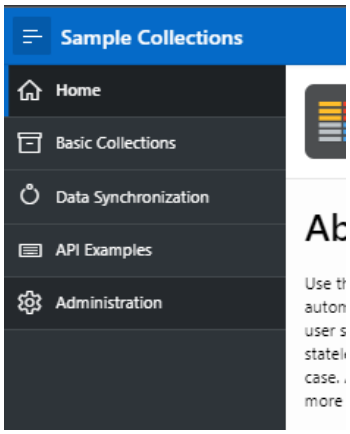
9.1 Kako urejamo menije

Navigacijski meni se avtomatično kreira, ko kreiramo aplikacijo. Sestavljen je na osnovi strani, ki smo jih naredili v aplikaciji. Način in mesto prikaza menija pa lahko tudi spremenimo, če spremenimo atribute na strani uporabniškega vmesnika (angl. User interface). Oracle APEX omogoča tri vrste menijev:

- stranski meni (angl. Side Menu),
- meni na vrhu strani (angl. Top Menu),
- veliki meni (angl. Mega Menu).

9.2 Stranski meni

Stranski meni lahko odpiramo in zapiramo s klikom na ikono menija v glavi strani. Posamezni elementi navigacije so prikazani v drevesni strukturi. Podobno kot glavne elemente, lahko odpiramo tudi podrejene elemente menija. Odpremo lahko celoten meni ali pa ga skrčimo samo v ikono.

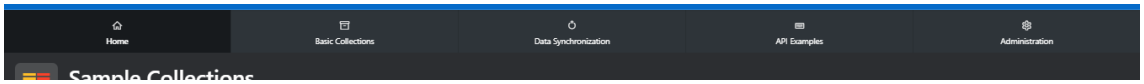


Slika 9.1: Razširjen stranski meni.

Privzeto je stranski meni skrit. V Template Options pa lahko nastavimo tudi tako, da je v primeru skritega menija prikazana ozka vrstica z ikonami.

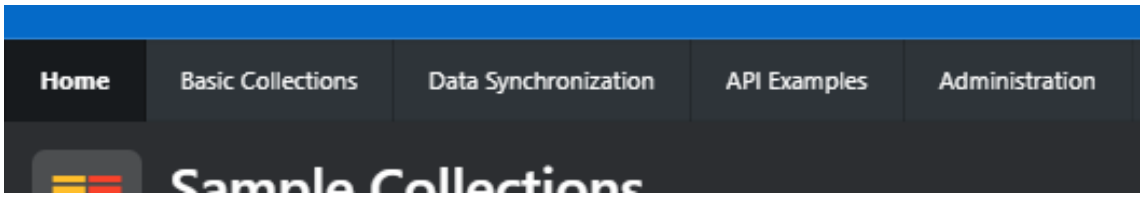
9.3 Meni na vrhu strani

Meni na vrhu strani se prikazuje na vrhu aplikacije. Oracle APEX ponuja dve predlogi za prikaz menija. Meni je lahko v obliki zavihkov *Top Navigation Tabs* ali prikazan v celoti *Top Navigation Menu.* Prvi tip je primeren za aplikacije, ki imajo v meniju malo elementov. Prikazan je na sliki 9.2.



Slika 9.2: Meni z zavihki na vrhu strani.

Meni na vrhu strani doda menijsko vrstico. Ta je primerna za bolj kompleksne aplikaije, ki ima več nivojev hierarhije v navigaciji. Prikazan je na sliki 9.3.

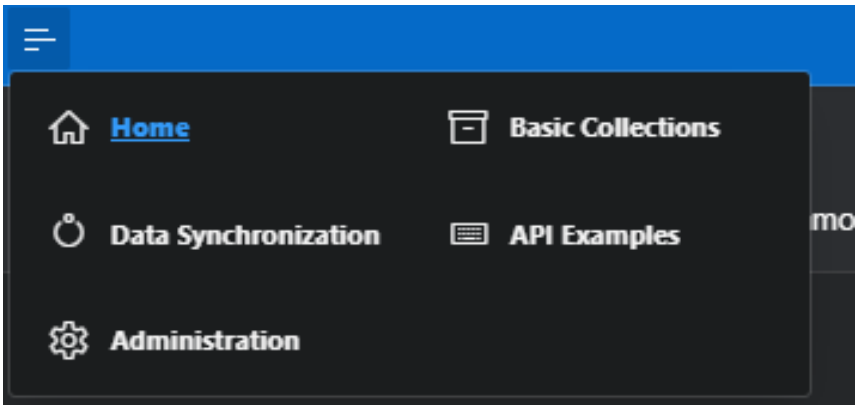


Slika 9.3: Meni na vrhu strani.

Meni na vrhu strani z zavihki se v primeru manjših zaslonov ali mobilnih naprav avtomatično postavi na dno zaslona. Meni na vrhu strani z menijsko vrstico, pa je podoben kot v večini namiznih aplikacij.

9.4 Veliki meni

Veliki meni je prikazan kot pojavno okno, ki ga lahko odprete ali zaprete s klikom na ikono v glavi strani. Tovrstni meniji so koristni predvsem za prikaz vseh navigacijskih elementov naenkrat.



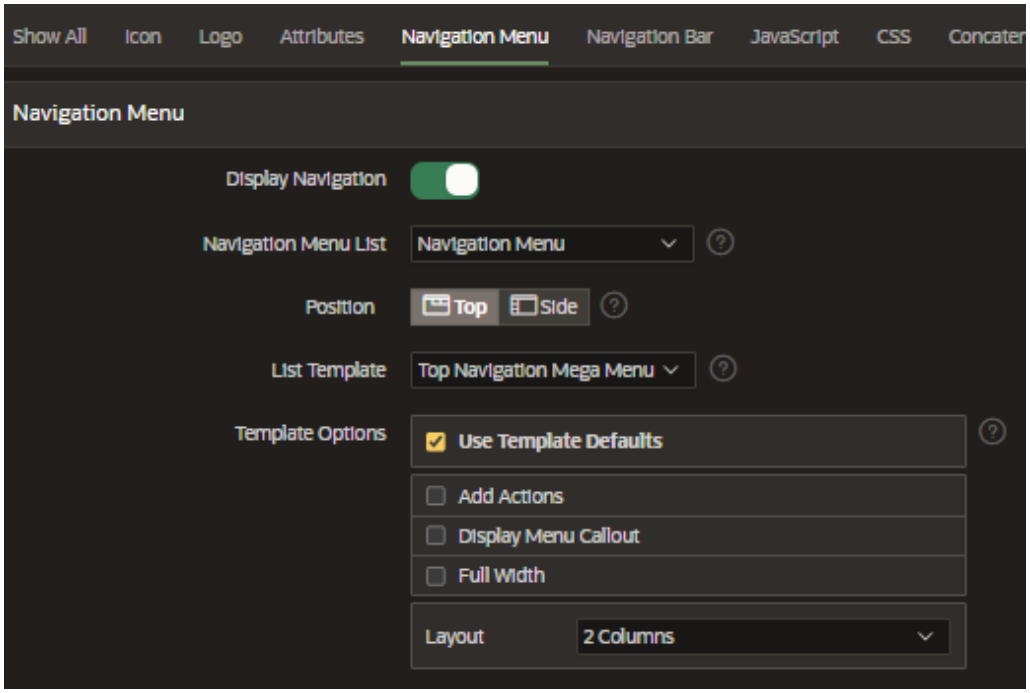
Slika 9.4: Zgornji veliki meni.

Dodatno lahko določimo število stolpcev v meniju.

9.5 Urejanje menijskih seznamov

Elementi menija so običajen seznam, ki ga lahko urejamo. Do seznama lahko dostopamo preko *Shared Components* v navigaciji. V Shared Components nato izberemo *Navigation Menu* ali *List*. Spreminjamo lahko tudi pozicijo in način prikaza menija, tako da v *Shared Components* izberemo

User Interface Attributes. V zavihku *Navigation menu* lahko spremenimo lastnosti, kot prikazuje slika 9.5.



Slika 9.5: Lastnosti navigacijskega menija.

- Display Navigation – vklop / izklop navigacije.
- Navigation Menu List – določitev seznama elementov navigacijskega menija.
- Position – izbira zgornje *Top* in stranske *Side* pozicije navigacijskega menija.
- List Template – izbira predlog prikaza navigacijskega menija.
- Template Options – nastavitev predloge navigacijskega menija.

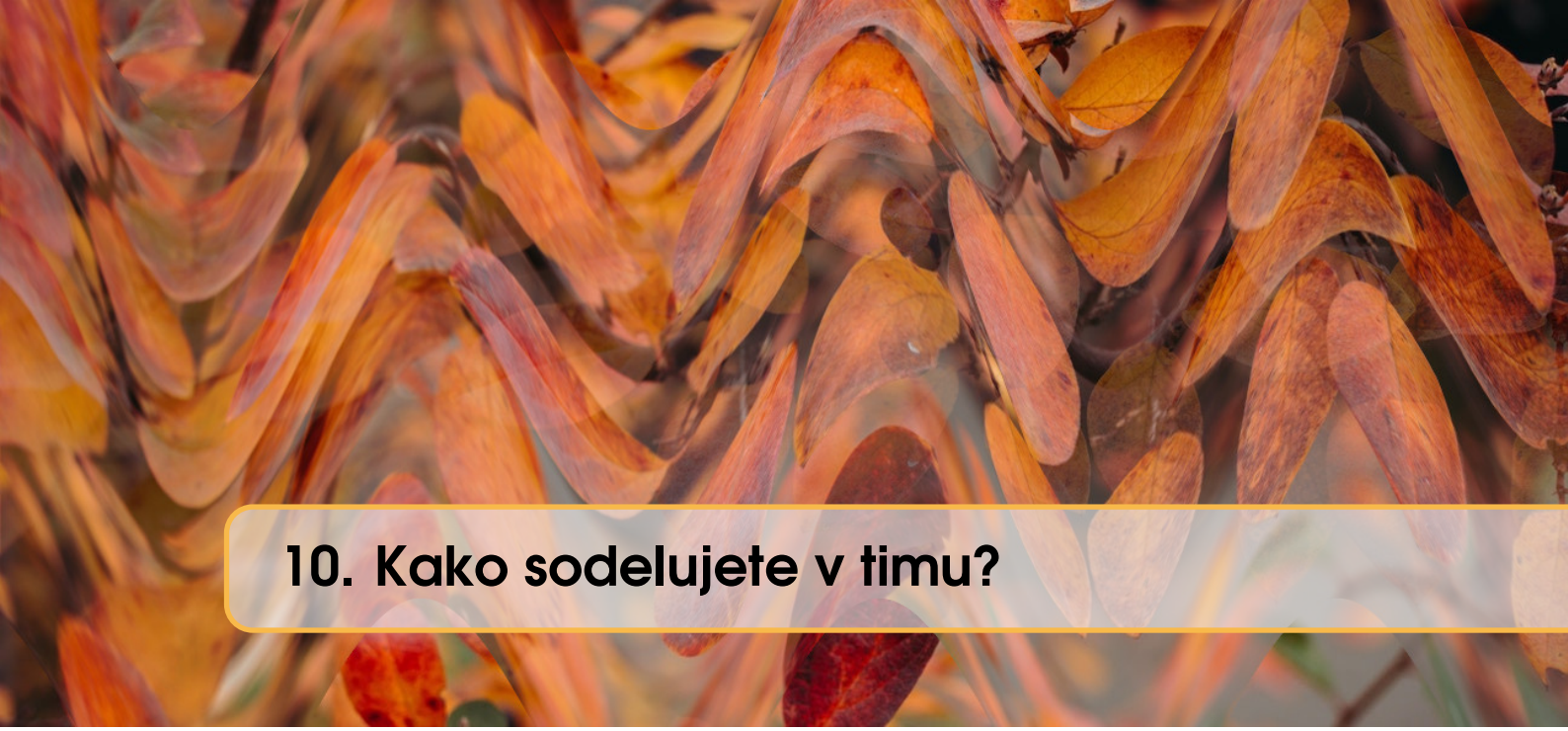
S potrditvijo na *Apply changes* nato zaženemo aplikacijo in preverimo spremembe.

9.6 Vprašanja

1. Najpogosteje je meni v aplikacijah Oracle APEX pozicioniran v zgornji levi del zaslona. Katera vrsta menija je to?
2. Kje lahko spreminjamo lastnosti elementov menija?
3. Katera vrsta menija omogoča prikaz vseh elementov naenkrat, brez da bi morali razširiti elemente za prikaz?

9.7 Odgovori

1. Stranski meni.
2. Posamezni elementi menija sestavljajo seznam, ki ga lahko urejamo podobno kot ostale sezname. Do teh seznamov lahko dostopamo preko *Shared Components* v *Navigation region*.
3. Veliki ali Mega meni omogoča prikaz vseh elementov menija naenkrat, brez dodatnih klikov za razširanje.



10. Kako sodelujete v timu?

P. STANISZEWSKI, M. SOŃTA, A. KIERZKOWSKI; PREVOD R. LESKOVAR

10.1 Sodelovalno ustvarjanje znanja je bistvo razvoja malokodnih rešitev

Ljudje smo naravno programirani za sodelovanje in sodelovanju vedno dajemo prednost (npr. [1]; [3]). Vendar je individualni način sodelovanja bolj različen kot sta palca na rokah posameznika. Programska podpora in platforme za sodelovanje spodbujajo kolektivno izmenjevanje znanja. Glavne značilnosti platform za sodelovanje so:

- produktivnost - posamezniki uporabljajo kolektivno inteligenco za razvoj boljšega izdelka in
- motivacija - proces aktivira ustvarjalne energije in skupnostni pristop k znanju.

10.2 Povezanost preko spleta

Oraclov APEX je platforma, ustvarjena za mreženje in interakcijo pri razvoju aplikacij. Ta pristop utemeljuje definicijo platforme sodelovalnega aktivizma. Vendar ta vrsta sodelovalne izkušnje zahteva "poseben način biti skupaj" ([3]). Del sodelovalne izkušnje je možnost ogleda, komentiranja in soustvarjanja vsebine za skupno ustvarjanje znanja ter spodbujanje ustvarjalnosti in duha z oblikovanjem družbenih odnosov. To je druga dimenzija sodelovalne uporabe APEX-a. Skupnost Oracle APEX ima letno na desetine dogodkov v okviru regionalnih združenj. Primeri:

- APEX World, Nizozemska
- APEX World, APEX Alpe Adria, Avstrija/Hrvaška/Slovenija
- APEX World, APEX Connect, Nemčija
- APEX World, ODTUG Kscope, ZDA

Čeprav je APEX lastniška programska oprema, je Oracle večkrat zatrdil, da bo za vedno ostala brezplačna in tako sledi smernicam spodbujanja duha skupnosti.

Nekateri raziskovalci (npr.[3]) menijo, da nekateri načini skupnega bivanja na spletu ponujajo velike priložnosti in imajo ambicijo doseči zrelost digitalnega aktivizma. Eden od vidikov te zrelosti je tudi pridobitev skupine ljudi, ki še niso usposobljeni za razvojno delo v informatiki.

10.3 Tehnološka podkovanost za sodelovanje

Večina akademskih člankov o APEX-u pokriva primere poslovne uporabe, primere medoddelčnih prizadevanj za ustvarjanje aplikacij, ki predstavljajo preplet človeških virov, družbene odgovornosti,

komunikacij z različnimi deležniki in podobno.

Zaradi tega pristopa, dostopnosti do platforme in širokega nabora izobraževalnih vsebin, je to tudi priložnost za vstop v svet malokodnega programiranja in osveščanje o zmogljivosti orodja APEX. APEX odpira priložnost različnim strokovnjakom, ki niso iz sveta IT, da sodelujejo pri projektih razvoja rešitev in da povečajo svojo tehnološko podkovanost. IT kompetence ustvarjajo bolj zaposljive deležnike za vse, ki sodelujejo v procesu razvoja programske opreme.

Načini sodelovanja je običajno povezan z metodologijo projekta. Tudi, če so malokodne platforme (npr. APEX), zasnovane za zelo hitro doseganje ciljev, običajno potrebujemo več kot enega razvijalca za izvedbo projekta v predvidenem času in s predvidenim finančnim vložkom. V takšni situaciji potrebujemo pravo orodje za pomoč razvijalcem pri sodelovanju, reševanju izzivov in iskanju priložnosti. Glede na obseg in naše zahteve ali zahteve naše organizacije je lahko tehnološka podkovanost povezana z vidiki, kot so:

- standardizacija – želimo zagotoviti, da so vsi elementi aplikacije skladni. Gre za standarde uporabniškega vmesnika, konvencije poimenovanja, tehnike programiranja, slog kodiranja in vse druge standarde, ki so značilni za izbrano tehnologijo. Na primer, v okolju APEX lahko preprosto razmišljate o različnih načinih poimenovanja elementov, procesov ali validacij ali upoštevate standarde pri izbiri številk strani.
- pridobivanje kompetenc – razvoj aplikacije z drugimi razvijalci pomaga pri izboljšanju veščin. Drugi člani tima so morda bolj napredni v vidikih, ki vam niso najbolj všeč. Tudi pri uporabi malokodne platforme kot je APEX, ki teoretično vsakemu uporabniku omogoča, da je celovit (ang. full-stack) razvijalec, lahko najdemo specializacije. Nekateri ljudje se na primer počutijo veliko bolj udobno pri delu z bazo podatkov, drugi s funkcijami APEX-a, tretji pa z oblikovanjem in jeziki vmesnega sloja. Večji tim pomaga članom razviti veščine, ki jih prej niso imeli in povečati splošno kakovost aplikacije.
- čas – naravno je, da lahko več razvijalcev hitreje dostavi aplikacije, kar ima za posledico boljšo učinkovitost tima ali oddelka, odgovornega za določen sistem ali platformo.
- kakovost – večja ekipa lahko izvede postopek pregleda kode in preveri kakovost elementov, ki jih razvijejo člani tima. Poveča se kakovost končnega izdelka in omeji število napak.
- povratne informacije – sodelovanje pomeni tudi komunikacijo med razvijalci in uporabniki. Koristno je imeti informacije uporabnikov o kakovosti aplikacije, prijavljenih napakah ali zahtevah za spremembe.

Oracle APEX kot zrela platforma pokriva različne strategije sodelovanja z vgrajenimi funkcijami, pa tudi z možnostmi, ki jih je mogoče razširiti z zunanjimi orodji, kot so paketi za sodelovanje, aplikacije za upravljanje opravil ali orodja za preverjanje kode.

Tudi, če smo šele začeli svojo pot z APEX, je bistveno vedeti, kako lahko sodelujemo z drugimi uporabniki z vgrajenimi funkcijami, da bo takšno delo bolj produktivno. Lažje bomo obravnavali tipične težave, kot so dodelitve nalog, delo na točno določenih straneh/funkcijah ali nadzor statusa načrtovanega dela.

Najpomembnejši del na samem začetku je komunikacija. Če sodeluje več kot ena oseba, bi morali imeti načine za načrtovanje dela za naslednje dni, obveščanje o statusu ter nadzor nad časom in roki. Nič ni pomembnejšega od rednih srečanj in komunikacije, če kaj ni jasno. Dobro je razmišljati o dnevnih sestankih, tedenskem načrtovanju ali drugih rednih dogodkih. Uporabite lahko tudi katero od znanih metodologij, kot je Scrum, da imate takšne navade zapisane na enem mestu. Samo začnite s tem in našli boste najboljši način sodelovanja za svojo ekipo.

Ko na začetku razmišljate o orodjih za sodelovanje, nehote pomislite na preglednice za spremljanje napredka. Seveda je na trgu veliko namenskih aplikacij za upravljanje opravil, kot so Jira, Monday, ClickUp, Redmine itd., in seveda lahko z naraščajočim številom razvijalcev razmislite o uporabi ene od naštetih. Vendar pa je večino teh dejanj mogoče dobro izvajati s funkcijami, vključenimi v platformo Oracle Application Express.

10.4 Opis funkcij

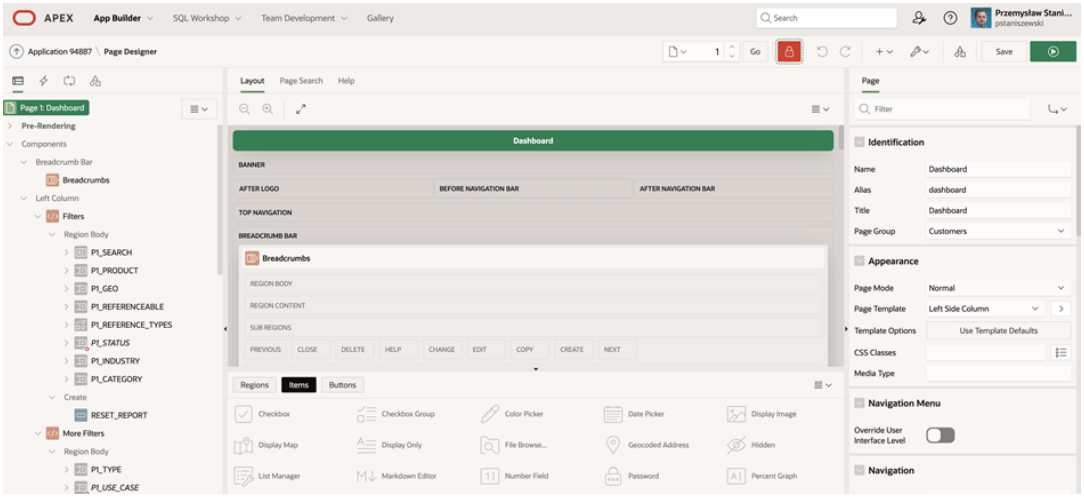
10.4.1 Zaklepanje strani

Funkcija zaklepanja strani preprečuje, da bi dva li več razvijalev istočasno urejali eno stran v aplikaciji. Ko eden od razvijalcev zaklene stran, drugi razvijalci ne morejo spremeniti ničesar na zaklenjeni strani. Razvijalci aplikacije v APEX-u tako učinkovito preprečijo konflikte v skupini razvijalcev. Slika 10.1 prikazuje pregled zaklenjenih in odklenjenih strani. Do te strani pridete, če vstopite v razvojno okolje aplikacije in izberete tabelaričen pregled vseh strani aplikacije.

Page	Name	Alias	Updated	Updated By	Type	Group	Lock	Run
0	0	0	3.1 years ago	-	Global Page	Unassigned		
1	Dashboard	dashboard	9 seconds ago	pstaniszewski@pretius.com	Report	Customers		
2	Customer Details	customer-details	2 years ago	-	DML Form	Customers		
3	Categories	categories	1 seconds ago	pstaniszewski@pretius.com	Interactive Report	Administration		
4	Category Details	category-details	3.1 years ago	-	DML Form	Administration		
5	Customer Types	customer-types	1.6 years ago	-	Interactive Report	Administration		
6	Status Details	status-details	3.1 years ago	-	DML Form	Administration		
7	Maintain Update	maintain-update	2 years ago	-	DML Form	Customers		
8	Contact Types	contact-types	1.6 years ago	-	Interactive Report	Administration		
9	Contact Type Details	contact-type-details	3.1 years ago	-	DML Form	Administration		
10	Validate Customer	validate-customer	2 years ago	-	Report	Customers		
11	Build Options	build-options	1.6 years ago	-	Interactive Report	Administration		
12	Administration	settings	2 years ago	-	Report	Administration		
13	Customer Usage Metrics	customer-usage-metrics	2 years ago	-	Legacy Calendar	Customers		

Slika 10.1: Zaklepanje strani v APEX-u.

Za zaklepanje določene strani kliknite ikono ključavnice na pregledu. V kolikor pa ste že začeli urejati določeno stran, lahko aktivirate funkcijo zaklepanja tudi v tem koraku (slika 10.2).



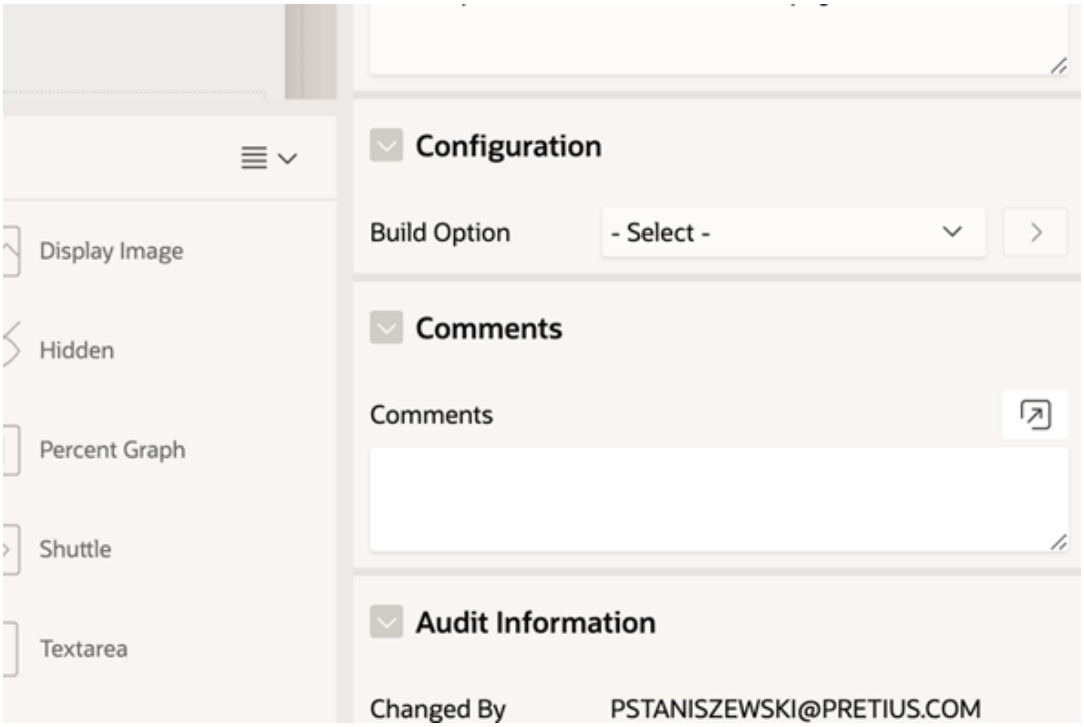
Slika 10.2: Zaklepanje strani, ki jo trenutno urejate.

Pri zaklepanju ali odklepanju strani vedno lahko napišete komentar za druge razvijalce. Običajno v komentar napišete razlog, trajanje, pogoje, ki so privedli do aktivacije te funkcije in podobno.

10.4.2 Komentarji

Razvijalci aplikacije v APEX-u lahko komentirajo strani, regije, polja na strani in mnoge druge elemente strani. Komentarji naj drugim razvijalcev dajo vpogled v kontekst, razlago za odločitev

ali prikazujejo sled spremembam. Z uporabo komentarjev dosežemo boljšo seznanjenost skupine razvijalcev in boljšo komunikacijo v skupini. Slika 10.3 prikazuje dodajanje komentarja.

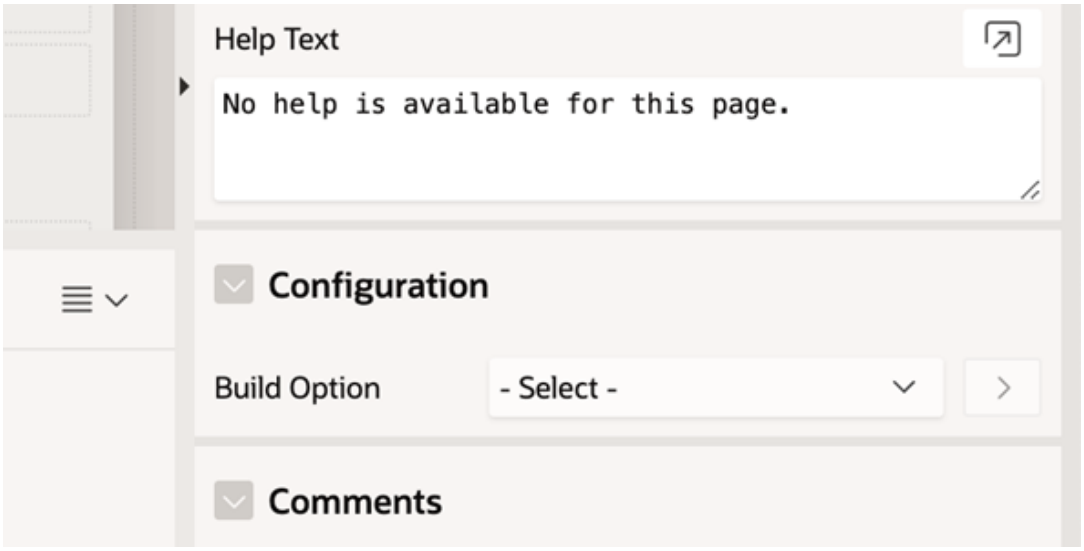


Slika 10.3: Dodajanje komentarja.

Dokumentiranje je pomembno ne le za člane trenutne skupine razvijalcev, temveč tudi za bodoče vzdrževalce. Čeprav malokodno razvojno okolje s številnimi čarovniki nadomesti potrebo po zelo detajnem dokumentiranju, predstavlja dodajanje komentarjev dobro prakso.

10.4.3 Opcije za gradnjo aplikacije

Opcije za gradnjo aplikacije v APEX-u so naslednja funkcija, ki omogoča boljše sodelovanje razvijalcev v skupini. Opcije dovoljujejo nastavitve, ki so namenjene razvojnemu, testnemu ali produkcijskemu okolju. Na ta način lahko enostavneje upravljamo pravilne nastavitve za določeno okolje. Slika 10.4) prikazuje nastavitev za eno stran, ki je lahko v enem okolju aktivna, v drugem okolju pa je označena kot komentar in ni izvedljiva.



Slika 10.4: Določitev opcije strani kot komentirana ali izvedljiva.

Komentiranje se lahko nanaša na celotno stran ali samo na polje oz. element strani. Če določimo, da je stran, polje ali proces komentar, se ta element ne izvozi. Ta situacija se pogosto zgodi takrat, ko je zaradi pogodbenih obveznosti potrebno namestiti aplikacijo, neka stran ali element strani pa še ni do konca razvit. Namesto, da bi stran ali element razvijalec brisal, ga komentira (ga naredi neizvedljivega).

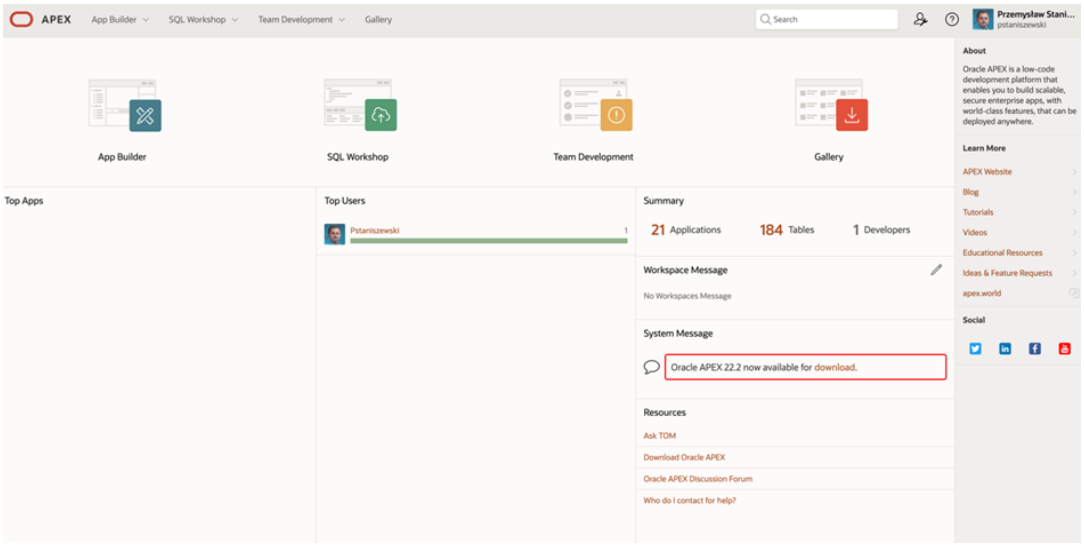
Če razvoj aplikacije teče v delovnem toku nenehne integracije, dostave ali namestitve (ang. CI/CD, continuous integration, continuous delivery, continuous development) je določitev opcij za gradnjo resnično koristna. S tem se boste srečali v profesionalnem razvoju aplikacij z APEX-om.

10.4.4 Razvoj v timu

Oracle APEX ima posebno fukcijo, ki je namenjena podpori projektnega dela v timu razvijalcev - Team Development. Zagotavlja kontrolirano in sodelovalno okolje. Pri projektih razvoja rešitev z APEX ta funkcija omogoča:

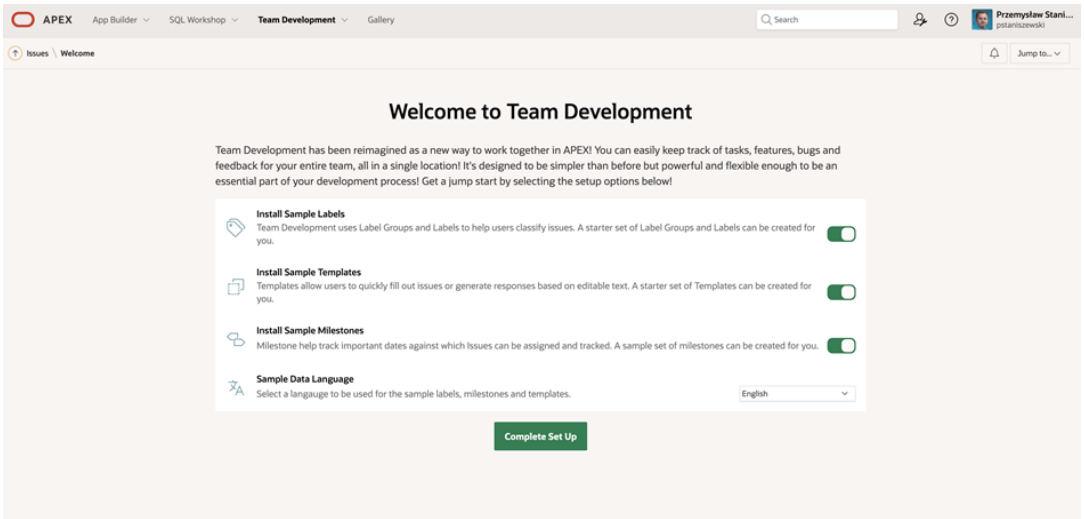
- določitev mejnikov projekta in faze projekta,
- kreiranje, dodeljevanje, spremljanje nalog ter problemov,
- določitev klasifikacije z uporabo skupin oznak in oznak,
- kreiranje predlog za sporočanje težav,
- povezovanje težav z aplikacijo in stranmi,
- komentiranje in diskutiranje o problemih vključno s spremljanjem statusa odpravljanja napak,
- naročanje na različne pojave problemov in
- spremljanje odzivov uporabnikov in odločanje za morebitne popravke in podobno.

Funkcija Team Development je ena od štirih glavnih zavihkov (slika 10.5), ki so na voljo v integriranem razvojnem okolju. Dostopna je od različice APEX 19.2 dalje. V najnovejših različicah ni na voljo takoj, ampak zahteva aktiviranje.



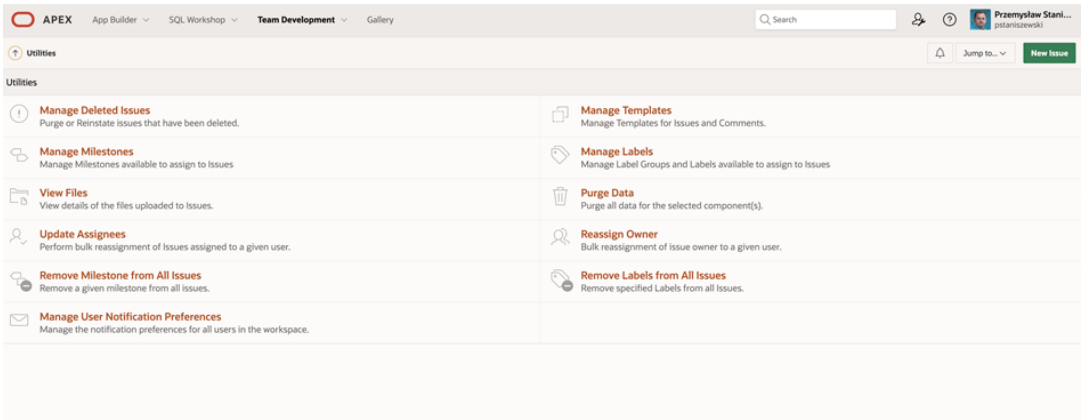
Slika 10.5: Funkcija Team Development.

Za aktiviranje funkcije “Team Development” v glavnem meniju najprej kliknite to izbiro. Ob prvem kliku so odpre namestitvena stran. Če želite, lahko vključite primere težav, mejnikov, predlog in oznak (Issues, Milestones, Templates, Labels). Lahko pa zgolj aktivirate funkcijo (slika 10.6).



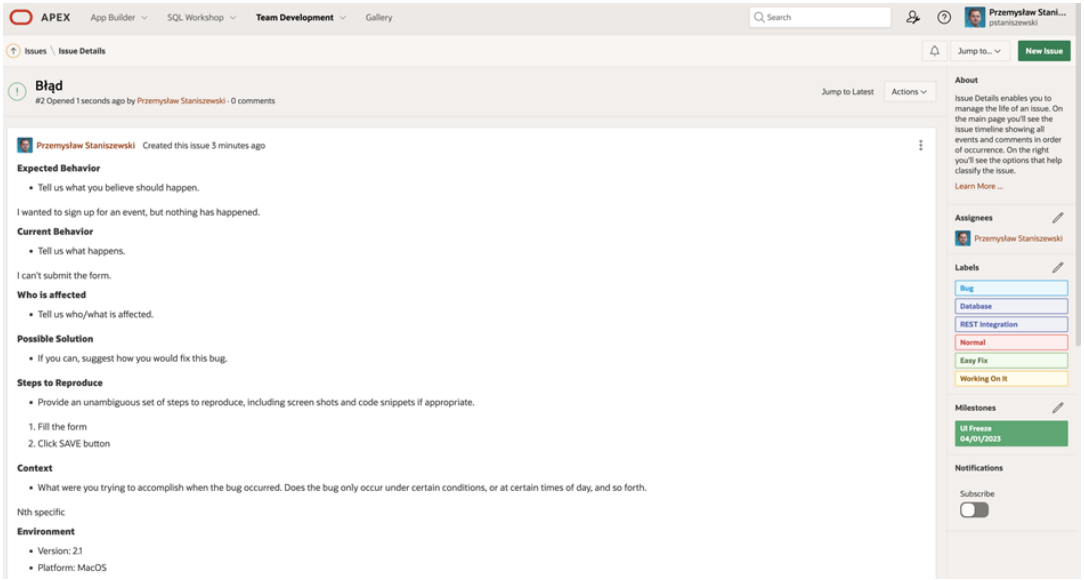
Slika 10.6: Aktiviranje funkcije Team Development.

Praktiki predlagajo, da za funkcijo “Team Development” najprej prilagodite slovar izrazov, ki so pomembni v izvedbi projekta. V razdelku Utilities definirajte vaše težave, projektne faze, mejnike in predloge za tipično komuniciranje v vašem projektu (slika 10.7).



Slika 10.7: Prilagoditev funkcije Team Development.

Najpomembnejši del te funkcije je razdelek Issue (težava), kjer lahko najdete vse prijavljene težave, jih kategorizirate in dodelite določenim osebam. Vsako težavo je mogoče dodeliti enemu ali več razvijalcem, ki bo na voljo za dodajanje komentarjev, razpravo o njej in spreminjanje njenega statusa. Če je potrebno, lahko uporabniki naložijo datoteke in v bistvu vodijo vso komunikacijo v zvezi s težavami s funkcijo "Team Development". Pomembno je tudi, da je vse težave mogoče povezati z določeno aplikacijo in stranjo. Tako je možno enostavno začeti iskanje vzrokov in spremljati povezanost težav (napak) z deli aplikacije. Status težave je preprosta različica delovnega toka odpravljanja težav, ki jo je mogoče najti v številnih sistemih za upravljanje opravil – uporabniki jih lahko prijavijo, označijo kot "v reševanju", se vrnejo k poročevalcu in zaprejo z informacijo o uspehu ali različnih razlogih za zaprtje (slika 10.8).



Slika 10.8: Komunikacija o težavi v funkciji Team Development.

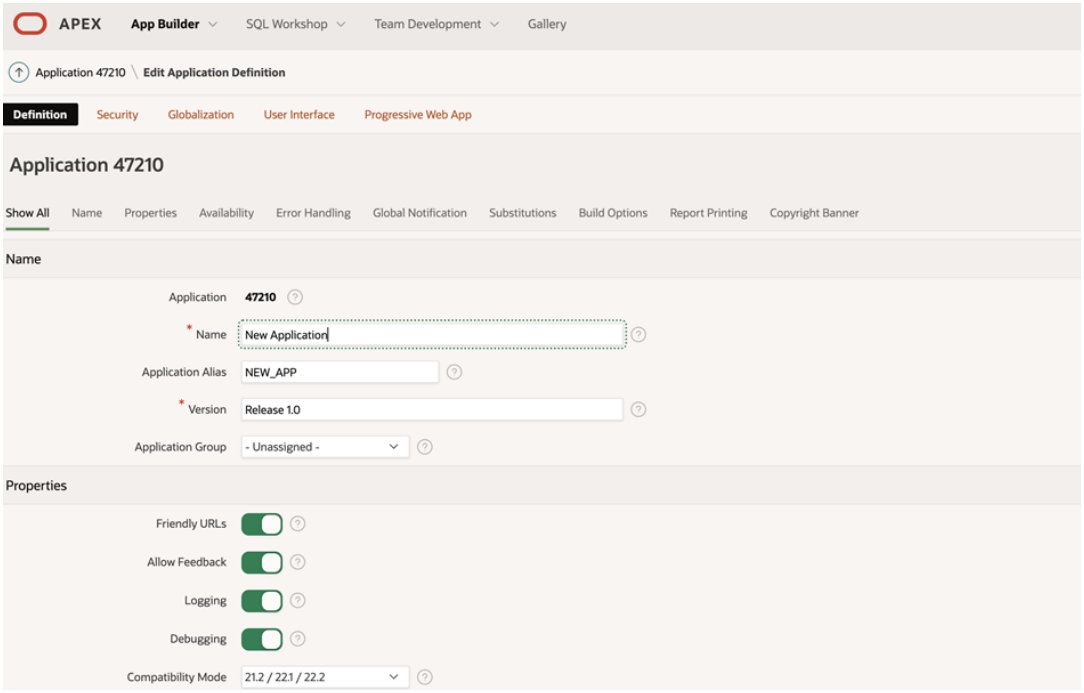
Razvijalci se lahko tudi naročijo komunikacijo o določeni težav. To zagotavlja sporočanje po e-pošti v trenutku, ko se s težavo nekaj dogaja. APEX omogoča razvijalcem, da omejijo število e-poštnih sporočil tako, da jim pošilja kot urne ali dnevne povzetke dogodkov v zvezi s težavo.

Izkušeni razvijalci v APEX-u zelo priporočajo, da tudi začetniki začnejo uporabljati "Team Development", če projektni tim nima drugega sistema za upravljanje opravil ali, če za upravljanje projekta uporabljate kar preglednice.

10.4.5 Odziv

Funkcija odzivanja ima t.i. strani za odzive (Feedback Pages) so vgrajene v Oracle APEX. Namenjene so končnim uporabnikom za sporočanje razvijalcem o svoji izkušnji s funkcionalnostjo in zmogljivostjo aplikacije. Strani za odzivanje je možno prilagajati posamezni aplikaciji. Lahko omogočite sporočanje o napakah, zahtevah za nove funkcije ali splošne odzive oz. vtise.

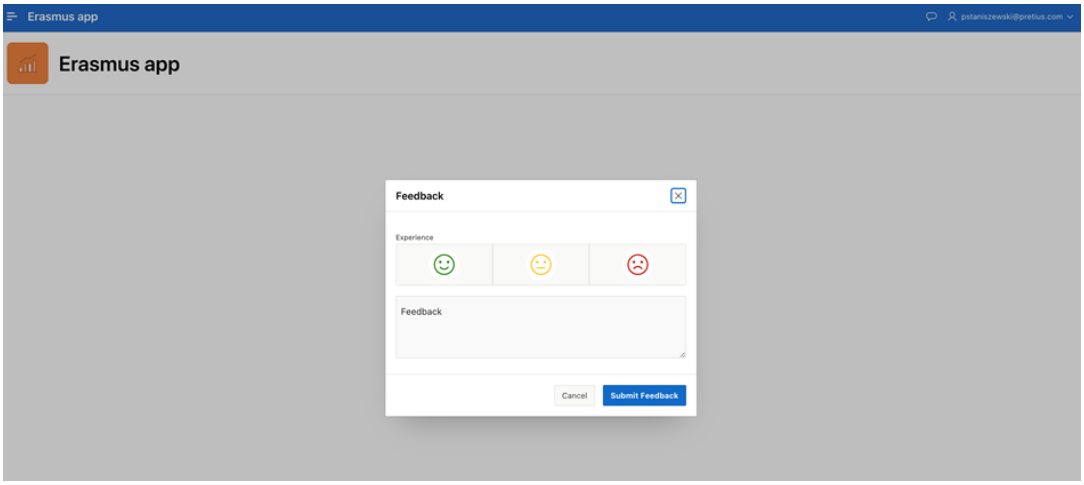
Ključno je pridobiti informacije od končnih uporabnikov in jih obravnavati kot podaljšek razvojne ekipe. Bolj kot tim sodeluje ne le interno, ampak tudi z dejanskimi uporabniki aplikacije, boljšo kakovost aplikacije bomo imeli. Timu omogoča tudi, da aplikacijo prilagodi dejanskim potrebam in pomaga graditi zaupanje med razvojno skupino in uporabniki (glejte sliko 10.9, ang. Allow feedback).



Slika 10.9: Dovoljenje za odzive uporabnikov (v *Application definition*.

Če želite omogočiti strani za odzive, morate med ustvarjanjem aplikacije to označiti. Če aplikacija že obstaja, pojdite na "Edit Application Definition" in označite "Allow Feedback" v razdelku Properties.

Po aktivaciji čarovnik doda namensko ikono v meniju aplikacije. Ko jo uporabnik klikne, lahko pošlje povratne informacije razvijalcu (slika 10.10).



Slika 10.10: Oddajanje odziva v aplikaciji.

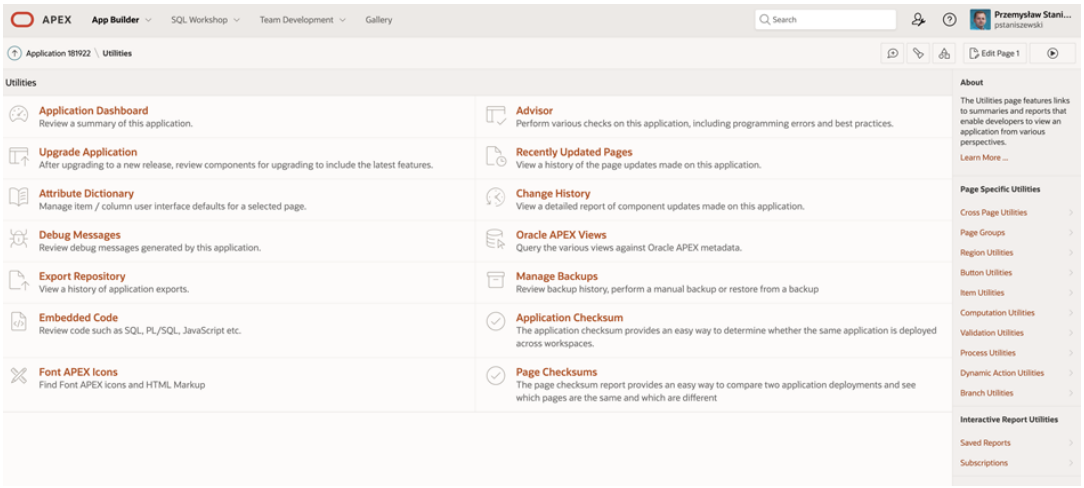
Izgled strani s povratnimi informacijami je mogoče prilagoditi, saj gre za standardno stran APEX-a. Vse informacije, ki jih posreduje končni uporabnik, so na voljo v razdelku Team Development in jih ekipa lahko analizira (možno jim je slediti tudi na ravni posamezne aplikacije v njenem razdelku Administration). Poleg tega razvijalci prejmejo podrobne informacije v zvezi s prijavljenimi povratnimi informacijami (ID aplikacije, številka strani, podatki uporabnika, informacije o njegovem okolju itd.) in jih lahko spremenijo v težavo (Issue).

10.4.6 Standardizacija

Med razvojem aplikacij v timu je zelo pomembno tudi ohranjanje skladnosti z razvojnimi standardi. Kot pri vsakem vidiku sodelovanja je komunikacija ključ do uspeha – na začetku projekta se mora ekipa dogovoriti o načinu razvoja in uporabe komponent APEX. Najpomembnejši vidiki tega vključujejo:

- dogovor o poimenovanju (za elemente, procese, validacije, dinamična dejanja itd.),
- številčenje in poimenovanje strani (t.i. alias),
- dovoljeni elementi (npr. za slovarje uporabljamo samo interaktivne mreže, vsak obrazec mora biti prikazan kot modalna stran),
- standardi UX/UI (npr. videz in občutek tipičnih strani, politika poravnave za različne časovne podatke podatkov, skladni formati podatkov, priporočene ikone za določena dejanja itd.) in
- konsistentnost podatkov (npr. uporaba skupnih slovarjev, skupne komponente ipd.).

APEX ponuja nekaj rešitev za pomoč timom pri standardizaciji. Večino jih je mogoče najti v razdelku “Utilities” na ravni izbrane aplikacije (slika 10.11).



Slika 10.11: Menu “Utilities”.

Najpomembnejši na začetku razvoja so:

- poročila za več strani (Cross-page reports, Page Specific Utilities) – ta omogočajo razvijalcem primerjavo različnih elementov (atributi strani, poročila, gumbi, polja itd.) v aplikaciji in hkratno posodobitev brez ročnega spreminjanja atributov na več mestih .
- zgodovina sprememb/nedavno posodobljene strani – omogoča sledenje dejanjem razvijalcev
- definicija privzetih vrednosti (slovar atributov) – omogoča ustvarjanje skupnega slovarja za ohranjanje doslednega poimenovanja, oblikovanja, tipov podatkov itd. za elemente, ki se ponavljajo na več mestih v aplikaciji. Primer: “Datum posodobitve” lahko obstaja v različnih poročilih ali elementih v aplikaciji, vendar želimo vedno imeti isto oznako in obliko datuma za vsak pojav.

Seveda APEX ponuja številne dodatne funkcije in strategije za ohranjanje doslednosti na ravni baze podatkov in aplikacije. Za tiste, ki bi radi razširili svoje znanje, preberite dokumentacijo APEX in preverite naprednejše možnosti, kot so globalna stran, načrti, privzete nastavitve uporabniškega vmesnika, glavna tema aplikacija in mehanizem naročanja, pogledi APEX-a ali vtičniki (ang. Global Page, Blueprints, User Interface Defaults, Master Theme Application and Subscribe mechanism, APEX Views, Plug-ins).

10.5 Zaključki

Kot zrela razvojna platforma Oracle APEX ponuja številne funkcije za pomoč pri sodelovanju. Razvijalci v malokodnem okolju lahko uporabimo ter profesionalno ustvarimo in vzdržujemo aplikacije, ki temeljijo na APEX tudi brez razširjene platforme ali njene integracije z zunanjimi orodji. Začetniki v APEX in novih strategijah sodelovanja pri programiranju naj začnejo z osnovnimi pristopi, ki so opisani v tem poglavju:

- komunikacija in redni sestanki tima (v živo ali na daljavo – uporabite orodja, kot so Google Meet, Zoom, MS Teams, BBB za video in glasovno sestankovanje ter Slack, Google Chat ali druga aplikacije za hitro sporočanje)
- začnite uporabljati orodja za sodelovanje, npr. APEX Team Development za načrtovanje razvoja in preverjanje napredka
- začnite dokumentirati svojo kodo in standarde - verificirajte jih

Sodelovanje pri razvoju z malokodnim orodjem včasih dojemamo kot pretiravanje, vendar je vredno poskusiti. Od sodelovanja lahko pričakujemo ravno toliko koristi kot pri standardnem, običajnem programiranju. Večino tipičnih strategij sodelovanja, kot so “pregled kode”, “programiranje v parih” ali “skupinsko programiranje” je mogoče uporabiti tudi pri projektih razvoja z malokodnim pristopom, čeprav pregledani element ni koda, temveč proces, stran ali konfiguracija.

Odvisno od obsega projekta ali standardov, ki se izvajajo v organizaciji, boste v določenem trenutku morda morali razširiti svoj proces sodelovanja. Na primer, pri velikih projektih boste morali začeti uporabljati repozitorije izvorne kode, avtomatizirana orodja za gradnjo in namestitev ali programsko opremo, namenjeno testiranju in preverjanju kakovosti vaše kode. Uporaba elementov profesionalnih procesov CI/CD, kot so Git, Docker, Flyway, Liquibase, Jenkins, Gradle, SonarQube, Selenium, Cypress in številnih drugih orodij je možna tudi pri razvoju aplikacij z APEX. V mnogih situacijah pa bodo funkcije, vključene v APEX, dovolj za ustvarjanje aplikacije, ki spremeni igro vam v korist.

10.6 Vprašanja

1. Kaj je funkcija zaklepanja strani in zakaj je pomembna?
2. Kako lahko razvijalci komunicirajo med razvojem aplikacije v APEX-u?
3. Kaj lahko člani razvojnega tima pridobijo z uporabo funkcije Team Development?

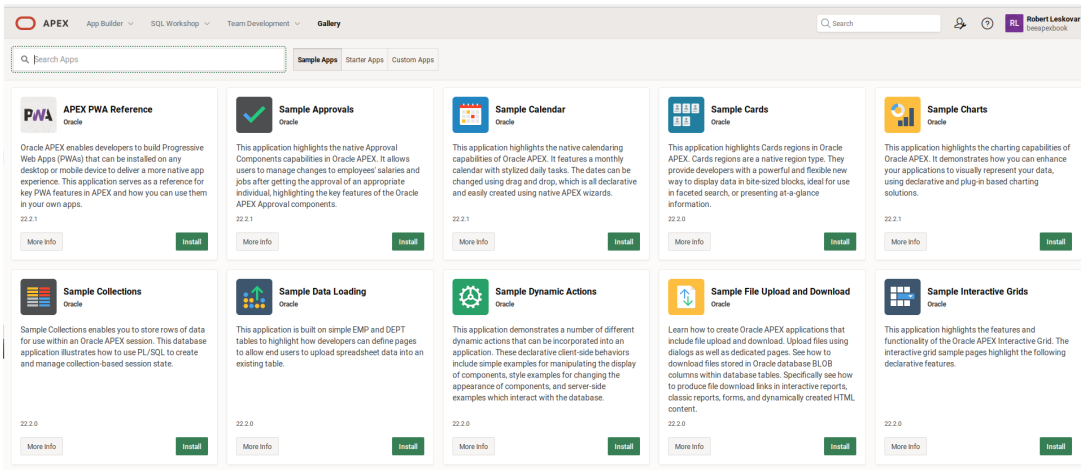
10.7 Odgovori

1. Zaklepanje strani je uporabna funkcija, ker več razvijalcem preprečuje sočasno urejanje iste strani in s tem preprečuje konflikte.
2. Razvijalci lahko med razvojem komunicirajo tako, da dodajajo komentarje stranem, regijam, elementom in mnogim drugim elementom aplikacije. Komentarji se lahko uporabijo za zagotavljanje konteksta za druge razvijalce, razlago razlogov za oblikovalsko odločitev ali preprosto dokumentiranje sprememb v aplikaciji. Z uporabo komentarjev lahko razvijalci bolje komunicirajo med seboj in zagotovijo splošno kakovost programske opreme.
3. Funkcija Team Development omogoča članom skupine upravljanje razvojnih projektov, zlasti: definiranje mejnikov in stopenj projekta, ustvarjanje, dodeljevanje in spremljanje nalog/težav, uporabo klasifikacij z uporabo skupin oznak in oznak, ustvarjanje predlog težav, ustvarjanje nalog ali poročanje dosledni, povezujejo težave z aplikacijo in stranjo, komentirajo in razpravljajo o obstoječih težavah, spreminjajo njihov status in spremljajo napredek faze projekta ter se naročijo na različne vrste težav.

11. Kako izkoristite galerijo aplikacij in vtičnikov?

VJERAN STRAHONJA, DIJANA OREŠKI; PREVOD R. LESKOVAR

To poglavje prikazuje vzorčne/zagonske aplikacije in vtičnike v APEX-u. Vsak delovni prostor v APEX-u je vsebuje te apliacije in vtičnike. Slika 11.1 prikazuje seznam vzorčnih aplikacij.



Slika 11.1: Seznam vzorčnih aplikacij v APEX-u.

Zagonske aplikacije so samostojni, popolnoma neodvisni primeri rešitev, ki zadovoljujejo preproste zahteve in ne terjajo kompleksnih rešitev. Vzorčne aplikacije so namenjene prikazu specifičnih funkcij in učenju. Vtičniki omogočajo uporabnikom dodatno funkcionalnost.

11.1 Kako namestite vzorčne in zagonske aplikacije?

Uporabniki lahko prenesejo vzorčne in začetne aplikacije iz galerije. Če je to omogočil skrbnik delovnega prostora, je mogoče aplikacije namestiti z uporabo povezav v razdelku Gallery.

Za **namestitev** aplikacij iz galerije sledite naslednjim korakom:

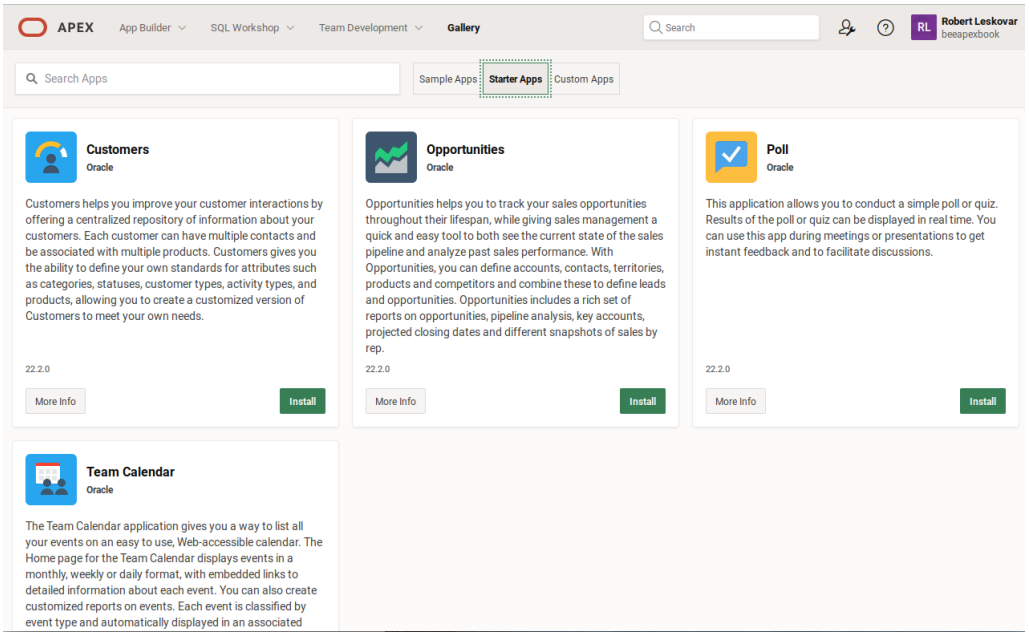
1. prijavite se v delovni prostor
2. izberite **Gallery** in aplikacije se prikažejo na karticah prve strani
3. izberite vzorčne **Sample Apps** ali zagonske **Starter Apps** aplikacije; na karticah se prikažejo vse aplikacije; aplikacije, ki jih še niste namestili imajo gumb Install; že nameščene aplikacije

imajo gumba Run in Remove App
4. izberite **Install App** za želeno aplikacijo
5. neobvezno razširite Advanced Settings
   a. določite ali identifikacijko številko (Application ID) avtomatsko generira APEX ali jo boste vpisali sami
   b. po potrebi izberite shemo, v kateri so tabele (Parsing Schema)
6. kliknite gumb Install Application za dokončanje namestitve

Aplikacijo torej brišete iz vaše galerije s klikom na gumb Remove App icon v razdelku Gallery. Aplikacije, ki jih prenesete iz repozitorija GitHub tega nimajo gumba. V tem primeru jih boste brisali s klikom na povezavo Delete this Application v razdelku App Builder > aplikacija.

11.2 Zagonske aplikacije

Zagonske aplikacije ponujajo samostojne rešitve za izpolnjevanje enostavnih zahtev. Te aplikacije je mogoče uporabiti "iz škatle" ali pa jih izboljšate z lastnimi funkcijami (slika 11.2).



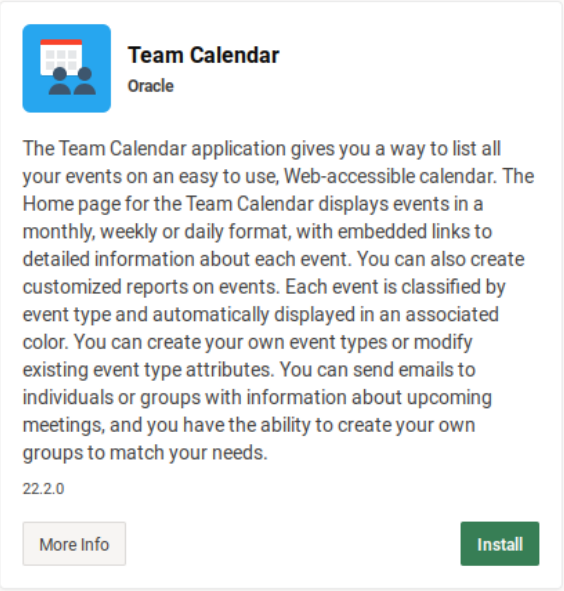
Slika 11.2: Zagonske aplikacije v galeriji.

Primeri zagonskih aplikacij so:

- **Customers** – Z zagotavljanjem konsolidirane baze podatkov o strankah aplikacija Customers omogoča izboljšanje interakcij s strankami. Vsaka stranka se lahko poveže z več kontakti in artikli. Aplikacija omogoča, da sami definiramo kategorije, statuse, vrsto strank, tipe aktivnosti in artiklov. Tako lahko uporabnik aplikacijo prialgodi svojim potrebam.
- **Opportunities** – Ta aplikacija ponuja upravljanje prodaje s hitro in preprosto rešitvijo za ogled trenutnega stanja prodajnega cevovoda in analizo pretekle uspešnosti prodaje. Pomaga vam slediti prodajnim priložnostim skozi njihov življenjski cikel. Potencialne stranke in priložnosti lahko določite tako, da združite račune, stike, regije, blago in tekmece. Priložnosti ponujajo obsežno zbirko informacij o priložnostih, raziskavah načrtov, pomembnih računih, pričakovanih zaključnih datumih in drugih zaznavah prodajnikov.
- **Poll** – Uporabniki lahko s to aplikacijo ustvarijo hitro anketo ali test. Rezultati ankete ali kviza so vidni takoj. To orodje se lahko uporablja za olajšanje mnenjskih raziskav in pridobivanje takojšnjih povratnih informacij med sestanki ali predstavitvami.

- **Team Calendar** - S pomočjo aplikacije Team Calendar lahko uporabniki preprosto upravljajo vse dogodke na spletnem koledarju. Aplikacija je enostavna za uporabo. Domača stran koledarja tima prikazuje dogodke v mesečnem, tedenskem ali dnevnem stilu skupaj z integriranimi povezavami do bolj podrobnosti posameznega dogodka. Poleg tega lahko oblikujete poročila o dogodkih. Vsak dogodek je kategoriziran in samodejno označen s svojo barvo. Uporabniki lahko razvijejo nove tipe dogodkov ali spremenijo atribute že obstoječih tipov dogodkov. Uporabniki imajo možnost, da ustvarijo lastne skupine, ki bolje ustrezajo njihovim potrebam in lahko posameznikom ali skupinam pošljejo e-pošto o prihajajočih sestankih.

Kot primer namestitve navajamo aplikacijo Team Calendar (slika 11.3).

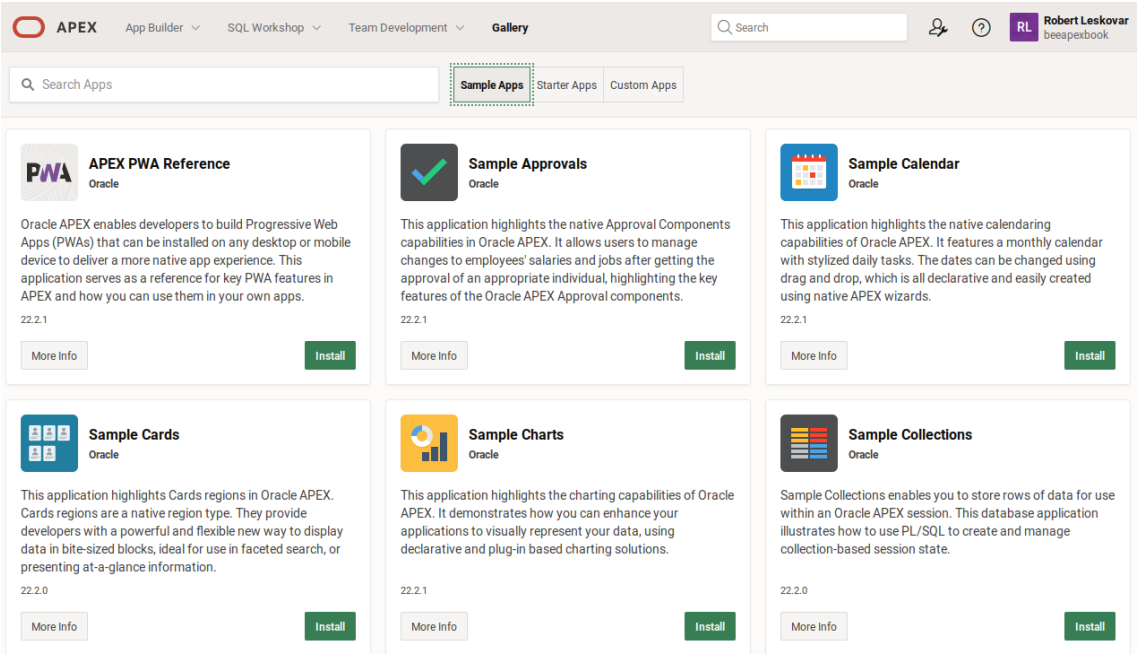


Slika 11.3: Namestitev aplikacije Team Calender.

Kliknite gumb Install za namestitev ali More Info za več informacij.

11.3 Vzorčne aplikacije

Vzorčne aplikacije poudarjajo določene zmogljivosti in delujejo kot priročnik za razvijalce (slika 11.4).



Slika 11.4: Vzorčne aplikacije.

Primeri vzorčnih aplikacij so:

- **APEX PWA Reference** - Za izboljšanje uporabniške izkušnje lahko razvijalci ponudijo tako imenovane Progressive Web Apps (PWA). Te aplikacije uporabniki prenesejo na svoje namizne računalnike ali na mobilne naprave. Prikazane so glavne prednosti PWA z APEX-om.
- **Sample Approvals** - Aplikacija obravnava poslovne procese, ki vključujejo odobravanje (npr. predložitev obračuna poslovne poti, odobritev izplačila stroškov). V aplikaciji je uporabljena komponenta Consent, ki omogoča potrjevanje različnih vlog, prošenj, zahtev in podobno.
- **Sample Calendar** - Aplikacija prikazuje vgrajene funkcije koledarja Oracle APEX. Vključen je mesečni koledar s stiliziranimi dnevnimi opravili. Za posodobitev datumov se lahko uporabi pristop "povleci in spusti", ki je popolnoma deklarativen in se hitro ustvari s čarovniki v APEX-u.
- **Sample Cards** – Ta aplikacija se osredotoča na kartice (Cards) kot vgrajeno komponento APEX-a. Kartice predstavljajo regije, ki jih je možno enostavno opremiti z različnimi podatki. To razvijalcem omogoča izjmne prilagoditve pri iskanju ali predstavitvi podatkov.
- **Sample Charts** – Aplikacija prikazuje funkcije grafikonov Oracle APEX. Pojasnjuje, kako lahko uporabite deklarativna orodja in orodja za grafikone, ki temeljijo na vtičnikih, da izboljšate svoje aplikacije, tako da vizualno prikazujejo vaše podatke.
- **Sample Collections** – Aplikacija uporabnikom omogoča shranjevanje vrstic podatkov v vzorčne zbirke za uporabo v seji Oracle APEX. Prikazano je ustvarjanje in upravljanje stanja seje na podlagi zbirke z uporabo PL/SQL.
- **Sample Data Loading** - Aplikacija uporablja dve tabeli: EMP (zaposleni) in DEPT (oddelki). Prikazujejo kako lahko razvijalci oblikujejo strani, ki uporabnikom omogočajo dodajanje podatkov preglednice v tabelo, ki že obstaja.
- **Sample Dynamic Actions** – Aplikacija prikazuje vrsto dinamičnih dejanj (ang. Dynamic actions), ki jih je mogoče vključiti v aplikacije. Ta deklarativna dejanja na strani odjemalca (spletnega brskalnika) vključujejo preproste primere za spreminjanje načina prikaza komponent, slogovne primere za prilagajanje videza komponent in primere na strani strežnika za interakcijo z bazami podatkov.

- **Sample File Upload and Download** – S to aplikacijo se lahko naučite uporabljati vgrajeno funkcijo APEX-a za nalaganje in prenos datotek. Za nalaganje datotek uporabite namenske strani in pogovorna okna. Oglejte si postopek za prenos datotek iz stolpcev tipa v tabelah baze podatkov. Zlasti preglejte kako narediti povezave za prenos datotek v obrazcih, interaktivnih poročilih, tradicionalnih poročilih in dinamični vsebini HTML.
- **Sample Interactive Grids** – Aplikacija demonstrira funkcije Oracle APEX v interaktivnih obrazcih (ang. Interactive Grid).
- **Sample Maps** – V tej aplikaciji so na voljo številni primeri ogledov koordinatnih podatkov na zemljevidu. Uporabljene so funkcije toplotnega zemljevida (ang. heat map), črt, poligonov ali oznak zemljevida. Zmogljivosti Oracle Spatial (na voljo v vsaki bazi Oracle) je mogoče zlahka integrirati z regijo zemljevida v APEX-u za izvajanje "Iskanja znotraj razdalje", "Iskanja najbližjega soseda" ali drugih prostorskih analiz.
- **Sample Master Detail** – Aplikacija prikazuje zmožnosti Oracle APEX pri upravljanju obrazcev, ki vsebujejo glav in podrobnosti (npr. račun, dobavnica). V aplikaciji so na voljo štiri možne postavitve glavne strani s podrobnostmi. Prvi dve postavitvi uporabljata interaktivne mreže, ki jih je mogoče urejati, za prikaz glave in podrobnosti na eni strani. Zadnji dve postavitvi predstavljata glavo in podrobnosti na dveh straneh z uporabo kombinacije interaktivnih mrež z možnostjo urejanja, obrazcev, starinskih poročil in modalnih pojavnih oken.
- **Sample Reporting** – Aplikacija prikazuje funkcije poročanja v Oracle APEX. Z uporabo SQL lahko uporabniki deklarativno sestavijo interaktivna poročila, interaktivne mreže, poročila o fasetiranem iskanju, poročila s karticami in klasična poročila.
- **Sample REST Services** – Aplikacija prikazuje, kako uporabljati Oracle APEX za povezavo z zunanjimi storitvami REST. Primeri v tej aplikaciji demonstrirajo filtriranje podatkov, dodajanje številčenja strani in izdelavo preprostega tabelaričnega poročila o podatkih storitve REST.
- **Sample Trees** – V tej aplikaciji se lahko naučite uporabite poizvedb v SQL za izdelavo drevesne strukture. Prikazano je več postavitev drevesne strukture.
- **Universal Theme Reference** - Aplikacija uporabnikom predstavi univerzalno temo, ki omogoča preprost način krmarjenja med številnimi predlogami, možnostmi predlog in slogi tem. Primeri prikazujejo, kako lahko uporabniki hitro upravljajo postavitev strani in ustvarijo dih jemajočo aplikacijo.

11.4 Vtičniki, ang. Plug-ins

Vtičniki omogočajo hitro razširitev ogrodja Application Express z vrstami elementov po meri, vrstami regij, procesi in dinamičnimi dejanji. Ko so enkrat definirane, se komponente, ki temeljijo na vtičnikih, ustvarijo in vzdržujejo podobno kot standardne komponente APEX-a. Vtičniki omogočajo razvijalcem, da ustvarijo zelo prilagojene komponente za izboljšanje funkcionalnosti, videza in uporabniške izkušnje. Vtičniki uporabnikom omogočajo dodajanje dodatnih funkcij aplikacijam APEX, ki niso vgrajene v platformo. Skupnost APEX je razvila široko paleto različnih vtičnikov, ki so uporabni za razširitev obstoječih ali za dodajanje novih funkcij. Obstaja na stotine vtičnikov za Oracle Apex. Običajno jih razvščamo v pet skupin:

- Dinamična dejanja
- Regije
- Postavke
- Procesi
- Preverjanje pristnosti

APEX ponuja različne vtičnike znotraj vsake kategorije. Vtičniki omogočajo deklarativno dodajanje novih funkcij vaši aplikaciji in s tem izboljšanje vgrajenih vrst. Ker so vtičniki namenjeni ponovni uporabi, jih lahko razvijalci vključijo v druge aplikacije in delijo s skupnostjo razvijalcev v APEX-u. Vtičnike je možno izvoziti in uvoziti v delovnih prostorih. Koraki za implementacijo vtičnika so:

- dodajte vtičnik v delovni prostor vaše aplikacije ali ga ustvarite od začetka
- če želite uporabiti vtičnik, uredite ali ustvarite tip dinamičnega dejanja, element, regijo, proces ali avtorizacijsko shemo.
- zaženite svojo aplikacijo, da preverite funkcionalnost vtičnika..

Osrednje mesto za razvijalce za skupno rabo in prenos vtičnikov je repozitorij v Oracle Technology Network.

- Če si želite ogledati repozitorij za vtičnike, pojdite na stran Shared Components:
  - kliknite App Builder na glavni strani delovnega prostora.
  - izberite aplikacijo
  - izberite Shared Components na domači strani urejevalnika aplikacije
- pojavi se stran z deljenimi komponentami
- izberite Plug-ins v menuju Other Components
- kliknite gumb View Plug-in Repository

Prikazan je repozitorij vtičnikov. Do vtičnika dostopate takole:

- pojdite na stran Shared Components
  - kliknite App Builder
  - izberite aplikacijo
  - izberite Shared Components
- izberite Plug-ins v menuju Other Components. Zavihek Plug-ins je izbran, ko je stran naloženaPrikazani so vsi vtičniki.

Ustvarjanje vtičnikov vključuje naslednje korake:

- izberite stran Shared Components
  - kliknite App Builder
  - izberite aplikacijo
  - izberite Shared Components
- izberite Plug-ins v menuju Other Components
- klinite gumb Create; zažene se čarovnik za izdelavo vtičnikov
- kliknite gumb Next po tem, ko ste izbrali metodo za kreiranje vtičnika
- določite Name:
  - Name – vpišite naziv vtičnika
  - Internal Name - vpišite interni naziv vtičnika, ki mora biti unikatno v aplikaciji
  - Type - določite vrsto vtičnika;
  - Category - kategorija
- določite Source:
  - PL/SQL Code - vpišite kodo anonimnega bloka v PL/SQL za generiranje strani, preverjanje, izvajanje in obdelavo povratnih klicev tega vtičnika
  - Do not validate PL/SQL code - če želite samo razčleniti kodo PL/SQL med izvajanjem, izberite to možnost, sicer se vtičnik zgradi, ko je koda razčlenjena
- določite Callbacks (povratne klice), konfigurirajte atribute. Tip vtičnika določa atribute, ki so prikazani. Označite atribute, ki jih ima vtičnik. Na voljo je pomoč za izbiro atributov.
- določite User Interfaces:
  - Desktop
  - Mobile
- določite Standard Attributes
- določite Information:
  - Version - vpišite verzijo vtičnika
  - About URL - vpišite URL, na katerem avtor vtičnika ponuja več informacij
- Kliknite gumb Create Plug-in. Zgoraj navedeno velja za verzijo APEX 22.1. Novejša verzija 23.1 omogoča še enostavnejše kreiranje vtičnika.

Uvoz vtičnika izvedete takole:

- izberite stran Shared Components
  - kliknite App Builder
  - izberite aplikacijo
  - izberite Shared Components
- izberite Plug-ins v menuju Other Components
- izberite Import; prikažejop se vtičniki
- določite Specify File:
  - Import file - vnesite pot ali kliknite ime datoteke, v kateri je vtičnik
  - File Type - izberite vtičnik.
  - File Character Set - določite nabor znakovuvožene datoteke
  - kliknite Next.
- kliknite Next pri For File Import Confirmation
- kliknite Install Plug-in za namestitev.

Izvoz vtičnika izvedete takole:

- izberite stran Shared Components
  - kliknite App Builder
  - izberite aplikacijo
  - izberite Shared Components
- izberite Plug-ins v Other Components
- pri Tasks izberite Export Plug-in
- na strani Export Plug-in
  - Application - določite aplikacijo za izvoz
  - Plug-in - izberite vtičnik
  - File Format - izberite format izvoza
  - izberite Export
- izberite Export.

V nadaljevanju je prikazan primer vtičnika, ki se nanaša na posodobljeno vzorčno aplikacijo za koledar in vtičnik za dinamično dejanje. Ta primer vtičnika lahko uporabite za oblikovanje dni v novem elementu izbirnika datuma. Uporabniki ga lahko onemogočijo, dodajo opise orodij ali razrede vsakemu dnevu v dnevni mreži izbirnika datumov. Za uporabo tega vtičnika ni predpogojev. Kako ga uporabljati?

- ustvarite aplikacijo
- uvozite vtičnik v Shared Components - Plug-in
- dodajte nov izbirnik datuma
- dodajte dinamično dejanje ob Page Load in kot dejanje izberite uvoženi vtičnik
- nastavite URL oblike standarda iCalendar ali vnesite poizvedbo SQL za oblikovanje datumskega zapisa
- izberite Affected Element pri želenem izbirniku datuma in zaženite stran

To poglavje predstavlja nekaj prednosti vtičnikov. Vgrajene funkcije APEX-a ne morejo zadostiti vsem željam različnih uporabnikov in to vrzel zapolnjujejo vtičniki. Vtičniki omogočajo preprostejše dodajanje določenih funkcij aplikacijam in prihranijo čas razvijalcu. Posledično so stroški razvoja nižji. Poleg tega vtičniki uporabnikom omogočajo prilagajanje. Večina vtičnikov omogoča vklop in izklop določenih nastavitev. Odstranjevanje vtičnika je preprost postopek in ga izvedete, če je to potrebno.

11.5 Vprašanja

1. Kaj so vzorčne aplikacije in kakšne so njihove prednosti?
2. Kaj so zagonske aplikacije in kakšne so njihove prednosti?
3. Kakšne so prednosti vtičnikov?

11.6 Odgovori

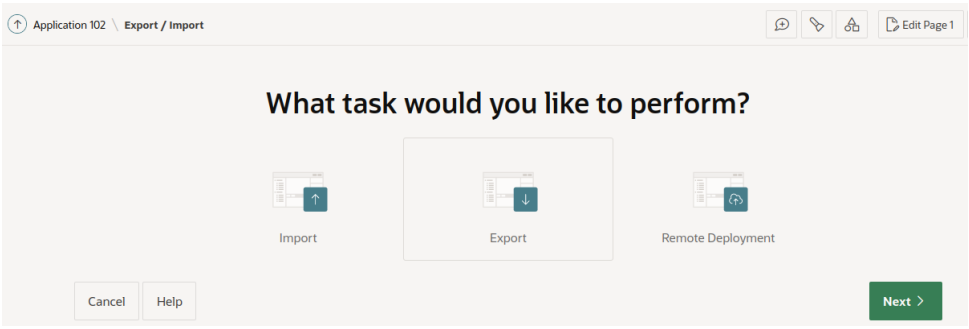
1. Vzorčne aplikacije so ustvarjene za poudarjanje določenih zmožnosti in delujejo kot priročnik za razvijalce za uporabo določene funkcije.
2. Zagonske aplikacije ponujajo samostojne rešitve točk, ustvarjene pa so za izpolnjevanje enostavnih zahtev, ki ne zahtevajo kompleksnih pristopov.
3. Vtičniki razvijalcem omogočajo ustvarjanje visoko prilagojenih komponent za izboljšanje funkcionalnosti, videza in uporabniku prijaznosto aplikacijo. Vtičniki uporabnikom omogočajo dodajanje dodatnih funkcij aplikacijam APEX, ki niso vgrajeni v platformo.

12. Kako upravljate paketne in večjezične aplikacije?

R. LESKOVAR, U. RAJKOVIČ, A.BAGGIA; PREVOD U. RAJKOVIČ

12.1 Aplikacija in paketna aplikacija

V poglavju 5 smo predstavili, kako ustvariti prvo aplikacijo kot razvijalec začetnik. Za prenos aplikacije v nov delovni prostor je treba aplikacijo izvoziti in nato uvoziti. Čarovnik za izvoz v APEX-u zapiše besedilno datoteko, polno ukazov SQL v: a) razvijalčev računalnik ali b) v oddaljeni delovni prostor prek storitve REST (glejte sliko 12.1).



Slika 12.1: Čarovnik za uvoz in izvoz aplikacije.

Na voljo je več možnosti za izvoz, kot so razdelitev v več datotek, izbira nabora znakov za datoteke, izbira ali zavrnitev javnih ali zasebnih poročil in podobno. V večini primerov začetnik preprosto pusti, da APEX uporabi privzete nastavitve. Nato se razvijalec prijavi v nov delovni prostor in zažene čarovnika za uvoz, da naloži predhodno izvoženo besedilno datoteko. Za zagon aplikacije v novem okolju mora razvijalec ustvariti vsaj prazne tabele z enakimi imeni, kot so bila uporabljena v prvotnem delovnem prostoru. Tako se definicije tabel, pogledov, prožilcev, zaporedij, funkcij, postopkov in paketov ne prenesejo z aplikacijo, če uporabimo čarovnika za izvoz. Tudi sami podatki se ne prenesejo na ta način. Prenesejo pa se strani in vgrajena logika.

V poglavju 11 smo predstavili aplikacije, ki jih je Oracle vključil v APEX. Te aplikacije so razvrščene v dve skupini: a) aplikacije za produktivnost in b) vzorčne aplikacije. Aplikacije za produktivnost so namenjene specifičnim poslovnim potrebam. Vgrajene funkcije bolj ali manj v celoti izpolnjujejo zahteve, ki jih postavljajo realni poslovni problemi. Vzorčne aplikacije prikazujejo različne funkcionalnosti. Obe vrsti aplikacij lahko spodbudita razvijalca, da jih prilagodi

in zadosti poslovnim potrebam. Takšne aplikacije predstavljajo primere paketnih aplikacij. Opazite lahko, da te aplikacije v večini primerov nameščajo tabele, zaporedja, prožilce, funkcije, postopke, pakete in vpisujejo podatke v tabele.

Najkrajša definicija **paketne aplikacije** pravi, da gre za vnaprej pripravljeno aplikacijo, ki jo je mogoče namestiti in konfigurirati v delovnem prostoru Oracle APEX. Poraja se vprašanje: "Kako lahko razvijalec ustvari po meri pripravljeno aplikacijo?"

Naredimo to korak za korakom:

- opredelite področje uporabe
- ustvarite in izvedite skripto za definiranje tabel
- ustvarite in izvedite skripto za vstavljanje vzorčnih podatkov
- ustvarite aplikacijo
- izvedite prilagoditve v aplikaciji
- pripravite zapakirano aplikacijo in jo preizkusite v novem delovnem prostoru

V prvem delu tega poglavja bomo predvsem uporabljali čarovnike v APEX-u. Nato bomo uporabili preproste stavke SQL INSERT, ki jih čarovniki v APEX-u ne podpirajo. Za ustvarjanje stavkov INSERT lahko bralec uvozi podatke v preglednico in nato izvozi stavke INSERT z drugimi orodji (npr. SQL Developer ali TOAD). Prilagoditve bodo prikazane kot ustvarjanje seznamov vrednosti (ang. lists of values - LOV) in vključevanje le-teh v ustvarjeno stran aplikacije.

V drugem delu je prikazano ustvarjanje pakirane aplikacije, njen izvoz in uvoz v nov delovni prostor.

V tretjem delu poglavja so podana navodila za izdelavo večjezične aplikacije. Kot dodatno študijsko gradivo so na voljo vse tri izvožene aplikacije in videoposnetek, ki podrobno prikazuje postopek razvoja.

12.2 Aplikacija

12.2.1 Področje uporabe

Aplikacija je namenjena upravljanju delovnih mest in kompetenc za kadrovsko službo. V podjetju je lahko nekaj sto delovnih mest. Eno delovno mesto lahko vključuje določeno podmnožico vseh znanih kompetenc. Zahtevano raven kompetenc za vsako delovno mesto predpiše kadrovska služba. Ravni usposobljenosti so opisne: pomagati, uporabljati, obvladati, prilagajati in inovirati. Ideja je torej razviti aplikacijo za pripravo in upravljanje profilov delovnih mest. Aplikacija uporablja tri tabele: JOB (tabela delovnih mest), COMPETENCE (tabela kompetenc) in JOB_COMPETENCE (vmesna tabela, ki povezuje zahtevane kompetence z izbranimi delovnimi mesti) z minimalnim številom podatkovnih polj (zaradi preprostosti in jedrnatosti). Vsaka tabela ima primarni ključ, tabela JOB_COMPETENCE pa ima dva tuja ključa za povezavo vsake vrstice z vrsticami tabel JOB in COMPETENCE. Tuji ključi zagotavljajo skladnost podatkov, in sicer vrstica v tabeli JOB_COMPETENCE lahko obstaja le, če v tabelah JOB in COMPETENCE obstajajo vrstice z istimi primarnimi ključi. Da zagotovimo doslednost, so imena tabel opremljena s predpono “CH12_”.

12.2.2 Ustvarjanje tabel

Tabele lahko ustvarimo v APEX-u

- brez znanja jezika za opisovanje podatkov s klikom v brskalniku predmetov (ang. Object browser). Ponazoritev tega postopka z veliko podatkovnih polj bi zavzela veliko prostora;
- s poznavanjem jezika za definicijo podatkov DDL v SQL Workshop > SQL Commands z vnosom sintaktično pravilnih ukazov;
- z osnovnim znanjem o modeliranju podatkov (glejte poglavje 2) in Quick SQL.

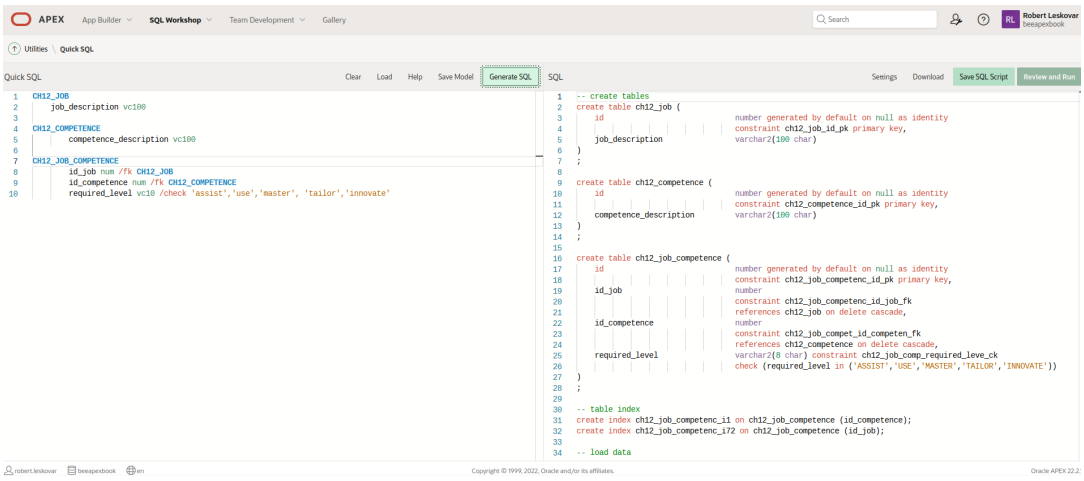
Da zagotovimo širše razumevanje in jedrnatost, bomo uporabljali Quick SQL (Quick SQL poiščite pod SQL Workshop > Utilities > Quick SQL). Vstavljeni bodo naslednji stavki:

```
CH12_JOB
  job_description vc100

CH12_COMPETENCE
  competence_description vc100

CH12_JOB_COMPETENCE
  id_job num /fk CH12_JOB
  id_competence num /fk CH12_COMPETENCE
  required_level vc10 /check 'assist','use','master', 'tailor','innovate'
```

Po vnosu zgornjega besedila in pritisku na gumb **Generate SQL** se prikaže naslednji zaslon (glejte sliko 12.2):

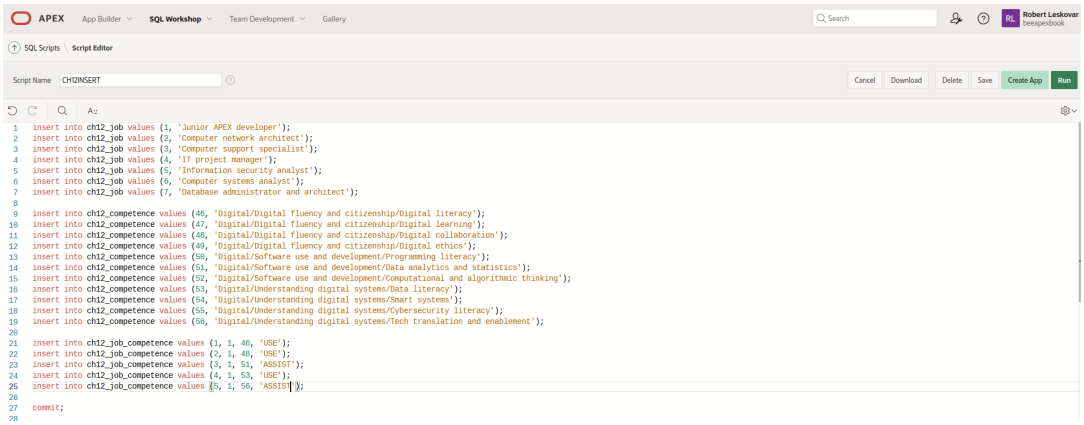


Slika 12.2: Pretvarjanje Quick SQL v ukaze SQL.

Nato kliknite gumb **Save SQL Script** in skripto poimenujte CH12CREATE. Skripto lahko sedaj izvedemo. Vrnite se v Scripts, poiščite ikono Run v vrstici, kjer je prikazana skripta CH12CREATE. Kliknite ikono **Run**. Ustvarijo se tri tabele in trije indeksi. Tabele lahko preverite v brskalniku objektov Object browser.

12.2.3 Vstavljanje podatkov

V programu APEX izberite SQL Workshop > SQL Scripts > Create. Zdaj je čas, da napišete nekaj ukazov SQL INSERT za vnos podatkov v tabele. Tabela CH12_JOB bo vsebovala sedem vrstic, od katerih bo vsaka opisovala drugo delovno mesto. Tabela CH12_COMPETENCE bo imela enajst vrstic z opisom enajstih digitalnih kompetenc, ki jih opredeljuje McKinseyjeva DELTA. Vstavili bomo pet kompetenc za delovno mesto "Junior Apex developer". Skripto poimenujmo CH12INSERT in jo napišimo (glejte sliko 12.3).

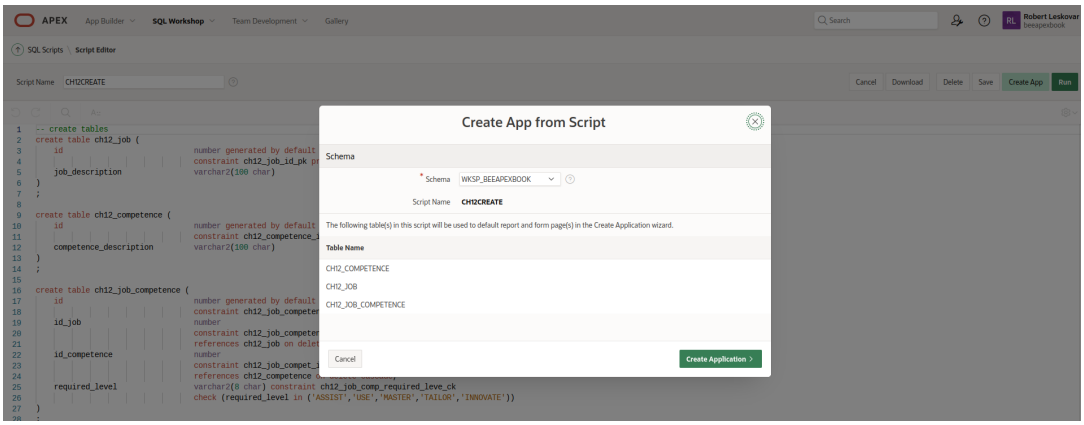


Slika 12.3: Skripta CH12INSERT vstavi podatke v tri tabele in izvede transakcijo.

Kliknite gumb **Create**. Zaženite skripto CH12INSERT. Vrnite se v Scripts in kliknite ikono **Run** v vrstici, kjer je prikazana skripta CH12INSERT. Kliknite gumb **Run now**. Ustvari se štiriindvajset vrstic. Vsebino tabel lahko preverite tudi v pregledovalniku predmetov Object browser.

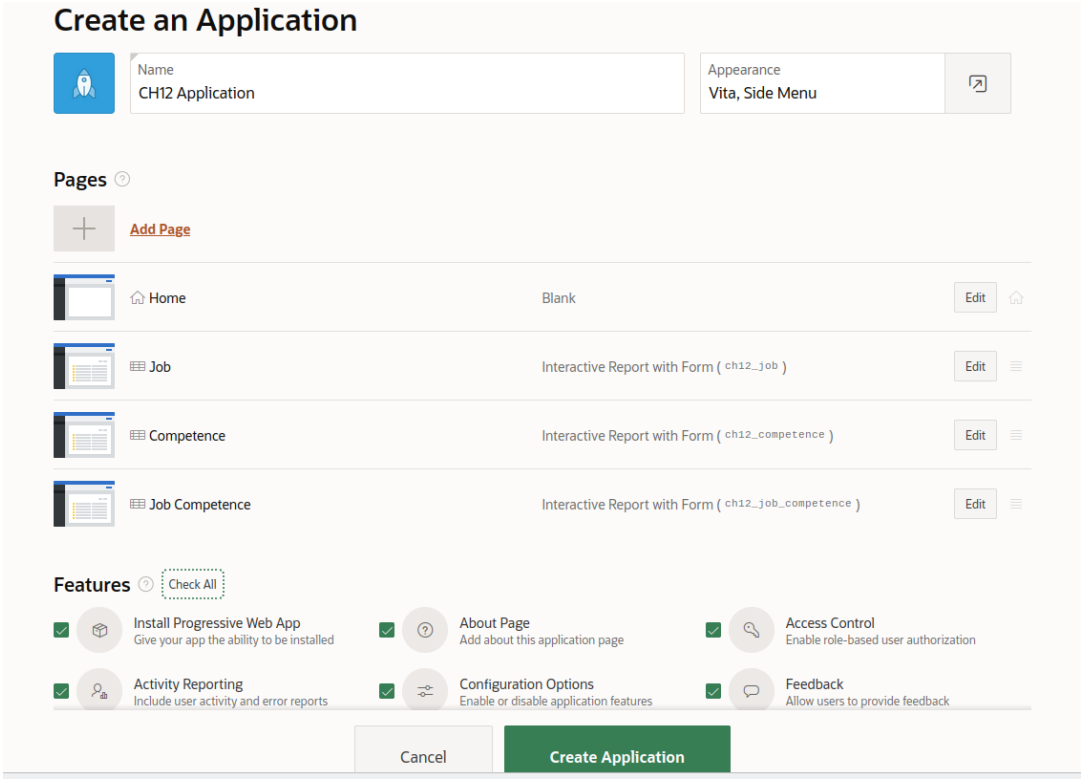
12.2.4 Ustvarjanje aplikacije

V programu APEX pojdite v SQL Workshop > Scripts in uredite skripto CH12CREATE. Kliknite gumb **Create App** (glejte sliko 12.4). Opazite lahko, da so na seznamu tri tabele.



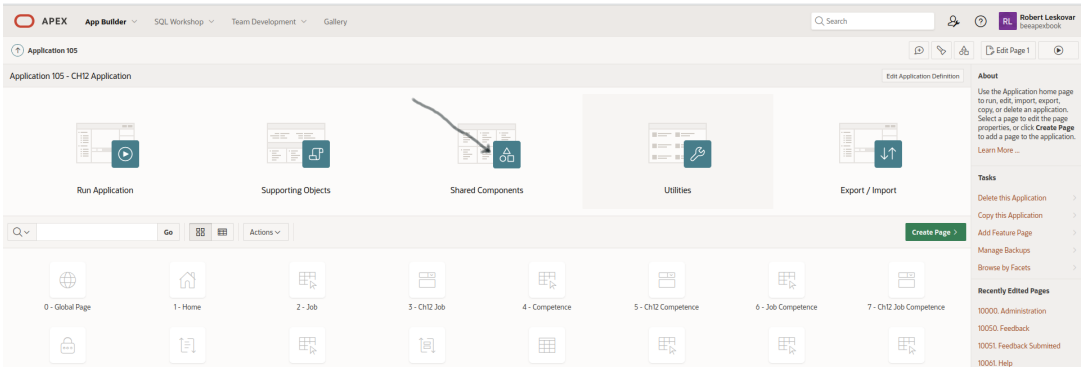
Slika 12.4: Ustvarjanje aplikacije iz skripte.

Kliknite gumb Create Application. Vnesite ime aplikacije (npr. CH12 Application), kliknite Check all features in nato gumb Create Application (glejte sliko 12.5).



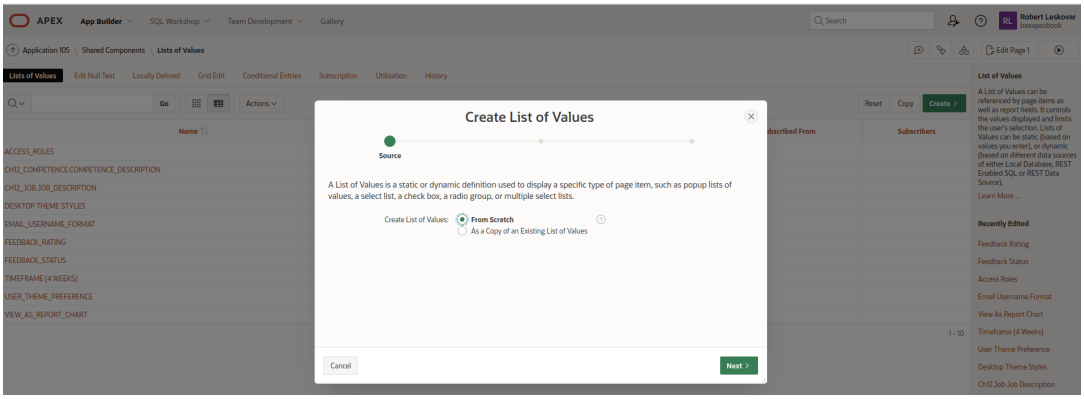
Slika 12.5: Izbira imena aplikacije in lastnosti.

Po končanem generiranju lahko zaženemo aplikacijo. V novejših različicah APEX generator ustvari tudi dva seznama vrednosti (CH12_COMPETENCE.COMPETENCE_DESCRIPTION in CH12_JOB.JOB_DESCRIPTION), skrajšano LOV. Seznam vrednosti (LOV) so komponente na strani, kjer je prikazana ena vrednost, ki jo uporabnik razume, medtem ko stran dejansko obdeluje ustrezen primarni ključ. Na primer: na obrazcu strani se prikaže priimek zaposlenega, ustrezna identifikacijska številka zaposlenega pa se shrani, ko je pritisnjen gumb za potrditev. Ustvarjeni elementi LOV temeljijo na tabelah, zato ni potrebno predhodno znanje SQL. V terminologiji APEX se imenujejo dinamični, saj bi sprememba v tabeli ustvarila nov seznam vrednosti. Ustvarjene LOV lahko najdete v razdelku Shared Components > List of Values, kjer lahko določite njihove nastavitve. V tem poglavju se bomo naučili, kako ustvariti sezname vrednosti na podlagi preproste poizvedbe SQL in kako ustvariti sezname vrednosti na podlagi statičnih vrednosti. Z uporabo seznama vrednosti (LOV) na straneh aplikacije bomo naredili aplikacijo bolj odporno na napake in bolj prijazno do uporabnika. Premaknite se na Shared Components (glejte sliko 12.6).



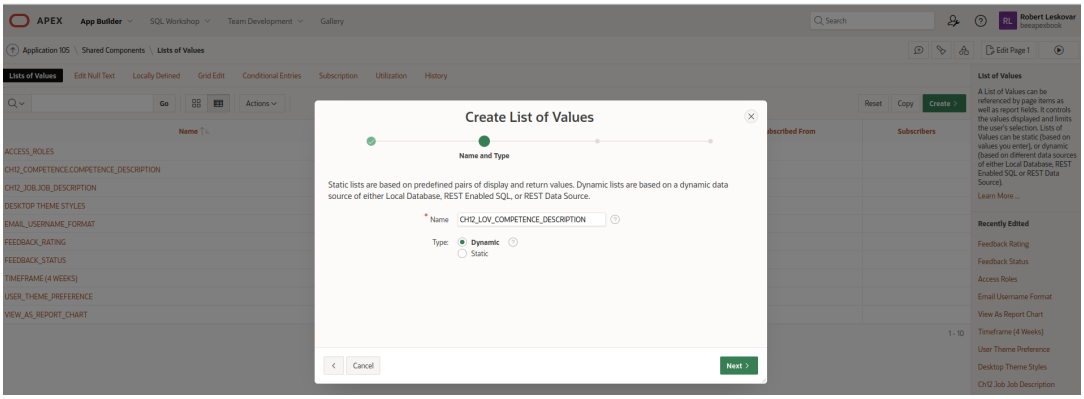
Slika 12.6: Izbira skupnih komponent.

V razdelku Shared Components izberite List of Values (v razdelku Other Components) in kliknite gumb Create. Seznam vrednosti bomo ustvarili od začetka (glejte sliko 12.7).



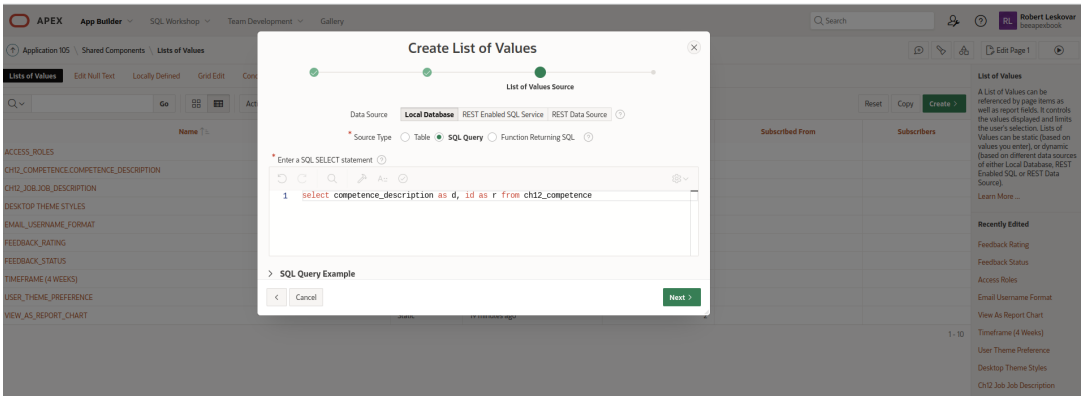
Slika 12.7: Ustvarjanje seznama vrednosti od začetka.

Poimenujte seznam vrednosti kot CH12_LOV_COMPETENCE_DESCRIPTION in izberite dinamični tip Dynamic type (glejte sliko 12.8)



Slika 12.8: Opredelitev imena in vrste: CH12_LOV_COMPETENCE_DESCRIPTION je dinamičen.

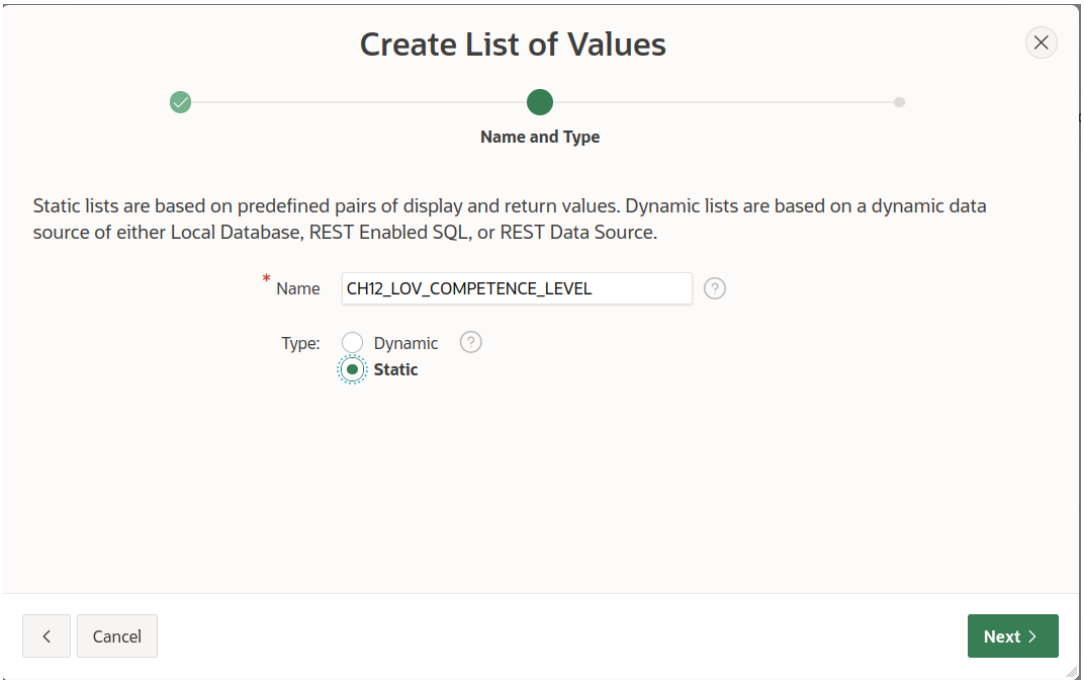
Kot vrsto vira izberite poizvedbo SQL (SQL query) in vnesite poizvedbo “select competence_description as d, id as r from ch12_competence” kot stavek SQL select (glejte sliko 12.9).



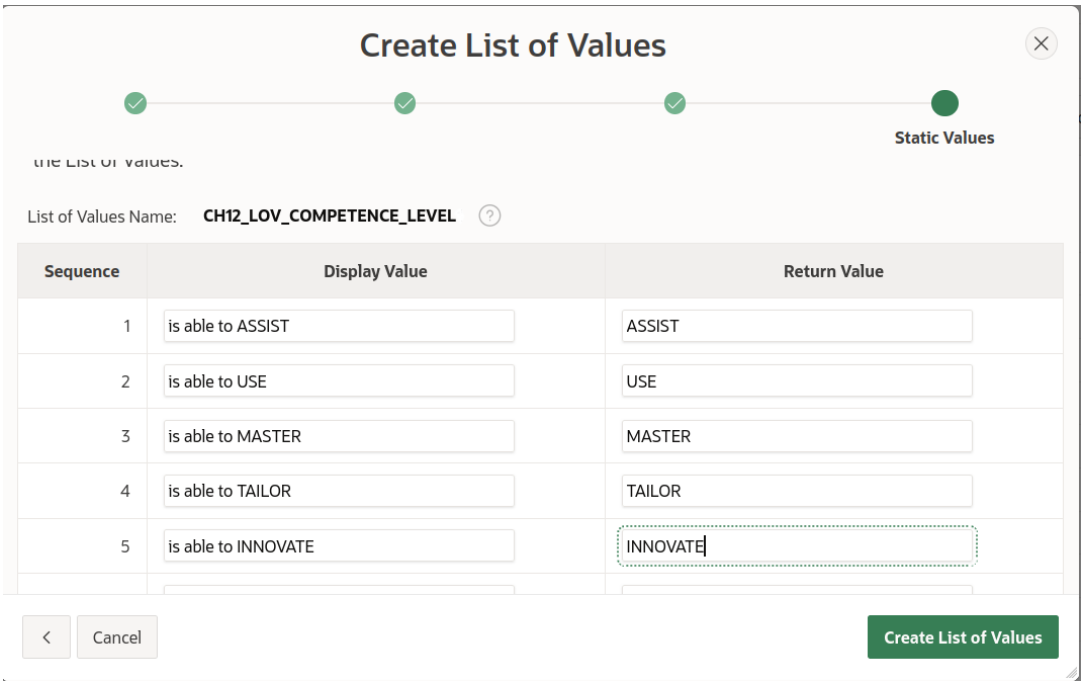
Slika 12.9: Vnos ukaza SQL SELECT.

Naslednji LOV bo statičen, poimenovan CH12_LOV_COMPETENCE. Ta seznam vrednosti bo omogočil prikaz opisa kompetenc (stolpec za prikaz, ang. display column) na strani obrazca, medtem ko bo vrnil ID (stolpec za vrnitev, ang. return column).

Naslednji seznam vrednosti bo statičen, poimenovan CH12_LOV_KOMPETENCE_LEVEL in ustvarjen, kot je prikazano na slikah 12.10 in 12.11.

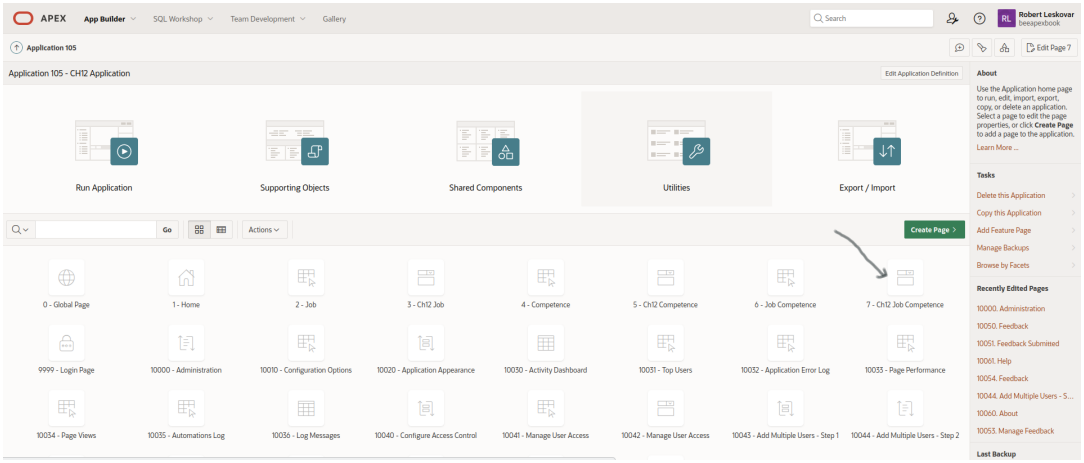


Slika 12.10: Določimo ime in tip seznama vrednosti: CH12_LOV_COMPETENCE_LEVEL je statična (ang. static).



Slika 12.11: Določimo ime in tip seznama vrednosti: CH12_LOV_COMPETENCE_LEVEL je statična.

Pojdite na strani z aplikacijami (glejte sliko 12.12). Izberite stran 7 (Ch12 Job Competences).



Slika 12.12: Izberite stran 7 (Ch12 Job Competences).

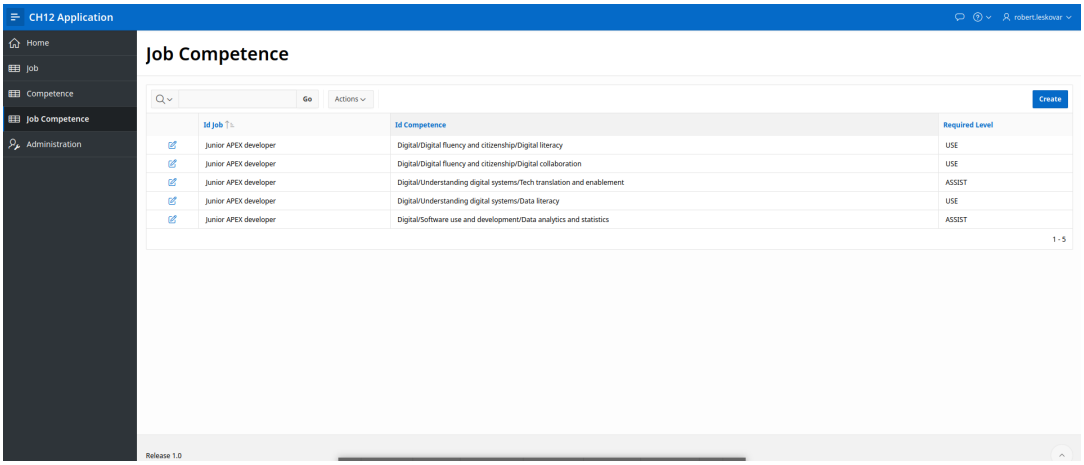
Spremenite element strani P7_ID_COMPETENCE:

- izberite “Select list” v Identification
- izberite “Shared components” v seznamu vrednosti (List of values)
- seznam vrednosti je CH12_LOV_COMPETENCE_DESCRIPTION List of values

Spremenite element strani P7_REQUIRED_LEVEL:

- izberite “Select list” v Identification
- izberite “Shared components” v seznamu vrednosti (List of values)
- seznam vrednosti je CH12_LOV_COMPETENCE_LEVEL List of values

Shranite stran 7, pojdite na stran 6 in zaženite aplikacijo. Izgled je prikazan na sliki 12.13.

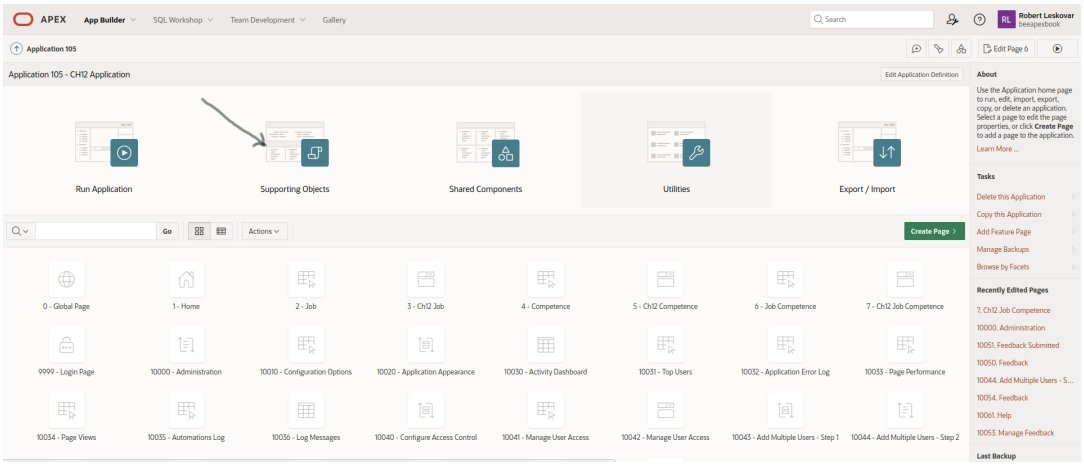


Slika 12.13: Izberemo stran 7 (Ch12 Job Competences).

Zdaj uredite prvi vnos. Na obeh seznamih so pričakovani podatki.

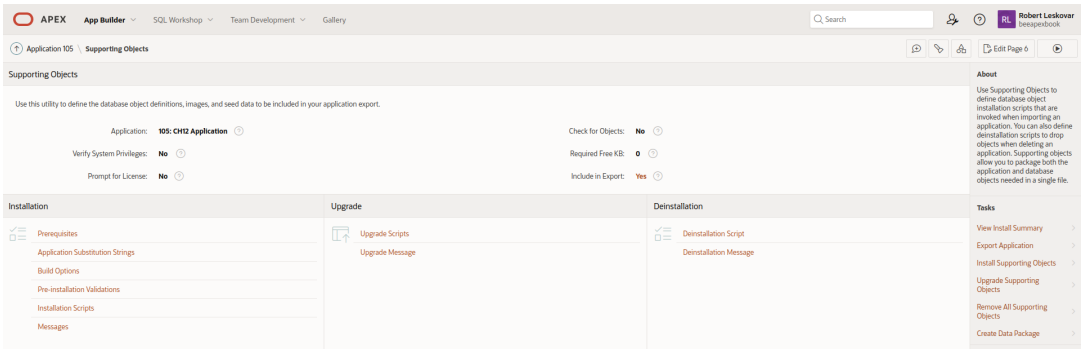
12.3 Paketna aplikacija

Ker smo že ustvarili nekaj skript (CH12CREATE, CH12INSERT), bomo paketno aplikacijo ustvarili zelo hitro. Pojdite v aplikacijo in izberite Supporting Objects (glejte sliko 12.14).



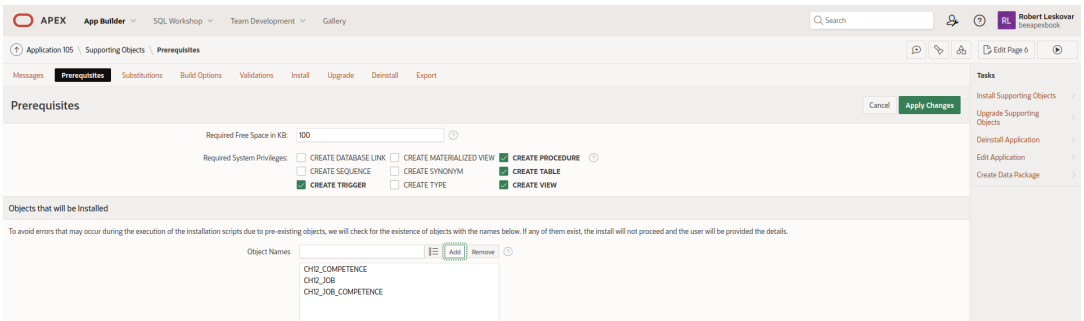
Slika 12.14: Izberemo Supporting Objects.

Zdaj lahko nastavimo lastnosti namestitve (predpogoji, nadomestni nizi, možnosti gradnje, potrditve pred namestitvijo, skripte in sporočila za namestitev), nadgradnjo (skripte za nadgradnjo, sporočilo o nadgradnji) in odstranitev aplikacije (skripte za odstranitev, sporočilo o odstranitvi). Primer vstopa v nastavitve je prikazan na sliki 12.15).



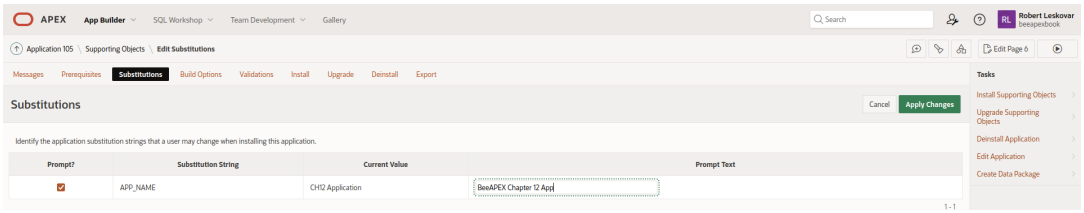
Slika 12.15: Nastavljanje predpogojev.

Za zaščito pred nenamernim brisanjem obstoječih podatkov bomo uporabili preverjanje obstoja treh tabel (glejte sliko 12.16). Kliknite gumb Apply changes.



Slika 12.16: Nastavitev preverjanja obstoja treh tabel.

V vrstici Application Substitution String bomo vprašali, ali naj bo ime aplikacije “BeeAPEX Chapter 12 App” (glejte sliko 12.17).



Slika 12.17: Nastavitev poziva za preimenovanje aplikacije.

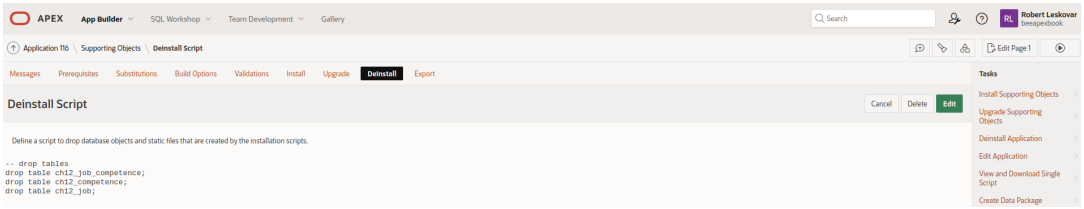
Kliknite gumb Apply changes.

Za paketno aplikacijo bomo uporabili tudi naslednjo nastavitev v Supporting Objects:

- predpogoji (glejte sliko 12.16)
- deli nizov za zamenjavo aplikacije (glej sliko 12.17)
- namestitvene skripte: ker smo že pripravili skripte CH12CREATE in CH12INSERT, jih bomo uporabili (glejte sliko 12.18)
- skripte za odstranitev aplikacije (glejte sliko 12.19)
- nastavljanje sporočil: pozdravno sporočilo na “Chapter 12: ...”, sporočilo o licenci na “CC BY”, sporočilo o uspešni namestitvi na “Success!”, sporočilo o neuspešni namestitvi na “Failure”, sporočilo ob odstranitvi aplikacije “Application CH12 deinstallled”

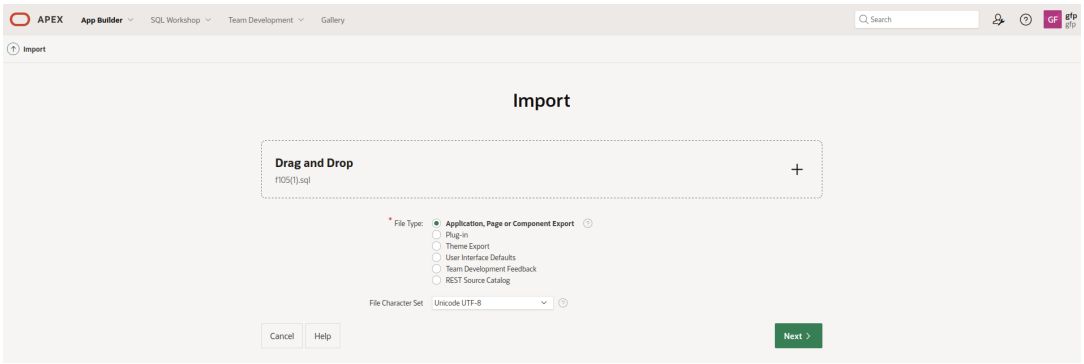


Slika 12.18: Nastavitev namestitvenih skript.



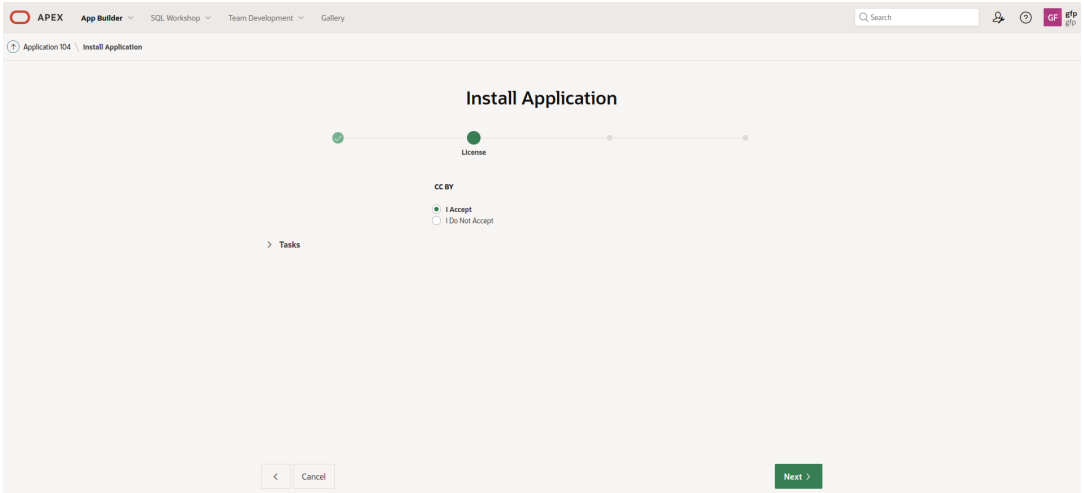
Slika 12.19: Nastavitev skript za odstranjevanje aplikacije.

Zdaj izvozite zapakirano aplikacijo s čarovnikom (glejte sliko 12.1). Sprejmite vse nastavitve izvoza, ki jih predlaga APEX. Izvoženo aplikacijo shranite na računalnik in si zapomnite ime datoteke. Prijavite se v nov delovni prostor. Kliknite čarovnika za uvoz v novem delovnem prostoru in povlecite ime datoteke v območje povleci in spusti (glejte sliko 12.20).

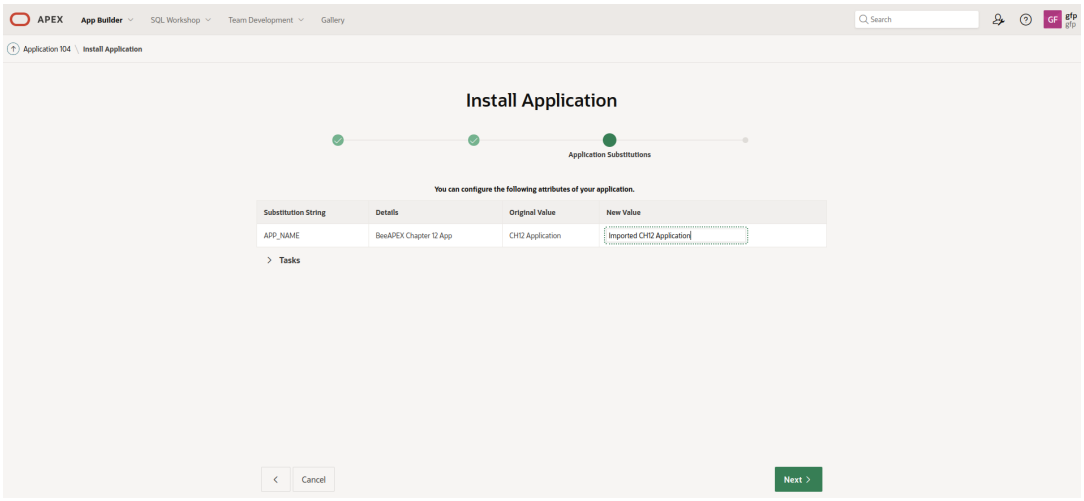


Slika 12.20: Uvoz aplikacije v drug delovni prostor.

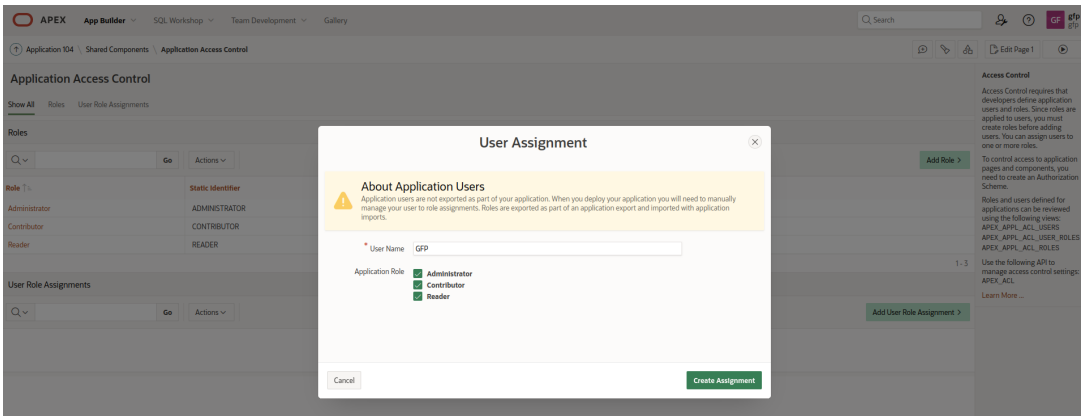
Nadaljujte s klikom na gumb Next (tudi za potrditev uvoza datotek) in kliknite gumb Install application. Z gumbom Next potrdite namestitev podpornih objektov, sprejmite licenco (glejte sliko 12.21), preimenujte aplikacijo v "Imported CH12 Application"(glejte sliko 12.22) in kliknite gumb Install. Aplikacije še ne zaženite, ker morate posebej nastaviti dovoljenja za novega uporabnika. Uporabniki in uporabniška dovoljenja se iz varnostnih razlogov ne izvozijo (opomba: za prenos uporabnikov in dovoljenj bi se lahko poglobili v API APEX in napisali skripte za namestitev in odstranitev). Pojdite do skupnih komponent (Shared components) v uvoženi aplikaciji. izberite nadzor dostopa do aplikacije in trenutno prijavljenemu uporabniku dodelite ustrezna dovoljenja. V našem primeru bomo uporabniku APEX dodali vloge administratorja GFP, sodelavca in bralca (glejte sliko 12.23). Potrdite s klikom na gumb Add Assignement. Sedaj zaženite aplikacijo kot uporabnik z novimi pooblastili.



Slika 12.21: Licenčna pogodba.



Slika 12.22: Preimenovanje uvožene aplikacije.



Slika 12.23: Dodajanje vloge uporabniku.

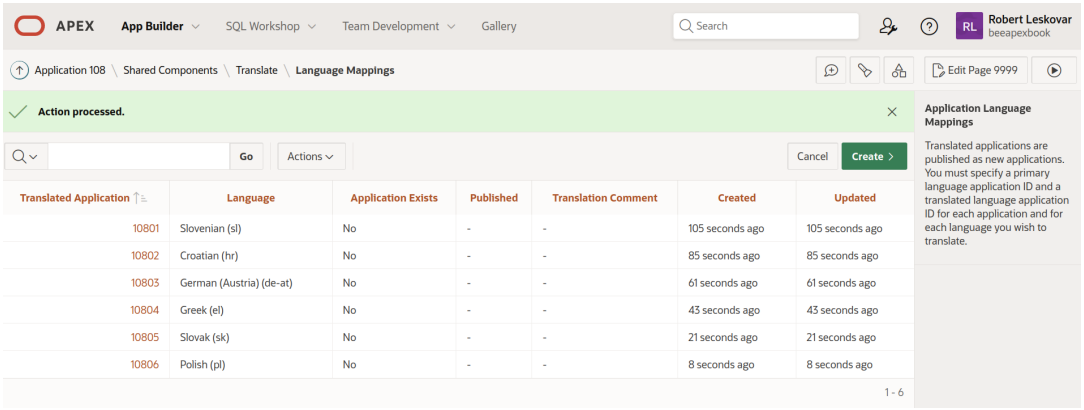
12.4 Večjezična aplikacija

To podpoglavje obravnava le en vidik večjezičnih aplikacij - besedilo na straneh aplikacije, kot so poročila in obrazci. Glavni razlogi za prilagoditev aplikacije določenemu jeziku so: a) uporabniki obvladajo samo druge jezike in b) obstajajo zahteve organizacij ali držav po podpori več kot enega jezika. Obseg prilagoditve je lahko omejen na prevod oznak elementov strani, sporočil aplikacije, notranjih sporočil APEX-a, oblik številk in datumov ali tako zapleten, kot je prevod besedila, shranjenega v tabelah podatkovne zbirke (npr. status, stopnja, ocena). Zaradi specifičnega kulturnega in jezikovnega konteksta je lahko dobeseden prevod nizov besedila smešen, žaljiv ali nestrokoven. Lokalizacija aplikacije je torej veliko več. APEX ima veliko možnosti za uporabo prevodov (na podlagi primarnega jezika aplikacije, brskalnika, preferenc aplikacije, elementa, seje). V nadaljevanju so na voljo najpreprostejša navodila za implementacijo več jezikov v aplikaciji s spreminjanjem primarnega jezika aplikacije. Koraki so naslednji:

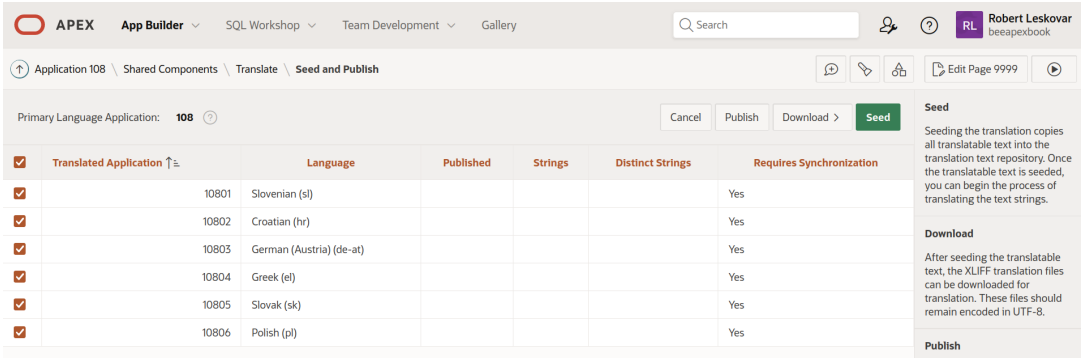
- pojdite v Shared components > Globalization > Application Translations > Define application languages: kliknite gumb Create; za vsak jezik določite unikatno celo številsko vrednost (tj. dodajte dve številki k številki aplikacije, kot so 10801 za sl; 10802 za hr; 10803 za de-at; 10804 za el; 10805 za sk; 10806 za pl). Slika 12.24 prikazuje vse nastavljene jezike.
- pojdite v Shared components > Globalization > Application translations > Seed translatable text. Izberite vse jezike, kot je prikazano na sliki 12.25. Kliknite gumb Seed in počakajte, da

se proces zaključi.

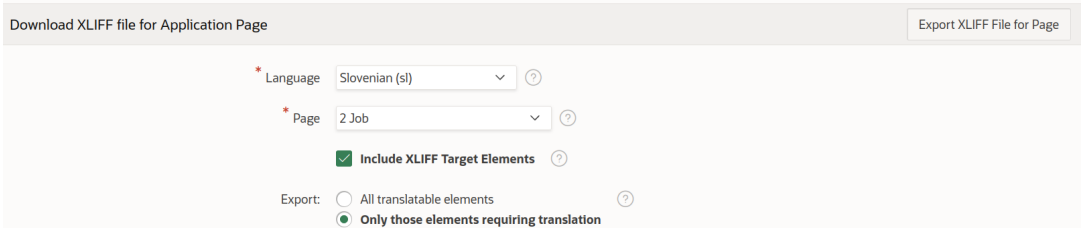
- pojdite v Shared components > Globalization > Application Translations > Download XLIFF translation files; izberemo lahko prenos vseh elementov, ki se lahko prevajajo, ali samo tistih, ki zahtevajo prevod. Primer na sliki 12.26 prikazuje izvoz strani 2, slovenski jezik in elemente, ki zahtevajo prevajanje.
- urejanje izvožene datoteke v želenem urejevalniku (Notepad++, Kate, Sublime itd.). V demonstracijske namene smo spremenili le nekaj "ciljnih" označenih nizov: v vrsticah 48, 52, 56, 60, 72 in 76 (glejte sliko 12.27). Shranite spremembe.
- pojdite v Shared components > Globalization > Translate application > Apply XLDIFF translation files > Upload Files. Izberite datoteko (glejte sliko 12.28) in kliknite Upload. Izberite ustrezen jezik v razdelku "Apply to Translation" (glejte sliko 12.29). Kliknite gumb Apply checked. Za objavo prevodov kliknite Publish.
- pojdite v Shared components > Globalization > Translate application > Publish translations. Izberite vse jezike, ki ste jih prevedli, in kliknite Publish (glejte sliko 12.30). Počakajte, da prejmete sporočilo, da je aplikacija uspešno objavljena.
- pojdite v Shared components > Globalization > Globalization attributes. Change Application Primary Language to new translation (glejte sliko 12.31). Kliknite Apply changes.



Slika 12.24: Določitev jezikov za prevajanje.



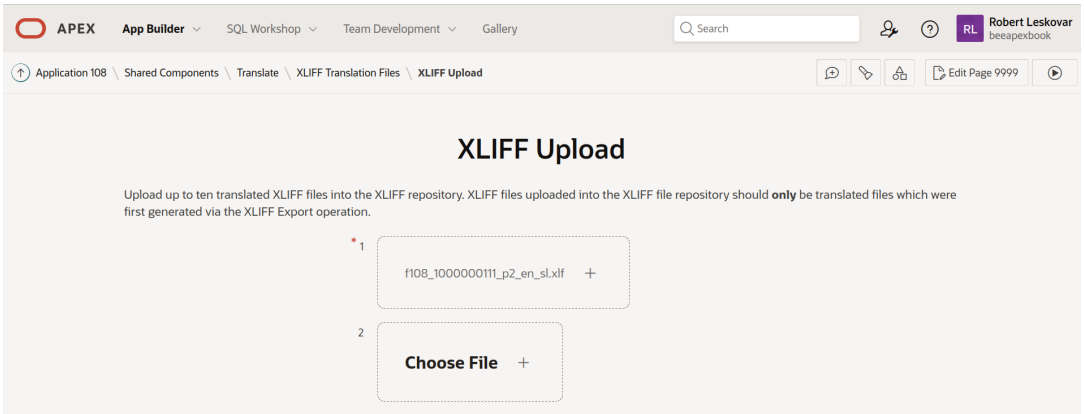
Slika 12.25: Besedilo, ki ga je mogoče prevesti.



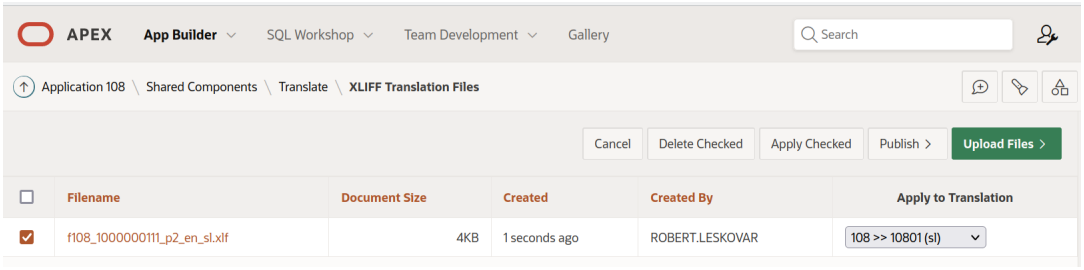
Slika 12.26: Izvoz nizov besedila za določen jezik in stran.

```
45  </trans-unit>
46  <trans-unit id="S-13-23279527479873043-108">
47  <source>Create</source>
48  <target>Ustvari</target>
49  </trans-unit>
50  <trans-unit id="S-20-23278046036873035-108">
51  <source>Ch12 Job</source>
52  <target>Ch12 Del. mesto</target>
53  </trans-unit>
54  <trans-unit id="S-20-23280741017873048-108">
55  <source>Breadcrumb</source>
56  <target>Drobtina</target>
57  </trans-unit>
58  <trans-unit id="S-143-23278046036873035-108">
59  <source>Job</source>
60  <target>Del. mesto</target>
61  </trans-unit>
62  <trans-unit id="S-146-23278230062873035-108">
63  <source>The maximum row count for this report is #MAX_ROW_COUNT# rows.  Please apply a filter to reduce the num
64  <target>The maximum row count for this report is #MAX_ROW_COUNT# rows.  Please apply a filter to reduce the num
65  </trans-unit>
66  <trans-unit id="S-147-23278230062873035-108">
67  <source>No data found.</source>
68  <target>No data found.</target>
69  </trans-unit>
70  <trans-unit id="S-149-23278547600873039-108">
71  <source>ID</source>
72  <target>Šifra</target>
73  </trans-unit>
74  <trans-unit id="S-149-23278989319873042-108">
75  <source>Job Description</source>
76  <target>Opis delovnega mesta</target>
77  </trans-unit>
78  </body>
79  </file>
80  </xliff>
```

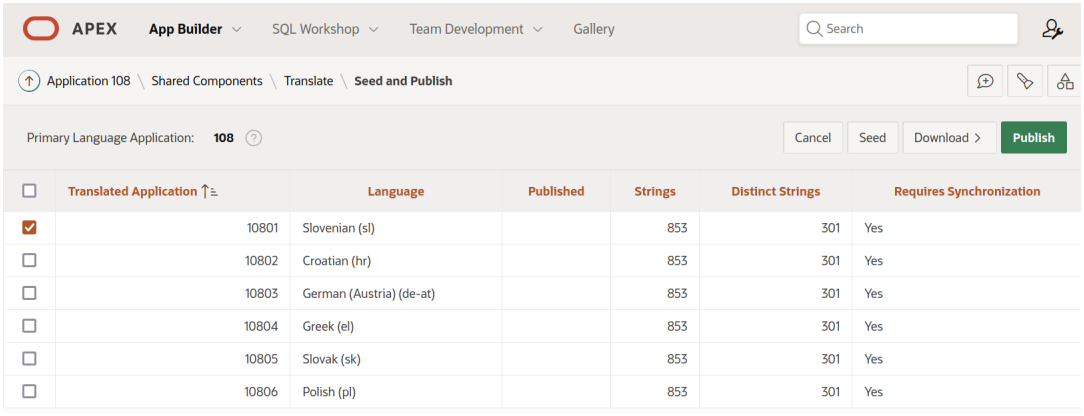
Slika 12.27: Prevod označenih “ciljnih” nizov v vrsticah 48, 52, 56, 60, 72 in 76.



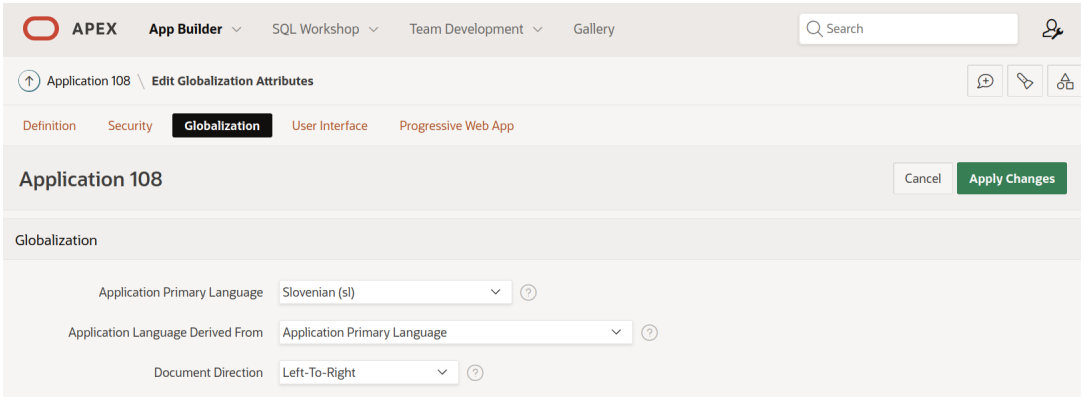
Slika 12.28: Nalaganje prevajalskih datotek XLDIFF.



Slika 12.29: Udejanjanje sprememb in objava.

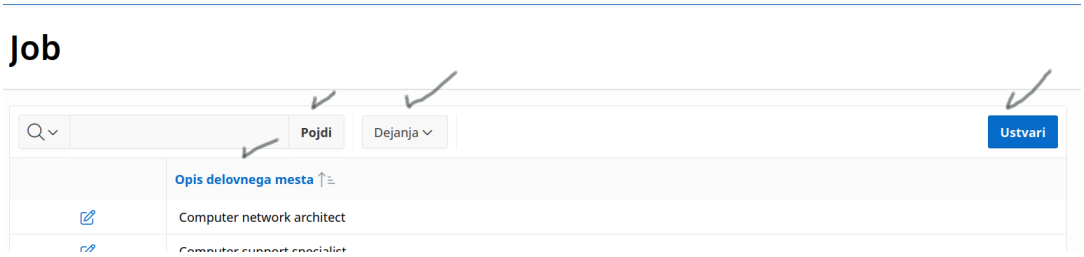


Slika 12.30: Končna objava prevoda aplikacije.



Slika 12.31: Nastavitev primarnega jezika aplikacije.

Zdaj preverite stran 2 v prevedeni aplikaciji (glejte sliko 12.32).



Slika 12.32: Prevedena stran.

Sklepne opombe Razvoj običajno poteka v testnem okolju. Pri selitvi pregledane aplikacije v nov delovni prostor so paketne aplikacije zelo priročne, saj skrajšajo čas selitve. Zaradi varnostnih razlogov se uporabniki in njihove vloge ne prenesejo. Odobritev uporabnikov v novem delovnem prostoru je seveda resna naloga. V tem poglavju so navedena tudi osnovna navodila za večjezične aplikacije. Opozoriti je treba, da bi bilo pri velikih aplikacijah in aplikacijah s kritičnimi nalogami prevajanje bolj zapleteno, in sicer z uporabo zmogljivih orodij za samodejno prevajanje ter strokovnjakov za jezik in kulturo.

12.5 Dodatno učno gradivo

Na voljo je naslednje dodatno učno gradivo:

- izvožene aplikacije
- video vodiči

Vse dodatno učno gradivo je na voljo na javni strani projekta BeeAPEX. Prijavite se kot gost (geslo ni potrebno). Učbenik poiščite v razdelku Knjige (Book section), skripte v mapi Part 1 > Chapter12 v razdelku Scripts section, video vodiče pa v Collection of videoguides. Gradivo za kratke tečaje v razdelku Short courses.

12.5.1 Izvožene aplikacije

Obstajajo tri povezane aplikacije:

- prvotna aplikacija. Preverite, ali obstajajo tabele, ki se začnejo s CH12. Če tabele obstajajo, jih izbrišite s skripto CH12DROP. Nato uvozite aplikacijo iz datoteke CH12_Application_initial.sql. Ko uvozite aplikacijo, definirajte tabele. Izvedite skripto CH12CREATE in po želji napolnite tabele s skripto CH12INSERT. Pojdite v uvoženo aplikacijo > hared Components > Application Access Control. Izberite Add User Role Assignment za trenutno prijavljenega uporabnika. Potrdite vsa dejanja z vlogo Administrator; stavke select, insert, update in delete z vlogo Contributor; in izberite samo stavke select z vlogo Reader.
- pakirana aplikacija. Preverite, ali obstajajo tabele, ki se začnejo s CH12. Če tabele obstajajo, jih izbrišite s skripto CH12DROP. Nato uvozite aplikacijo iz datoteke CH12_Application_packaged.sql. Pojdite v uvoženo aplikacijo > Shared Components > Application Access Control. Izberite Add User Role Assignment za trenutno prijavljenega uporabnika. Potrdite vsa dejanja z vlogo Administrator; stavke select, insert, update in delete z vlogo Contributor; in izberite samo stavke select z vlogo Reader.
- pakirana večjezična aplikacija. Preverite, ali obstajajo tabele, ki se začnejo s CH12. Če tabele obstajajo, jih izbrišite s skripto CH12DROP. Nato uvozite aplikacijo iz datoteke CH12_Application_packaged.sql. Pojdite v uvoženo aplikacijo > Shared Components > Application Access Control. Izberite Add User Role Assignment za trenutno prijavljenega uporabnika. Potrdite vsa dejanja z vlogo Administrator; stavke select, insert, update in delete z vlogo Contributor; in izberite samo stavke select z vlogo Reader.

12.5.2 Video vodiči

Obstajajo naslednji video vodiči:

- ustvarjanje začetne aplikacije
- uvoz začetne aplikacije
- kopiranje začetne aplikacije in ustvarjanje paketne aplikacije
- uvoz paketne aplikacije
- uvoz paketne večjezične aplikacije

12.6 Vprašanja

1. Kaj je paketna aplikacija in kakšne so njene prednosti?
2. Kako izvozite paketno aplikacijo, da vključuje definicije objektov v bazi podatkov in podatke?
3. Kakšna je vloga datoteke XLDIFF pri ustvarjanju večjezične aplikacije?

12.7 Odgovori

1. Paketna aplikacija je vnaprej pripravljena aplikacija, ki jo je mogoče namestiti in konfigurirati v delovnem prostoru Oracle APEX. Prednosti paketne aplikacije so enostavna namestitev in takojšnja razpoložljivost vnaprej pripravljenih podatkov.
2. Paketno aplikacijo je potrebno izvoziti s čarovnikom. Vključimo lahko skripte, ki definirajo objekte v bazi podatkov (npr. tabele) in navodila za vstavljanje podatkov v tabele. Seznam skript in vrstni red njihovega izvajanja določimo v čarovniku. Pakirana aplikacija lahko vsebuje tudi skripte za odstranitev aplikacije.
3. Datoteka XLDIFF vsebuje oznake in tako omogoča enostaven dostop do besedila v nizih v več jezikih. Datoteka lahko vsebuje celotno aplikacijo ali samo določeno stran v izbranem jeziku. Datoteka s prevedenimi nizi se uvozi v aplikacijo. Repozitorij prevodov je treba objaviti, da je na voljo končnemu uporabniku.

II Razvoj aplikacije v APEX-u

13 Intranetne novice za zaposlene 225
R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

14 GreenDi - katalog rastlin 237
Vjeran Strahonja, Dijana Oreški, Darko Andročec, Ana Kutnjak; prevod R. Leskovar . . . . . .

15 GreenDi - avtorizacija in upravljanje uporabnikov 245
Vjeran Strahonja; prevod R. Leskovar . . . . . .

16 Sistem malih inovacij 251
R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

17 Vodenje poslovnih procesov 261
R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

18 GreenDi – menjava rastlin in semen . 293
Vjeran Strahonja, Dijana Oreški, Darko Andročec, Ana Kutnjak; prevod R. Leskovar . . . . . .

19 Sistem za ocenjevanje knjig 299
A. Kutnjak, L. Hrustek, A. Baggia and R.Leskovar; prevod R. Leskovar . . . . . .

20 Materialna kosovnica in kalkulacija stroškov 311
R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

21 Vodenje prehrane in diete 327
A. Angeioplastis, G. Myllis, A. Tsimpiris, D. Varsamis; prevod R. Leskovar . . . . . .

22 Razporejanje uradnih ur 347
J. Mańko, M. Soňta, R. Leskovar; prevod R. Leskovar .

23 Primer telekomunikacijskih storitev . . 365
Veronika Šalgová, Jozef Kostolný, Michal Mrena, Michal Kvet, Miroslav Potočár; prevod R. Leskovar . . . . . .

24 Najem vozila 389
A. Angeioplastis, G. Myllis, A. Tsimpiris, D. Varsamis; prevod R. Leskovar . . . . . .



13. Intranetne novice za zaposlene

R. LESKOVAR, U. RAJKOVIČ, A.BAGGIA; PREVOD R. LESKOVAR

13.1 Poslovni vidik primera

Srednje veliko podjetje s sedemsto zaposlenimi je uveljavljen igralec v svetovni industriji vrhunske opreme za šport in prosti čas ter naprednih kompozitov. Sedež je v Sloveniji, hčerinske družbe pa so v ZDA, Kanadi, Nemčiji in na Japonskem. Njihovi letni prihodki znašajo približno 90 milijonov evrov. Večina zaposlenih se nahaja v Sloveniji na eni lokaciji. Podjetje je imelo v času te študije štiri divizije in vsak je imel več oddelkov. V preteklosti so uporabili interni polletni bilten za sporočanje različnih dosežkov, zgodb o uspehu, napovedi novih izdelkov in podobno. Nato so izdelali statično (samo HTML) intranetno stran za sporočanje pomembnih novic zaposlenim. Toda ta priložnost za opolnomočenje zaposlenih in krepitev njihove pripadnosti podjetju je bila kratkotrajna. Idiličen vrt se je spremenil v problem.

13.2 Definicija problema

Postopek objavljanja intranetnih novic je bil ohlapno opredeljen. Zaposleni v oddelku za informacijsko tehnologijo (IT) je prejel zahtevo za objavo novic od izdajateljev – vodij divizij in oddelkov. Zahteva je bila posredovana po elektronski pošti, po telefonu ali osebno in to pogosto s skromnim gradivom (besedilo, slike, video). Datum začetka in konca novic ni bil določen. Pogostost zahtev za objavo novic se je povečala na približno sto novic na mesec in nekatere zahteve so se prekrivale (dva ali več vodij oddelkov je imelo podobne ali enake teme). Zaposleni v IT je moral rešiti vse težave med vodji in z vodji. Začetni in končni datumi za pojav novic so bili tako pogosto na ramenih skrbnika v IT, kar je povzročilo slabo razpoloženje vodij in zaposlenih v IT. Tudi upravljanje statične strani je postalo zamudno in podvrženo napakam. Rešitev je zahtevala tako spremembo postopka kot tudi novo založniško platformo.

Sprememba postopka objavljanja in nova platforma Za razvoj intranetne aplikacije je bilo izbrano orodje za malokodno programiranje. Podjetje že uporablja Oracle bazo podatkov, zato je bil Oracle APEX izbran kot primerno razvojno okolje (hiter razvoj, ni dodatnih stroškov z licencami). Opredeljene so bile tri vrste uporabnikov platforme. *Skrbnik* ima dostop do vseh podatkov ter ima pravice za upravljanje uporabnikov in njihovih vlog. Poleg osnovnih podatkov o zaposlenih podjetje spremlja tudi njihovo zaposlitev po oddelkih in lokacijah. Vsak zaposleni ima določeno vlogo, ki se lahko sčasoma spremeni. Druga vrsta uporabnika je *založnik* (izdajatelj), ki nima pravice do upravljanja podatkov o zaposlenih in določanja privilegijev, ima pa pravico

objavljati novice. Tretja vrsta uporabnika je *gledalec/bralec* novic, ki ima pravico brati internetne novice.

Opredelitev zahtevanih lastnosti Na podlagi spremenjenega postopka bo platforma uporabnikom v posameznih vlogah zagotavljala:

- *bralec*: omogočena je prijava, dostop do nadzorne plošče, ogled novic in prilog k novicam
- *založnik*: omogočena je prijava, dostop do nadzorne plošče, ogled in objavljanje (urejanje) novic ter dodajanje prilog objavljenim novicam
- *skrbnik*: omogočena je prijava, dostop do nadzorne plošče, urejanje novic, ogled in upravljanje oddelkov, vlog, zaposlenih, vlog zaposlenih, upravljanje aplikacije

13.3 Primeri uporabe

13.3.1 Opis primerov uporabe

Založniška platforma omogoča dostop trem različnim vrstam zaposlenih: skrbnik, bralec in založnik. Vsak ima različne privilegije na intranetnem portalu. Založnik lahko objavi novice, bralec lahko bere novice in skrbnik lahko upravlja platformo. Vsak od teh primerov uporabe zahteva, da se uporabnik prijavi na intranetno platformo (glejte tabele 13.1, 13.2 in 13.3).

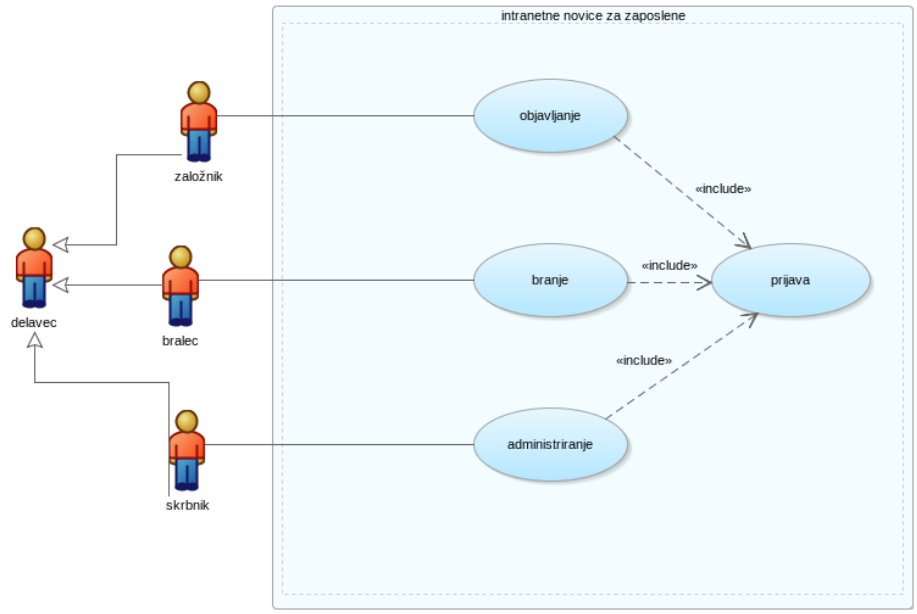
13.3.2 Delno strukturiran opis

Povzamemo lahko tri različne uporabniške zgodbe ali primere uporabe: objavljanje, branje in upravljanje.

Dodajanje priloge novici je dejansko razširitev zgornjega primera uporabe. Zato bi to lahko obravnavali kot nov primer uporabe. Zgolj zaradi jedrnatosti tega poglavja nadaljujemo z drugimi primeri uporabe, čeprav bo v aplikaciji razvita funkcija dodajanja prilog k novici.

13.3.3 Diagram primera uporabe

Opisani primeri uporabe so prikazani na diagramu primerov uporabe (slika 13.1).



Slika 13.1: Diagram primerov uporabe.

Zaradi jedrnatosti je opuščen primer uporabe “dodajanje priloge” na sliki 13.1.

Tabela 13.1: Opis primera uporabe: objavljanje novice na intranetu.

Ključni pojem	**Vrednost**
ID:	*Ch13-01*
Naslov:	*Objava novice na intranetnem portalu*
Opis:	*Odgovorna oseba v kadrovskem oddelku, ki ima vlogo založnika, uporabi intranetno aplikacijo v APEX-u za objavo novice. Vsaka novica ima definiran datum objave in umika objave. Privzeti datum začetka objave je trenutni datum.*
Glavni igralec:	*Zaposleni z vlogo založnika*
Predhodni pogoji:	*Uporabnik mora biti zabeležen v tabeli zaposlenih in mora imeti vlogo založnika. Omogočen mora biti dostop do intranetne aplikacije.*
Pogoji po zaključku:	*Po uspešni objavi je novica z vsemi prilogami dostopna vsem uporabnikom s pravico branja.*
Glavni del:	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v intranetno aplikacijo.* 2. *Izberite Objavo novice (ang. Publish news).* 3. *Vpišite naslov novice (ang. Title), opis)(ang. description) in datum umika novice (ang. end-date).* 4. *Potrdite ustvarjanje novice.* 5. *Dodajte priloge (ang. Add attachment(s)).* 6. *Vnesite podrobnosti (ang. attachment details).* 7. *Potrdite dodajanje priloge (ang. attachment).* 8. *Preglejte objavljeno novico.*
Razširitve:	• *1a. Prijava ni uspela.* • *1a* Razširitev:* • *1a1. Prikaži opis napake.* • *1a2. Odpri okno za prijavo.* • *4a. Napaka v tipu podatka.* • *4a* Razširitev:* • *4a1. Prikaži opis napake.* • *7a. Napaka v tipu podatka.* • *7a* Razširitev:* • *7a1. Prikaži opis napake.*
Pogostost uporabe:	*Založniki objavijo okoli 1000 novic letno, v poprečju 5 na dan*
Status:	*Zaključen*
Lastnik:	*Zaposleni z vlogo založnika*
Prioriteta:	*zmerna*

Tabela 13.2: Opis primera uporabe: branje intranetne novice

Ključni pojem	**Vrednost**
ID:	*Ch13-02*
Naslov:	*Branje novice na intranetu*
Opis:	*Uporabnik bere novice*
Glavni igralec:	*Uporabnik intraneta*
Predhodni pogoji:	*Uporabnik mora biti zabeležen v tabeli zaposlenih in mora v aplikaciji imeti vlogo bralca. Omogočen mora biti dostop do intranetne aplikacije.*
Pogoji po zaključku:	-
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v intranetno aplikacijo.* 2. *Izberite pregled novic (ang. View news)* 3. *Izberite prenos (ang. Download) za vpogled v prilogo.*
Razširitve:	• *1a. Prijava ni uspela.* • *1a* Razširitev:* • *1a1. Prikaži opis napake.*
Pogostost uporabe:	*Zaposleni dnevno berejo intranetne novice.*
Status:	*<Zaključen*
Lastnik:	*Uporabnik intraneta*
Prioriteta:	*nizka*

Tabela 13.3: Opis primera uporabe: upravljanje intranetne aplikacije.

Ključni pojem	**Vrednost**
ID:	*Ch13-03*
Naslov:	*Upravljanje intranetne aplikacije*
Opis:	*Upravljanje intranetne aplikacije*
Glavni igralec:	*Skrbnik intranetne aplikacije*
Predhodni pogoji:	*Uporabnik mora biti zabeležen v tabeli zaposlenih in mora imeti vlogo skrbika. Omogočen mora biti dostop do intranetne aplikacije*
Pogoji po zaključku:	*Podatki o uporabnikih, vlogah in zaposlenih so pripravljeni za zaposlene, ki imajo predvidene pravice.*
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v intranetno aplikacijo.* 2. *Uredite podatke o oddelkih.* 3. *Uredite podatke o vlogah.* 4. *Uredite podatje o zaposlenih in njihovih vlogah.* 5. *Objavite novico.* 6. *Preglejte objavljeno novico.* 7. *Administrirajte aplikacijo.*
Razširitve:	• *1a. Prijava ni uspela.* • *1a* Razšititev:* • *1a1. Prikaži opis napake.* • *1a2. Odpri okno za prijavo.* • *2a. Napaka v tipu podatka.* • *2a* Razšititev:* • *2a1. Prikaži opis napake.* • *3a. Napaka v tipu podatka.* • *3a* Razšititev:* • *3a1. Prikaži opis napake.* • *4a. Napaka v tipu podatka.* • *4a* Razšititev:* • *4a1. Prikaži opis napake.* • *5a. Napaka v tipu podatka.* • *5a* Razšititev:* • *5a1. Prikaži opis napake.*
Pogostost uporabe:	*Pogostost uporabe je odvisna od fluktuacije zaposlenih in sprememb pri dodeljevanju vlog. Ocena števila sprememb je 5 tedensko.*
Status:	*Zaključen*
Lastnik:	*Skrbnik aplikacije*
Prioriteta:	*visoka*

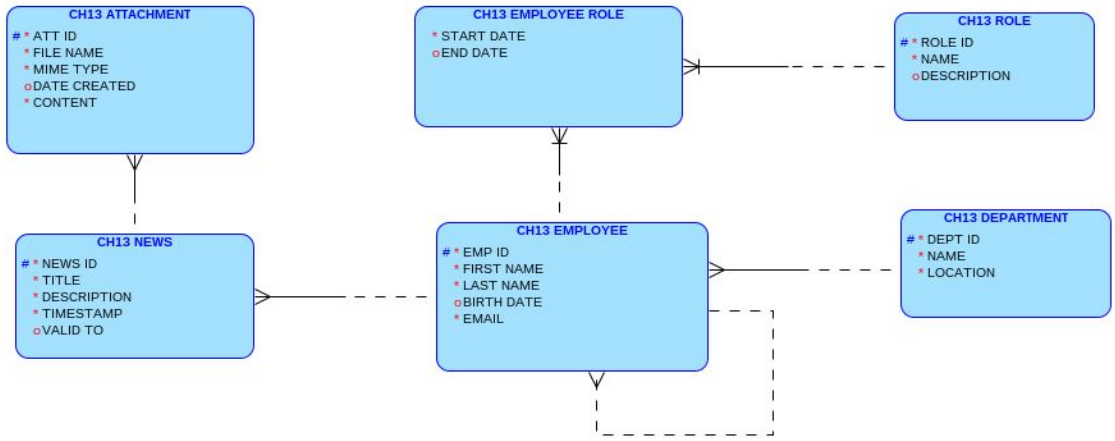
13.4 Model podatkov

13.4.1 Opis modela podatkov

V logičnem podatkovnem modelu je šest entitet. Entiteta CH13 DEPARTMENT (oddelek) ima tri atribute: ID, naziv in lokacijo oddelka. Vsak oddelek ima lahko veliko zaposlenih. Osnovni podatki o zaposlenih so shranjeni v entiteti CH13 EMPLOYEE: ID, ime in priimek, datum rojstva in e-pošta. Zaposleni je lahko vodja. Vsak zaposleni ima samo enega vodjo, vodja pa ima lahko več zaposlenih. Vsak delavec ima lahko svojo vlogo. Vloge so določene z ID, imenom in opisom. Za vsakega zaposlenega in vsako vlogo se določita začetni in končni datum. Tako ima podjetje pregled zgodovine zaposlitev. Sčasoma ima lahko zaposleni več vlog, več zaposlenim pa se lahko dodeli ena vloga. Kljub temu definirane vloge ni mogoče prenesti ob premestitvi na drugo delovno mesto. Zaposleni lahko v daljšem časovnem obdobju objavljajo več novic. Vsaka novica je označena z ID, naslovom, opisom, začetnim in končnim datumom. Vsaka novica ima lahko več priponk. Priloga je definirana z ID, imenom datoteke, vrsto priponke (tip MIME - Multipurpose Internet Mail Extensions), datumom kreiranje in vsebino.

13.4.2 Logični model podatkov

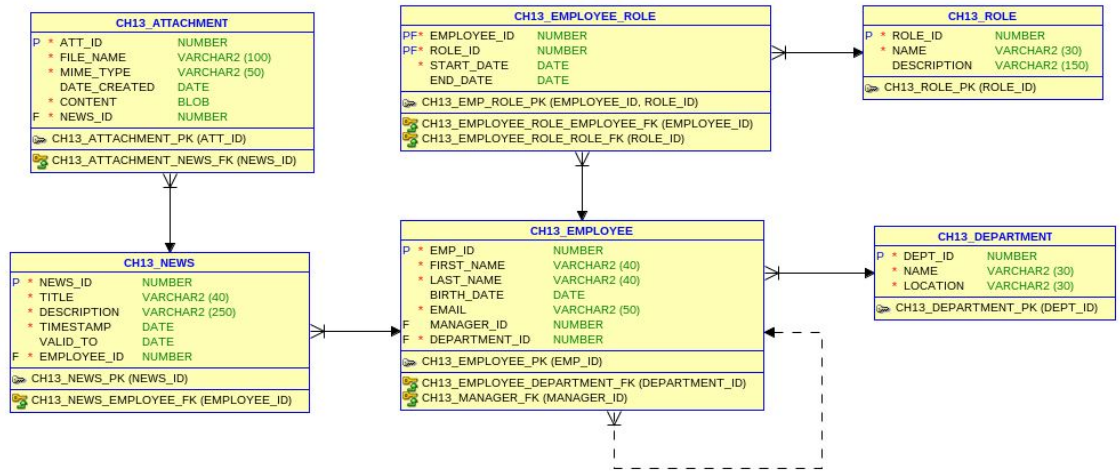
Zgoraj opisane entitete in povezave so prikazane na logičnem modelu podatkov (slika 13.2).



Slika 13.2: Logični model podatkov.

13.4.3 Relacijski model podatkov

Preslikavo logičnega modela v relacijski model je možno izvesti z orodjem Oracle SQL Data Modeler. Funkcija se imenuje *Engineering to relational*. Rezultat preslikave, to je relacijski model podatkov, je prikazan na sliki 13.3.



Slika 13.3: Relacijski model podatkov.

Oracle SQL Data Modeler lahko na zahtevo generira tudi skript za kreiranje tabel, sekvenc in prožilcev. V orodju izberemo vse tabele in uporabimo funkcijo *File* > *Export* > *DDL File*. Del generiranega skripta zgleda takole:

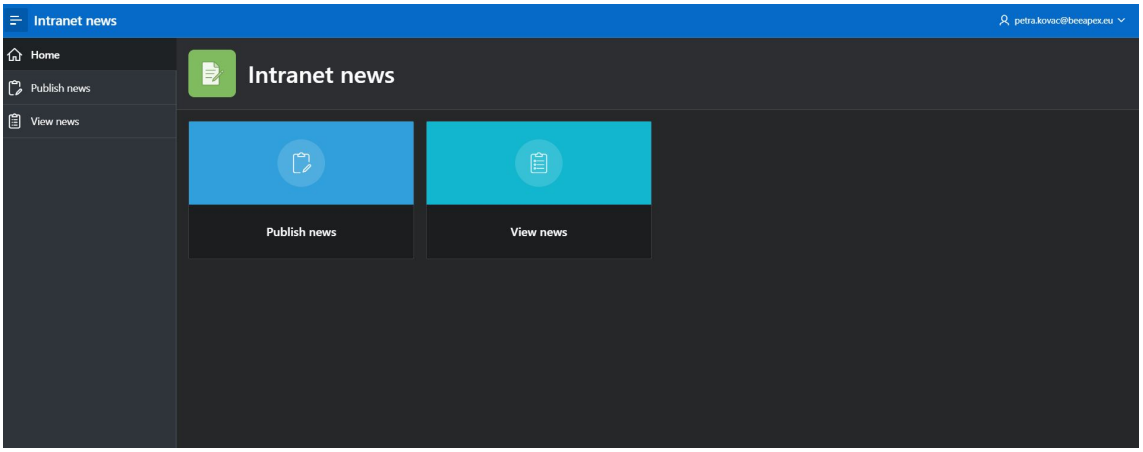
```
CREATE TABLE CH13_NEWS (
NEWS_ID NUMBER NOT NULL
...
);

CREATE TABLE CH13_DEPARTMENT (
DEPT_ID NUMBER NOT NULL
...
);
```

Zdaj je čas, da tabele kreiramo v bazi Oracle. V APEX-u uvozimo skript in ga izvedemo.

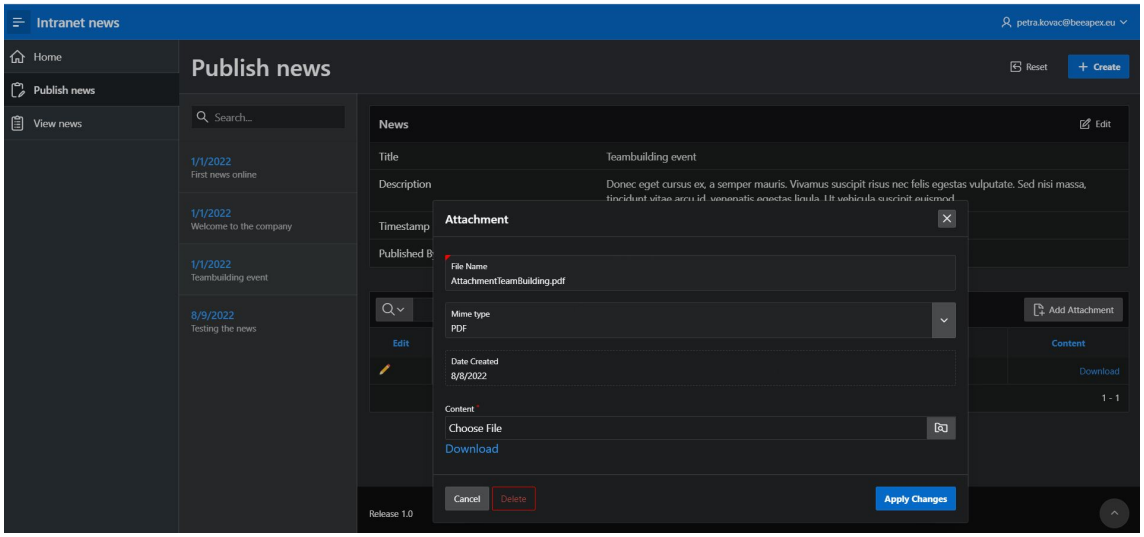
13.5 Uporabniški vmesniki

Na naslednjih slikah so prikazani uporabniški vmesniki za vse tri vloge. Na sliki 13.4 je prikazana pregledna plošča uporabnika, ki ima vlogo založnika.



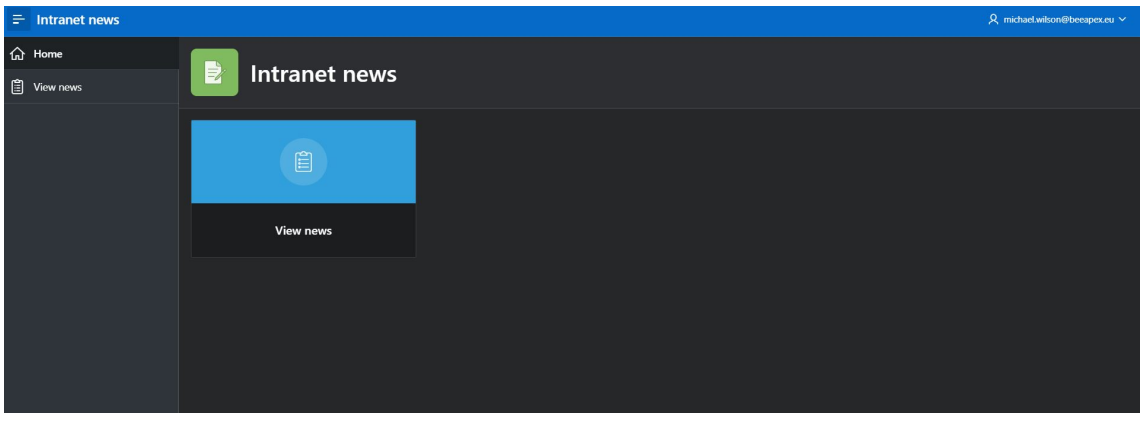
Slika 13.4: Pregledna plošča uporabnika z vlogo založnika.

Stran, ki omogoča objavo novic s prilogami je prikazana na sliki 13.5.



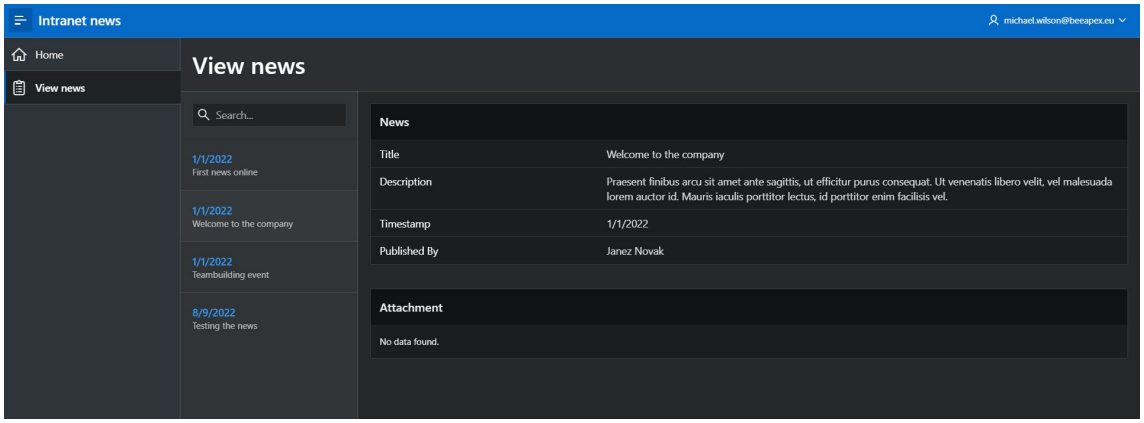
Slika 13.5: Uporabniški vmesnik za objavljanje novic s prilogami.

Pregledna plošča uporabnika z vlogo bralca ima v menujski izbiri le Domov (ang. Home) ter Pregled novic (ang. View news). Prikazana je na sliki 13.6.



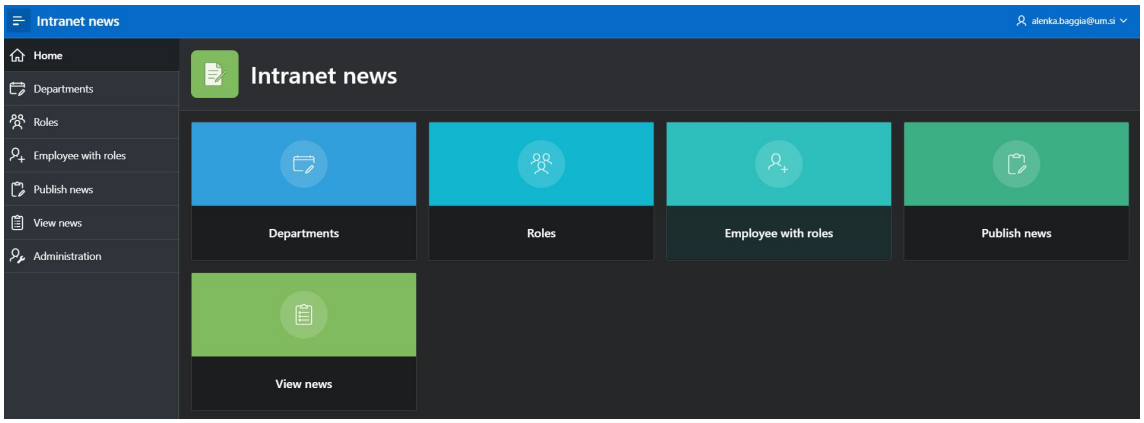
Slika 13.6: Pregledna plošča uporabnika z vlogo bralca.

Najpomembnejša funkcija vmesnika za bralca je dostop do novic (slika 13.7).



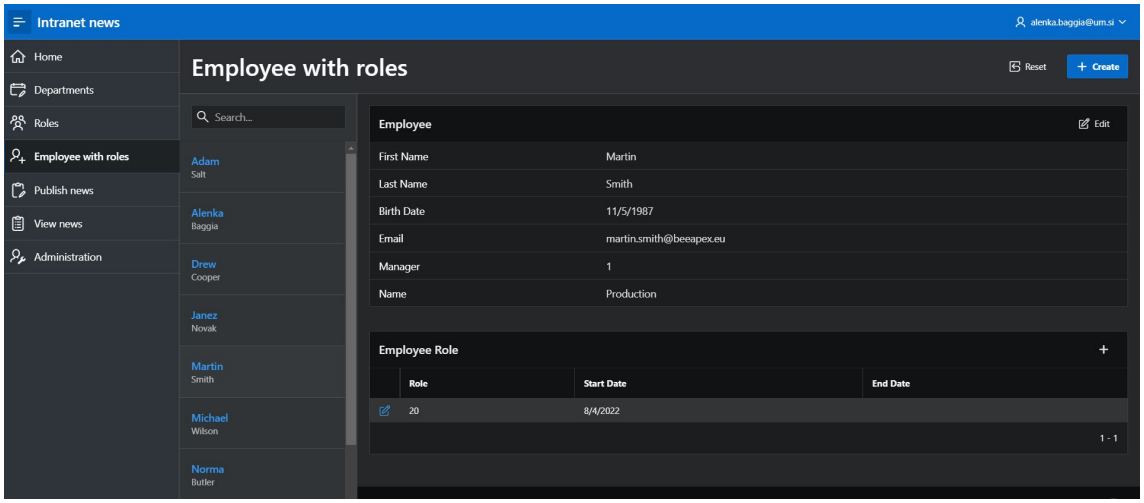
Slika 13.7: Uporabniški vmesnik za branje novic.

Skrbnik aplikacija ima na voljo funkcije za urejanje novic (vključno s prilogami), urejanje oddelkov, vlog, dodeljevanje vlog uporabnikom (slika 13.8). Skrbniška stran (Administration) je na voljo uporabnikom, ki imajo vlogo skrbnika.



Slika 13.8: Pregledna plošča uporabnika z vlogo skrbnika.

Stran za udejanje vlog uporabnikov je prikazana na sliki 13.9 in je na voljo izključno skrbnikom aplikacije.



Slika 13.9: Urejanje vlog uporabnikov.

13.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija
- skripti za kreiranje in brisanje tabel ter za dodajanje testnih podatkov v tabele
- video vodič

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter13 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

13.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

13.6.2 Video vodiči

Video vodiči natančno prikazujejo razvoj aplikacije.

13.7 Vprašanja

1. Kako bi spremenili logični model podatkov, da bi vpeljali možnost, da določeno prilogo uporablja več novic?
2. kako se spremeni relacijski model podatkov, če je vpeljana možnost, da določeno prilogo uporablja več novic?
3. Katere vloge uporabnikov so definirane v poglavju 13 ter zakaj?

13.8 Odgovori

1. Dodali bi povezava ena-mnogo v smeri priloga - novica.
2. Par tabel priloga-novica bi bil spremenjen v tri relacijske tabele: priloga, novica in priloga_novica. Slednja bi imela vsaj dve polji: IDs priloge in ID novice. Vsako polje posebej bi imelo vlogo tujega ključa.
3. V poglavju 13 so definirane vloge: skrbnik, založnik in bralec. Vloge zagotovijo poobla-

ščenim upoprabnikom vse njim dovoljene funkcije ter hkrati preprečijo nepooblaščenim uporabnikom zlorabo določenih funkcij.



14. GreenDi - katalog rastlin

VJERAN STRAHONJA, DIJANA OREŠKI, DARKO ANDROČEC, ANA KUTNJAK; PREVOD R. LESKOVAR

14.1 Definicija problema

Skupina navdušencev, ki so se zbrali v neprofitni organizaciji GreenDI, je začela zvajati različne pobude, povezane z urbanim vrtnarjenjem, zdravim prehranjevanjem, gojenjem starih domačih sort, sodelovanjem z lokalnimi kmeti in podobno. Zato so se odločili za razvoj lastne platforme GreenDI za medsebojno izmenjavo semen in rastlin, da bi spodbudili biotsko raznovrstnost in povečali dejavnost članov. Za razvoj in uvedbo takšne platforme je treba določiti zahtevane lastnosti, pokazati primere uporabe in oblikovati model podatkov. Orodje za malokodni razvok, kot je npr. Oracle APEX pa mora omogočiti hiter razvoj in prilagajanje, ko se zahteve dopolnjujejo.

14.2 Primeri uporabe

Odprti del platforme ima naslednje značilnosti:

- Uporabniki GreenDI niso omejeni na ožje geografsko področje, pač pa so lahko kjerkoli. Ker pa gre za izmenjavo rastlin, se pričakuje oblikovanje lokalnih mrež, povezanih z določenim geografskim območjem.
- Platforma je razdeljena na več tematskih enot, kot so tradicionalne in avtohtone sorte in semena (zelenjava, sadje, zdravilna zelišča, notranje in okrasne rastline), z idejo širitve na industrijske in druge rastline.
- Osnova odprtega dela platforme je katalog rastlin z nazivi v jeziku uporabnika, latinščini in angleščini, klasifikacijo rastline (taksonomija), opisom, habitati, gojenjem in uporabo, fotografijami in drugimi informacijami.
- Osnovne informacije o rastlini so na voljo za iskanje in ogled brez registracije.
- Katalog rastlin GreenDI je povezan z več odprtimi spletnimi stranmi, zbirkami podatkov in zunanjimi storitvami, ki vsebujejo informacije, povezane z rastlinami. Uporabljajo se naslovi URL, hiperpovezave in druge vrste označevalnih podatkov.
- Katalog vključuje rastline, ki se izmenjujejo na platformi GreenDI, pa tudi tiste, ki še niso bile izmenjane ali vključene v ponudbo. Katalog ima tudi povezave s komercialnimi dobavitelji, s katerimi imajo člani dobre izkušnje in ki spodbujajo ekološko kmetovanje.

Upravljanje kataloga lahko kratko opišemo takole:

- katalog rastlin upravljajo skrbniki platforme GreenDi,

- upravljanje kataloga se nanaša na vnašanje in spreminjanje podatkov o rastlinah,
- nekatere funkcije upravljanja kataloga lahko izvajajo samo samo avtorizirani uporabniki, ki jih določijo skrbniki platforme,
- napredne funkcije kot je npr. določitev hiperpovezav, so na voljo le skrbnikom platforme in
- vsi registrirani uporabniki(člani) lahko delijo svoja mnenja o rastlini, dodajajo nasvete in podobno, pri čemer prvi prototip te funkcije še ne bo podpiral.

14.2.1 Opis primera uporabe

14.2.2 Delno strukturiran opis

Odprti del platforme GreenDI omogoča vsakemu uporabniku (tudi brez registracije), da si iz katerekoli lokacije ogleda podatke v katalogu rastlin. Rastline so razvrščene v kategorije, ki uporabniku omogočajo lažjo navigacijo. Za vsako rastlino je prikazan naziv, taksonomija, opis, habitati, gojenje in uporaba, fotografije in druge osnovne informacije ter povezave do zunanjih virov podatkov.

14.2.3 Delno strukturiran opis

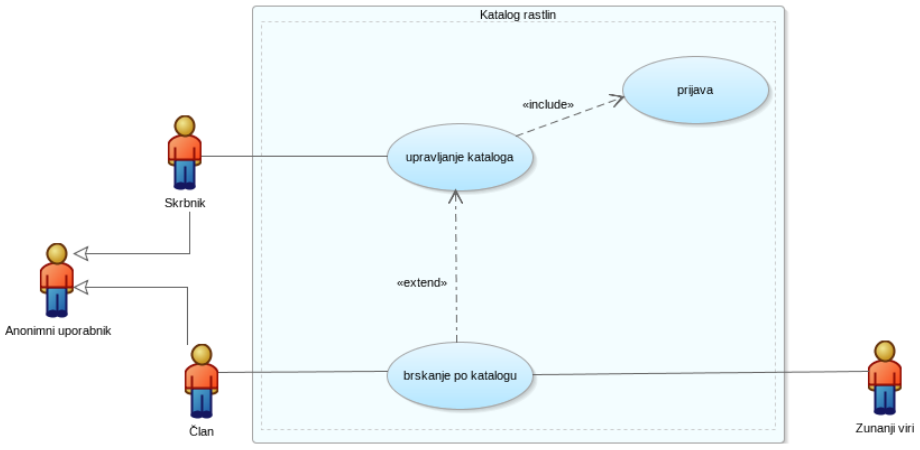
Tabela 14.1 prikazuje primer uporabe na platformi GreenDI.

14.2.4 Diagram primera uporabe

Opis primerov uporabe lahko prikažemo še grafično na diagramu primerov uporabe (slika 14.1). Na desni strani je zunanji vir prikazani igralec, ki lahko dopolni podatke o rastlinah. Obstaja več primerov zunajih virov, ki omogočajo odprt dostop:

1. Global Biodiversity Information Facility, kratko GBIF) omogoča prost dostop do podatkov o biotski raznolikosti, vključno o rastlinah. Prav tako vključuje dostop do številnih drugih virov kot so herbariji in raziskovalne ustanove. Delež prekrivanja GBIF z The International Plant Names Index, kratko IPNI je 97%.
2. WFO Plant List deluje od maja 2021 in omogoča uporabniku prijayno iskanje rastlin. Bazo je mogoče prenesti na lokalni računalnik ali pa za dostop uporabiti aplikacijski vmesnik (API) WFO Plant List.
3. Podatkovno zbirko Flora of Italy je možno prenesti s spletne strani University of Roma. Vključuje visokoločljive fotografije preko 10 tisoč rastlin iz Italije.

To je le nekaj primerov razpoložljivih virov podatkov s slikami rastlin in semen. V prototipni aplikaciji ne bo realizirana povezava z zunanjimi viri, ker to zahteva naprednejše znanje.



Slika 14.1: Diagram primerov uporabe.

Tabela 14.1: Opis primera uporabe: brskanje po katalogu rastlin

Ključni pojem	**Vrednost**
ID:	*ch14-01*
Naslov:	*Brskanje po katalogu rastlin*
Opis:	*Katalog rastlin je javno dostopen vsem. Aplikacija omogoča iskanje in filtriranje po vseh podatkovnih elementih.*
Glavni igralec:	*kdorkoli*
Predhodni pogoji:	*Brskanje po katalogu je omogočeno brez registracije uporabnika.*
Pogoji po zaključku:	*Če je brskanje po katalogu rastlin realizirano s klicem iz drugega primera uporabe, se v nadrejeni primer uporabe prenese ID izbrane rastline.*
Glavni del:	*Katalog rastlin*
Uspešni scenariji:	*Prvi scenarij - brskanje po katalogu rastlin:* 1. *Vsak uporabnik na domači strani odpre menu za brskanje, ki je realiziran kot iskalni obrazec.* 2. *Iskanje je omogočeno po naslednjih poljih: a) tematska enota - seznam skupin kot so zelenjava, sadje, zdravilna zelišča, notranje in okrasne rastline; b) vrsta rastline - hierarhično organizirana razvrstitev rastlin (vse privzeto ali določeno ime), c) naziv v uporabnikovem jeziku, d) latinski naziv e) angleški naziv* 3. *Rezultati iskanja so prikazani v obliki tabele.* 4. *Tabelo z rezultati je možno sortirati po več stolpcih.* 5. *Z izbiro povezave v vrstici se odpre novo okno, ki vsebuje: vrsta rastline, tematska enota, lokalna imena, angleška imena, latinska imena, opis, habitat, gojenje, uporaba, fotografije, hiperpovezave in status.* *Drugi scenarij - upravljanje kataloga:* 1. *Upravljanje kataloga lahko izvaja le uporabnik z vlogo skrbnika. Zato je prvi korak prijava v platformo (primer uporabe Prijava). Ob prijavi sistem preveri ali ima uporabnik vlogo skrbnika.* 2. *Upravljanje kataloga obsega dodajanje, spreminjanje in skrivanje podatkov o rastlinah.* 3. *Uporabnik potrdi ali zavrne spremembe.*
Razširitve:	• *Iskalni obrazec)* • *Tabela z rezultati, ki jo je možno urejati* • *Obrazec s podrobnim opisom rastline*
Pogostost uporabe:	*Predvidoma do največ 1000 iskanj na minuto.*
Status:	*V razvoju*
Lastnik:	*javni, anonimni uporabnik*
Prioriteta:	*visoka*

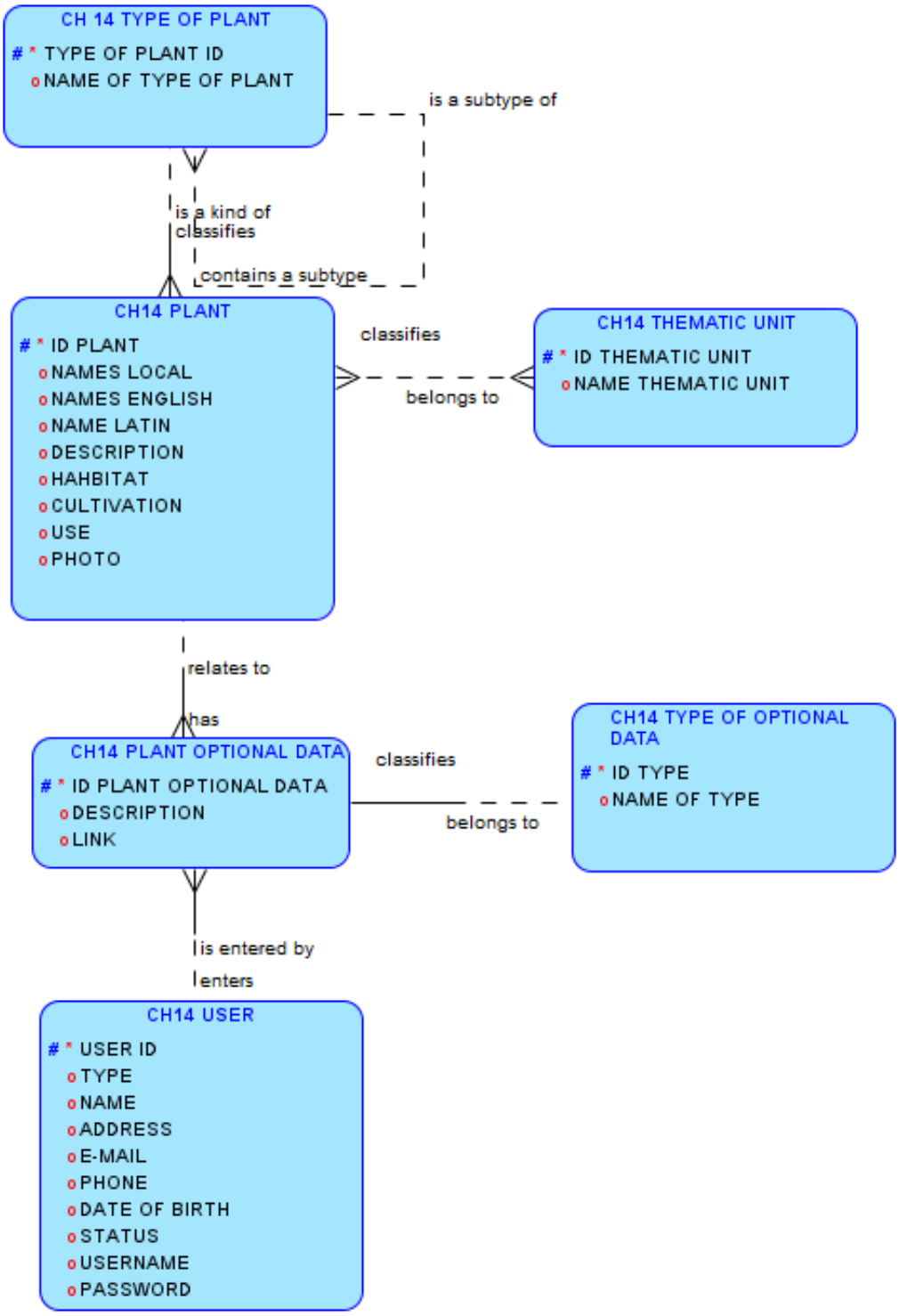
14.3 Model podatkov

14.3.1 Opis modela podatkov

Logični model podatkov vsebuje več entitet. Entiteta CH14 PLANT vsebuje osnovne podatke o rastlinah. Za eno rastlino je lahko na voljo več opcijskih podatkov (entiteta CH14 PLANT OPTIONAL DATA). Uporabnik (entiteta CH14 USER) lahko vnese več opcijskih podatkov. Opcijski podatki pripadajo eni od skupin opcijskih podatkov (CH14 TYPE OF OPTIONAL DATA). Rastlina pripada določeni vrsti (entiteta CH14 TYPE OF PLANT), ena vrsta pa ima lahko več rastlin. Vrste so lahko hierarhično povezane. Npr. nadrejena vrsta je sestavljena iz podvrst. Rastlina lahko pripada več tematskim enotam (entiteta CH14 THEMATIC UNIT), ena tematska enota pa povezuje več rastlin.

14.3.2 Logični model podatkov

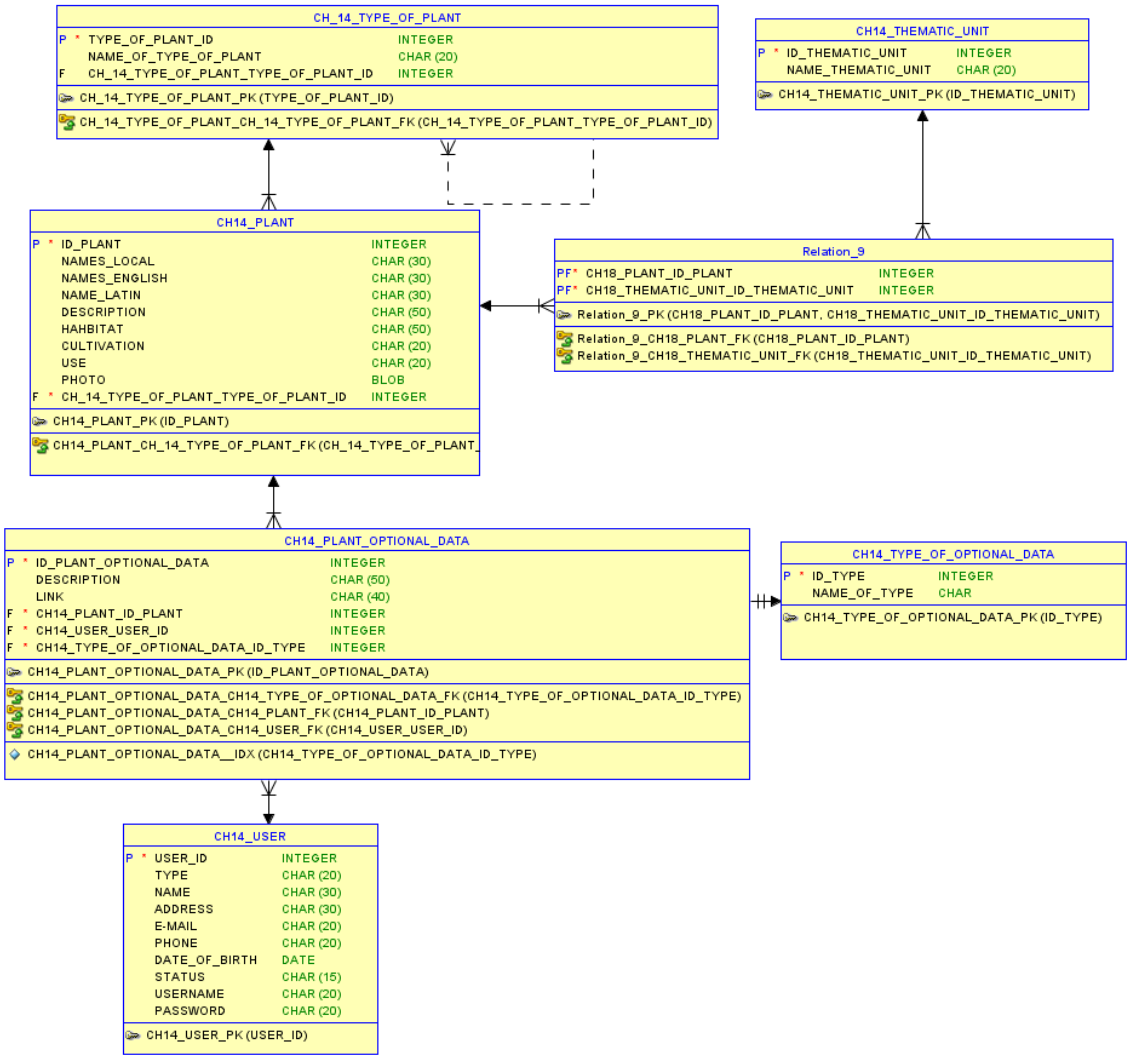
Logični model podatkov je prikazan na sliki 14.2.



Slika 14.2: Logični model podatkov.

14.3.3 Relacijski model podatkov

Preslikavo logičnega modela v relacijski model je možno izvesti z orodjem Oracle SQL Data Modeler. Funkcija se imenuje *Engineering to relational*. Rezultat preslikave, to je relacijski model podatkov, je prikazan na sliki 14.3.

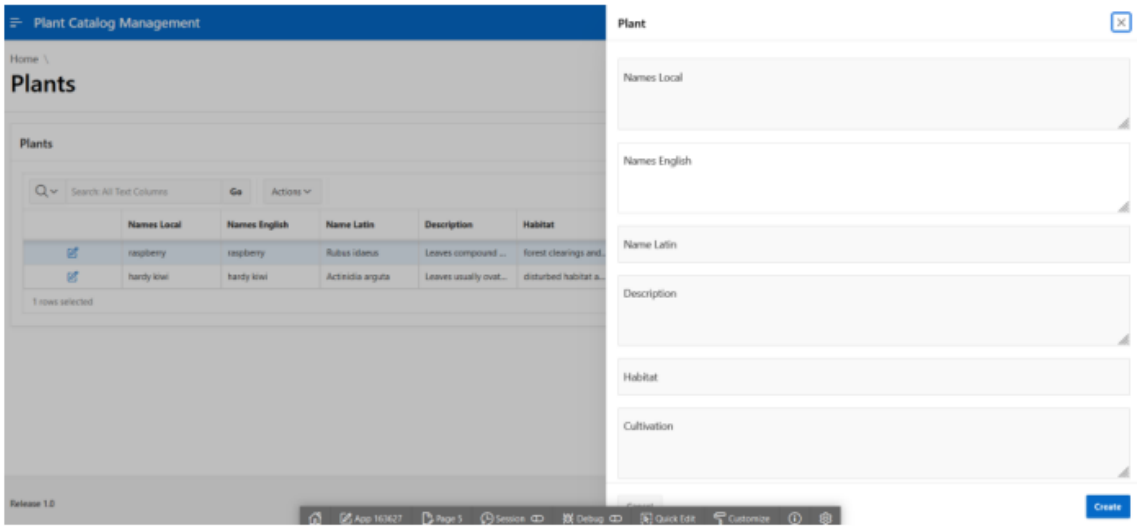


Slika 14.3: Relacijski model podatkov.

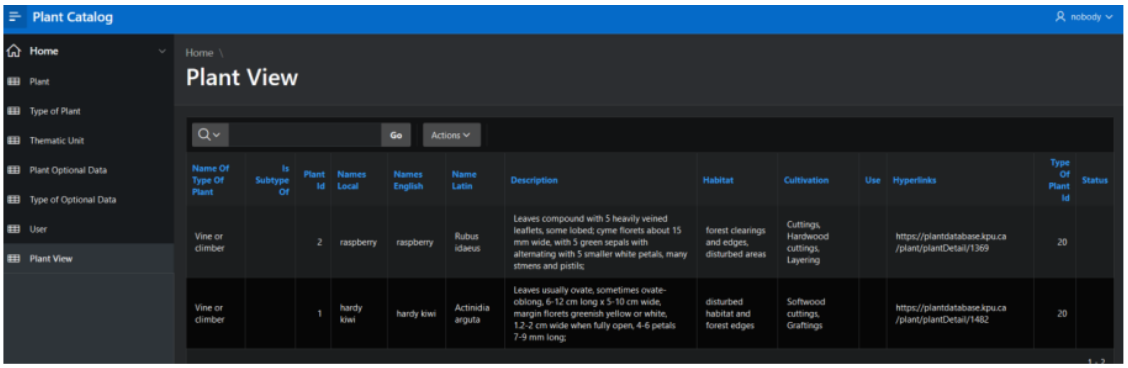
Z orodjem Oracle SQL Developer Data Modeler nato generiramo skript ((*File* > *Export* > *DDL File*). Nato v APEX-u uvozimo skript in ga izvedemo.

14.4 Uporabniški vmesniki

Izgled interaktivnega poročila in obrazca za osnovne podatke o rastlinah prikazuje slika 14.4, brskanje po raslinah, ki ga omogoča javni dostop, pa je prikazano na sliki 14.5.



Slika 14.4: Interaktivno poročilo (interactive grid) in obrazec za rastline.



Slika 14.5: Public part – open browsing of plant data.

14.5 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- skripti za kreiranje in brisanje tabel ter za dodajanje testnih podatkov v tabele
- izvožena aplikacija
- video vodič

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter14 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

14.5.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket in uporablja zgolj dve tabeli (ch14_type_of_plant and ch14_plants). Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo. Da bo izvožena aplikacija delovala v novem delovnem prostoru APEX-a (workspace) moramo dodati vlogo skrbnika novemu ali obstoječemu uporabniku še pred prvim zagonom. Izberite Shared Components > Application Access Control > Add User Role Assignment. Izberite uporabnika in mu dodelite vse ponujene vloge (Administrator, Contributor, Reader). V

produkcijskem okolju bodite pozorni na vloge, ki ste jih dodelili uporabnikom in se v primeru negotovosti posvetujte s strokovnjaki. Aplikacija ne Če uporabniku, ki se prijavi v aplikacijo, ni dodeljena vloga, se bo uvožena aplikacija zrušila. Po zrušitvi aplikacije je treba počistiti piškotek spletnega brskalnika (npr. Firefox: Nastavitve > Piškotki in podatki spletnega mesta > Upravljanje podatkov).

14.5.2 Video vodiči

Video vodiči natančno prikazujejo razvoj aplikacije.

14.6 Vprašanja

1. Raziščite kako bi bilo možno vzpostaviti povezavo z zunanjim virom podatkov.
2. Kateri so najpogostejši formati fotografij rastlin in semen?
3. Kakšen podatkovni tip uporablja baza Oracle za shranjevanje slik in video posnetkov?

14.7 Odgovori

1. Baze podatkov, ki jih je možno prenesti na lokalni računalnik imajo to slabost, da je potrebno posodabljanje. Zato je bolje raziskati možnost uporabe programskih vmesnikov. Kot primer navajamo aplikacijske vmesnike (API) za WFO Plant List. Opisi se nahajajo na tej povezavi.
2. Najpogostejši formati fotografij rastlin in semen so:
   a. JPEG (Joint Photographic Experts Group) - format omogoča stiskanje, zato so datoteke manjše. Izgubo natančnosti lahko izbiramo. Večja natančnost pomeni večjo velikost datoteke.
   b. BMP (Bitmap) - format se najpogosteje uporablja na operacijskih sistemih MS Windows. Ne podpira stiskanja, zato so datoteke bistveno večje kot pri formatih npr. JPEG ali PNG.
   c. PNG (Portable Network Graphics) - format omogoča stiskanje brez izgube kakovost slike. Omogoča tudi prosojno ozadje in se zato pogosto uporablja za spletno grafiko in digitalno umetnost.
   d. TIFF (Tagged Image File Format) - this format is commonly used for storing high-quality images, such as those used in printing. It supports lossless compression, multiple layers, and can handle both grayscale and color images. format se običajno uporablja za shranjevanje visokokakovostnih slik za tiskanje na papir ali plastiko. Podpira stiskanje brez izgub, več plasti, sivine in barvne odtenke.
3. Fotografijo, video ali podobne objekte tv bazi Oracle lahko shranjujemo kot zunanje datoteke. Kadar pa jih hočemo shraniti v polje tabele, uporabimo podatkovni tip BLOB (Binary Large Object). BLOB lahko shranjuje do 4 Gigabyte velike datoteke.



15. GreenDi - avtorizacija in upravljanje uporabnikov

VJERAN STRAHONJA; PREVOD R. LESKOVAR

15.1 Poslovni vidik primera

Zgoščen vpogled poslovni primer platforme GreenDi je opisan v poglavju 14. Osredotoča se na odprti del platforme GreenDi. Napredne funkcionalnosti so na voljo le registriranim uporabnikom (članom in skrbnikom). Omogočiti je treba funkcionalnost registracije uporabnikov, prijavo registriranih uporabnikov v sistem z avtentikacijo in avtorizacijo za opravljanje določenih aktivnosti. Potrebne funkcionalnosti vključujejo suspendiranje uporabnika pod določenimi pogoji in spremljanje zgodovine podatkov o registriranem uporabniku.

15.2 Definicija problema

Za avtorizacijo in upravljanje uporabnikov je značilno:

- za globlji vpogled v podatke o rastlinah in drugih naprednih funkcionalnostih je potrebna prijava v platformo, kar je dovoljeno le registriranim uporabnikom (članom),
- prijava je enotna za vse napredne funkcionalnosti,
- osnovni mehanizem za preverjanje pristnosti uporabnika sta uporabniško ime in geslo,
- informacije o trenutnem geslu niso vidne skrbniku ali kateremu koli drugemu uporabniku,
- postopek registracije vključuje vnos zahtevanih podatkov, kot so ime, naslov, e-pošta, telefon, uporabniško ime in geslo,
- neobvezni podatki so interesi uporabnikov, datum rojstva itd.
- po uspešni registraciji nov uporabnik avtomatsko pridobi privzeti tip člana in status aktivnega člana
- spremembo vloge skrbnika (administratorja) lahko izvede le drug aktivni skrbnik.
- z registracijo se član strinja s pogoji uporabe,
- če član krši pogoje uporabe, administratorji smejo prekiniti članstvo,
- registracija in izbris članstva sta prostovoljni in jo opravi uporabnik sam,
- vsaka sprememba katerega koli uporabniškega podatka se zabeleži v dnevniku sprememb,
- administrator lahko pregleda zgodovino sprememb vsakega uporabnika, razen gesla,
- uporabnik si lahko ponastavi geslo sam,
- sistem pošlje uporabniku začasno geslo preko elektronske pošte ali SMS-a,
- izbirna funkcionalnost: poleg uporabniškega imena in gesla je izbirno možna avtentikacija s računi Facebook, Google ali ali drugimi storitvami. Primer slednje je spletna aplikacija, ki

uporablja OAuth 2.0 za dostop do Googlovih API-jev.

15.3 Primeri uporabe

15.3.1 Opis primera uporabe

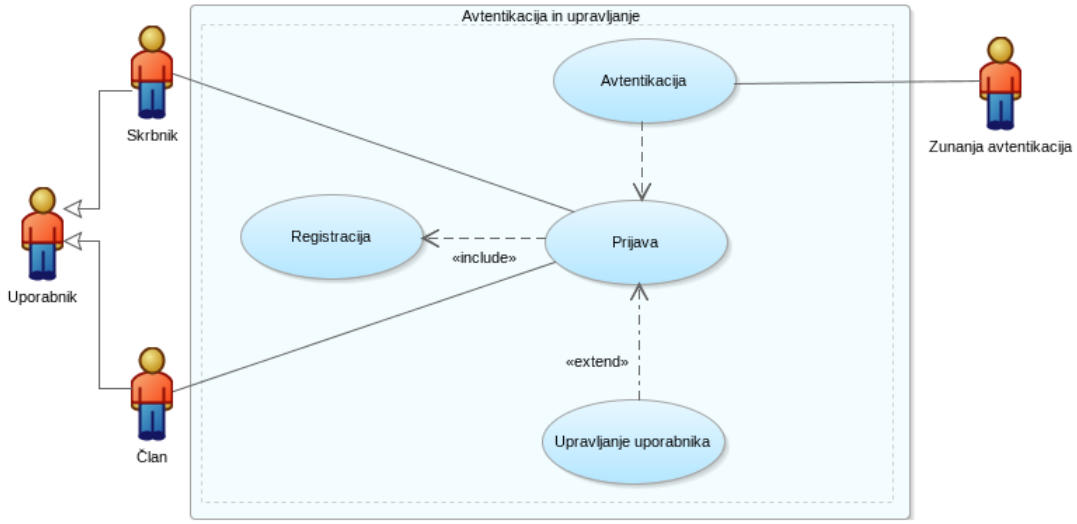
Diagram primera uporabe opisuje komunikacijo igralcev in drugih sistemov v okolju, ki ga razvijamo. Primeri uporabe predstavljajo funkcionalnost avtorizacije in upravljanja uporabnikov, ki je opisana na sliki 15.1.

15.3.2 Delno strukturiran opis

Tabela 15.1 prikazuje primer uporabe na platformi GreenDI.

15.3.3 Diagram primera uporabe

Opisani primer uporabe je prikazan na diagramu primera uporabe na sliki 15.1.



Slika 15.1: Diagram primera uporabe - avtorizacija in upravljanje uporabnikov na platformi GreenDi.

15.4 Model podatkov

15.4.1 Opis modela podatkov

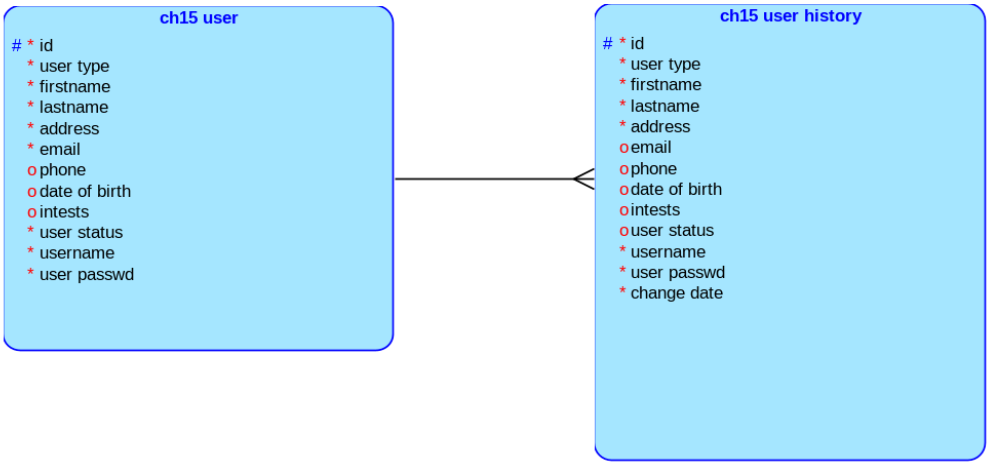
Logični model (entitetno relacjski model) ima dve entiteti: uporabnik in zgodovina uporabnika. Atributi uporabnika so, tip (član ali administrator), ime, priimek, naslov, e-pošta, telefon, datum rojstva (ni obvezen podatek), interesi (ni obvezen podatek), status (aktiven, suspendiran, neaktiven; uporabniško ime (ni vidno skrbniku, kriptirano) in geslo (ni vidno skrbniku, kriptirano). Uporabnik ima lahko več zapisov v zgodovini (dnevnik sprememb). V zgodovini uporabnika se shranijo vsi podatki o uporabnikovih preteklih lastnostih, datum nastanka spremembe (datum in čas) in ID uporabnika, ki je sprožil spremembo.

15.4.2 Logični model podatkov

Logični model podatkov je prikazan na sliki 15.2.

Tabela 15.1: Opis primera uporabe: avtorizacija in upravljanje uporabnikov.

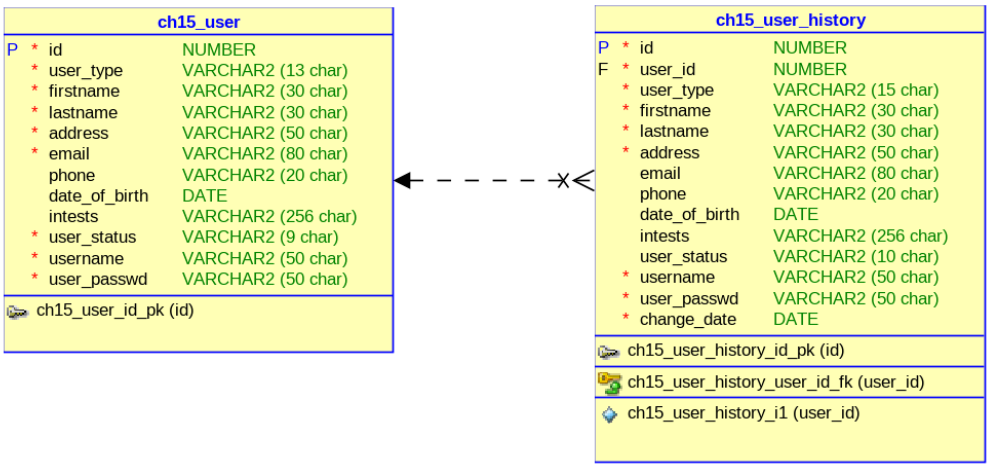
Ključni pojem	**Vrednost**
ID:	*ch15-01*
Naslov:	*Avtorizacija in upravljanje uporabnikov*
Opis:	*Avtorizacija in upravljanje uporabnikov.*
Glavni igralec:	*uporabnik, član, administrator*
Predhodni pogoji:	*Uporabnik s pravicami upravljanja se mora uspešno prijaviti aplikacijov.*
Pogoji ob zaključku:	*Po uspešni spremembi se uporabniki lahko prijavijo z novimi pravicami.*
Glavni del	-
Uspešen scenarij:	1. Za uporabo naprednih funkcij se mora uporabnik prijaviti. a. Pri prvi prijavi mora se mora uporabnik registrirati in vnesti obvezne podatke kot so ime, naslovm e-pošta, telefon. Lahko vnese tudi izbirne poratke kot so interesi, rojstni datum in podobno. b. Uporabnik določi uporabniško ime in geslo. Podatki se shranijo v posebno tabelo. c. Uporabnik sprejme pogoje uporabe. d. S tem postane aktiven član. 2. Uporabnik lahko svoje podatke spreminja sam. 3. Geslo si uporabnik ponastavi sam. Po e-pošti dobi novo začasno geslo. 4. Geslo lahko uporabniku spremeni tudi skrbnik. 5. Sprememba gesla se zapiše v dnevnik sprememb. Skrbnik laho pogleda zgodovino sprememb, ne vidi pa gesel. 6. Skrbnik lahko dodeli poljubnemu uporabniku vlogo skrbnika in mu spremeni status aktivnosti. 7. Možen je vpogled in iskanje v tabeli uporabnikov. 8. Ob vsaki prijavi aplikacija preveri pravice uporabnika.
Razširitve:	• *Prijavni obrazec* • *Obrazec za registraracijo* • *Obrazec za urejanje uporabnikov* • *Obrazec z detajli uporabnika* • *Obrazec z vsebino dnevnika sprememb*
Pogostost uporabe:	*Približno 100 dostopov dnevno.*
Status:	*V razvoju*
Lastnik:	*javni, anonimni uporabnik*
Prioriteta:	*visoka*



Slika 15.2: Logični model podatkov.

15.4.3 Relacijski model podatkov

Preslikavo logičnega modela v relacijski model je možno izvesti z orodjem Oracle SQL Data Modeler. Funkcija se imenuje *Engineering to relational*. Rezultat preslikave, to je relacijski model podatkov, je prikazan na sliki 15.3.

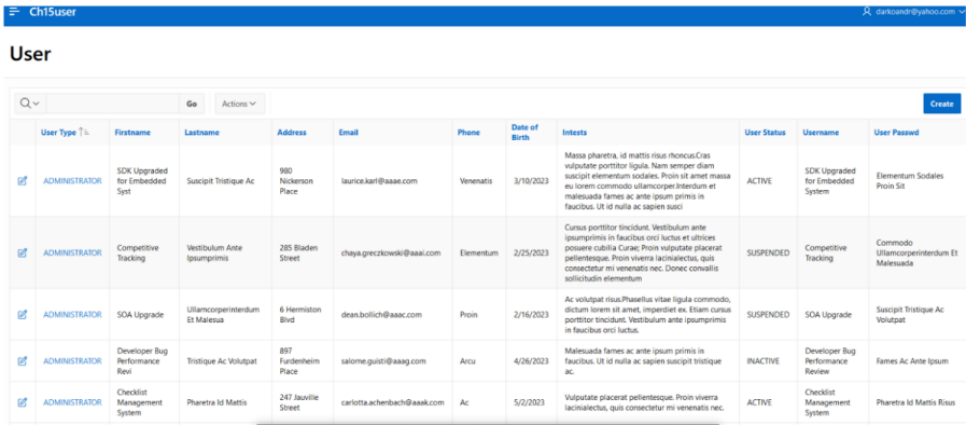


Slika 15.3: Relacijski model podatkov.

Z orodjem Oracle SQL Developer Data Modeler nato generiramo skript ((*File* > *Export* > *DDL File*). Nato v APEX-u uvozimo skript in ga izvedemo.

15.5 Uporabniški vmesniki

Zgodovina sprememb je prikazana na sliki 15.4.



Slika 15.4: Zgodovina sprememb podatkov uporabnika.

Zgodovina sprememb podatkov uporabnika je prikazana v tabelarični obliki. Napredne funkcionalnosti so prikazane v poglavju 18.

15.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- skripti za kreiranje in brisanje tabel ter za dodajanje testnih podatkov v tabele
- izvožena aplikacija
- video vodič

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter15 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

15.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket in uporablja zgolj dve tabeli (ch15_user in ch15_user_history). Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

15.6.2 Video vodiči

Video vodiči natančno prikazujejo razvoj aplikacije.

15.7 Vprašanja

1. Kako APEX zagotavlja avtentikacijo in avtorizacijo?
2. Kako je možno vključiti zunanjo avtentikacijo?
3. Kakšne so prednosti za uporabnike, ko se prijavijo v APEX-ovo aplikacijo s poverilnicami za Facebook ali Google?

15.8 Odgovori

1. APEX ima vgrajene mehanizme za avtentikacijo in avtorizacijo. Poleg tega omogoča povezavo z zunanjimi storitvami avtentikacije in avtorizacije. Vgrajena avtentikacija temelji na uporabniškem imenu in gesli.Avtorizacijo omogočajo vgrajene avtorizacijske sheme. Vnaprej pripravljene vloge so skrbik, založnik in bralec. Prav tako je možno na posamezni

strani ali komponenti strani določiti pravico dostopa za določeno vlogo.

2. Uporabite lahko možnosti prijave v Facebook, Google in druge storitve. Številne priljubljene platforme ponujajo razvijalske API-je (vmesnike za programiranje aplikacij), ki vam omogočajo integracijo njihove funkcije prijave v vašo aplikacije ali spletna mesta. Na primer, Facebook ponuja Facebook Login API, ki uporabnikom omogoča prijavo v vašo aplikacijo ali spletno mesto s svojimi Facebook poverilnicami. Podobno Google ponuja Google Sign-In API, ki uporabnikom omogoča prijavo z njihovimi Google računi. Če želite uporabiti te možnosti prijave, morate običajno registrirati svojo aplikacijo ali spletno mesto na portalu za razvijalce ustrezne platforme. Ta postopek registracije običajno vključuje pridobitev ključev API ali ID-jev strank, ki jih boste nato uporabili v svoji kodi za preverjanje pristnosti uporabnikov in upravljanje postopka prijave.
3. Z vključitvijo možnosti prijave z navedenih storitev lahko svojim uporabnikom ponudite priročen in znan način prijave v vašo aplikacijo ali spletno mesto, ne da bi morali ustvariti nove račune ali si zapomniti dodatne poverilnice za prijavo. Ni se treba registrirati in vnašati imena, naslova in podobnih podatkov, ne ustvarjajo se nova uporabniška imena in gesla, ki si jih je treba zapomniti.



16. Sistem malih inovacij

R. Leskovar, U. Rajkovič, A.Baggia; prevod R. Leskovar

16.1 Poslovni vidik primera

V sodobnem podjetju so inovativne ideje ključ do uspeha in izboljšanja poslovne uspešnosti. Zaradi poznavanja poslovnih procesov so zaposleni verjetno najboljši vir inovativnih idej. Te ideje vključujejo različne predloge za optimizacijo procesov, izboljšave upravljanja odnosov s strankami, nadgradnjo zagotavljanja kakovosti in podobno. V konkretnem podjetju so se odločili, da bodo te male inovativne ideje zbirali in vpeljevali v prakso, inovatorje pa glede na vpliv inovacije na poslovanje nagradili.

16.2 Definicija problema

Inovativne ideje v podjetju so zbirali že prej, a po e-pošti in na papirnatih obrazcih. Pregled idej v podjetju je bil zamuden. Merila ocenjevanja za ocenjevanje idej niso bila opredeljena. Zaradi velikega števila potencialnih idej, ki bi lahko bile koristne za podjetje, je bil potreben dober pregled. V podjetju so ugotovili, da potrebujejo spletno platformo, ki bo omogočala zbiranje inovativnih idej. Zaposleni naj bi se prijavili v platformo ter vnesli svoje ideje s preprostimi opisi ali priponkami. Ker ideje nimajo enake vrednosti za vodstvo oziroma uspešnost poslovanja, jih bodo pregledali in ovrednotili ocenjevalci. Ko bodo ocenjevalci zaključili postopek, bo ideja ocenjena kot sprejeta, zavrnjena ali pa je potrebna popravkov. Najbolj inovativne ideje bodo nagrajene glede na prejeto oceno. Predlogi bodo razvrščeni v štiri skupine: zlate (nagrada 100 EUR), srebrne (nagrada 80 EUR), bronaste (nagrada 60 EUR), prispevki (nagrada 40 EUR) in zahvale (brez denarne nagrade). Ocenjevalci inovativnih idej bodo le vodje. Vodje bodo redno ideje ocenjevali enkrat mesečno. Vsi prijavljeni uporabniki bodo lahko pregledali ideje šele, ko bodo že ocenjene.

16.3 Primeri uporabe

16.3.1 Opis primera uporabe

Vsak zaposleni ima možnost predlagati ideje za inovacije preko spletnega obrazca. Ko so ideje zbrane, jih oceni vodstvo. Ideje so razvrščene glede na uporabnost in možnosti realizacije: Inovatorji za dobre ideje prejmejo denarno nagrado ali zahvalo.

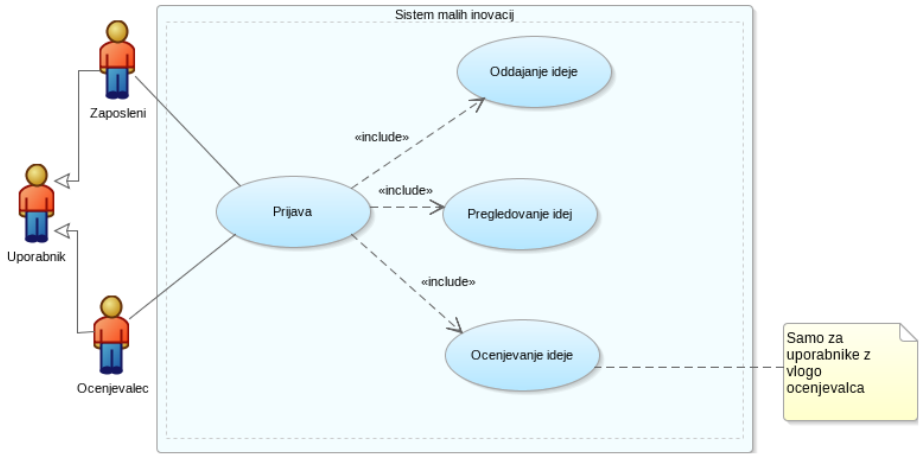
16.3.2 Delno strukturiran opis

Povzamemo lahko tri različne uporabniške zgodbe ali primere uporabe: oddaja ideje, pregled idej in ocenjevanje idej. Končni korak - plačilo nagrade za dobre ideje ni zajet v tem poslovnem primeru.

Dodajanje priloge ideji je pravzaprav razširitev zgoraj opisanega primera uporabe in bi ga lahko opisali kot nov primerek. Zaradi berljivosti nadaljujemo z drugima dvema glavnima primeroma uporabe, vendar bo aplikacija imela funkcijo za dodajanje prilog ideji.

16.3.3 Diagram primera uporabe

Opisani primeri uporabe so prikazani na diagramu primera uporabe na sliki 16.1.



Slika 16.1: Diagram primera uporabe.

Zaradi jedrnatosti smo razširitev primera uporabe Dodaj prilogo na sliki 16.1 opustili.

16.4 Model podatkov

16.4.1 Opis modela podatkov

V logičnem podatkovnem modelu so tri entitete. Entiteta CH16 IDEA ima osem atributov: ID, datum oddaje, naslov, opis ideje, ime datoteke ideje, tip MIME, odločitev in datum odločitve. Vsako idejo predlaga en zaposleni. V entiteti CH16 EMPLOYEE so shranjeni osnovni podatki o zaposlenih: ID, ime in priimek. Zaposleni je lahko vodja drugim zaposlenim. Vsak zaposleni ima samo enega vodjo, vodja pa lahko vodi več zaposlenih. Vodje ocenjujejo ideje. En vodja lahko oceni veliko idej, eno idejo pa lahko oceni samo en vodja. Vsaka ideja lahko prejme eno nagrado. Entiteta CH16 AWARD ima tri atribute: ID, opis nagrade in znesek v evrih. Ena vrsta nagrade se lahko podeli več idejam. Pravila ocenjevanja niso vključena v podatkovni model, ker bi to pomenilo tveganje spremembe aplikacije vsakič, ko se pravila spremenijo. S predlaganim podatkovnim modelom smo zagotovili: robustnost aplikacije, hiter razvoj in uvajanje, nizke stroške razvoja in vzdrževanja ter fleksibilnost.

16.4.2 Logični model podatkov

Opisani logični model podatkov je predstavljen na sliki 16.2.

Tabela 16.1: Opis primera uporabe: oddajanje ideje

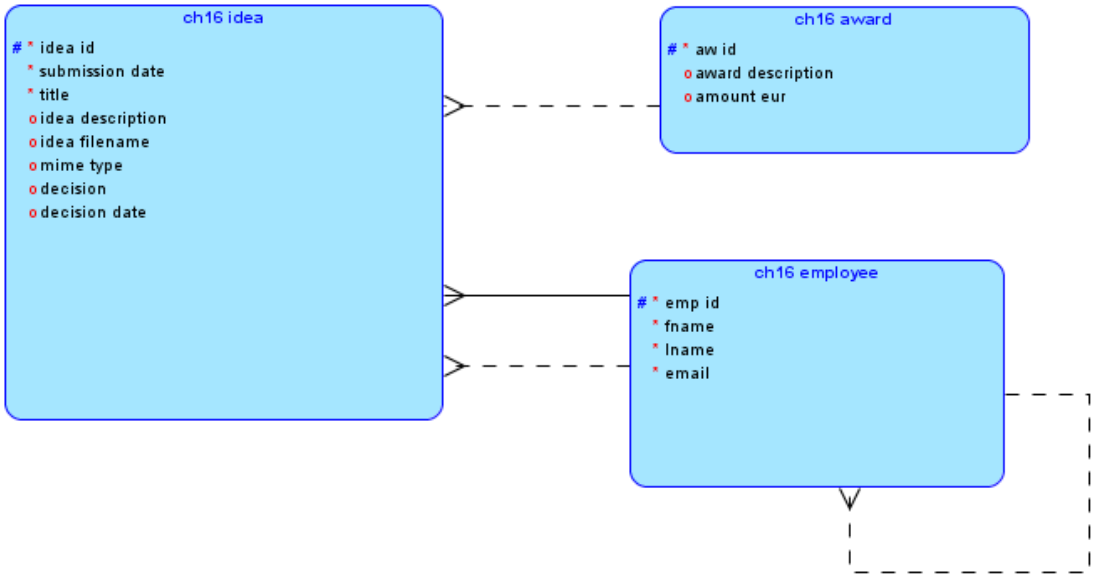
Ključni pojem	**Vrednost**
ID:	*Ch16-01*
Naslov:	*Oddaja ideje*
Opis:	*Zaposleni se prijavi na platformo in vpiše idejo. Vsaka ideja ima naslov in kratek opis. Opis je datoteka (besedilna, grafika ali stisnjena), ki jo zaposleni naloži s svojega službenega ali domačega računalnika.*
Glavni igralec:	*Zaposleni*
Predhodni pogoji:	*Zaposleni potrebuje uporabniški račun za aplikacijo Sistem malih inovacij v Oracle APEX. Omogočen mora biti dostop do spletne aplikacije.*
Pogoji po zaključku:	*Po uspešni oddaji predloga male inovacije, je ta dostopna ocenjevalcem.*
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v apliacijo Sistem malih inovacij* 2. *Na menuju izberite ideje (Ideas)* 3. *Izberite dodajanje nove ideje (Add new idea)* 4. *Vnesite naslov (Title) in dodajte prilogo (attachment)* 5. *Vpišite ime datoteke in izberite tip* 6. *Potrdite dodajanje s klikom na gumb (Create)* 7. *Preglejte vnesene podatke o inovativni ideji*
Razširitve:	• *1a. Prijava ni uspela.* • *1a* Razširitev:* • *1a1. Prikaži opis napake.* • *1a2. Odpri okno za prijavo.* • *4a. Napaka v tipu podatka.* • *4a* Razširitev:* • *4a1. Prikaži opis napake.* • *7a. Napaka v tipu podatka.* • *7a* Razširitev:* • *7a1. Prikaži opis napake,*
Pogostost uporabe:	*V poprečju zaposleni oddajo tri inovativne ideje na teden*
Status:	*Zaključen*
Lastnik:	*Zaposleni z dostopom do aplikacije*
Prioriteta:	*Zmerna*

Tabela 16.2: Opis primera uporabe: pregledovanje idej.

Ključni pojem	**Vrednost**
ID:	*Ch16-02*
Naslov:	*Pregled idej*
Opis:	*Zaposleni se prijavi na platformo in pregleda statuse oddanih predlogov malih inovacij.*
Glavni igralec:	*Zaposleni*
Predhodni pogoji:	*Zaposleni potrebuje uporabniški račun za aplikacijo Sistem malih inovacij v Oracle APEX. Omogočen mora biti dostop do spletne aplikacije.*
Pogoji po zaključku:	-
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v apliacijo Sistem malih inovacij* 2. *Na menuju izberite ideje (Ideas)* 3. *Preglejte poročilo o statusu oddanih predlogov* 4. *Izberite pregled (Overview) na menuju* 5. *Preglejte poročilo o oddanih idejah v grafični obliki*
Uspešen scenarij:	• *1a. Prijava ni uspela.* • *1a* Razširitev:* • *1a1. Prikaži opis napake.*
Pogostost uporabe:	*Možnost vpogleda je omogočena zaposlenim, ocenjevalcem in administratorjem. Predvidoma bo okoli 5 vpogledov dnevno.*
Status:	*Zaključen*
Lastnik:	*Zaposleni z uporabniškim računom.*
Prioriteta:	*Nizka*

Tabela 16.3: Opis primera uporabe: ocenjevanje ideje.

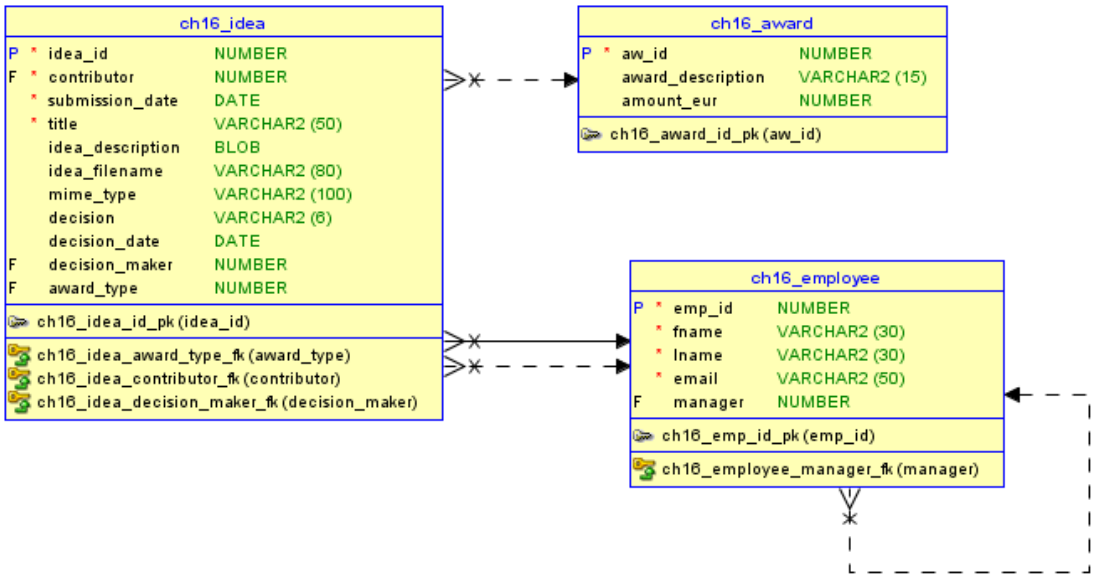
Ključni pojem	**Vrednost**
ID:	*Ch16-03*
Naslov:	*Ocenjevanje ideje*
Opis:	*Ocenjevalec se prijavi na platformo, pregleda idejo in jo oceni tako, da izbere stopnjo koristnosti.*
Primarni igralec:	*Ocenjevalec*
Predhodni pogoji:	*Zaposleni mora imeti vlogo ocenjevalca (CH16_Reviewer). Omogočen mora biti dostop do intranetne aplikacije.*
Pogoji po zaključku:	*Izbrana ideja je ocenjena z definirano nagrado.*
Glavni del	*Scenariji*
Uspešen:	1. *Odprite spletni brskalnik in se prijavite v apliacijo Sistem malih inovacij* 2. *Na menuju izberite ocenjevanje (Review)* 3. *Izberite idejo in dodajte oceno s klikom na gumb za urejanje (Edit)* 4. *Izberite splošno oceno - sprejem, zavrnitev ali popravek (Accept, Reject, Revise)* 5. *Izberite vrsto nagrade - zlata, srebrna, bronasta, prispevek ali zahvala (Gold, Silver, Bronze, Contributor, Thank you)* 6. *Potrdite oceno z gumbom (Confirm)*
Extensions:	• *1a. Prijava ni uspela.* • *1a* Razširitev:* • *1a1. Prikaži opis napake.* • *1a2. Odpri okno za prijavo.* • *2a. Napaka v tipu podatka.* • *2a* Razširitev:* • *2a1. Prikaži opis napake.* • *3a. Napaka v tipu podatka.* • *3a* Razširitev:* • *3a1. Prikaži opis napake.* • *4a. Napaka v tipu podatka.* • *4a* Razširitev:* • *4a1. Prikaži opis napake.* • *5a. Napaka v tipu podatka.* • *5a* Razširitev:* • *5a1. Prikaži opis napake.*
Pogostost uporabe:	*Vodje ocenjujejo prispele ideje enkrat mesečno.*
Status:	*Zaključen*
Lastnik:	*Ocenjevalec*
Prioriteta:	*Visoka*



Slika 16.2: Logični model podatkov.

16.4.3 Relacijski model podatkov

Preslikavo logičnega modela v relacijski model je možno izvesti z orodjem Oracle SQL Data Modeler. Funkcija se imenuje *Engineer to relational*. Rezultat preslikave, to je relacijski model podatkov, je prikazan na sliki 16.3.



Slika 16.3: Relacijski model podatkov.

Oracle SQL Data Modeler lahko na zahtevo generira tudi skript za kreiranje tabel, sekvenc in prožilcev. V orodju izberemo vse tabele in uporabimo funkcijo *File* > *Export* > *DDL File*. Del generiranega skripta zgleda takole:

```
CREATE TABLE CH16_AWARD (
    AW_ID NUMBER NOT NULL
```

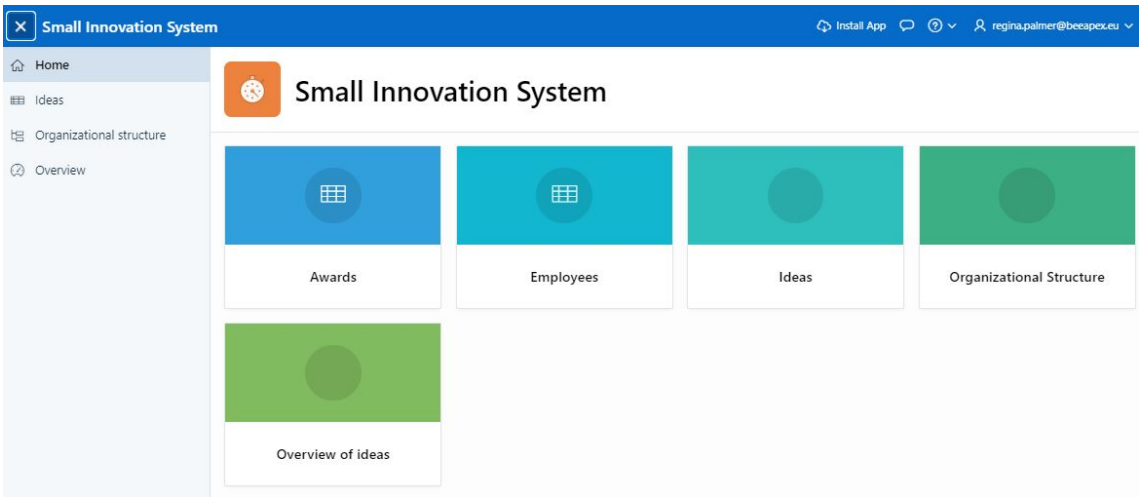
```
...
);

CREATE TABLE CH16_EMPLOYEE (
    EMP_ID NUMBER NOT NULL
...
);
```

Zdaj je čas, da tabele kreiramo v bazi Oracle. V APEX-u uvozimo skript in ga izvedemo.

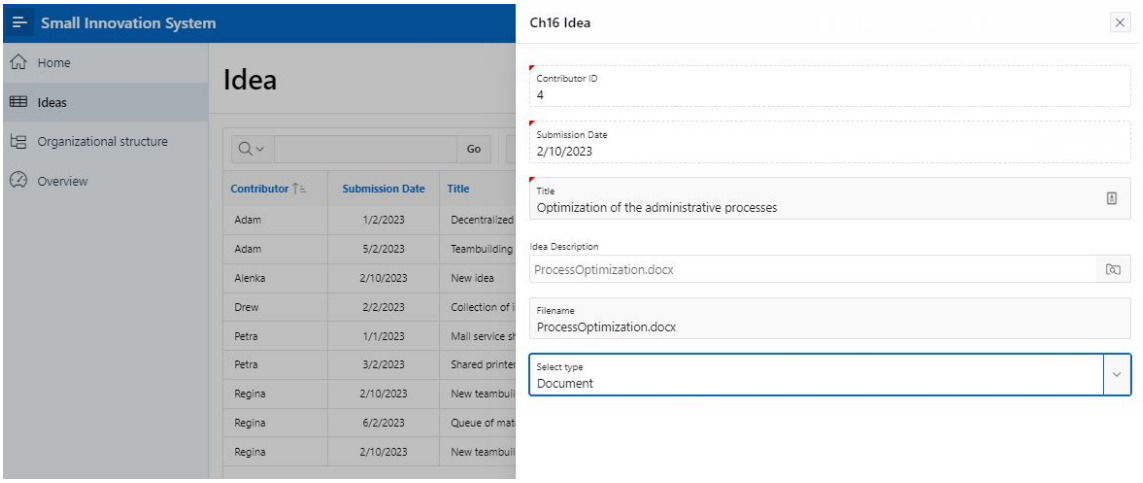
16.5 Uporabniški vmesniki

Na naslednjih slikah prikazujemo uporabniški vmesnik za vlogo zaposlenega in ocenjevalca. Pregledna plošča zaposlenega je prikazana na sliki 16.4.



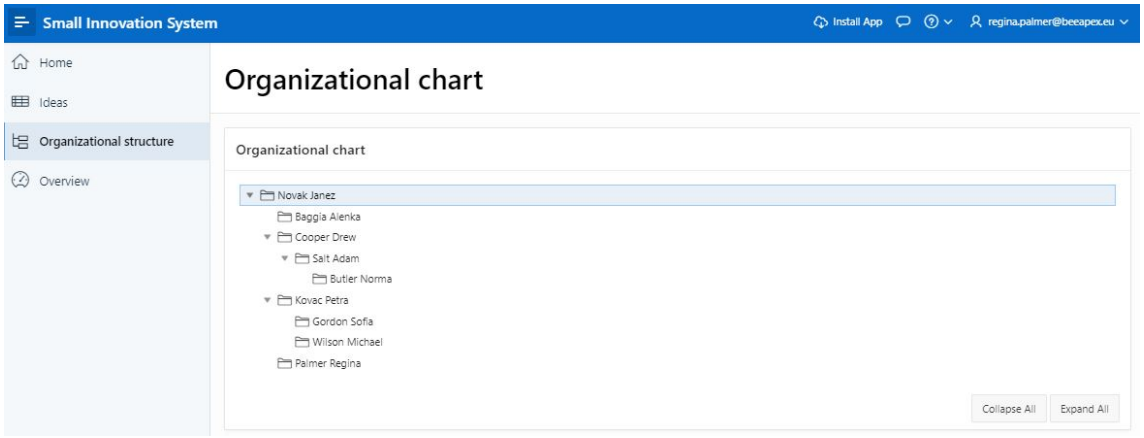
Slika 16.4: Pregledna plošča za zaposlenega.

Stran za oddajanje ideje s prilogami je prikazana na sliki 16.5.



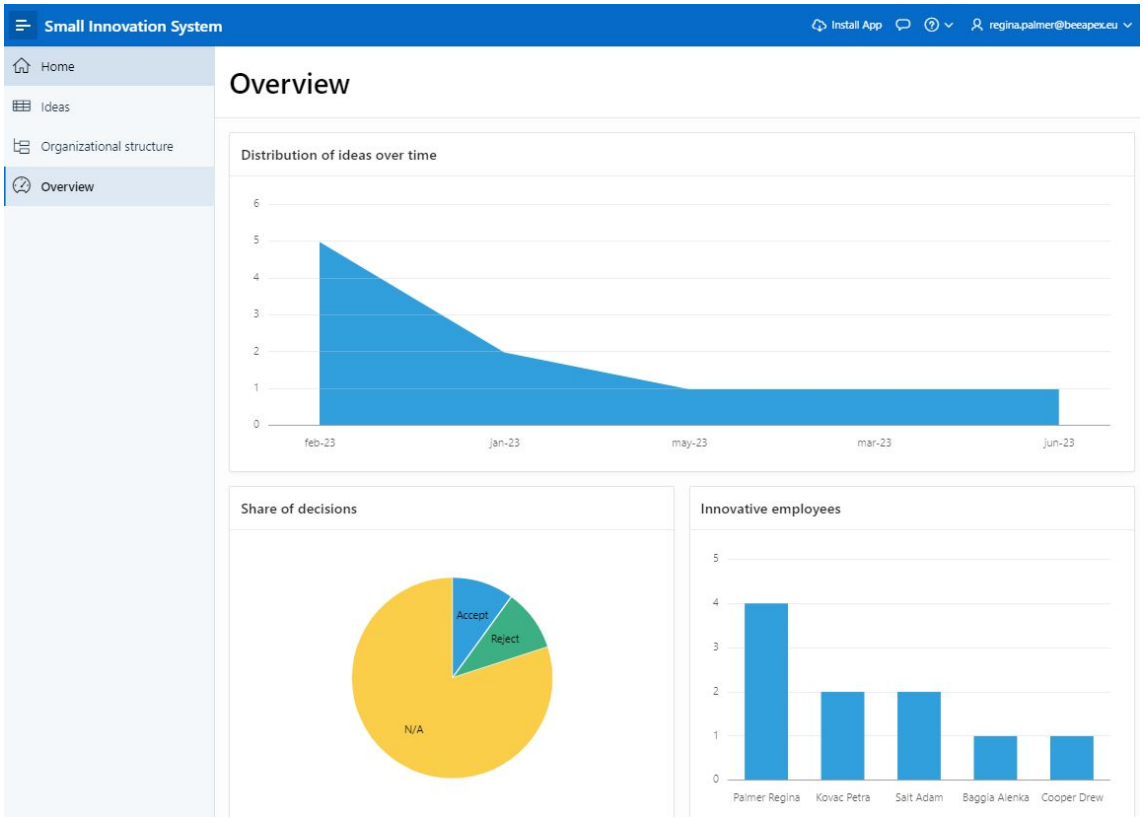
Slika 16.5: Oddajanje ideje s prilogami.

Zaposleni lahko pregleda organizacijsko strukturo podjetja (slika 16.6).



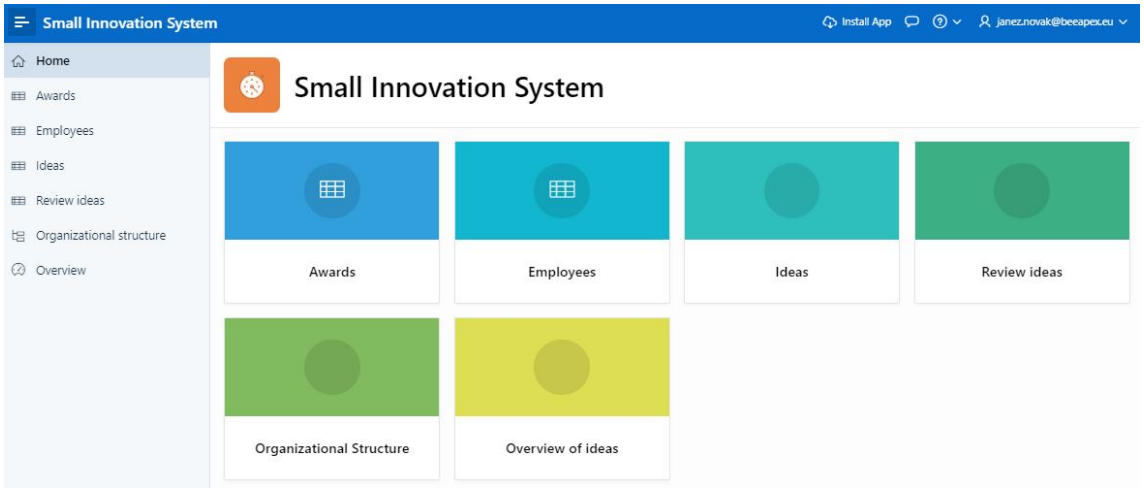
Slika 16.6: Organizacijska struktura podjetja.

Vsak zaposleni lahko na pregledni plošči pogleda grafikone, ki prikazujejo porazdelitev idej po času oddaje, delež sprejetih, zavrnjenih ter popravkov. Prav tako je prikazan podatek o zaposlenem, ki je oddal največ predlogov inovacij (slika 16.7).



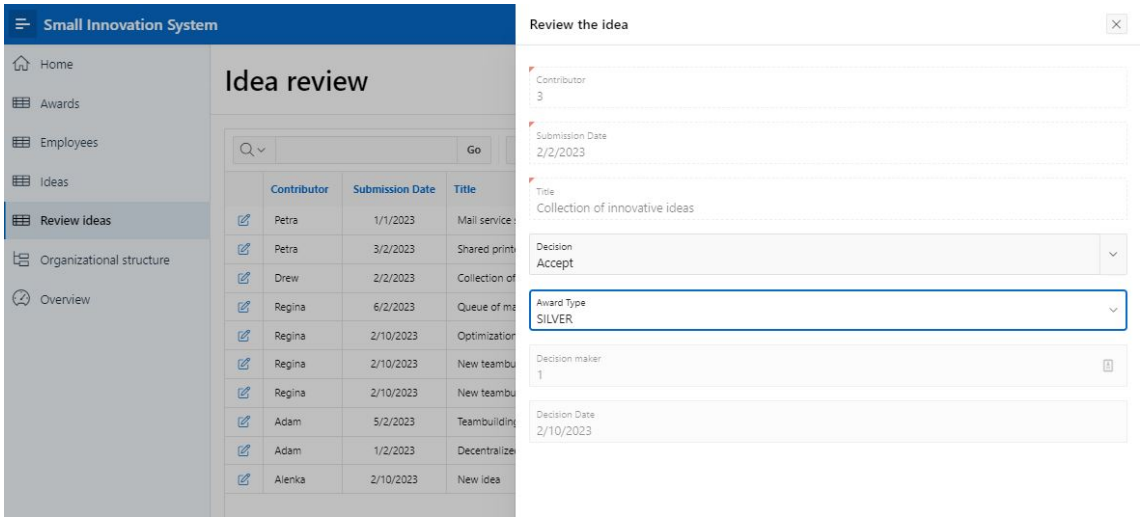
Slika 16.7: Vpogled v oddane inovacijske ideje.

Ocenjevalec lahko spreminja podatke o nagradah in zaposlenih (slika 16.8). Stran za administracijo aplikacije je dostopna le skrbniku (ne ocenjevalcem).



Slika 16.8: Pregledna plošča ocenjevalca.

Stran za ocenjevanje idej je prikazana na sliki 16.9. Dostop do te strani je omogočen samo ocenjevalcem. Ocenjevalec izbere ponujene možnosti odločitve in nagrade, kadar je taka odločitev.



Slika 16.9: Stran za ocenjevanje idej.

16.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- skripti za kreiranje in brisanje tabel ter za dodajanje testnih podatkov v tabele
- izvožena aplikacija
- video vodič za generiranje aplikacije iz skripta

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter16 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

16.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih

objektov, ki so specifični za to konkretno aplikacijo.

Izvožena aplikacija je preskušena in bo delovala v novem delovnem prostoru, če je izpolnjen naslednji pogoj:

- pred zagonom aplikacije je potrebno dodati uporabnika v APEX.

Če uporabniku ni dodeljena ustrezna vloga, se aplikacija zruši. Če se vam to zgodi, počistite piškotke v spletnem brskalniku, na primer v brskalniku Firefox: Settings > Cookies and Site Data > Manage Data.

16.6.2 Video vodiči

Video vodič natančno prikazuje razvoj aplikacije.

16.7 Vprašanja

1. Kje moramo definirati vloge za uporabnike v tej aplikaciji?
2. Katera povezava v logičnem modelu podatkov omogoča prikaz organizacijske strukture (organigram) v tej aplikaciji?
3. Zakaj so nekatera polja na strani za pregled idej onemogočena v tej aplikaciji?

16.8 Odgovori

1. V tej aplikaciji niso definirane vloge uporabnikov v modelu podatkov, temveč uporablja mehanizem vlog, ki je vgrajen v APEX.
2. Rekurzivna povezava na entiteti (eniteta povezana s samo seboj) omogoča predstavitev organizacijske strukture.
3. Nekatera polja so onemogočena, da bi uporabniku zagotovili boljšo uporabniško izkušnjo in konsistentnost podatkov. Uporabnik ne more vnašati podatkov, ki so že shranjeni v bazi podatkov.



17. Vodenje poslovnih procesov

R. LESKOVAR, U. RAJKOVIČ, A.BAGGIA; PREVOD R. LESKOVAR

17.1 Poslovni vidik primera

Večina te knjige poudarja pomen podatkov in pristopa, pri katerem razvoj spletnih aplikacij z APEX-om temelji na podatkovno vodenem pristopu. Toda pristop, ki temelji na podatkih, ima dvojčka: procesno voden razvoj aplikacije. Kot že ime pove, se procesno vodene aplikacije osredotočajo na procese. Tako kot dobro vodena podatkovno temelječa aplikacija ne more zanemariti procesnega vidika, tako tudi procesno vodena aplikacija ne more zanemariti podatkovnega vidika. Implicitna ali eksplicitna vključitev obeh poskrbi, da aplikacija ustreza namenu. Za razvijalca je pomembno, da ima poglobljen vpogled v poslovne procese v organizaciji. Vsaka vrsta organizacije, ne glede na velikost, letni zaslužek, lastništvo, panogo in poslanstvo, izvaja poslovne procese. Skupine poslovnih procesov, ki se izvajajo v večini organizacij, so:

- Prodaja in trženje vključujeta podprocese, kot so pridobivanje in ohranjanje strank, pridobivanje potencialnih strank, upravljanje odnosov s strankami in tržne akcije.
- Upravljanje dobavne verige vključuje podprocese, kot so koordinacija in upravljanje pretoka blaga in storitev, nabava, proizvodnja in distribucija.
- Finančno upravljanje vključuje podprocese, kot so upravljanje finančnih virov organizacije, računovodstvo, proračun in finančno poročanje.
- Upravljanje človeških virov vključuje podprocese, kot so upravljanje delovne sile organizacije, zaposlovanje, usposabljanje in razvoj zaposlenih.
- Storitev za stranke vključuje podprocese, kot so zagotavljanje podpore in pomoči strankam, reševanje pritožb in reševanje težav.
- Operativno vodenje vključuje podprocese, kot so upravljanje vsakodnevnih poslovnih operacij, proizvodnja, logistika in upravljanje zalog.
- Informacijska tehnologija vključuje podprocese, kot je upravljanje virov in sistemov informacijske tehnologije organizacije, strojne opreme, programske opreme in upravljanje podatkov.

Ti procesi veljajo za kritične, saj podpirajo temeljne funkcije organizacije in omogočajo doseganje njenih ciljev. Izboljšanje učinkovitosti in uspešnosti teh procesov lahko pomembno vpliva na uspešnost celotne organizacije. V poskusu jasnega razumevanja in definicije tipičnega poslovanja je OASIS OPEN razvil univerzalni poslovni jezik, ki opisuje procese in poslovne dokumente (glejte [5]). Industrija programske opreme se je začela v poznih devetdesetih letih prejšnjega stoletja z integracijo procesov, izumom posebnih jezikov za opis in izvajanje procesov:

Business Process Execution Language (BPEL) in kasneje Business Process Modeling Language (BPML). Veliki igralci, kot so Microsoft, IBM, SAP, Oracle, pa tudi nešteto manjših, so prispevali k programskim platformam, katerih namen je bil zagotoviti gladko integracijo poslovnih funkcij. Poleg industrije programske opreme so se druge veje v osemdesetih letih prejšnjega stoletja osredotočile na procese s pobudami in standardi za upravljanje kakovosti.

Od leta 1990 imata področji upravljanja poslovnih procesov (ang. business process management, kratica BPM) in prenove poslovnih procesov (ang. business process re-engineering, kratica BPR) pomembno vlogo, ko želijo organizacije izboljšati svojo uspešnost. BPM se osredotoča na predstavitev in dokumentacijo delovnih tokov, operacij in procesov podjetja z uporabo diagramov, vizualizacijo poteka ali drugih vizualnih pripomočkov. Cilj je razumeti in izboljšati obstoječe procese ali oblikovati nove, da bi povečali učinkovitost in zmanjšali neučinkovitosti. BPR je sprejet kot temeljni premislek in radikalno preoblikovanje poslovnih procesov za doseganje bistvenih izboljšav v uspešnosti in produktivnosti. Vključuje analizo in oblikovanje delovnih tokov in procesov znotraj organizacije, s ciljem, da bi bila bolj učinkovita, uspešna in sposobna prilagajanja spreminjajočemu se poslovnemu okolju. BPR lahko vključuje tehnologije in informacijske sisteme za avtomatizacijo procesov in izboljšanje odločanja.

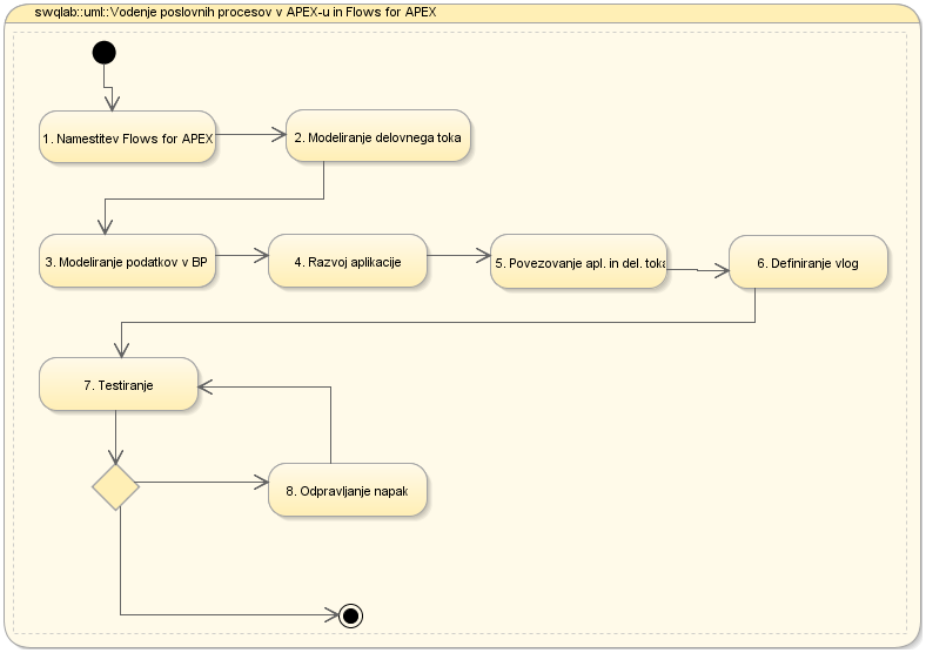
V BPM in BPR je mogoče uporabiti več diagramov:

- Diagrami poteka: preprost in intuitiven način za predstavitev procesa kot niza korakov, odločitev in zank.
- Diagrami stez: vrsta diagrama poteka, ki doda vizualno predstavitev tega, kdo ali kaj je odgovoren za vsak korak v procesu.
- Zemljevidi procesa: visokonivojski pogled na ravni korakov in dejavnosti, vključenih v proces, ki pogosto vsebuje vhode, izhode in odločitvene točke.
- BPMN (model in zapis poslovnih procesov [4]): industrijski standardni zapis, ki se uporablja za modeliranje poslovnih procesov; standardiziral ga je Object Management Group (OMG).
- EPC (veriga procesov, ki temelji na dogodkih): vrsta diagrama poteka, ki namesto dejavnosti uporablja dogodke kot gonilo procesa.
- IDEF (definicija integracije za modeliranje funkcij): metoda, ki se uporablja v programskem inženirstvu za modeliranje in analizo procesov in sistemov.

Izbira vrste diagrama za predstavitev poteka dela je odvisna od ravni potrebnih podrobnosti, namena modela in ciljne publike. Na tej točki se bomo osredotočili samo na BPMN. Glavni razlogi so: a) BPMN ima standardno notacijo in je široko sprejet v različnih panogah in b) BPMN je implementiran kot vrsta diagrama modeliranja v Oracle Application Express (APEX) funkciji (ali razširitvi) imenovani *Flows for APEX*. Flows za APEX je odprtokoden, licenciran pod zelo permisivno licenco MIT. *Flows for APEX* razvijalcem omogoča gradnjo in uvajanje spletnih aplikacij z uporabo vizualnega vmesnika povleci in spusti. *Flows for APEX* omogoča avtomatizacijo in poenostavitev zapletenih poslovnih procesov. Procese razdeli na vrsto korakov in nalog ter uporabnike vodi skozi po teh korakih in nalogah.

Flows for APEX lahko vključuje različne elemente, kot so strani, obrazci, poročila in pogovorna okna in jih je mogoče konfigurirati tako, da vključujejo logiko razvejanja, pogojno razvejanje, preverjanje veljavnosti podatkov in obravnavanje napak. Vizualni vmesnik *Flows for APEX* razvijalcem olajša načrtovanje, gradnjo in testiranje poslovnih procesov brez potrebe po obsežnem kodiranju. Lahko se integrira z drugimi komponentami Oracle APEX, kot so storitve SQL, PL/SQL in REST, za ustvarjanje bogatih spletnih aplikacij, kot tudi za izdelavo rešitev po meri za specifične poslovne potrebe, kot je upravljanje delovnega toka, uvajanje strank in upravljanje podatkov. Na splošno *Flows for APEX* zagotavlja organizacijam način za racionalizacijo in avtomatizacijo zapletenih poslovnih procesov, izboljšanje produktivnosti ter zmanjšanje napak in neučinkovitosti. Odlične vadnice o *Flows for APEX* so na voljo na tej spletni strani. Torej, kako lahko povežemo APEX kot malokodno programsko okolje in *Flows for APEX* z vdelano standardno predstavitvijo BPMN?

Diagram aktivnosti (slika 17.1) prikazuje poenostavljeno predstavitev integracije Flows in aplikacije APEX.



Slika 17.1: Integracija Flows for APEX z aplikacijo APEX.

Koraki *3. modeliranje podatkov (ang. modeling data), 4. razvoj aplikacije (ang. developing application), 6. definiranje uporabniških vlog (ang. defining user roles), 7. testiranje (ang. testing)* in *8. odstranjevanje napak iz aplikacije (ang. removing errors from application)* so običajne za vse aplikacije APEX. Korak *1. namestitev Flows for APEX (ang. installing Flows for APEX)* je podobna namestitvi katere koli druge pakirane aplikacije. Prave novosti so *2. modeliranje delovnega toka (ang. modeling the workflow)* in *5. povezovanje aplikacije APEX z delovnim tokom (ang. linking APEX application with workflow)*.

Bralcem, ki niso seznanjeni z diagrami za upravljanja poslovnih procesov (BPM) in modeliranja poslovnih procesov (BPMN), svetujemo, da poiščejo zadevne učbenike in preučijo že omenjeno specifikacijo BPMN [4]. Kot kratek opomnik za vse bralce, kratko predstavljamo le najpogosteje uporabljene elemente diagramov BPMN:

- Začetni dogodek (ang. Start event): označuje začetno točko procesa.
- Naloga (ang. Task): predstavlja eno enoto dela, kot je dejavnost ali korak v procesu.
- Prehod (ang. Gateway): predstavlja točko odločitve v procesu, kot je razcep v toku ali pogojna veja.
- Zaporedje (ang. Sequence): povezuje elemente v diagramu in predstavlja tok nadzora za prehod od enega elementa do drugega.
- Končni dogodek (ang. End event): označuje konec procesa.
- Bazen/Steza (ang. Pool/Lane): predstavlja združevanje povezanih nalog, ki se uporabljajo za določanje vlog in odgovornosti različnih udeležencev v procesu.
- Tok sporočil (ang. Message flow): predstavlja tok sporočil med udeleženci v procesu, kot je komunikacija med dvema sistemoma.
- Podatkovni objekt (ang. Data object): predstavlja del podatkov, ki se uporablja ali generira v procesu, kot je račun ali zapis stranke.
- Podatkovna shramba (ang. Data Store(): predstavlja vsebnik za shranjevanje podatkov, ki se uporabljajo ali proizvajajo v procesu, kot je baza podatkov ali datotečni sistem.

Te elemente je mogoče kombinirati za ustvarjanje podrobnih in natančnih modelov poslovnih procesov, ki organizacijam omogočajo, da prepoznajo področja za izboljšave in optimizirajo svoje delovne tokove.

17.2 Priprava za delo z vtičnikom Flows for APEX

17.2.1 Namestitev Flows for APEX

Prenesite datoteko zip iz tega URL in razpakirajte vse datoteke v svoj računalnik. Sledite navodilom na tej povezavi. Preden namestite vtičnik Flows for APEX, preverite zahteve. Zahteve so izpolnjene, če uporabljate brezplačen delovni prostor APEX, brezplačen/plačljiv račun OCI ali račun Oracle Academy. Dve dodatni nalogi po namestitvi sta:

- dodelite pravico "create job" delovnemu prostoru. Vi ali skrbnik instance APEX morate vzpostaviti povezavo z bazo podatkov (z vmesnikom OCI, SQL Developer, SQL Plus, TOAD) in izvesti ukaz. Tega privilegija ne morete dodeliti znotraj APEX.
- konfigurirajte delovni prostor, privzeto aplikacijo in podatke o uporabniku znotraj aplikacije "Flows for APEX". Ta korak lahko izvedete kadar koli pozneje, ko razvijete lastno procesno vodeno aplikacijo.

Čeprav se včasih imenuje funkcija ali vtičnik, bo namestitev ustvarila nove objekte baze podatkov (tabele, poglede) in aplikacijo, imenovano "Flows for APEX". Ta aplikacija je skrbniški vmesnik, ki vam omogoča:

- spremljanje primerkov delovnih tokov na nadzorni plošči (ang. Dashboard),
- vizualno ustvarjanje novih delovnih tokov (ang. Flow Management) in
- izvajanje delovnih tokov z ustvarjanjem primerkov, dokončanjem nalog, ki pripadajo primerkom in preizkušanjem delovnih tokov (ang. Flow Monitor).

17.2.2 Namestitev aplikacije Sample Process Flow

Primer upravljanja procesa je v isti datoteki zip Flows for APEX. Namestitev je enostavna. Primer ponazarja povrnitev stroškov zaposlenemu. Zaposleni pripravi zahtevek za povrnitev stroškov (npr. potovanja, plačila nočitve), ki ga potrdi vodja. Če je zahtevek zavrnjen, je delavec obveščen. Za odobrene zahtevke računovodstvo pripravi plačilo, izvede bančno transakcijo in nastavi status plačila. Če je zahtevek plačan, je primerek tega delovnega toka končan. Aplikacija je zelo podobna resničnemu procesu v kateri koli organizaciji. Ker je tako dobro dokumentiran, predstavlja odlično študijsko gradivo.

17.2.3 Preberite in vadite na priloženih primerih

Za Flows for APEX obstajajo odlična študijska gradiva. Začnite na tej strani. Za poglobitev poznavanja BPMN preštudirajte to stran in nato vodič za integracijo z APEX.

Z branjem in vadbo boste lahko razvili lastno aplikacijo, ki temelji na procesnem pristopu. Vse vaje zahtevajo nekaj ur zbranega študija. Bralce želimo spodbuditi k izkušnji učenja s praktičnim preskusom. Preostali del tega poglavja bo vseboval isto predlogo kot druga poglavja v tem učbeniku (od definicije problema do uporabe) z dodatno vključitvijo BPMN in integracijo aplikacije s Flows for APEX. Obseg aplikacije, razvite v poglavju 17, je skrčen na osnovno funkcionalnost in preskoči podrobnosti avtorizacije. V tem poglavju bomo pokazali:

- definiranje delovnega toka s Flows for APEX
- zagon primerka definiranega procesa znotraj razvite aplikacije
- spremljanje primerka primerka delovnega toka v razviti aplikaciji
- dokončanje vseh nalog primerka delovnega toka v razviti aplikaciji

Tudi tako skrčena aplikacija lahko predstavlja lep izziv za navdušenega učenca, kajne?

17.3 Definicija problema

Srednje veliko proizvodno podjetje prejema povpraševanja strank za izdelke po meri. Vsako povpraševanje ima lahko več povezanih dokumentov s specifikacijami, zahtevanimi standardi, shemami, skicami ipd. Stranka ima lahko več povpraševanj. Komercialist pripravi in razvrsti povpraševalne dokumente v tri kategorije: proizvodno, finančno in poslovno. Za oceno ene poizvedbe mora komercialist pripraviti več dokumentov za odločevalce. Vodja proizvodnje in njegova ekipa ocenita zmožnost proizvodnje. Zagotavljajo strokovno mnenje o sposobnosti podjetja, da proizvede želeni izdelek v želenem roku in količinah. Če je izdelek mogoče izdelati, potem ekipa oceni predvideno trajanje proizvodnega procesa v dnevih. Finančni direktor in njegova ekipa ocenita finančni vidik poizvedbe. Podajo strokovno mnenje o sposobnosti podjetja za financiranje in oceno pričakovanega dobička. Končno oceno in odločitev sprejme izvršni direktor s svojo ekipo. Upoštevajo izvedljivost (proizvodno, finančno), ocenijo pomembnost kupca in se odločijo glede povpraševanja. Glede na zmožnosti podjetja za proizvodnjo in financiranje poslovne priložnosti ter poslovne perspektive se generalni direktor odloči, kako bo odgovoril kupcu: a) poslal ponudbo s ceno in pričakovanim rokom dobave ali b) sporočil kupcu, da proizvajalec ne more izvesti posla.

Slabe poslovne odločitve (pošljanje poslovnega predloga ali zavrnitev potencialnega posla) imajo lahko katastrofalne posledice za proizvodno podjetje: izgubljeni zaslužek, nizek dobiček, slab ugled zaradi prekoračenih dobavnih rokov in celo stečaj. Predstavljena situacija povzroča preglavice vsem vpletenim: komercialistu, vodji proizvodnje, finančnemu direktorju in izvršnemu direktorju. Oddelek IT je odgovoren za zagotavljanje uporabniku prijazne, zanesljive in varne programske podpore za opisani potek dela. Njihov problem je razviti takšno podporo s Flows za APEX.

Če želite razviti tako aplikacijo, morate imeti nameščen Flows for APEX. V tem poglavju bomo prikazali analizo poslovnega primera (opis, polstrukturiran opis, diagram primera uporabe, procesni model, podatkovni model) in razvili delujočo aplikacijo. Bralec se bo naučil:

- kako definirati model procesa (delovni tok) za ta poslovni primer,
- kako ustvariti podatkovni model in razviti aplikacijo v APEX-u in
- kako povezati razvito aplikacijo z definiranim delovnim tokom.

Upoštevajte, da avtorizacija uporabnikov ne bo detajlno prikazana, ker bi to poglavje še dodatno razširilo.

17.4 Primeri uporabe

17.4.1 Opis primera uporabe

V štirih primerih uporabe sodelujejo štirje akterji. Vsak udeleženec je odgovoren za en primer uporabe: komercialist pripravi dokumentacijo v zvezi s povpraševanjem, vodja proizvodnje zagotovi oceno proizvodnih vidikov, finančni direktor zagotovi oceno finančnih vidikov in izvršni direktor sprejme odločitev. Diagram primerov uporabe (glejte sliko 17.2) predpostavlja, da se bo oblikovalec delovnega toka odločil, ali naj proizvodni in finančni vodja delujeta vzporedno ali zaporedno. Če delata vzporedno in se eden ali oba odločita, da zahtev poizvedbe ni mogoče izpolniti, se pojavi nekaj nepotrebnega truda (stroškov), vendar je delovni tok mogoče dokončati hitreje. Če se oblikovalec delovnega toka odloči za zaporedni pristop, bo delovni tok zahteval več časa, vendar bo manj nepotrebnega truda.

17.4.2 Delno strukturiran opis

UML predpostavlja, da ima vsak primer tudi delno strukturiran besedni opis. Za štiri primere uporabe so besedni opisi prikazani v tabelah 17.1, 17.2, 17.3 in 17.4.

Tabela 17.1: Opis primera uporabe: priprava dokumentacije za povpraševanje.

Ključni pojem	**Vrednost**
ID:	*ch17-01*
Naslov:	*Priprava dokumentacije za povpraševanje*
Opis:	*Komercialist doda novo povpraševanje in pripadajočo dokumentacijo*
Glavni igralec:	*Komercialist*
Predhodni pogoji:	Potencialni kupec je že poslal dokumentacijo, povezano s povpraševanjem
Pogoji po zaključku:	-
Glavni del	-
Uspešen scenarij:	1. Dodaj novo povpraševanje 2. Dodaj pripadajoče dokumente kot prilogo povpraševanju 3. Potrdi vnos z gumbom “Confirm” ali prekini z gumbom “Cancel”
Razširitve:	-
Pogostost uporabe:	*Približno 250 letno oz. 5 tedensko.*
Status:	[V razvoju]
Lastnik:	Komercialist
Prioriteta:	*Visoka*

Tabela 17.2: Opis primera uporabe: ocenitev proizvodnega vidika povpraševanja.

Ključni pojem	**Vrednost**
ID:	*ch17-02*
Naslov:	*Ocenitev proizvodnega vidika povpraševanja*
Opis:	*Vodja proizvodnje in njegova/njena ekipa podajo strokovno mnenje, ali je podjetje sposobno izdelati izdelek in če da, koliko časa (v dnevih) bi trajala izdelava dane količine. Utemeljitev mnenja je obvezna.*
Glavni igralec:	*Vodja proizvodnje*
Predhodni pogoji:	Dokumentacija o povpraševanju je pripravljena
Pogoji po zaključku:	-
Glavni del	-
Uspešen scenarij:	1. Vnesi mnenje 2. Potrdi vnos z gumbom “Confirm” ali prekini z gumbom “Cancel”
Razširitve:	-
Pogostost uporabe:	*približno 250 letno oz. 5 tedensko.*
Status:	[V razvoju]
Lastnik:	Vodja proizvodnje
Prioriteta:	*Visoka*

Tabela 17.3: Opis primera uporabe: ocenitev finančnega vidika povpraševanja.

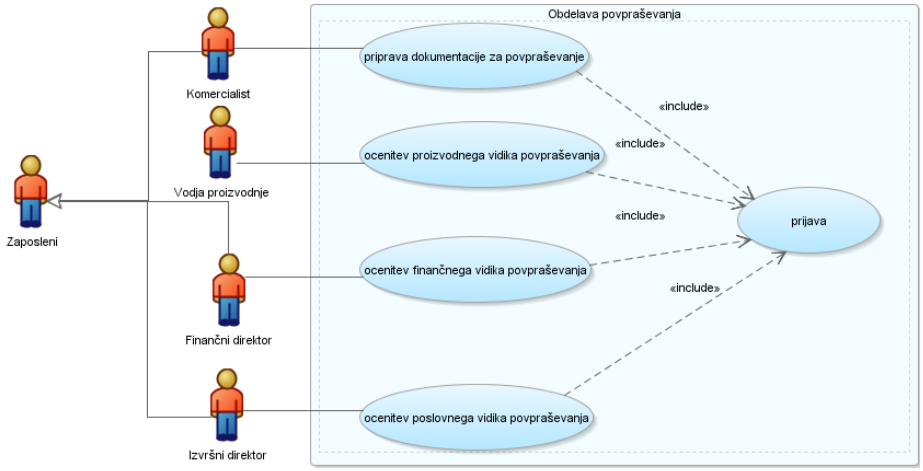
Ključni pojem	**Vrednost**
ID:	*ch17-03*
Naslov:	*Ocenitev finančnega vidika povpraševanja*
Opis:	*Finančni direktor in njegova/njena ekipa podajo strokovno mnenje, ali je podjetje sposobno financirati izdelavo in če da, kolikšen bo predvideni dobiček. Utemeljitev mnenja je obvezna.*
Glavni igralec:	*Finančni direktor*
Predhodni pogoji:	Dokumentacija o povpraševanju je pripravljena
Pogoji po zaključku:	-
Glavni del	-
Uspešen scenarij:	1. Vnesi mnenje 2. Potrdi vnos z gumbom “Confirm” ali prekini z gumbom “Cancel”
Razširitve:	-
Pogostost uporabe:	*približno 250 letno oz. 5 tedensko.*
Status:	[V razvoju]
Lastnik:	Finančni direktor
Prioriteta:	*Visoka*

Tabela 17.4: Opis primera uporabe: ocenitev poslovnega vidika povpraševanja.

Ključni pojem	**Vrednost**
ID:	*ch17-04*
Naslov:	*Ocenitev poslovnega vidika povpraševanja*
Opis:	*Izvršni direktor in njegova/njena ekipa podajo strokovno mnenje, ali je na povpraševanje odgovorijo s ponudbo ali se zahvalijo za interes. Utemeljitev mnenja je obvezna in jo zahteva nadzorni odbor.*
Glavni igralec:	*Izvršni direktor*
Predhodni pogoji:	Primeri uporabe 17.1, 17.2 in 17.3 so zaključeni.
Pogoji po zaključku:	-
Glavni del	-
Uspešen scenarij:	1. vnesi mnenje 2. potrdi vnos z gumbom “Confirm” ali prekini z gumbom “Cancel”
Razširitve:	-
Pogostost uporabe:	*približno 250 letno oz. 5 tedensko.*
Status:	[V razvoju]
Lastnik:	Izvršni direktor
Prioriteta:	*Visoka*

17.4.3 Diagram primera uporabe

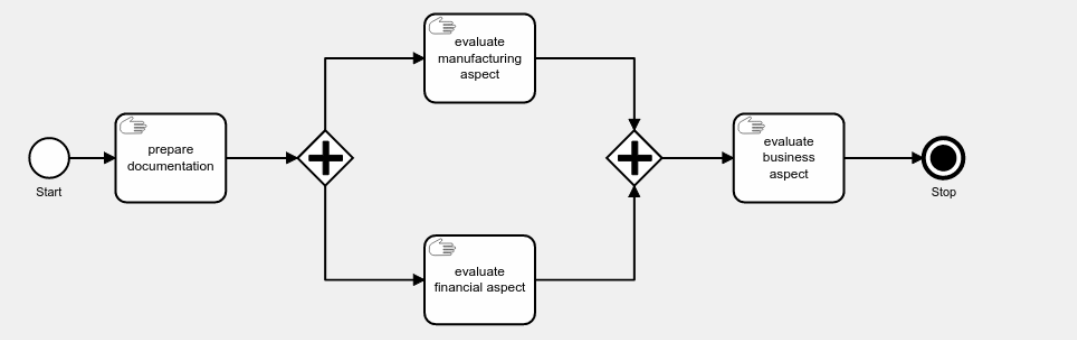
Diagram primerov uporabe lahko prikazuje tudi posploševanje. V danem kontekstu ima zaposleni štiri posebne primere vodij področij. Vsak vodja je povezan z enim primerom uporabe. Vsi štirje primeri uporabe so prikazani na sliki 17.2.



Slika 17.2: Diagram primerov uporabe.

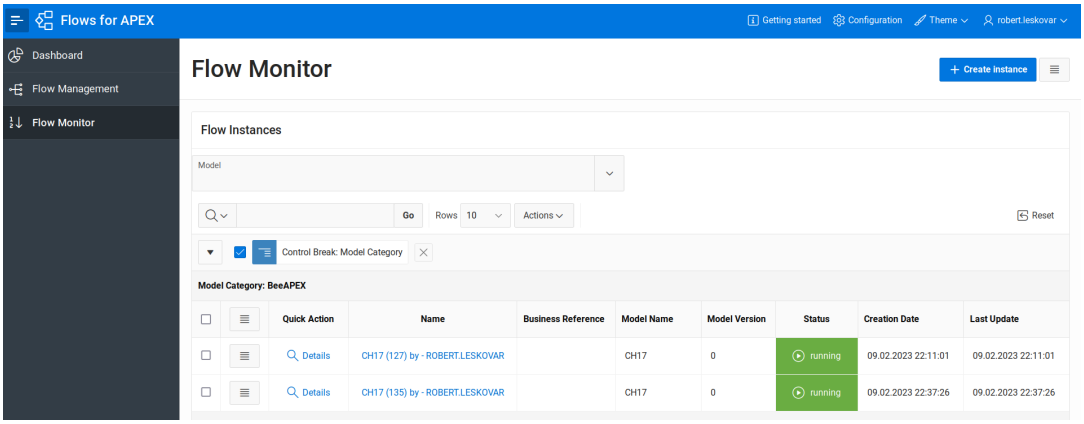
17.5 Model delovnega toka

Začetni model delovnega toka prikazuje nalogo priprave dokumentacije, ki se nato razdeli na vzporedni prehod z dvema nalogama (proizvodna in finančna ocena). Delovni tok se nadaljuje z zadnjo nalogo (ocena poslovnega vidika), ko sta obe prejšnji nalogi končani (slika 17.3).



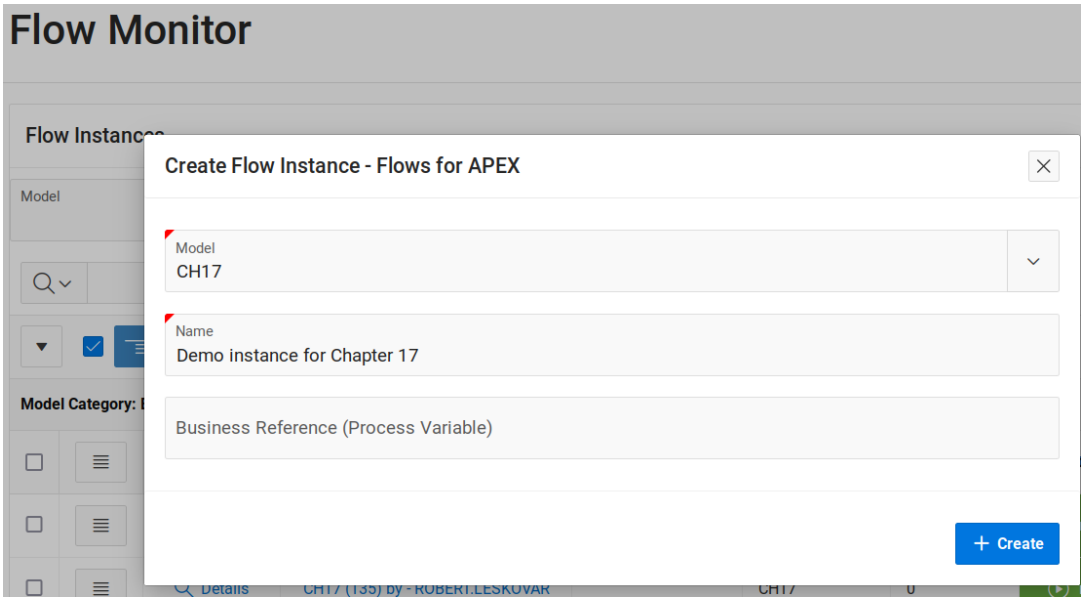
Slika 17.3: Delovni tok - obdelava povpraševanja - diagram BPMN.

Vse štiri naloge so definirane kot “ročna naloga” (ang. “Manual task”), končni dogodek (Stop) pa kot “Zaključi končni dogodek” (ang. “Terminate end event”). Delovni tok lahko takoj preizkusimo v Flow Monitorju (v aplikaciji “Flows for APEX”). Naš model delovnega toka shranimo kot CH17. Nato se pomaknite v Flow Monitor. Kliknite gumb “Create instance” (slika 17.4).



Slika 17.4: Ustvarjanje primerka (instance) delovnega toka.

Izberite model (CH17) in vnesite unikatno ime primerka delovnega toka CH17 (slika 17.5).



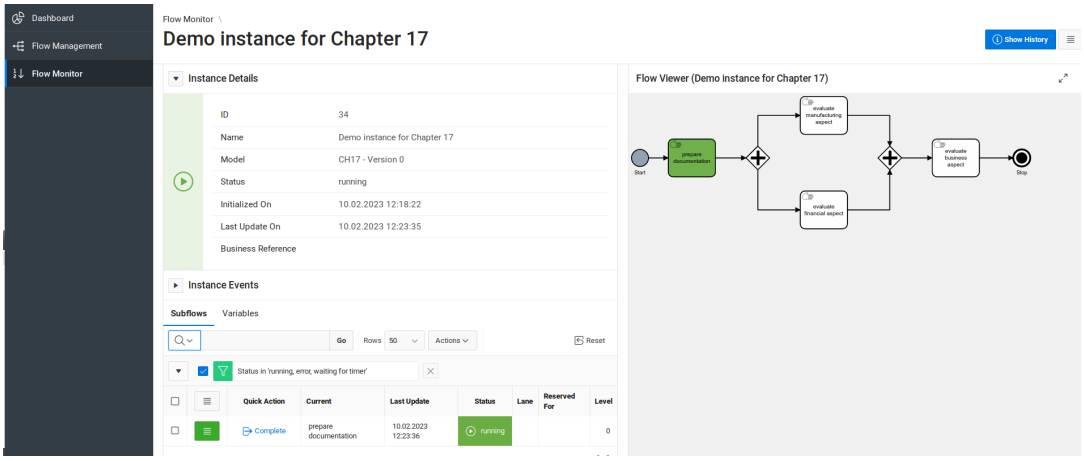
Slika 17.5: Ustvarjanje primerka delovnega toka CH17.

Pojavi se nov primerek. Kliknite označeno ikono na sliki 17.6 in izberite “Start”.



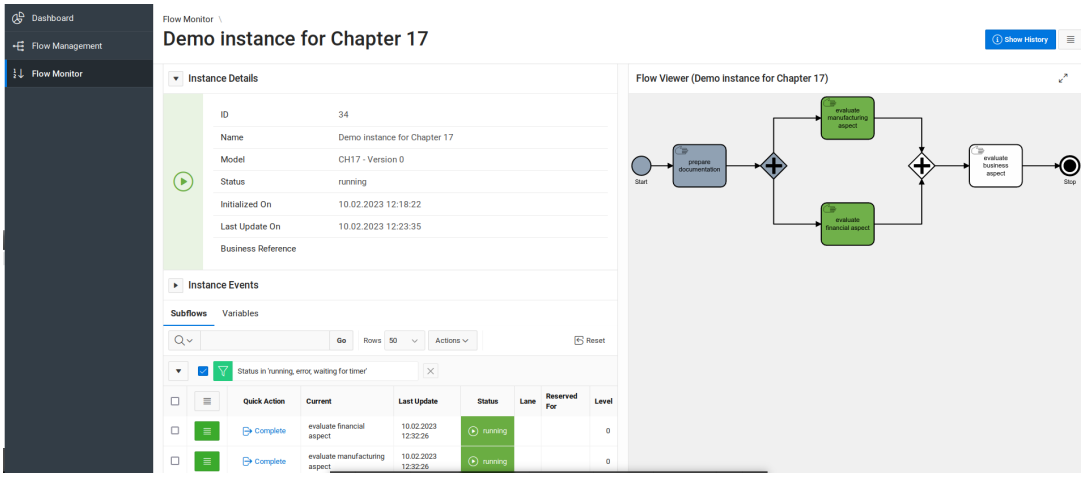
Slika 17.6: Zagon primerka delovnega toka.

Status primerka se spremeni v “running”. Kliknite “Details” za vpogled ta primerek. Nastavite prikaz “side-by-side” in kliknite “Complete” v stolpcu “Quick Action” (slika 17.7).



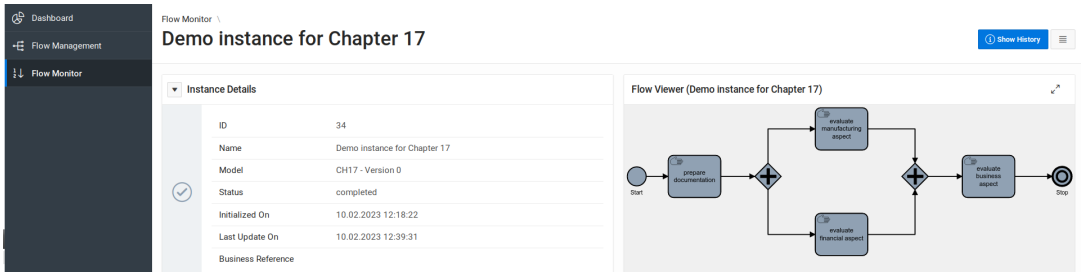
Slika 17.7: Zaključek prve naloge v primerku delovnega toka.

Delovni tok se nadaljuje s paralelnimi vrati in prehodom na dve nalogi (slika 17.8).



Slika 17.8: Izvajanje nalog v paralelnih vratih.

Kliknite “Complete” v stolpcu “Quick Action” pri obeh nalogah. Nadaljujte z zadnjo nalogo in jo zaključite (slika 17.9).



Slika 17.9: Zaključek vseh nalog v primerku delovnega toka.

Celotno zgodovino primerka si ogledamo, ko kliknemo na “Show history”. Dokazali smo, da je delovni tok CH17 izvedljiv in da lahko modeliramo podatke in aplikacijski vmesnik.

17.6 Model podatkov

17.6.1 Opis modela podatkov

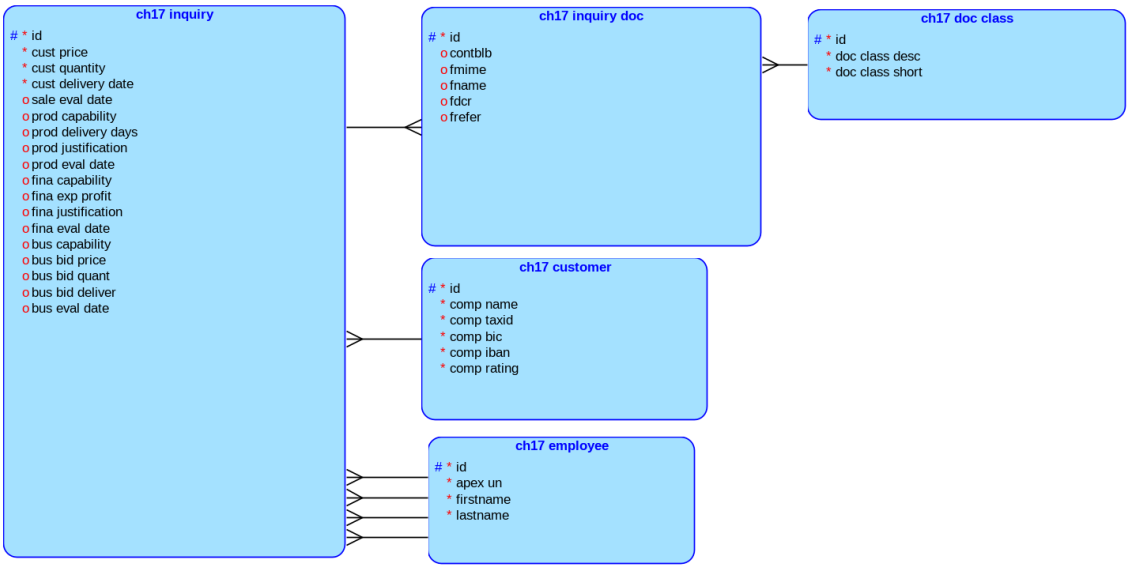
Najpomembnejša entiteta je **ch17 inquiry**. Primerek **ch17 inquiry** je povezan z enim primerkom **ch17 custimer**, medtem ko ima stranka lahko več povpraševanj. Vsak primerek **ch17 inquiry** mora imeti vsaj tri primerke **ch17 document**. Vsak primerek **ch17 document** pripada enemu primerku **ch17 document class**.

Atributi za entiteto **ch17 inquiry** so: ID, zahteve kupca (cena na naročilo, količina, datum dobave), proizvodni vidiki (zmogljivost, dobavni rok za zahtevano količino, utemeljitev), finančni vidiki (finančna zmožnost, pričakovan dobiček za zahtevano količino) in poslovni vidiki (odločitev o ponudbi ali opustitev priložnosti, utemeljitev).

Da bo ta primer čimbolj osredotočen na osnovni problem, bomo določili le omejen nabor atributov za ostale entitete.

17.6.2 Logični model podatkov

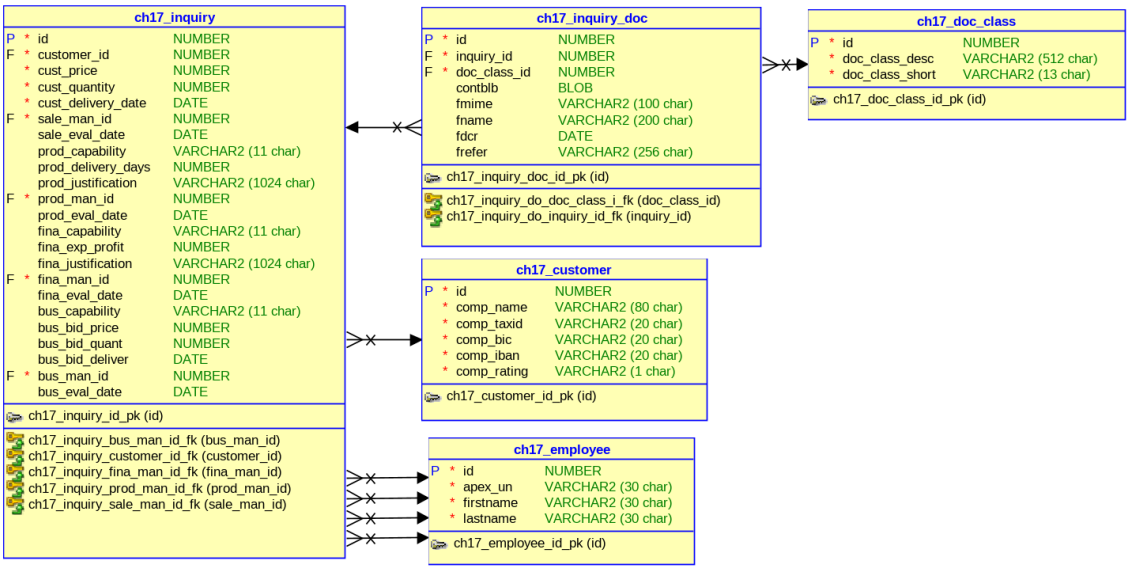
Logični model podatkov je prikazan na sliki 17.10.



Slika 17.10: Logični model podatkov.

17.6.3 Relacijski model podatkov

Relacijski model podatkov je prikazan na sliki 17.11.



Slika 17.11: Relacijski model podatkov.

17.6.4 Skript Quick SQL za generiranje skripta v SQL

Quick SQL je priročno orodje za pospešitev razvoja podatkovnega modela. Uporablja se lahko za izdelavo prototipov z ustvarjanjem skripta SQL znotraj APEX (generiraj aplikacijo iz skripta) ali za uporabo obratnega inženiringa v logični model podatkov z uvozom skripta SQL v SQL Developer Data Modeler in pretvorbo relacijskega v logični model. Podatkovni model, predstavljen v tem poglavju, je mogoče opisati z naslednjim skriptom Quick SQL:

```
ch17_employee
        apex_un vc30 /nn,
        firstname vc30 /nn,
        lastname  vc30 /nn

ch17_customer
        comp_name vc80 /nn,
        comp_taxid vc20 /nn,
        comp_bic vc20 /nn,
        comp_iban vc20 /nn,
        comp_rating vc1 /check 'A','B','C','D' /nn

ch17_doc_class
        doc_class_desc vc512 /nn,
        doc_class_short vc13 /check 'MANUFACTURING','FINANCIAL','BUSINESS',

ch17_inquiry
        customer_id num /fk ch17_customer /nn,
        cust_price  num /nn,
        cust_quantity num /nn,
        cust_delivery_date date /nn,
        cust_product_item vc50 /nn,
        sale_man_id num /fk ch17_employee /nn,
        sale_eval_date date,
        prod_capability vc11 /check 'YES','NO','CONDITIONAL','N.A.',
```

```
        prod_delivery_days num,
        prod_justification vc1024,
        prod_man_id num /fk ch17_employee /nn,
        prod_eval_date  date,
        fina_capability vc11 /check 'YES','NO','CONDITIONAL','N.A.',
        fina_exp_profit num,
        fina_justification vc1024,
        fina_man_id num /fk ch17_employee /nn,
        fina_eval_date date,
        bus_capability vc11 /check 'YES','NO','CONDITIONAL','N.A.',
        bus_bid_price num,
        bus_bid_quant num,
        bus_bid_deliver date,
        bus_man_id num /fk ch17_employee /nn,
        bus_eval_date date

ch17_inquiry_doc
        inquiry_id /fk ch17_inquiry /nn,
        doc_class_id /fk ch17_doc_class /nn,
        contblb blob,
        fmime vc100,
        fname vc200,
        fdcr date,
        frefer vc256
```

Ena od prednosti Quick SQL je njegova kompaktnost in preprostost. Število vrstic v Quick SQL je za polovico manjše od skripta SQL in število znakov je manj kot ena tretjina skripta SQL. Čas in trud za učenje Quick SQL sta dobra naložba. APEX ponuja tudi uporabniku prijazen urejevalnik s samodokončanjem, vendar bi prednosti Quick SQL bolj izkoristil uporabnik, ki že pozna SQL.

17.6.5 Sekvence in dve shranjeni funkciji

V tem poslovnem primeru nas zahteve aplikacij prisilijo, da uporabimo nekaj lahko razumljivih konceptov baze podatkov Oracle:

- mehanizem zaporedja (sekvenc) - generira cola števila in sicer tako, da podamo začetno vrednost, največjo vrednost, prirastek itd. S tem mehanizmom bo naša aplikacija ustvarila edinstveno številko za identifikacijo povpraševanja in sestavila ime primerka delovnega toka.
- Shranjene funkcije v jeziku PL/SQL sprejmejo vhodne parametre in vrnejo eno vrednost. Naša aplikacija bo uporabljala dve uporabniško definirani funkciji: eno za pridobitev velikosti dokumenta, shranjenega v stolpcu BLOB, in eno za štetje števila dokumentov, povezanih z določenim povpraševanjem.

Zaporedje in shranjene funkcije lahko definiramo v SQL Workshop > SQL Command. Zaporedje ima ime, najmanjšo vrednost, največjo vrednost, prirastek, začetno vrednost in nekatere druge lastnosti, ki za nas trenutno niso pomembne:

```
CREATE SEQUENCE   "CH17_SEQ_INQUIRY"  MINVALUE 1 MAXVALUE 999999999999
  INCREMENT BY 1 START WITH 100 CACHE 1
  NOORDER  NOCYCLE  NOKEEP  NOSCALE  GLOBAL;
```

Definirali bomo tudi dve shranjeni funkciji, ki sta v kodi razloženi s komentarji - dva pomišljaja na začetku vrstic sta komentarja. Prva funkcija se imenuje **ch17_doc_bytesize**:

```
create or replace function ch17_doc_bytesize(xdocid in number) return number
```

```
-- Function ch17_doc_bytesize returns the size of the BLOB, stored in contblb
-- column in the table ch17_inquiry_doc for particular row.
-- Function parameter 1 is the ID of of ch17_inquiry_doc table.
-- The length of contblb column is provided by another system level PL/SQL
-- function called dbms_lob.getlength.
is
wdoc_size number := 0;
wblob blob;
Begin
  select contblb into wblob from ch17_inquiry_doc where id = xdocid;
  wdoc_size := dbms_lob.getlength(wblob);
  RETURN  wdoc_size;
END ch17_doc_bytesize;
```

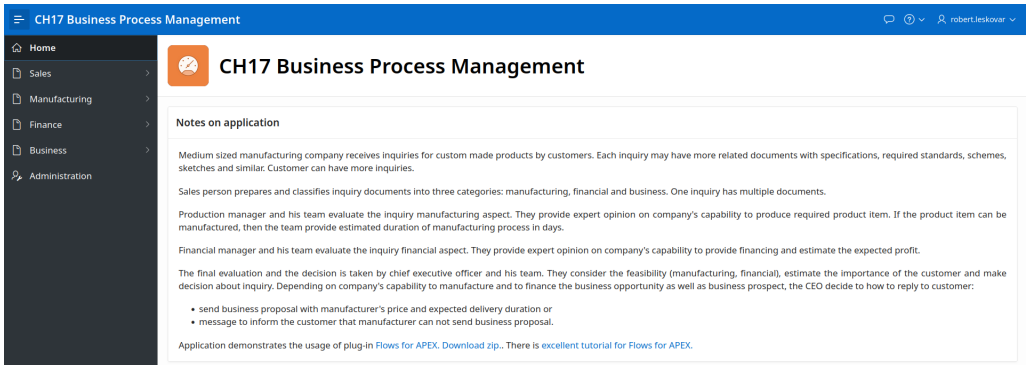
Druga funkcija se imenuje **ch17_count_docs**:

```
create or replace function ch17_count_docs(xinqid in number) return number
-- Function ch17_count_docs returns the number of documents associated with
-- particular inquiry. Function parameter 1 is the ID of inquiry.
-- If the length of the BLOB in the table ch17_inquiry_doc is greater than 0,
-- then we assume that the content is stored. So, function adds 1 to variable
-- wnumdocs, which is used as a counter. When all matching rows in the
-- table ch17_inquiry_doc are examined, than counter function returns counter.
is
wnumdocs number := 0;
CURSOR c_blob is
  select contblb from ch17_inquiry_doc where inquiry_id = xinqid;
Begin
  FOR r_blobs IN c_blob
    LOOP
      if dbms_lob.getlength(r_blobs.contblb) > 0 then
        wnumdocs := wnumdocs + 1;
      end if;
    END LOOP;
  RETURN  wnumdocs;
END ch17_count_docs;
```

17.7 Uporabniški vmesniki

Predlagamo, da bralec namesti paketno aplikacijo za to poglavje pred načrtovanjem aplikacije. Povezava do paketne aplikacije je v dodatnem učnem gradivu za to poglavje. Številke spletnih strani, ki jih navajamo v tem poglavju se torej nanašajo na pripravljeno paketno aplikacijo. Bralec lahko seveda razvije in uporabi druge številke strani, da zagotovi enako funkcionalnost, kot je opisana v tem razdelku.

Aplikacijski vmesnik omogoča funkcionalnost štirim igralcem: komercialistu, vodji proizvodnje, finančnemu direktorju in izvršnemu direktorju. Uporabimo čarovnika za ustvarjanje aplikacij in označimo vse možne funkcije (progresivna spletna aplikacija, stran o aplikaciji, nadzor dostopa, poročanje o dejavnosti, konfiguracijske možnosti, povratne informacije in izbira sloga teme, skupaj 21 strani). Privzeto je ustvarjena tudi prijavna stran. Domača stran vsebuje čisti HTML za opis poslovne situacije. Zagotavlja povezave do zahtevanega vtičnika in vadnice (glejte sliko 17.12, stran 1 v paketni aplikaciji).



Slika 17.12: Domača stran aplikacije “CH17 Business Process Management”.

Nekateri LOV-ji in vtičniki so bili nastavljeni ali vključeni pred razvojem aplikacije v ožjem smislu.

17.7.1 Seznam vrednosti (List of values) v deljenih komponentah (Shared Components)

V tej aplikaciji so opredeljeni naslednji seznami vrednosti:

- CH17_LOV_CAPABILITY: tip je static, prikazne vrednosti so (YES, NO, CONDITIONAL, Not Available), vrnjena vrednost je lahko (YES, NO, CONDITIONAL, N.A.). Ta seznam vrednosti je uporabljen pri oceni povpraševanja z vidika proizvodnje, financ in poslovanja.
- CH17_LOV_CRATING: tip je static, prikazne vrednosti so (Excelent rating, Good rating, Edge acceptable rating, Not acceptable rating), vrnjena vrednost je lahko (A, B, C, D). Ta seznam vrednosti se uporabi za določitev ocene kupca.
- CH17_LOV_CUSTOMER: tip je dynamic, temelji na tabeli CH17_CUSTOMER, prikazna vrednost je DOC_CLASS_DESC in vrnjena vrednost je ID. Ta seznam vrednosti se uporablja na obrazcih in poročilih za boljšo berljivost in preprečevanje napačnega vnosa kupčevega ID.
- CH17_LOV_DOC_CLASS: tip je dynamic, temelji na tabeli CH17_DOC_CLASS, prikazna vrednost je COMP_NAME in vrnjena vrednost je ID. Ta seznam vrednosti se uporablja na obrazcih in poročilih za boljšo berljivost in preprečevanje napačnega vnosa razreda dokumenta.
- CH17_LOV_EMPLOYEE: tip je dynamic, temelji na tabeli CH17_EMPLOYEE:

```
SELECT FIRSTNAME || ' ' || LASTNAME as d,
id as r
FROM CH17_EMPLOYEE order by LASTNAME, FIRSTNAME
```

  Ta seznam vrednosti se uporablja na obrazcih in poročilih za boljšo berljivost in preprečevanje napačnega vnosa osebe, ki pripravlja in ocenjuje povpraševanje. Operator združevanja (ang. concatenate operator) poveže ime zaposlenega, presledek in priimek.
- CH17_LOV_INQUIRY: tip je dynamic, temelji na tabeli CH17_INQUIRY:

```
SELECT
'Instance:' || ch17_inquiry.id || ' (Customer: ' || comp_name || ',
   product:' || cust_product_item  as d,
ch17_inquiry.id as r
FROM ch17_inquiry, ch17_customer
WHERE ch17_inquiry.customer_id = ch17_customer.id;
```

  Ta seznam vrednosti se uporablja na obrazcih in poročilih za boljšo berljivost in preprečevanje napačnega vnosa ID. Operator združevanja (ang. concatenate) poveže več polj v tabeli.

17.7.2 Vtičniki v deljenih komponentah (Shared Components)

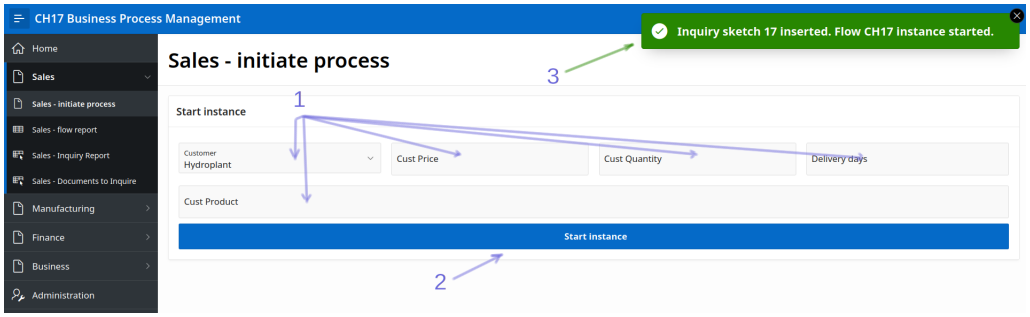
Uvoženi so naslednji vtičniki, ki so del Flows for APEX: Manage Flow Instance, Manage Flow Instance Step, Manage Flow Instance Variables in Viewer. Ti vtičniki so kopirani iz nameščenega vtičnika Flows for APEX.

17.7.3 Komercialist

Aplikacijski vmesniki komercialistu omogočajo:

- sprožitev primerka delovnega toka (slika 17.13, stran 2 v aplikaciji)
- nadzorovanje delovnega toka (slika 17.14, stran 5 v aplikaciji)
- potrditev, da so vsi dokumenti za nadaljnje vrednotenje shranjeni v bazi podatkov (slika 17.15, stran 4 v aplikaciji)
- vpogled v vsa povpraševanja (slika 17.17, stran 6 v aplikaciji)
- vpogled v vse dokumente v zvezi s povpraševanjem (slika 17.17, stran 8 v aplikaciji)
- nalaganje dokumenta, ki se nanaša na povpraševanje (sliki 17.18 in 17.20, strani 8 in 9 v aplikaciji)

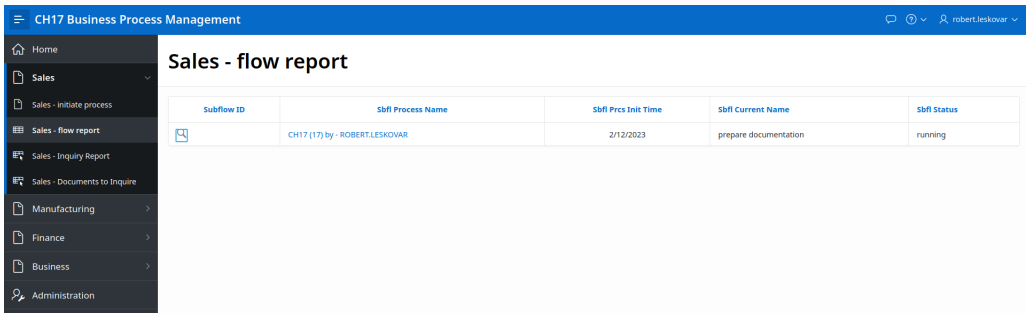
Komentarji k sliki 17.13 (številke in puščice) predstavljajo zaporedje dejanj: 1 (uporabnik vnese podatke), 2 (uporabnik potrdi vnesene podatke) in 3 (prikaže se sporočilo o uspehu ali neuspehu; podatkovna polja se počistijo za nov vnos).



Slika 17.13: Komercialist - začetek primerka delovnega toka.

Poleg vidnih komponent obstaja še en skriti element strani (P2_WID), ki preprosto dobi naslednjo številčno vrednost iz zaporedja CH17_SEQ_INQUIRY. To je bil razlog, da smo pred oblikovanjem strani definirali zaporedje CH17_SEQ_INQUIRY. Gumb “Start” izvede dva procesa: “Add inquiry” doda eno vrstico v tabelo CH17_INQUIRY in štiri vrstice v tabeli CH17_INQUIRY_DOC, medtem ko postopek “Create and start” izvede akcijo iz Flows for APEX (glejte podrobnosti v pakirani aplikaciji, stran 2, Procesi). Vrne se sporočilo o uspešnem zaključku, kot je prikazano na sliki 17.13.

Poročilo o poteku za komercialista je predstavljeno na sliki 17.14.



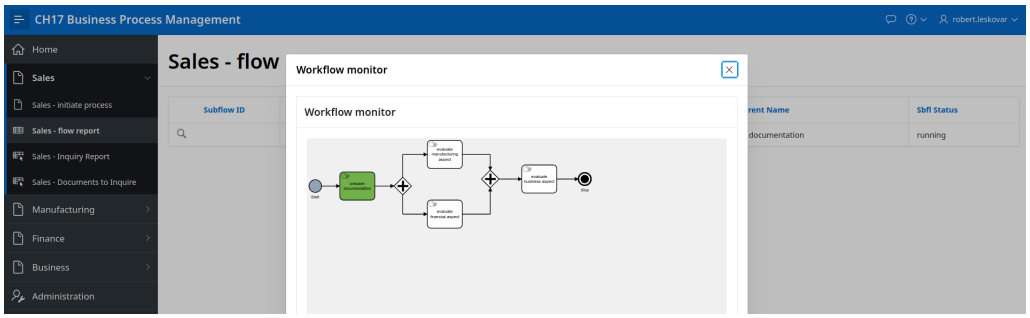
Slika 17.14: Komercialist - poročilo o poteku.

Komercialist ima klasično poročilo, ki temelji na poizvedbi SQL:

```
select sbfl_id, sbfl_prcs_id, sbfl_process_name, sbfl_prcs_init_ts,
          sbfl_current_name, sbfl_step_key, sbfl_status
    from flow_task_inbox_vw
    where sbfl_dgrm_name = 'CH17' and
      sbfl_current_name = 'prepare documentation'
```

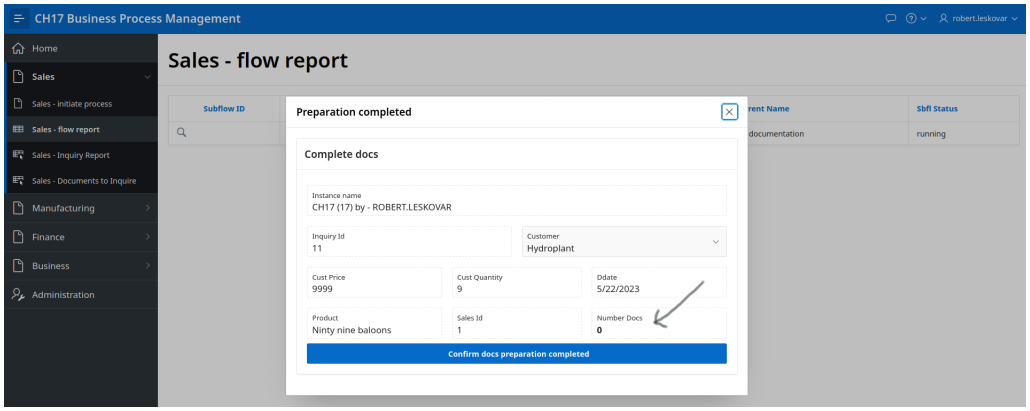
Poročilo prikaže samo primerke delovnega toka "CH17", ki imajo naziv "prepare documentation". Na strani 4 sta v aplikaciji dve povezavi (sslika 17.14):

1. Ikona za lupo v stolpcu "Subflow ID" prikazuje stanje primerka delovnega toka na diagramu (slika 17.15, stran 5 v aplikaciji).



Slika 17.15: Komercialist - poročilo o delovnem toku določenega primerka.

2. Stolpec "Sbfl Process Name" prikaže modalni obrazec, na katerem komercialist potrdi, da je dokumentacija pripravljena (slika 17.16, stran 7 v aplikaciji).



Slika 17.16: Komercialist - poročilo o delovnem toku za izbrani primerek.

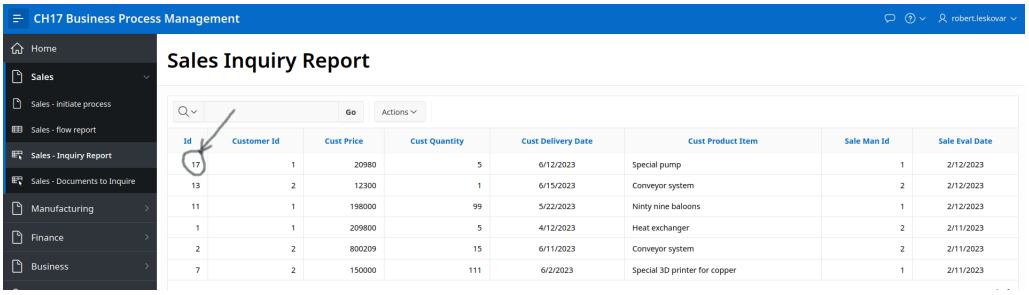
V tem trenutku dokumentacija še ni pripravljena (0 naloženih dokumentov). Vrednost števca je rezultat anonimnega bloka PL/SQL:

```
BEGIN
RETURN CH17_COUNT_DOCS(:P7_INQUIRY_ID);
END;
```

Funkcija CH17_COUNT_DOCS (glejte podrobnosti v paketni aplikaciji Supporting objects ali po namestitvi v Object Browser > Functions) prešteje vse velike binarne objekte (BLOB), shranjene v tabeli CH17_INQUIRY_DOC, ki so povezani z določenim povpraševanjem. Če je velikost BLOB-a večja od nič, je vsebina shranjena, sicer obstaja le prazen BLOB. Po

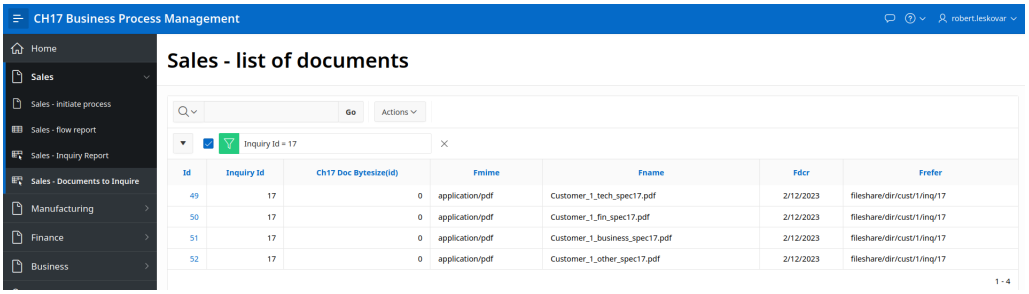
pravilih poslovanja morajo biti pripravljeni vsaj trije dokumenti (za proizvodnjo, finance in poslovodenje).

Komercialist ima pripravljeno interaktivno poročilo, ki mu omogoča vpogled v vsa povpračevanja (slika 17.17, stran 6 v aplikaciji).



Slika 17.17: Komercialist - poročilo o povpraševanjih.

Poročilo o prodajnem povpraševanju ponuja iskanje in različne akcije (izbira stolpcev, preurejanje, filtriranje, oblikovanje, grafikon, poročanje, izvoz itd.). Namen tega poročila je komerialistu zagotoviti ID povpraševanja, ta pa služi za povezovanje povpraševanja in z njim povezanih dokumentov (npr. 17, kot je prikazano na sliki 17.17). Naslednji korak za komercialista bi bila priprava tehničnih, finančnih, poslovnih in drugih specifikacij (glej sliko 17.18, stran 8 v aplikaciji).



Slika 17.18: Komercialist - seznam dokumentov.

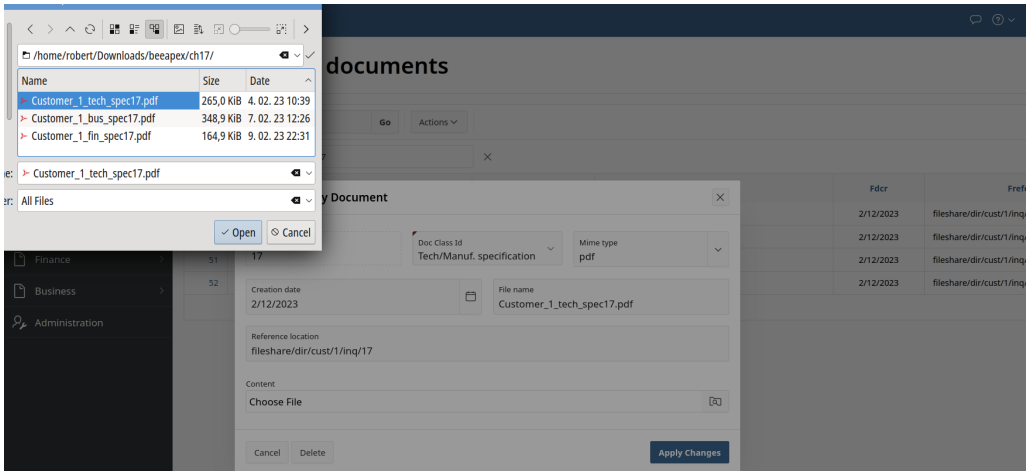
Slika 17.18 prikazuje filtrirane rezultate poizvebve (povpraševanje za ID je 17). Ker ni naložen noben dokument je v stolpcu “Ch17 Doc Bytesize(id)” prikazana nič. Stran 8 je interaktivno poročilo, ki temelji na naslednji poizvedbi:

```
select ID, INQUIRY_ID,DOC_CLASS_ID, CH17_DOC_BYTESIZE(ID), CONTBLB,
       FMIME,FNAME, FDCR, FREFER
from CH17_INQUIRY_DOC
```

Funkcija CH17_DOC_BYTESIZE, ki je napisana v PL/SQL, vrne velikost BLOB-a. V funkcijo posredujemo ID dokumenta v tabeli CH17_INQUIRY_DOC. Funkcija je definirana takole:

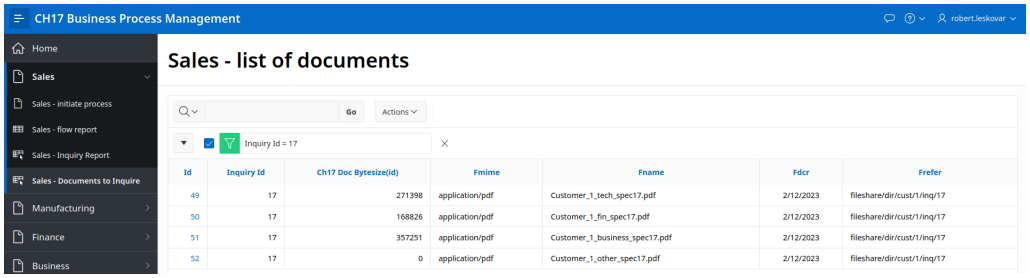
```
create or replace function CH17_DOC_BYTESIZE(xdocid in number) return number
is
  wdoc_size number := 0;
  wblob blob;
Begin
  select contblb into wblob from ch17_inquiry_doc where id = xdocid;
  wdoc_size := dbms_lob.getlength(wblob);
  RETURN wdoc_size;
END;
```

ID iz tabele CH17_INQUIRY_DOC je uporabljen kot povezava za modalno stran 9 (slika 17.19)).



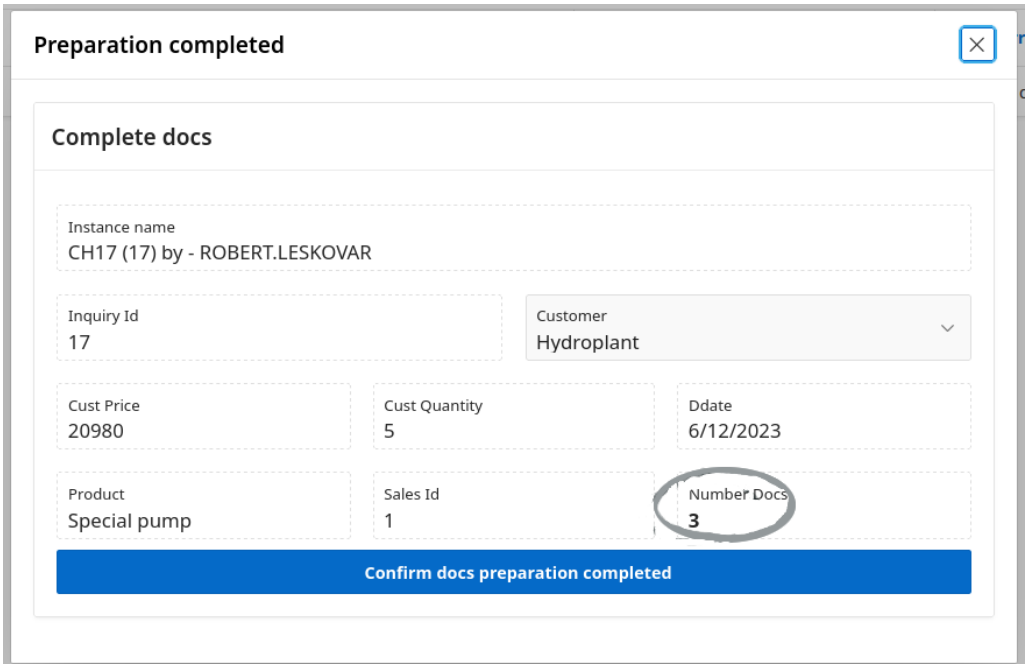
Slika 17.19: Komercialist - nalaganje dokumenta za povpraševanje 17.

Ko osvežimo stran 8, se lahko prepričamo, da je dokument resnično naložen (slika 17.20, stran 8 v aplikaciji).



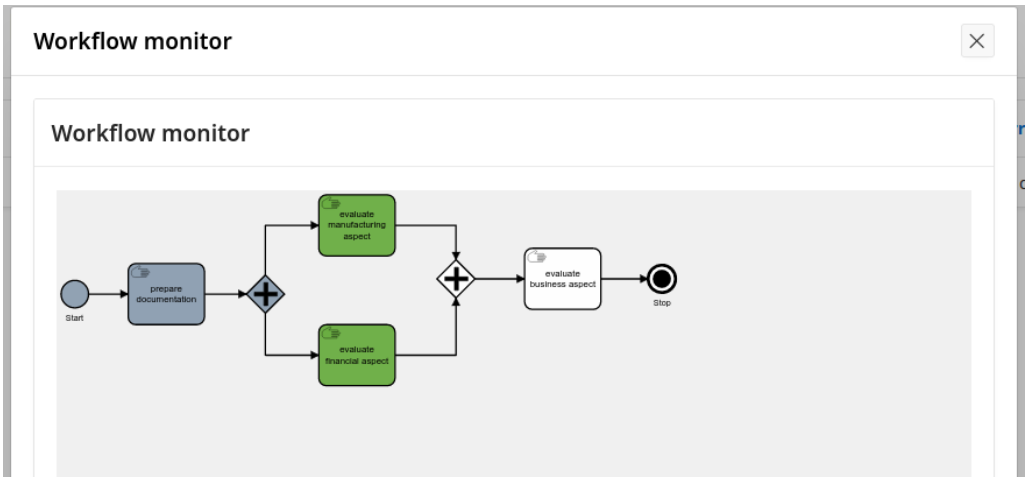
Slika 17.20: Komercialist - seznam naloženih dokumentov po osvežitvi.

Ko komercialist naloži vse zahtevane dokumente, mora to potrditi. S klikom na ime procesa se odpre modalna stran (slika 17.21, stran 8 v aplikaciji).



Slika 17.21: Komercialist - poročilo za določen primerek po nalaganju treh dokumentov.

S klikom na “Subflow ID” se komercialist lahko prepriča, da je pripravil dokumente in dokončal svojo nalogo (slika 17.22, stran 5 v aplikaciji).



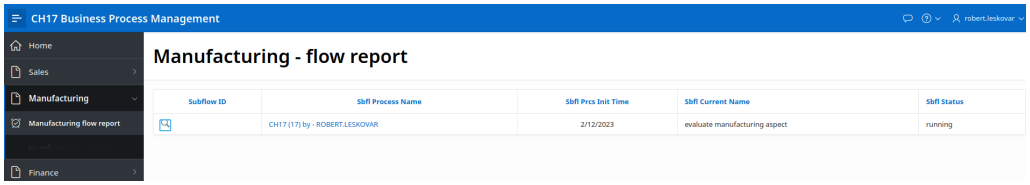
Slika 17.22: Komercialist - poročilo v delovnem toku za določen primerek po nalaganju treh dokumentov.

17.7.4 Vodja proizvodnje

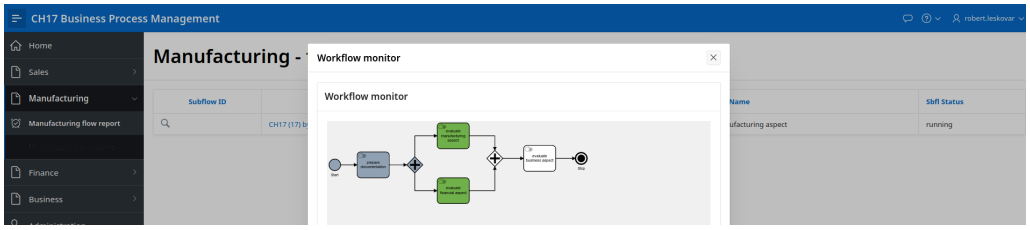
Vodja proizvodnje lahko :

- spremlja delovni tok (poročilo na sliki 17.23, stran 20 v aplikaciji in diagram na sliki 17.24, stran 5 v aplikaciji)
- vnese oceno povpraševanja z vidika proizvodnje (slika 17.25, stran 21 v aplikaciji)

S klikom na “Manufacturing flow report” vodja proizvodnje dobi poročilo o neocenjenih povpraševanjih (npr. poročilo na sliki 17.23, stran 20 v aplikaciji).

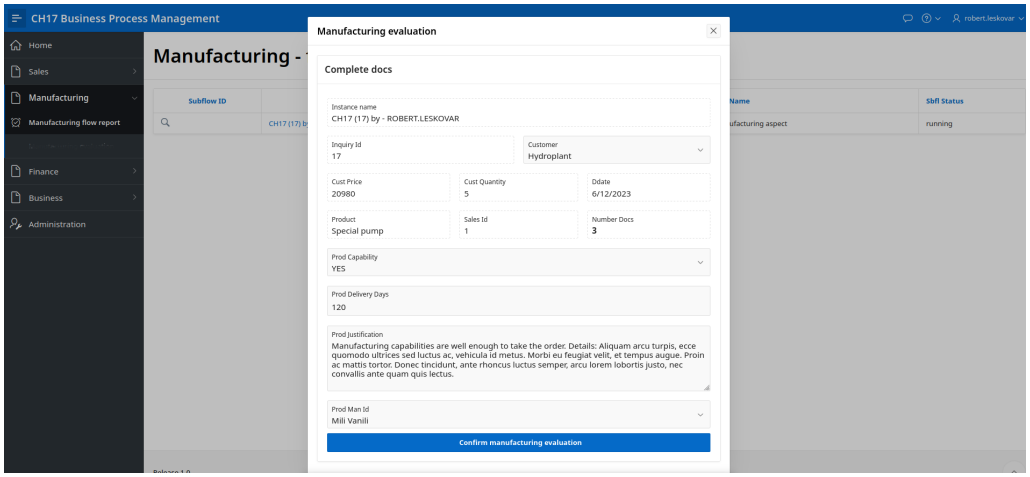


Slika 17.23: Vodja proizvodnje - poročilo o delovnem toku.



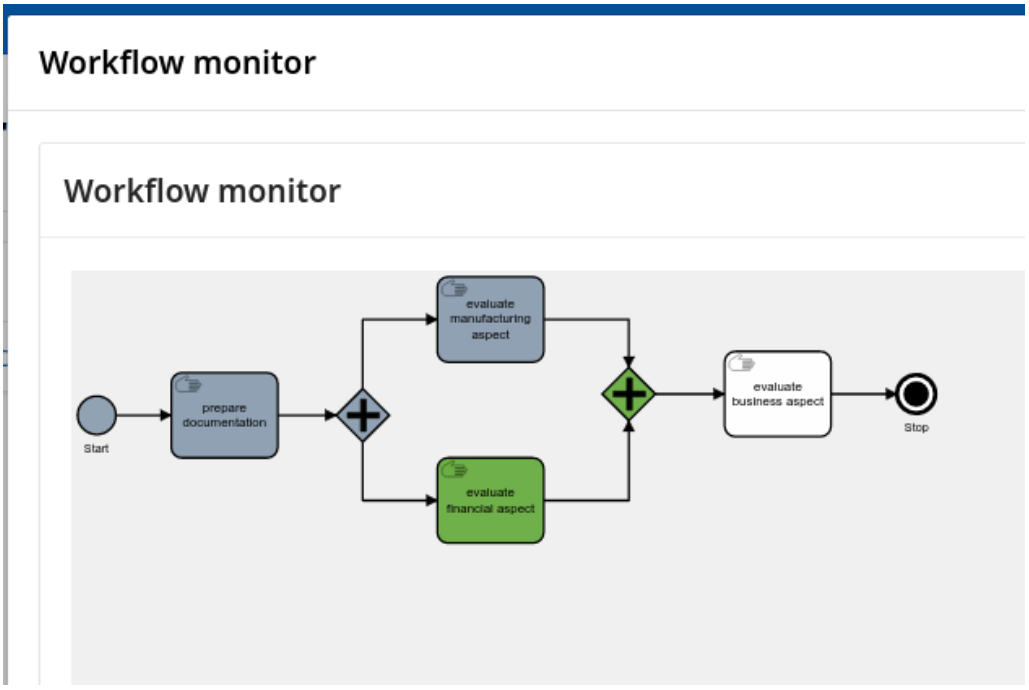
Slika 17.24: Vodja proizvodnje - diagram BPMN - stanje primerka.

Z izbiro ikone za lupo “Subflow ID” se odpre stran 5 (slika 17.24), ki prikazuje primer toka CH17. Primer je v vzporednih vratih, kar pomeni, da morata biti obe nalogi (proizvodna in finančna ocena) končani, preden se začne naslednja naloga (poslovna ocena). Vrednotenje vodja proizvodnje vnese tako, da klikne povezavo “Sbfl Process Name” na strani 20 (slika 17.25, stran 21 v aplikaciji).



Slika 17.25: Vodja proizvodnje - vnos ocene za povpraševanje.

Klik na gumb “Confirm manufacturing evaluation” povzroči, da delovni tok napreduje (slika 17.26).



Slika 17.26: Vodja proizvodnje - ocenjevanje zaključeno, primerek čaka na finančno oceno.

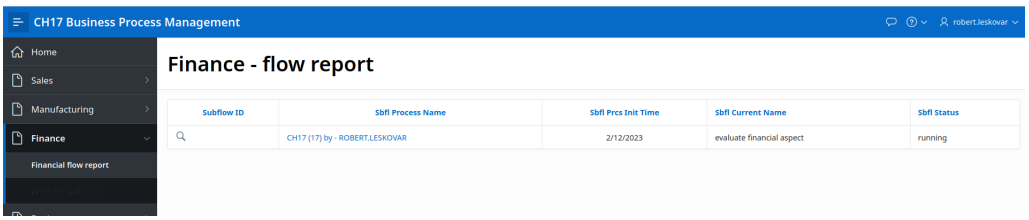
Ko osvežimo stran 20, poročilo pokaže, da ni čakajočih primerkov (ang. “Manufacturing has no flow instances”).

17.7.5 Vodja financ

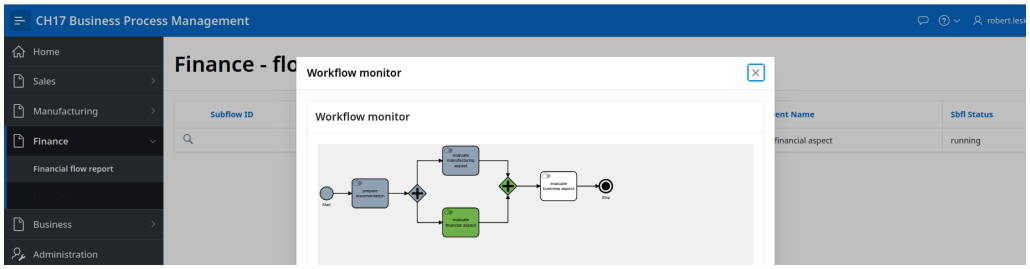
Finančni direktor lahko:

- spremlja delovni tok (poročilo na sliki 17.27, stran 30 v aplikaciji in diagram na sliki 17.28, stran 5 v aplikaciji)
- vnese oceno povpraševanja z vidika financiranje (slika 17.29, stran v aplikaciji).

Direktor financ s klikom na “Financial flow report” dobi poročilo o neocenjenih povpraševanjih (npr. poročilo na sliki 17.27, stran 30 v aplikaciji).

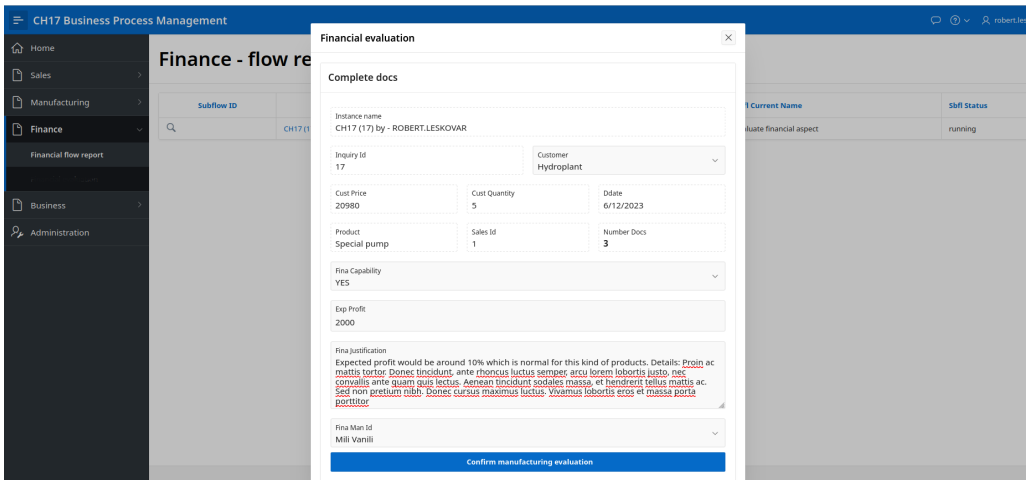


Slika 17.27: Finančni direktor - poročilo o delovnem toku.



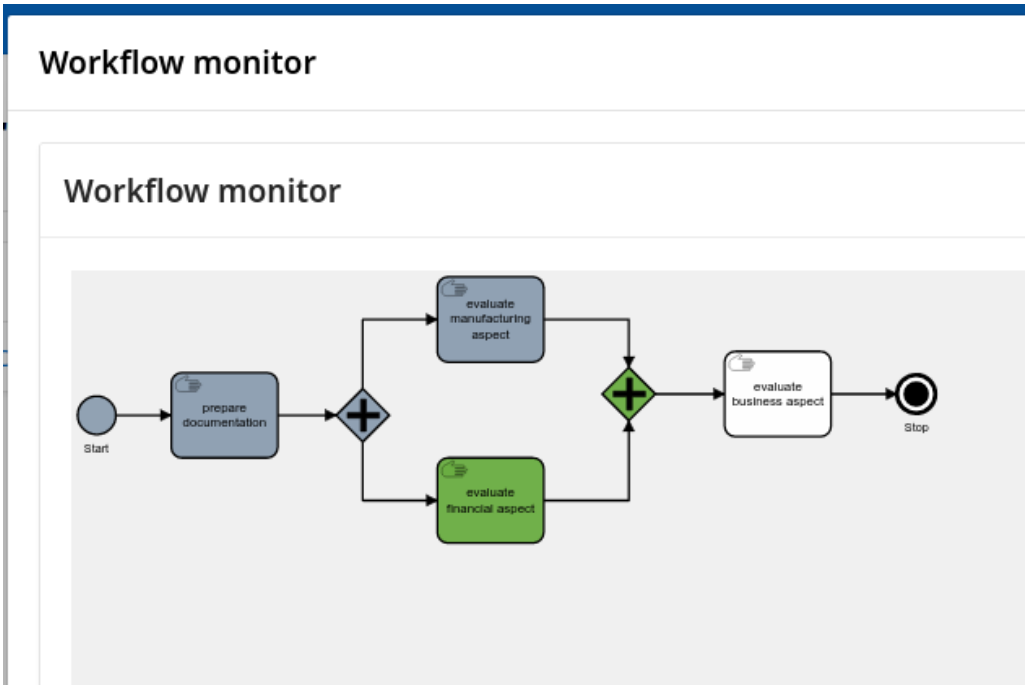
Slika 17.28: Finančni direktor - diagram BPMN - stanje primerka.

Z izbiro ikone za lupo “Subflow ID” se odpre stran 5 (slika 17.28), ki prikazuje primerek delovnega toka CH17. Finančno oceno direktor vnese s klikom na povezavo “Sbfl Process Name” na strani 30 (slika 17.29, stran 31 v aplikaciji).



Slika 17.29: Finančni direktor - vnos ocene za povpraševanje.

Klik na gumb “Confirm manufacturing evaluation” sproži napredovanje primerka do naslednje naloge, kot je to prikazano na sliki 17.30.



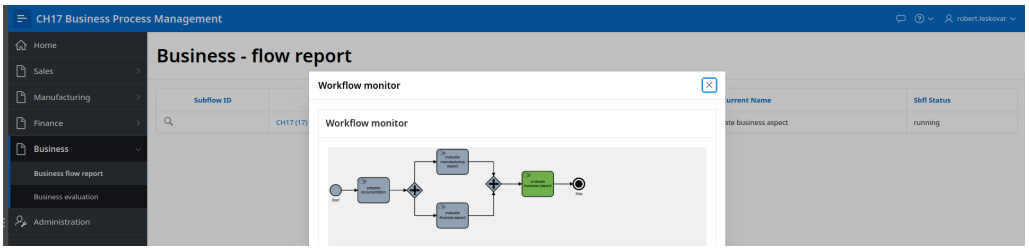
Slika 17.30: Finančni direktor - ocenjevanje zaključeno, primerek čaka na poslovno ocenjevanje.

Ko osvežimo stran 20, poročilo pokaže, da ni čakajočih primerkov (ang. “Finance has no flow instances”).

17.7.6 Izvršni direktor

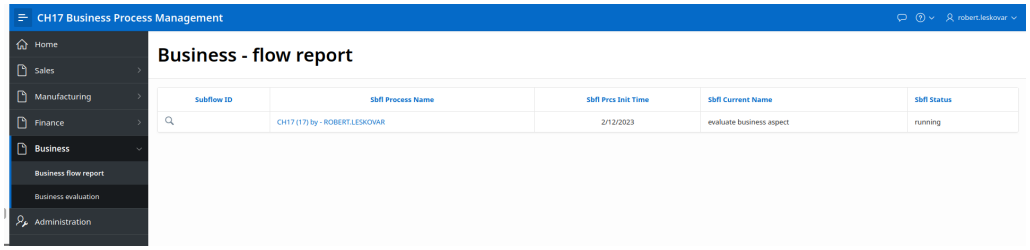
Izvršni direktor lahko:

- spremlja delovni tok (poročilo na sliki 17.32, stran 40 v aplikaciji in diagram na sliki 17.31, stran 5 v aplikaciji)
- vnese oceno povpraševanja z vidika poslovodstva (slika 17.33, stran 41 v aplikaciji)



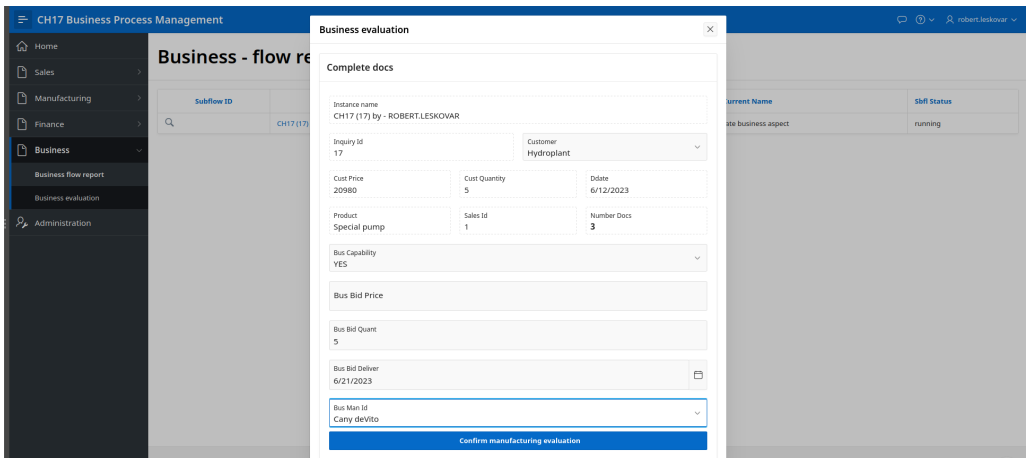
Slika 17.31: Izvršni direktor - diagram BPMN- stanje primerka.

Izvršni direktor s klikom na “Business flow report” dobi poročilo o neocenjenih povpraševanjih (npr. poročilo na sliki 17.32, stran 40 v aplikaciji).



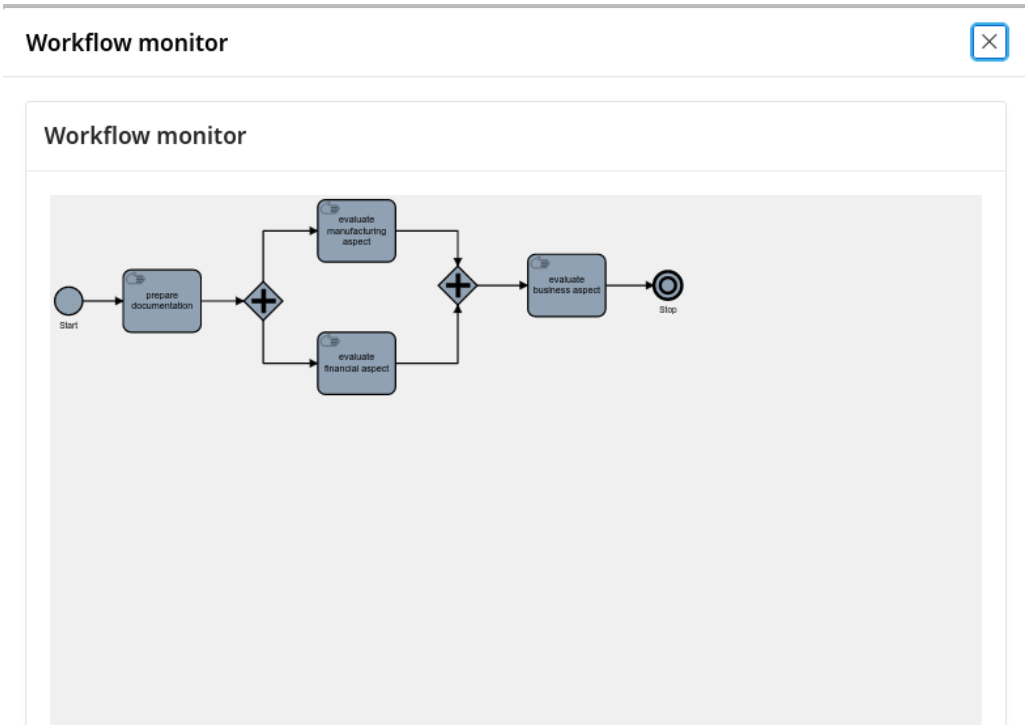
Slika 17.32: Izvršni direktor - poročilo o delovnem toku.

Z izbiro ikone za lupo “Subflow ID” se odpre stran 5 (slika 17.31), ki pokaže primerke delovnega toka CH17. Poslovno oceno direktor vnese s klikom na povezavo “Sbfl Process Name” na strani 40 (slika 17.33, stran 41 v aplikaciji).



Slika 17.33: Izvršni direktor - vnos poslovne ocene.

Klik na gumb “Confirm business evaluation” povzroči napredovanje primerka do zaključka, kot je to prikazano na sliki 17.34.



Slika 17.34: Izvršni direktor - poslovno ocenjevanje je zaključeno, primerek je zaključen.

Ko osvežimo stran 20, poročilo pokaže, da ni čakajočih primerkov (ang. “Business has no flow instances”).

17.8 Povezovanje aplikacije in vtičnika Flows for APEX

Aplikacija, razvita v poglavju 17, je uporabila dve funkciji, ki ju ponuja Flows for APEX:

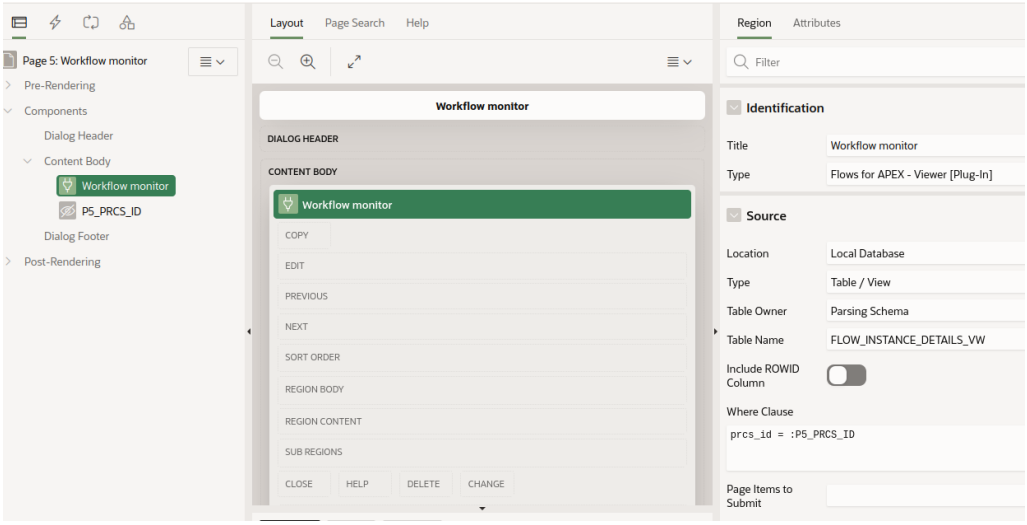
- zaženi primerek na strani 2
- prikaži stanje primerka na strani 5
- potisni primerek na naslednjo nalogo na straneh 7, 21, 31, 41

Zagon primerka zahteva aktiviranje procesa na strani. V identifikacijskem območju tipa procesa: “Flows for APEX - Managing Flow Instance [Plug-In]” morate izbrati akcijo “Create and Start”. Flow instance Name nastavimo na:

```
CH17 (&P2_WID.) by - &APP_USER.
```

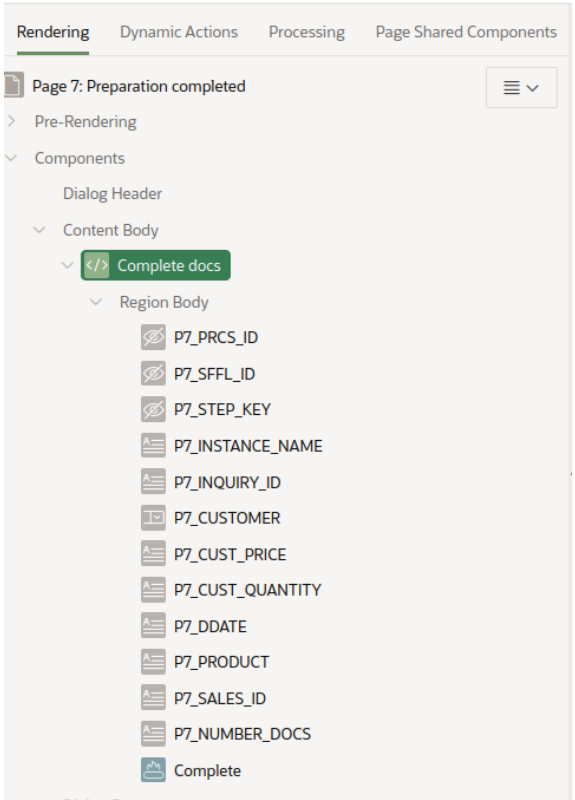
Ne pozabite, da postavka na strani P2_WID vsebuje unikatno številko, ki jo zagotavlja sekvenca. Ime je torej sestavljeni niz, ki vsebuje fiksni niz “CH17”, presledek, predklepaj, unikatne zaporedne številke, zaklepaj, presledek, fiksni niz “by - ” in trenutnega uporabnika apliakcije. Select Flow uporablja “Static text”, statično besedilo pa je “CH17”. Sporočilo o uspehu je “Flow CH17 instance started.” in sporočilo o napaki je “Flow CH17 instance NOT started.” .

Za predstavitev stanja primerka na strani 5 se uporablja čarovnik za stran vtičnika. Stran vsebuje pregledovalnik vtičnika in skrito polje - P5_PRCS_ID. Glejte nastavitve na sliki 17.35.



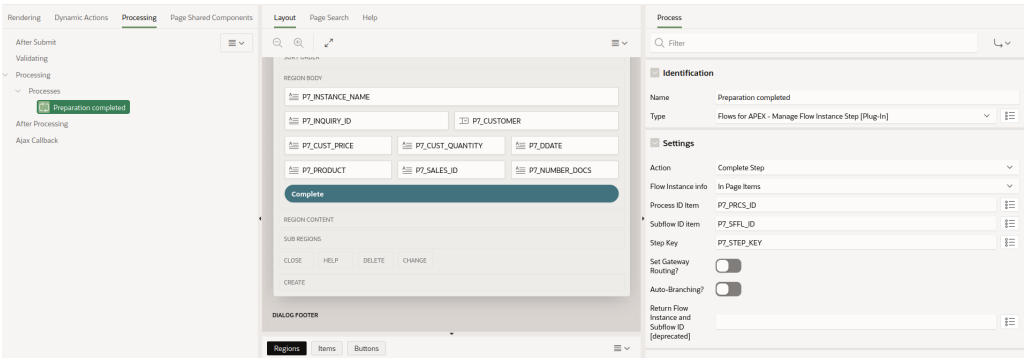
Slika 17.35: Prikaz stanja primerka na diagramu BPMN.

Če želite potisniti primerek naprej (npr. strani 7, 21, 31, 41), definirajte elemente strani, kot je prikazano na sliki 17.36.



Slika 17.36: Nastavitev polj na strani.

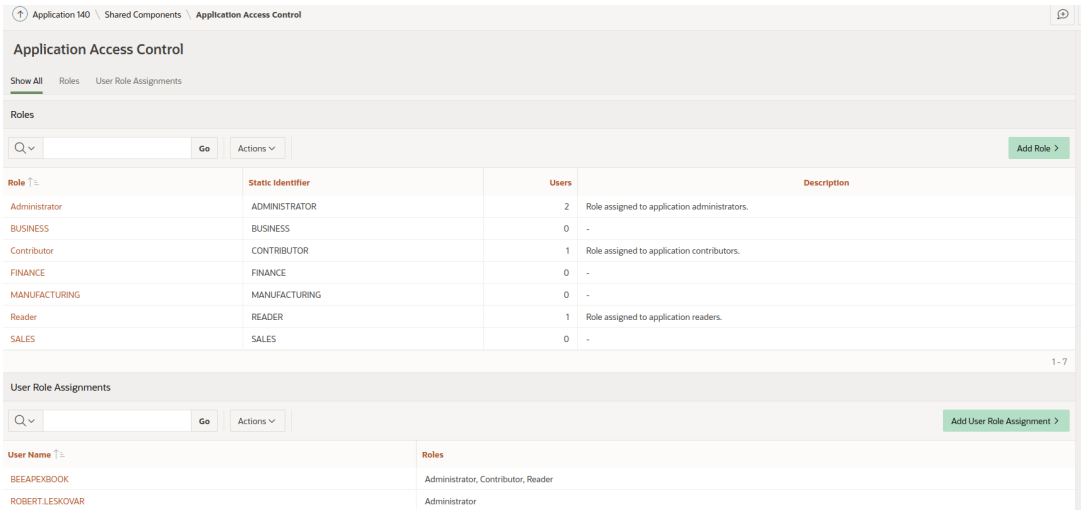
Samo prva štiri polja na straneh (od P7_PRCS_ID do P7_INSTANCE_NAME) so pomembna za integracijo Flows for APEX in aplikacije. Kreirati je potrebno nov proces, ki ga sproži pritisk na gumb. Proces je definiran kot vtičnik “Manage Flow Instance Step”. Določite Action, Process ID Item, Subflow ID item in Step Key, kot prikazuje slika 17.37.



Slika 17.37: Dokončanje koraka v Flows for Apex.

17.9 Definiranje uporabniških vlog (user roles)

Uporabniške vloge je mogoče definirati v *Application Access Control* v *Shared Components* (npr. slika 17.38).



Slika 17.38: Določitev vloge in uporabniške vloge v meniju Application Access Control.

V tem trenutku bomo preskočili podrobnosti in izvedbo za ta poseben primer. Za hiter pregled glejte poglavje 13 ali za temeljit vpogled preučite dokumentacijo o avtorizaciji v APEX-u.

17.10 Testiranje in odpravljanje napak

Za začetnike bi bil postopek testiranja zelo poenostavljen. Ko položite svoje roke na tipkovnico in miško za razvoj z APEX, ste na poti, da naredite napako in jo popravite. Ne bojte se delati napak. Naučite se iz vsake, ki ste jo rešili sami. Delite znanje v skupnosti razvijalcev APEX. Disciplina testiranja programske opreme je zelo obsežna in lahko se je naučite le, če delate napake. Ni druge poti. Razvijalci so prva obrambna linija pred programskimi napakami, pomanjkljivostmi in okvarami.

Presenetljivo je, da so neuspeli razvojni projekti programske opreme v večini primerov šli v napačno smer še preden je bila napisana ena vrstica kode. Aplikacija, razvita za to poglavje, ima tudi programske napake. Nekatere od njih bodo odkrili in razhroščii bistri bralci. Nekatere napake so tudi globoko v samem APEX-u. Pisci in razvijalci tega poglavja upamo, da bosta testiranje in odpravljanje napak bralcem predstavljala prijeten izziv.

Odstranjevanje napak iz programske opreme je včasih zabavno, včasih pa prekletstvo. Samo ne obupajte prehitro!

17.11 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija in skripti
- delovni tok, ki je imenovan CH17
- video vodič

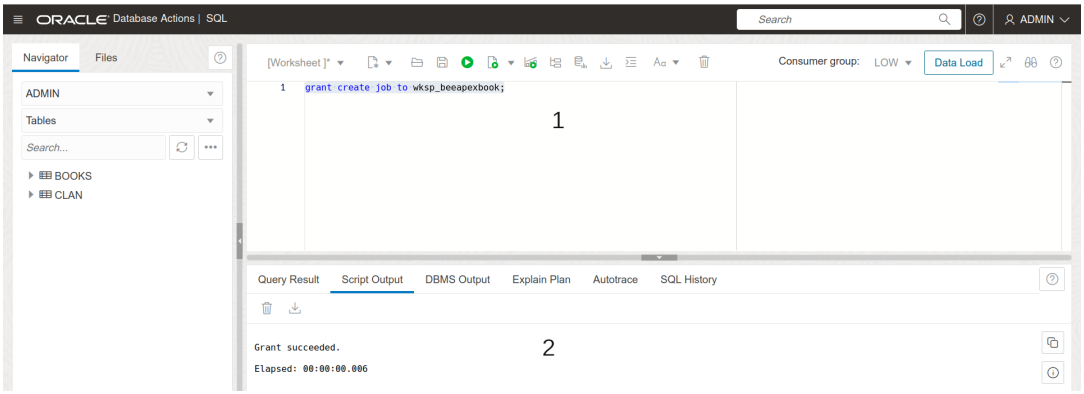
Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter17 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

17.11.1 Izvožena aplikacija

Izvožena aplikacija je pakirana. Namestitev ustvari tabele, zaporedje, funkcije in napolni podatke. Odstranitev briše vse objekte v bazi podatkov, ki se uporabljajo v tej aplikaciji.

Pakirana aplikacija je testirana in se bo izvajala v novem delovnem prostoru, če so izpolnjene naslednje zahteve:

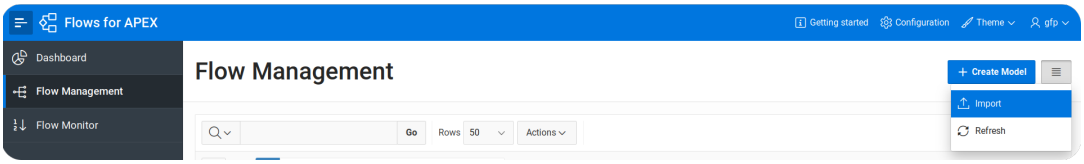
- pravica “create job” je dodeljena shemi, v kateri teče delovni prostor. Skrbnik baze lahko to pravico dodeli z ukazom “grant create job to <shema>”. Delovni prostor APEX-a je povezan z objektom, ki mu v terminiologiji baz podatkov rečemo shema. Če uporabljamo OCI (brezplačni, plačljivo ali preko Oracle Academy), potem lahko dostopamo do vmesnika Autonomous Database > specific instance > Database action ali preko spletnega vmesnika za skrbika baze podatkov. V tem vmesniku lahko izvajamo skripte in poizvedbe. Slika 17.39 predstavlja izgled vmesnika z omenjenim ukazom (1) in odzivom baze podatkov (2).



Slika 17.39: Dodelitev pravice “create job privilege” delovnemu prostoru.

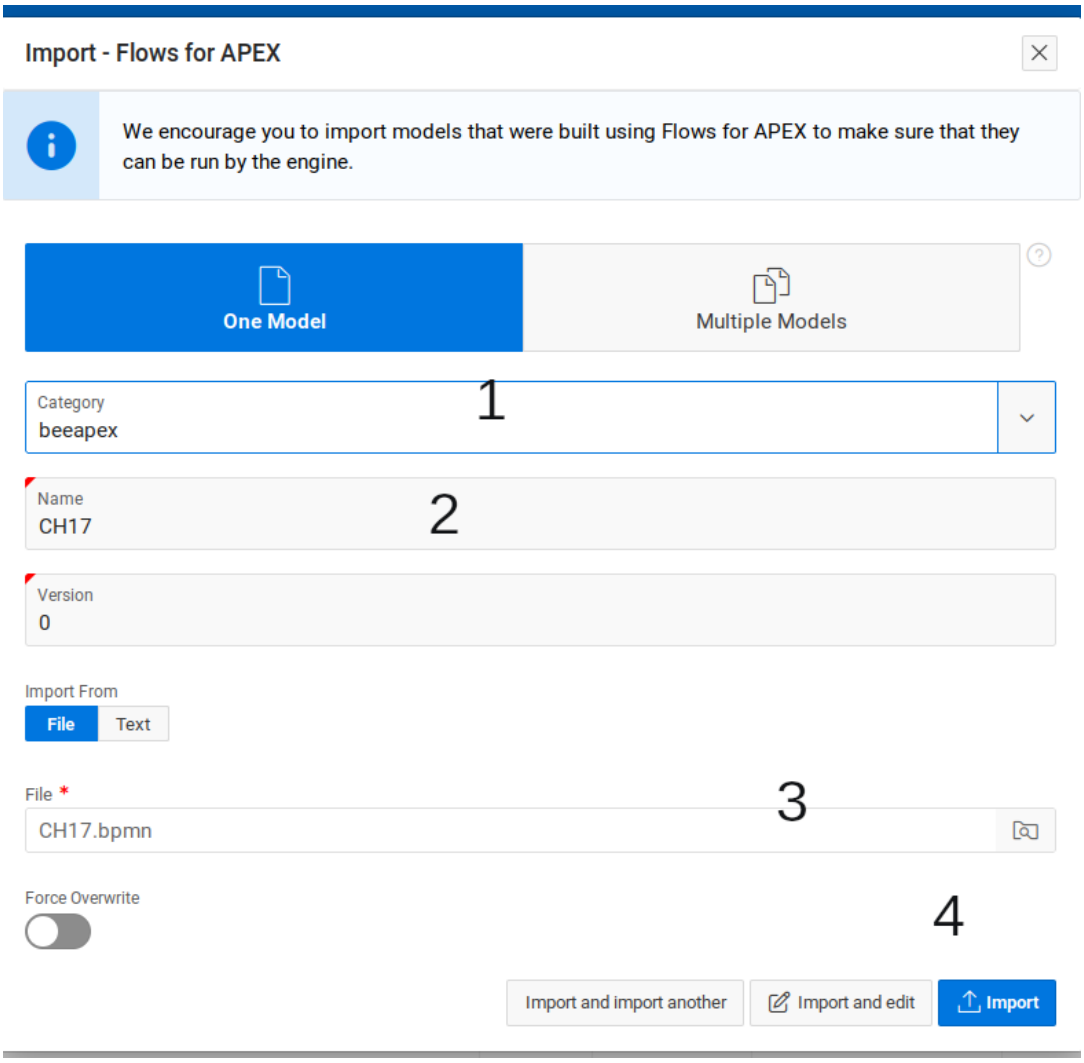
Če uporabljate druga orodja (SQL Developer, TOAD), je postopek zelo podoben. Izjema je seveda izgled uporabniškega vmesnika.

- Flows for APEX mora biti nameščen v novem delovnem prostoru (glejte prejšnja navodila).
- Delovni tok z nazivom CH17 morate uvoziti v Flows for APEX. Datoteka *CH17.bpmn* je shranjena v mapi Flows4APEX v pakirani aplikacija kot statična datoteka. Pojdite na Shared Components > Static Application Files in prenesite datoteko na vaš lokalni računalnik. Ista datoteka je dostopna tudi v učnih gradivih med skripti. Datoteko *CH17.bpmn* uvozite z vtičnikom Flows for APEX. Prijavite se v aplikacijo Flows for APEX. Izberite Flow management in uvozite datoteko (slika 17.40 - korak 1 slika 17.41 - korak 2).



Slika 17.40: Uvoz delovnega toka CH17 v Flows for APEX - korak 1.

Pomen številk na sliki 17.41 je: 1 (poljubna kategorija), 2 (ime modela mora biti natančno CH17), 3 (usmerite raziskovalca datotek na CH17.bpmn) in 4 (potrdite s klikom na Import)



Slika 17.41: Uvoz delovnega toka CH17 v Flows for APEX - korak 2.

- Po namestitvi pakirane aplikacije dodajte vlogo uporabniku v Shared components > Application Access Control > Add User Role Assignment

Če katerakoli od navedenih zahtev ni izpolnjena, se bo aplikacija sesula. Takrat je potrebno počistiti piškotke v spletnem brskalniku (primer za Firefox: Settings > Cookies and Site Data > Manage Data).

17.11.2 Video vodiči

Video vodič prikazuje začetne korake v razvoju aplikacije.

17.12 Vprašanja

1. Razložite kaj je Flows for APEX?
2. Katere artefakte modeliranja poslovnih procesov prikazujejo diagrami BPMN?
3. Katere funkcije Flows for APEX so uporabljene v tej aplikaciji?

17.13 Odgovori

1. Flows for APEX se imenujejo funkcija, vtičnik in aplikacija. Vsi trije pomeni so pravilni. Funkcija je mišljena, ko govorimo v kontekstu modeliranja procesa. Vtičnik se nanaša na razvoj aplikacije po meri. Aplikacija Flows for APEX je namenjena oblikovanju delovnih tokov in administriranju instanc v primeru težav.
2. Artefakti modela poslovnih procesov, predstavljeni z diagramom BPMN, so:
   - Začetni dogodek (ang. Start event): označuje začetno točko procesa.
   - Naloga (ang. Task): predstavlja eno enoto dela, kot je dejavnost ali korak v procesu.
   - Prehod (ang. Gateway): predstavlja točko odločitve v procesu, kot je razcep v toku ali pogojna veja.
   - Zaporedje (ang. Sequence): povezuje elemente v diagramu in predstavlja tok nadzora za prehod od enega elementa do drugega.
   - Končni dogodek (ang. End event): označuje konec procesa.
   - Bazen/Steza (ang. Pool/Lane): predstavlja združevanje povezanih nalog, ki se uporabljajo za določanje vlog in odgovornosti različnih udeležencev v procesu.
   - Tok sporočil (ang. Message flow): predstavlja tok sporočil med udeleženci v procesu, kot je komunikacija med dvema sistemoma.
   - Podatkovni objekt (ang. Data object): predstavlja del podatkov, ki se uporablja ali generira v procesu, kot je račun ali zapis stranke.
   - Podatkovna shramba (ang. Data Store): predstavlja vsebnik za shranjevanje podatkov, ki se uporabljajo ali proizvajajo v procesu, kot je baza podatkov ali datotečni sistem.
3. Za zagon primerka je bilo uporabljeno dejanje “Create and Start”. Za predstavitev primerka na diagramu BPMN je čarovnik v vtičniku uporabil komponento Viewer. Za pomik na naslednjo nalogo je bila v aplikaciji uporabljena komponenta “Manage Flow Instance Step”.



18. GreenDi – menjava rastlin in semen

VJERAN STRAHONJA, DIJANA OREŠKI, DARKO ANDROČEC, ANA KUTNJAK; PREVOD R. LESKOVAR

18.1 Poslovni vidik primera

Opis platforme GreenDi je kratko opisan v poglavjih 14 in 15. Poglavje 18 se nanaša na del platforme, ki je namenjen izmenjavi rastlin in semen. V tej izmenjavi lahko sodelujejo vsi registrirani uporabniki platforme, ki predstavijo svojo ponudbo ali odgovorijo na ponudbo nekoga drugega. Hkrati naj bi sistem omogočal iskanje ponudb po različnih filtrih, sortiranje po želenih stolpcih in imel vse vgrajene funkcije APEX-a. Poleg tega mora biti omogočena komunikacija med ponudnikom in drugimi uporabniki v zvezi s posamezno ponudbo. Komunikacija naj poteka kot klepet.

18.2 Definicija problema

Problem platforme GreeDI je, ker ne obstaja vrsta nujno potrebnih funkcij:

- osnovni funkcionalnosti izmenjave semen in rastlin sta oddaja lastnih ponudb (ang. ask) in odgovarjanje na ponudbe drugih (ang. bid), kar je rezervirano za člane,
- člani lahko iščejo po katalogu po različnih kriterijih in odprejo poročilo z odprtimi ponudbami za vsako vrsto rastlin iz kataloga,
- platforma ni komercialna, torej se rastline na platformi ne prodajajo ali kupujejo, ampak se izmenjujejo in podarjajo. Zato transakcije niso predmet obdavčitve,
- ponudba vsebuje ponujeno ali zahtevano rastlino, količino, kraj, ponudnika, opombe, datum ponudbe, rok veljavnosti ponudbe, navodila za prevzem ali pošiljanje, status itd.,
- ponudbe (bid ali ask) so prikazane v obliki tabele, z možnostjo razvrščanja po stolpcih ter iskanja in filtriranja po različnih kriterijih (vrsta rastline, datum ponudbe, član, kraj, status itd.),
- vsak član lahko odgovori na katero koli odprto ponudbo,
- pogovor o kateri koli ponudbi poteka v obliki javnega ali zasebnega klepeta,
- osnovna enota za pogovor (klepet) je sporočilo,
- sporočilo vsebuje podatke o članu, ki ga pošilja (samodejno generirano), besedilo sporočila in vidnost (javno, zasebno).
- sistem naj podpira vse faze upravljanja življenjskega cikla ponudb (ustvarjanje, spreminjanje, zapiranje, brisanje in arhiviranje). Ponudbo ustvarja, spreminja, zapira, briše in arhivira

lastnik ponudbe. Administrator lahko izbriše ponudbo, če krši določena pravila in lahko po določenem času arhivira neaktivne ponudbe.

18.3 Primeri uporabe

18.3.1 Opis primera uporabe

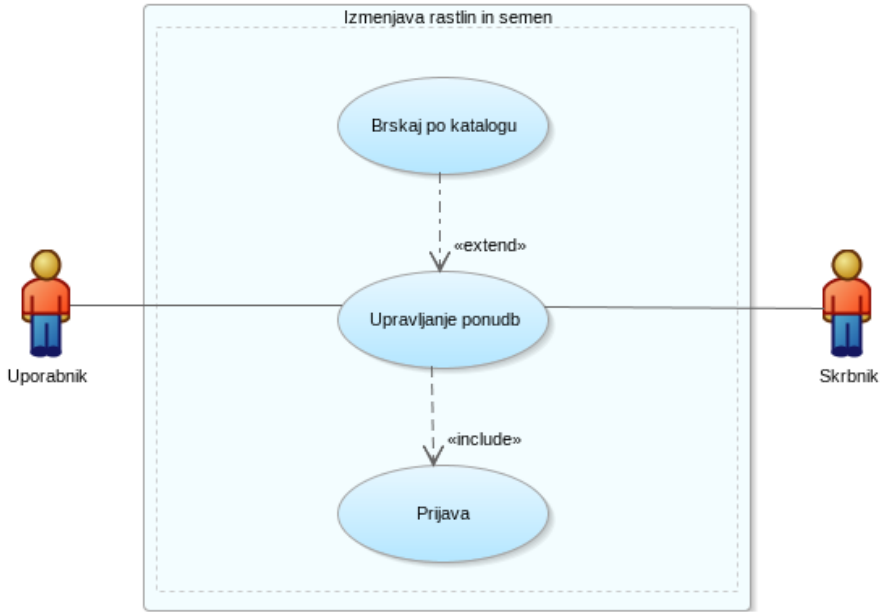
Komunikacija s primerom uporabe "Upravljanje ponudb" je na voljo vsakemu registriranemu uporabniku ali članu platforme GreenDi. Predpogoj za komunikacijo je uspešna prijava. Primer uporabe "Brskaj po katalogu rastlin" je razširitev "Upravljanja ponudb". Skrbnik lahko tudi sodeluje pri upravljanju ponudb, brez omejitev pregleduje vse ponudbe in klepete ter spreminja status posamezne ponudbe.

18.3.2 Delno strukturiran opis

Tabele 18.1 prikazuje primer uporabe.

18.3.3 Diagram primera uporabe

Opis je grafično prikazan na diagramu primera uporabe 18.1.



Slika 18.1: Diagram primera uporabe - izmenjava rastlin in semen.

18.4 Model podatkov

V tem delu prikazujemo modeliranje podatkov.

18.4.1 Opis modela podatkov

Logični podatkovni model (ang. Entity-Relationship model) je pravzaprav nadgradnja podatkovnega modela iz poglavij 14 in 15). Entitete, ki se navezujejo na rastline in tematske enote so povzete iz poglavja 14, uporabnik pa iz poglavja 15. Glavni način realizacije izmenjave sta ponudba in sporočilo. Ponudbo opisujejo naslednji atributi: ID, vrsta zapisa (ponudba ali povpraševanje), količina, lokacija, ponudnik, komentar, datum vnosa, veljanost ponudbe in status. Statusi ponudbe

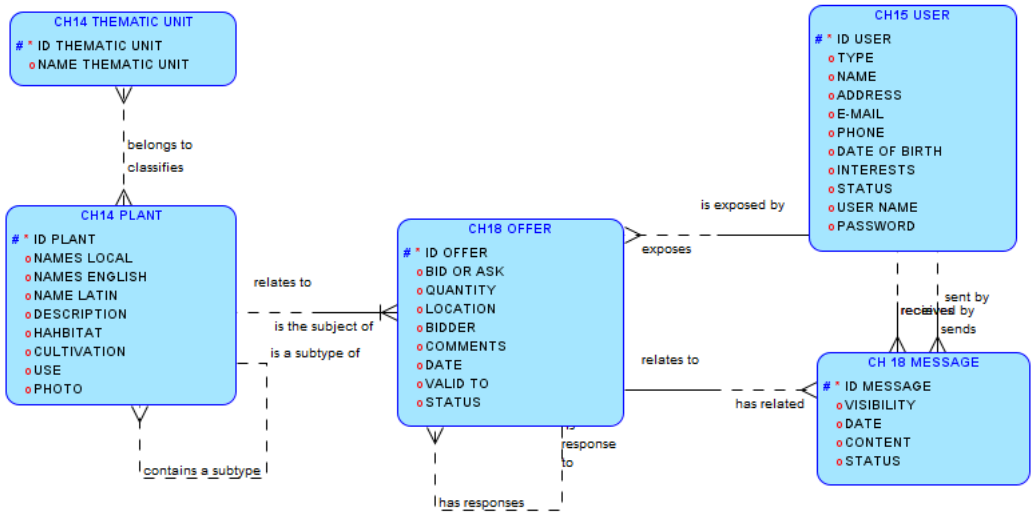
Tabela 18.1: Opis primera uporabe: izmenjava rastlin in semen.

Ključni pojem	**Vrednost**
ID:	*ch18-01*
Naslov:	*Izmenjava rastlin in semen*
Opis:	*Platforma omogoča ponujanje, povpraševanje in menjavanje rastlin in semen.*
Primerni igralec:	*Uporabnik*
Predhodni pogoji:	*Spletna stran z aplikacijo je dostopna, osnovne funkcije so namenjene splošni, anonimni javnosti brez registracije.*
Pogoji po zaključku:	*Če uporabnik vnese ponudbo, je shranjena v bazi*
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Uporabnik se pomika po katalogu, dokler ne najde rastline ali uporabi iskanja* 2. *Uporabnik odpre seznam ponudb za izbrano rastlino ali vse rastline* 3. *Uporabnik se pomika gor in dol po seznamu ponudb ali uporabi iskanje* 4. *Uporabnik odpre novo ponudbo za izbrano rastlino in vnese novo ponudbo* 5. *Uporabnik klikne na izbrano ponudbo in odpre pojavno okno s podrobnostmi o izbrani ponudbi* 6. *Uporabnik lahko odgovori na katero koli aktivno ponudbo s protiponudbo ali sprejemom ali začne klepet* 7. *Klepet je v obliki sporočil in odgovorov* 8. *Povezava je vidna ponudniku in osebam, ki sodelujejp v pogovoru*
Raazširitve:	• *seznam ponudb (poročilo)* • *detajli ponudbe (nova ali obstoječa)* • *klepet (seznam sporočil)*
Pogostost uporabe:	*Približno 100 dnevno.*
Status:	*V razvoju*
Lastnik:	*Uporabnik*
Prioriteta:	*Visoka*

so: aktivna, umaknjena, realizirana. Sporočilo je povezano s ponudbo, vsebuje pa podatek o vidnosti, datum sporočila, vsebino sporočila in status sporočila. Za posamezno rastlino lahko obstaja več ponudb. Uporabnik lahko objavi več ponudb, na ponudbo lahko uporabniki napišejo več sporočil.

18.4.2 Logični model podatkov

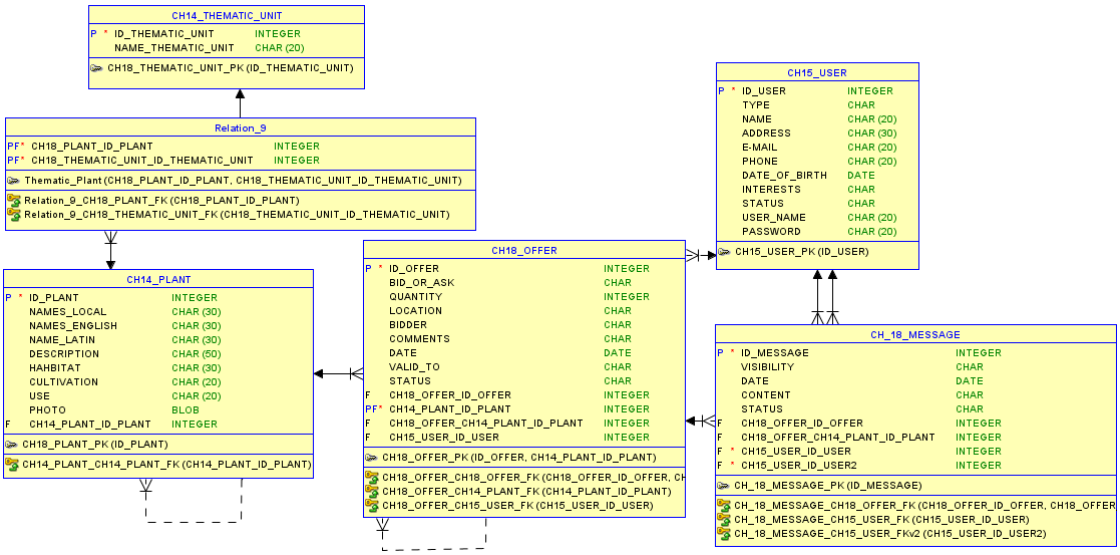
Logični model podatkov je prikazan na sliki 18.2.



Slika 18.2: Logični model podatkov.

18.4.3 Relacijski model podatkov

Preslikavo logičnega modela v relacijski model je možno izvesti z orodjem Oracle SQL Data Modeler. Funkcija se imenuje *Engineer to relational*. Rezultat preslikave, to je relacijski model podatkov, je prikazan na Sliki 18.3.

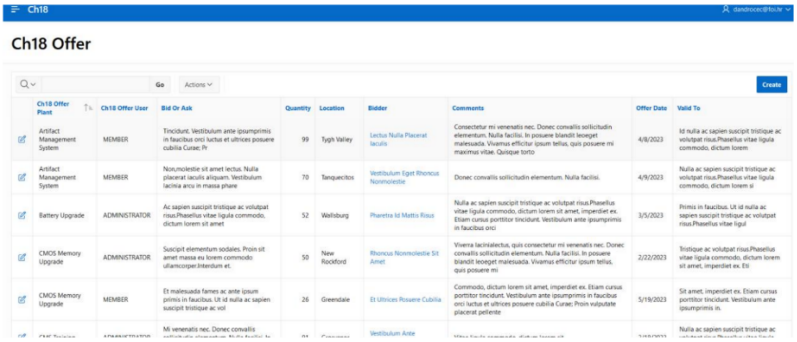


Slika 18.3: Relacijski model podatkov.

V orodju Oracle SQL Developer Data Modeler generiramo skript v SQL. Izberemo vse relacijske tabele na diagramu in sprožimo funkcijo *File* > *Export* > *DDL File*. Skript lahko uvozimo v APEX in ga tam izvedemo.

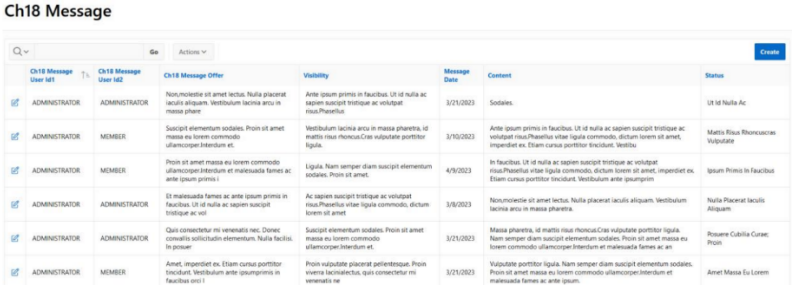
18.5 Uporabniški vmesniki

Interaktivno poročilo o ponudbah je prikazano na Sliki 18.4.



Slika 18.4: Interaktivno poročilo o ponudbah.

Obrazec za vnos sporočil je prikazan na Sliki 18.5.



Slika 18.5: Obrazec za vnos sporočil.

18.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija
- skripti za kreiranje in brisanje tabel ter za dodajanje testnih podatkov v tabele
- video vodič

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter18 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

18.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Namestitev kreira tabele in napolni testne podatke. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

18.6.2 Video vodiči

Video vodiči natančno prikazujejo razvoj aplikacije.

18.7 Vprašanja

1. Kako narediti ponudbo vidno šele od trenutka, kot to želi ponudnik?
2. Ali je možno to aplikacijo uporabiti tudi za izmenjavo drugih dobrin? Kako?
3. V relacijskem podatkovnem modelu ima v tabeli CH18_OFFER polje VALID_TO definiran tip CHAR. Kakšne so morebitne težave, če je datum zapisan kot znakovni tip podatka?

18.8 Odgovori

1. V tabelo CH18_OFFER dodamo polje, ki definira želeni začetek objave. Popravimo tudi obrazec in ga prikaženo. Na poročilu bi dopolnili poizvedbo s pogojem (where polje <= SYSDATE)
2. Da, je možno. Moderator bi odprl novo prazno ponudbo in vsi uporabniki, ki bi vnesli protiponudbo bi lahko sodelovali v klepetu.
3. Shranjevanje datumov v podatkovnem tipu CHAR lahko vodi do problema integritete podatkov, težav pro sortiranju, nezmožnost operacij kot so prištevanje, odštevanje časovnih enot, oblikovanju in podobno. Če je polje dejansko datum, potem ga je smiselno definirati kot DATE ali TIMESTAMP v izogib navedenim problemom.



19. Sistem za ocenjevanje knjig

A. KUTNJAK, L. HRUSTEK, A. BAGGIA AND R.LESKOVAR; PREVOD R. LESKOVAR

19.1 Poslovni vidik primera

Sistem za ocenjevanje knjig se nanaša na knjige, ki so naprodaj na poljubni spletni platformi. Podjetje (npr. Amazon) razmišlja o upravljanju kataloga razpoložljivih knjig tako, da bi se na podlagi ocen uporabnikov odločali o knjižni ponudbi (dodajanje ali umikanje knjig). S sistemom upravljanja pregledov podjetje zbira ocene uporabnikov, oblikuje bazo podatkov in se na podlagi zbranih podatkov odloča o nadaljnjem ukrepanju. V proces sta vključeni dve skupini uporabnikov: ocenjevalci oz. bralci knjig in skrbniki. Primer uporabe in ideja izhajata iz Kaggle – platforme spletne skupnosti za podatkovne znanstvenike in navdušence nad strojnim učenjem. Izvirni podatki so na voljo tukaj. Primeri uporabe in podatkovni model so temeljito spremenjeni in dopolnjeni. Cilj tega poslovnega primera je prikazati razvojne zmožnosti APEX-a, ki temeljijo samo na čarovnikih. Razen Quick SQL (skupaj 31 kratkih vrstic) razvijalec ne bo napisal nobene druge kode.

19.2 Definicija problema

Sistem za ocenjevanje knjig še ne obstaja, imel pa naj bi naslednje funkcije:

- glede na pestro ponudbo knjig v svetu se zbirajo podatki o njihovih ocenah in koristnosti teh ocen.
- platforma vključuje seznam knjig z informacijami o naslovu, avtorjih, opisu vsebine, založniku, datumu objave, kategorijah in ocenah.
- skrbnikove funkcije vključujejo dodajanje knjig v bazo podatkov platforme ter urejanje in izbiranje po kategorijah.
- ker zbirka podatkov uporabnikom omogoča dodajanje mnenj, si lahko skrbnik ogleda ocene in pregleda komentarje. Na podlagi tega skrbnik sprejema odločitve o dejanjih (dodajanje ali odstranjevanje knjige s police).
- uporabnik ima možnost registracije na platformi. To mu omogoča vpogled v podrobnosti knjig, pa tudi njihovo pregledovanje in komentiranje recenzij.
- ocenjevanje koristi uporabnikom. Uporabnik se na podlagi analize ocen knjige odloči za nakup knjige. Na podlagi obstoječih recenzij podjetje ukrepa v zvezi s knjigami, torej jih dodaja ali odstranjuje iz kataloga.

19.3 Primeri uporabe

19.3.1 Opis primera uporabe

Administrator je zadolžen za vnos ali spreminjanje podatkov o knjigah v okviru sistema za ocenjevanje. Ima omogočen vpogled v različne statistične kazalce ocen knjige. Amazonov sistem za upravljanje ocenjevanja temelji na registraciji uporabnikov na platformi. Z registracijo si uporabniki lahko ogledajo katalog knjig, ki so na voljo na platformi. Knjižni katalog vsebuje podrobne informacije o knjigah in je osnova za nadaljnje akcije na platformi. Knjige so razvrščene v kategorije, kar olajša iskanje in omogoča vnos osebnih preferenc uporabnika. Registriran uporabnik lahko doda oceno in komentira oceno.

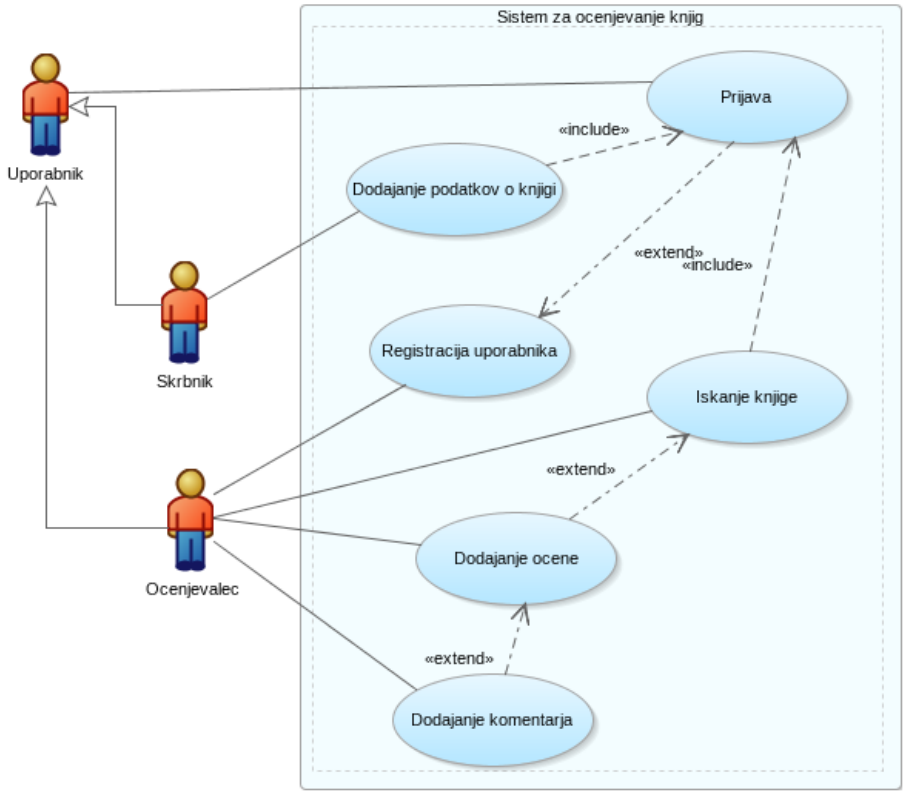
19.3.2 Delno strukturiran opis

Aktivnosti povezane z vnosom novih podatko o knjigi so dodeljene skrbniku. Dnevno obstoječo bazo dopolnjuje z novimi knjigami. Uporabnik opravi registracijo na platformi ocenjevanje knjig. Po uspešni registraciji uporabnik pregleda katalog knjig, njihove recenzije in na koncu doda svojo recenzijo in komentarje na recenzije. Tipični uporabnik se prijavi na platformo večkrat mesečno.

Opisi primerov uporabe so navedeni v tabeli 19.1.

19.3.3 Diagram primera uporabe

Opisana zgodba je prikazana na diagramu primera uporabe (slika 19.1).



Slika 19.1: Diagram primera uporabe.

Tabela 19.1: Opis primera uporabe: ocenjevanje knjig

Ključni pojem	**Vrednost**
ID:	*ch19-10*
Naslov:	*Ocenjevanje knjig*
Opis:	*sistem za ocenjevanje knjig je na voljo registriranim uporabnikom. Platforma omogoča pregledovanje knjig, filtriranje, sortiranje, klasificiranje, dodajanje recenzij in komentarjev o koristnosti recenzij.*
Primarni igralec:	*Registrirani uporabnik*
Predhodni pogoji:	Brskalnik na osebnem računalniku ali telefonu, uporabnik ima poverilnice za dostop, aplikacija je dostopna na strežniku.
Postconditions:	Če je dodana knjiga, recenzija ali komentar recenzije, so podatki shranjeni v bazi.
Glavni del	Scenariji
Uspešen scenarij:	Prvi scenarij (skrbnik doda novo knjigo): 1. Skrbnik se prijavi v aplikacijo 2. Skrbnik izbere obrazec za dodajanje knjige 3. Skrbnik vpiše naslov, avtorje, opis, založnika, datum izdaje in kategorijo 4. Skrbnik potrdi dodajanje z gumbom 5. Skrbnik se odjavi iz aplikacije Drugi scenarij (uporabnik doda recenzijo): 1. Uporabnik se registrira v aplikacijo 2. Uporabnik brska po knjiga in izbere želeno knjigo za recenziranje 3. Uporabnik vpiše podatke - recenzijo ali oceno koristnosti recenzije v obrazec 4. Uporabnik potrdi dodajanje z gumbom 5. Uporabnik se odjavi iz aplikacije Tretji scenarij (uporabnik komentira recenzijo): 1. Uporabnik se registrira v aplikacijo 2. Uporabnik brska po recenzijah in izbere želeno recenzijo 3. Uporabnik vpiše podatke - oceno koristnosti recenzije v obrazec 4. Uporabnik potrdi dodajanje z gumbom 5. Uporabnik se odjavi iz aplikacije
Razširitev:	• *Dodajanje knjige* • *Interaktivno poročilo o knjigah* • *Recenzije knjig* • *Komentar recenzij*
Pogostost uporabe:	*Večkrat dnevno*
Status:	[V razvoju]
Lastnik:	Registrirani uporabnik
Prioriteta:	*Visoka*

19.4 Model podatkov

19.4.1 Opis modela podatkov

Logični model podatkov sestavlja več entitet (Entity-Relationship model):

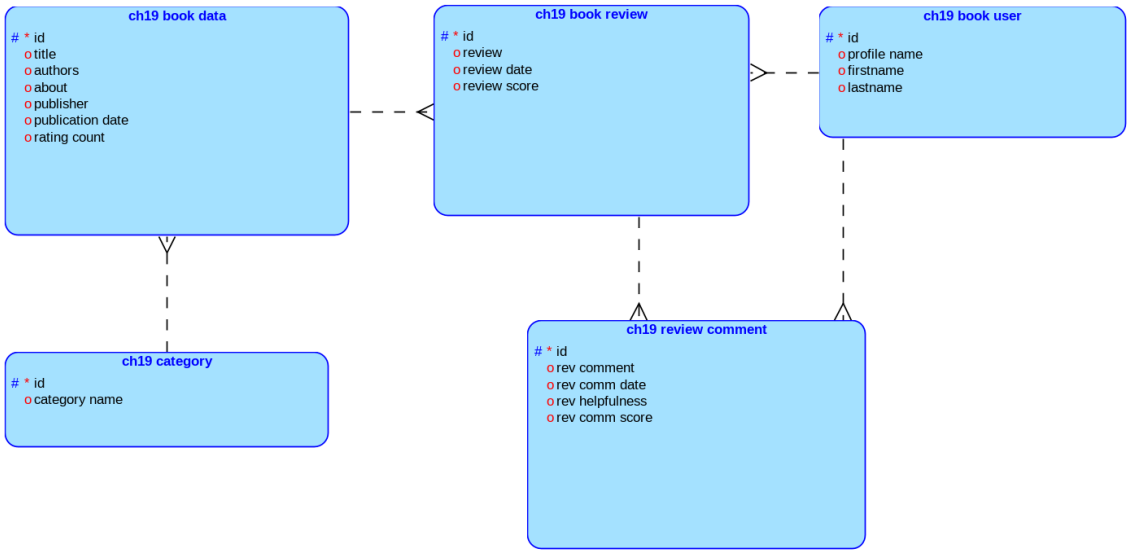
- ch19_category,
- ch19_book_data,
- ch19_book_review,
- ch19_book_user in
- ch19_book_review_comment.

Entiteta ch19_category ima samo en atribut, ki opisuje naziv kategorije. Entiteta ch19_book_data vsebuje naslov knjige, avtorje, kratek opis, založnik, datum izdaje in poprečno oceno. Entiteta ch19_book_user vsebuje podatke o uporabniku kot so ime profila, ime in priimek. Entiteta ch19_book_review je namenjena recenzijam (besedilo recenzijem datum in ocena). Ocena knjige je celo število med 1 in 5. Entiteta ch19_review_comment vsebuje odzive, komentarje na recenzije in je namenjena zaznavanju koristnosti recenzije. Koristnost recenzije je podana s celim številom med 1 in 5. Vsaka knjiga pripada samo eni kategorij, kategorijA vsebuje več knjig. Na knjigo je lahko podanih več recenzij, tudi uporabnik lahko napiše več recenzij za različne knjige. Vsaki entiteti smo določili primarni ključ, je je celo število.

Razvijalec ima več možnosti za ustvarjanje tabel. Ena je uporaba SQL Developer Data Modeler in načrtovanje logičnega modela od začetka, pretvorba v relacijskega in ustvarjanje skripte v SQL. Drugi način je ustvarjanje tabel v APEX, izvoz skriptov, ki ustvarjajo tabele, uporaba obratnega inženiringa v SQL Developer Data Modeler za ustvarjanje relacijskega modela in nato preoblikovanje v logični model. Tretji način je uporaba APEX-ovega Quick SQL. Ko s Quick SQL napišemo opis tabel, jih generator pretvori v skript z SQL sintakso.

19.4.2 Logični model podatkov

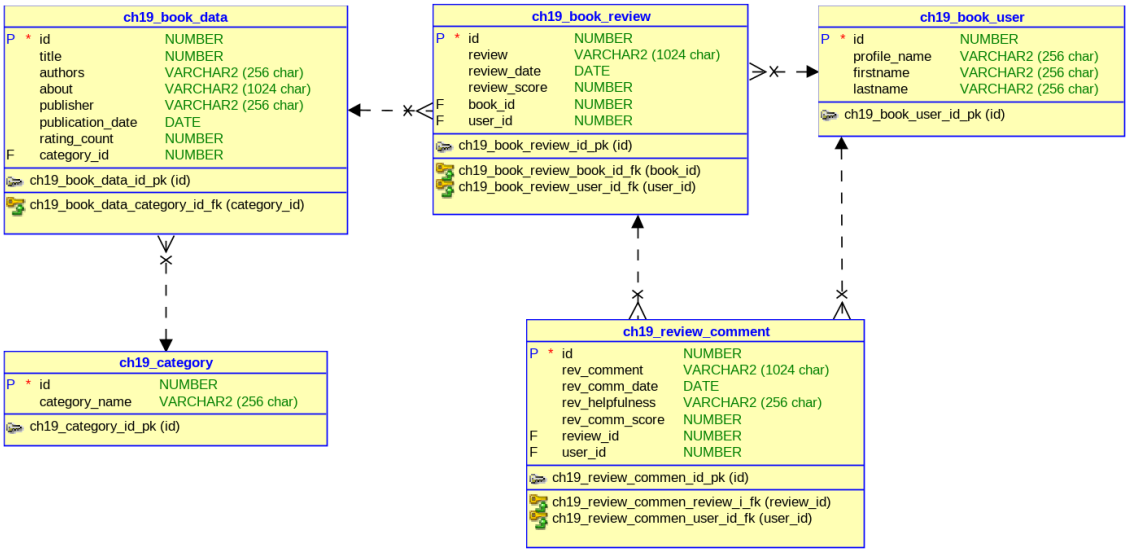
Logični model podatkov je prikazan na sliki 19.2.



Slika 19.2: Logični model podatkov.

19.4.3 Relacijski model podatkov

Samodejno pretvorbo iz logičnega podatkovnega modela v relacijski podatkovni model v Oracle SQL Data Modeler zagotavlja funkcija *Engineer to relational*. Rezultat pretvorbe je relacijski model podatkov (slika 19.3), pripravljen za izvoz kot skript SQL. .



Slika 19.3: Relacijski model podatkov.

V orodju Oracle SQL Data Modeler poženemo generiranje skripta SQL. Izberemo vse tabele in uporabimo funkcijo File > Export > DDL File. Generirani skript uvozimo v APEX in ga izvedemo.

19.4.4 Skript v SQL

Spodnji skript odraža tisto, kar smo oblikovali z Oracle SQL Developer Data Modeler:

```
1  create table ch19_category (
2          id number generated by default on null as identity
3          constraint ch19_category_id_pk primary key,
4          category_name varchar2(256 char));
5
6  create table ch19_book_user (
7          id number generated by default on null as identity
8          constraint ch19_book_user_id_pk primary key,
9          profile_name varchar2(256 char),
10         firstname varchar2(256 char),
11         lastname varchar2(256 char));
12
13 create table ch19_book_data (
14         id number generated by default on null as identity
15                 constraint ch19_book_data_id_pk primary key,
16         title varchar2(256 char),
17         authors varchar2(256 char),
18         about varchar2(1024 char),
19         publisher varchar2(256 char),
20         publication_date date,
21         rating_count number,
22         category_id      number
23         constraint ch19_book_data_category_id_fk
24         references ch19_category on delete cascade);
25
26 create table ch19_book_review (
```

```
        id number generated by default on null as identity
        constraint ch19_book_review_id_pk primary key,
        review varchar2(1024 char),
        review_date date,
        review_score  number
                constraint ch19_book_revie_review_scor_ck
                check (review_score in (1,2,3,4,5)),
        book_id   number
        constraint ch19_book_review_book_id_fk
        references ch19_book_data on delete cascade,
        user_id   number
                constraint ch19_book_review_user_id_fk
references ch19_book_user on delete cascade);

create table ch19_review_comment (
        id  number generated by default on null as identity
                constraint ch19_review_commen_id_pk primary key,
        rev_comment varchar2(1024 char),
        rev_comm_date date,
        rev_helpfulness varchar2(256 char),
        rev_comm_score number
                constraint ch19_review_c_rev_comm_scor_ck
                check (rev_comm_score in (1,2,3,4,5)),
        review_id number
                constraint ch19_review_commen_review_i_fk
                references ch19_book_review on delete cascade,
        user_id   number
                constraint ch19_review_commen_user_id_fk
                references ch19_book_user on delete cascade);
```

19.4.5 Skript Quick SQL

Najhitrejši način za generiranje skripta v SQL, ki bo ustvaril tabele, je v APEX-u dostopen v funkciji Quick SQL. Če spodnje vrstice vpišemo v urejevalnik Quick SQL, bomo pri generiranju SQL dobili enak rezultat, kot z orodjem Oracle SQL Developer Data Modeler:

```
ch19_category
        category_name vc256

ch19_book_user
        profile_name vc256
        firstname vc256
        lastname vc256

ch19_book_data
        title vc256
        authors vc256
        about   vc1024
        publisher  vc256
        publication_date date
        rating_count num
        category_id /fk ch19_category
```

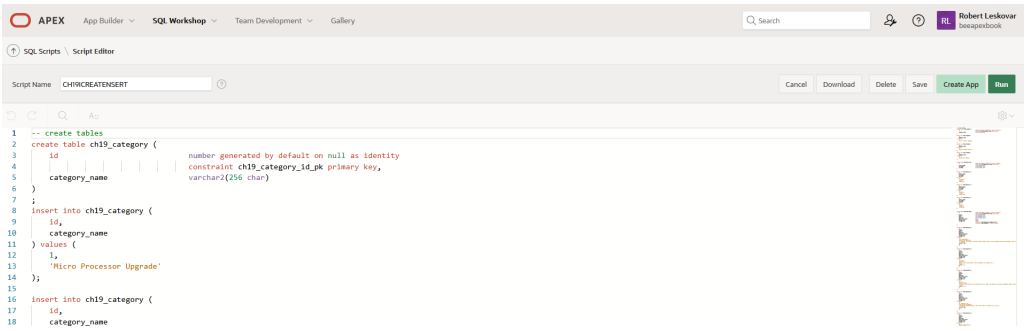
```
17
18 ch19_book_review
19         review vc1024
20         review_date date
21         review_score  num /check 1,2,3,4,5
22         book_id /fk ch19_book_data
23         user_id /fk ch19_book_user
24
25 ch19_review_comment
26         rev_comment vc1024
27         rev_comm_date date
28         rev_helpfulness vc256
29         rev_comm_score  num /check 1,2,3,4,5
30         review_id /fk ch19_book_review
31         user_id /fk ch19_book_user
```

Izvorna koda Quick SQL je zelo kratka in zgoščena. Za uporabo se moramo naučiti le nekaj osnovne sintakse, ob tem, da razumemo principe modeliranja podatkov.

19.5 Uporabniški vmesniki

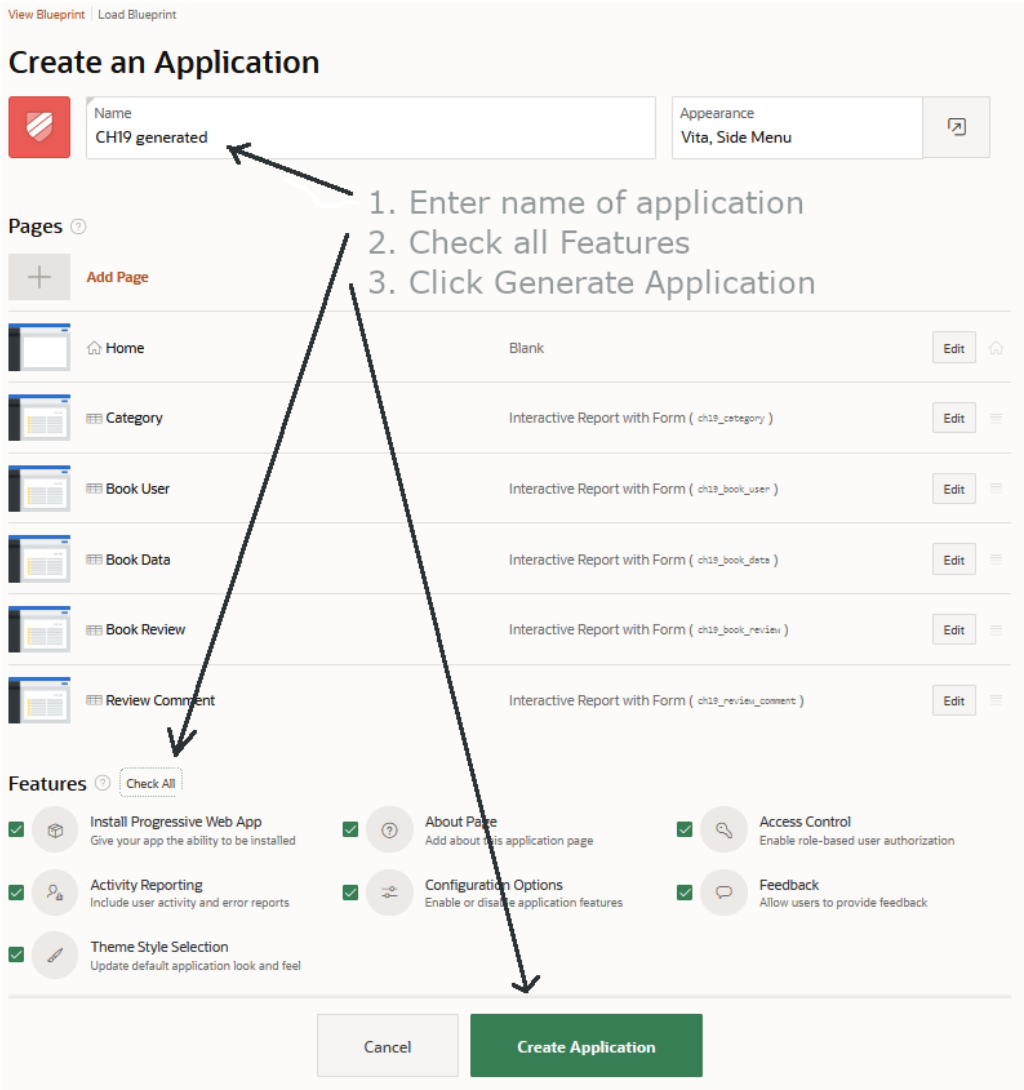
19.5.1 Administrator

V tej aplikaciji bomo generirali skript v SQL s Quick SQL. Quick SQL ima možnost generiranja naključnih podatkov in sicer tako, da pri generiranju skripta SQL na koncu napiše še ukaze za dodajanje zahtevanega števila vrstic v določeno tabelo. V dopolnilnem študijskem gradivu poiščite skript CH19CREATEINSERT.sql, ga uvozite v delovni prostor, ter zaženete, da ustvari tabele in jih napolni s testnimi podatki. Pri skriptu imate tudi možnost kreiranja aplikacije s klikom na “Create App” (slika 19.4).



Slika 19.4: Generiranje aplikacije s pomočjo skripta CH19CREATEINSERT - korak 1.

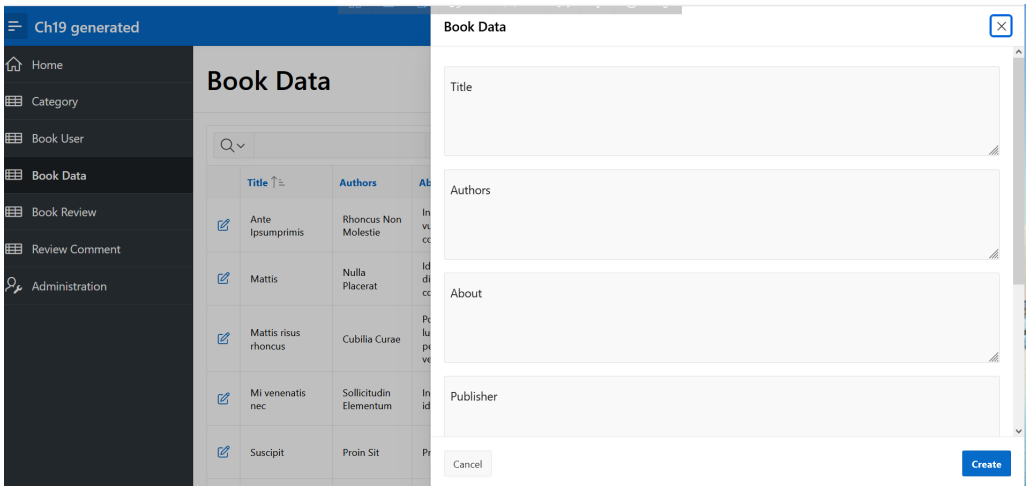
Vnesemo samo ime aplikacije, izberemo možnosti in kliknemo “Generate Application” (slika 19.5).



Slika 19.5: Generiranje aplikacije s pomočjo skripta CH19CREATEINSERT - korak 2.

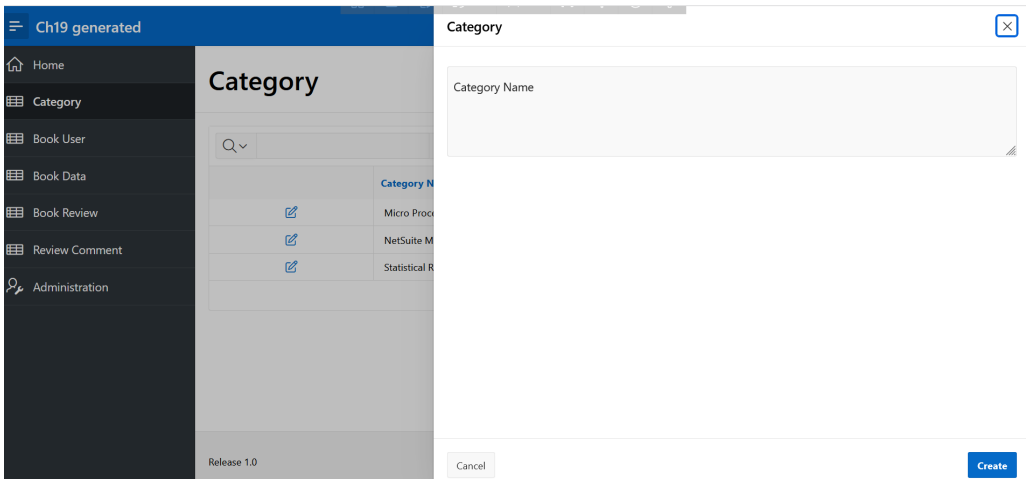
Delujoč prototip aplikacije je pripravljen v manj kot dveh minutah. Od tu dalje predstavljamo funkcije za skrbnika in končnega uporabnika. Do tega trenutka razvijalcu ni bilo treba napisati ene same samcate vrstice izvorne kode aplikacije (z izjemo Quick SQL) in razvijalec ni prilagodil nobenega aplikacijskega vmesnika. Dejansko kodo za aplikacijo so v celoti sestavili čarovniki v APEX-u.

Slika 19.6 prikazuje obrazec za vnos nove knjige, ki je namenjen skrbniku.



Slika 19.6: Skrbnik - dodajanje knjige.

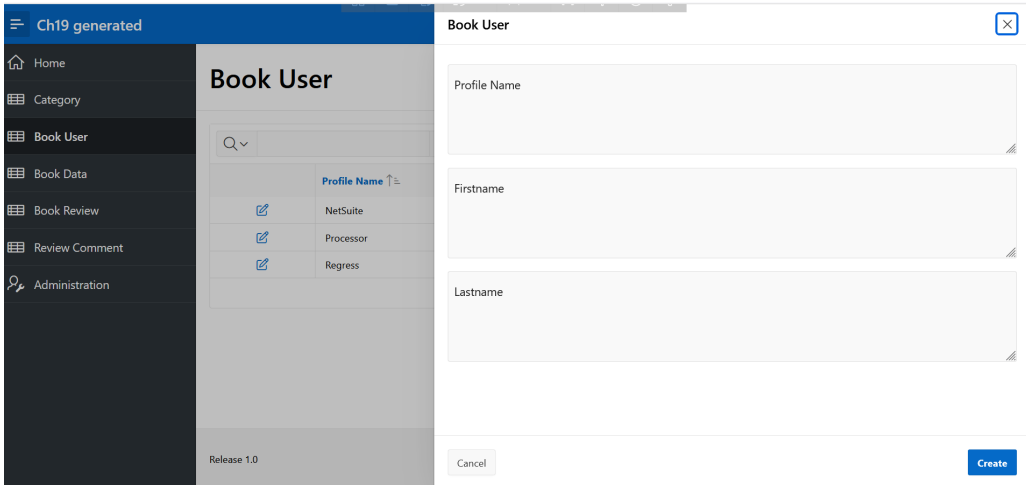
Slika 19.7 prikazuje obrazec za dodajanje kategorij knjig, ki je namenjen skrbniku.



Slika 19.7: Skrbnik - dodajanje kategorije.

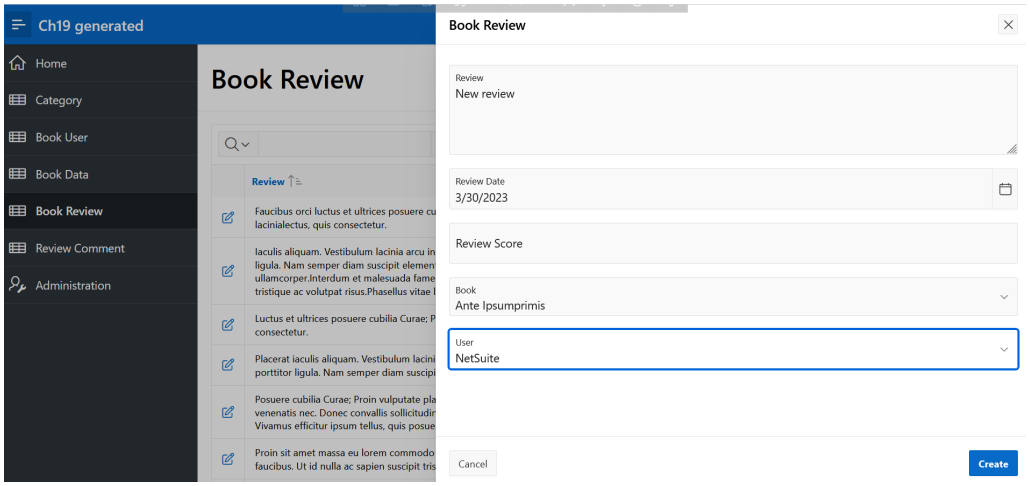
19.5.2 Uporabnik

Slika 19.8 prikazuje obrazec za registracijo uporabnika.



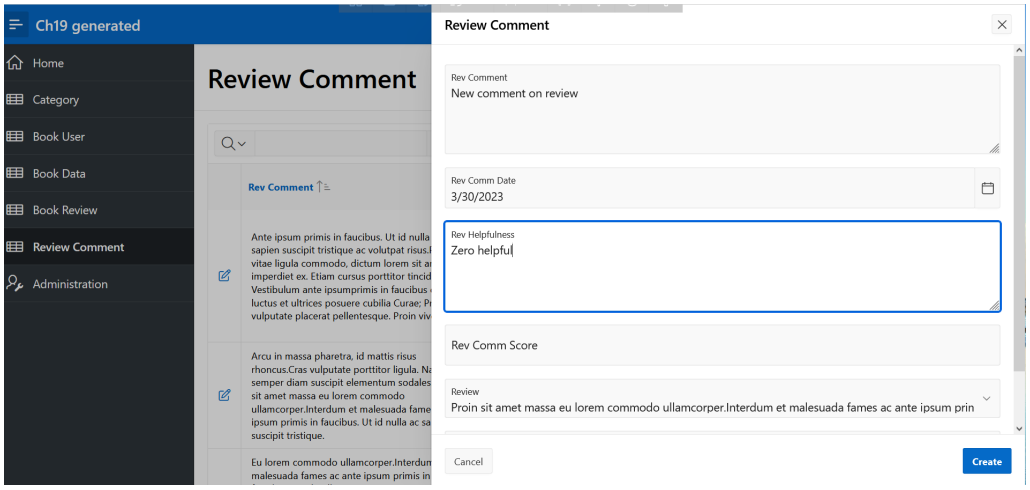
Slika 19.8: Registracija uporabnika.

Slika 19.9 prikazuje brskanje po knjigah in dodajanje recenzije.



Slika 19.9: Brskanje po knjigah in dodajanje recenzije.

Slika 19.10 prikazuje obrazec za komentiranje recenzije.



Slika 19.10: Obrazec za komentiranje recenzije.

S čarovnikom generirana poročila so privzeto tabelarična. Seveda jih lahko končni uporabnik brez znanja programiranja in brez vrstice napisane kode, pretvori v grafikone. Uporabi izključno zmožnosti, ki so vgrajene v poročilo. V poglavju 8 so osnovni napotki za transformacijo tabelaričnega poročila v grafikone.

19.6 Definiranje vlog uporabnikov

Vloge uporabnikov definiramo v “Application Access Control” v “Shared Components”. Če želite spoznati osnove te funkcije, poglejte poglavje 13, za vse podrobnosti pa preštudirajte APEX-ovo dokumentacijo o avtorizaciji.

19.7 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- skript za kreiranje in polnenje tabel
- skript za brisanje tabel
- izvožena pakirana aplikacija
- video vodič, ki prikazuje generiranje aplikacije iz skripta

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter19 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

19.7.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo testni podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

Pakirana aplikacija testirana in deluje. Pri prenosu v nov delovni prostoru morajo biti izpolnjeni naslednji pogoji:

- dodajte APEX-ovega uporabnika pred zagonom aplikacije. Samo v razvojnem in testnem okolju uporabite “Shared Components > Application Access Control > Add User Role Assignment”. Nato vnesite kreiranega uporabnika in mu dodajte vloge Administrator, Contributor in Reader. V produkcijskem okolju se morate nujno posvetovati z izkušenimi razvijalci aplikacij v APEX-u.

Če uporabnik nima ustrezne vloge, se bo aplikacija sesula. V primeru sesutja najprej počistite piškotke v brskalniku, npr. v Firefox: Settings > Cookies and Site Data > Manage Data.

19.7.2 Video vodiči

Video vodič natančno prikazuje vse korake v razvoju te aplikacije.

19.8 Vprašanja

1. Zakaj smo v tem primeru uporabili skript v Quick SQL?
2. Kako v enem koraku generiramo obrazcev in poročil?
3. Kako avtoriziramo uporabnika v APEX-ovi aplikaciji?

19.9 Odgovori

1. Quick SQL skript je bil predstavljen z namenom pohitritve razvoja modela podatkov. Skript Quick SQL na zahtevo ustvari skript SQL. Skript SQL je izvršljiv, kar pomeni, da ustvari objekte baze podatkov, kot so tabele in pogledi. V tej aplikaciji smo preko skripta Quick SQL zelo hitro zagnali čarovnika za generiranje aplikacije.
2. Več obrazcev in poročil je možno generirati v enem koraku, če napišemo skript z definicijo več tabel.
3. Za avtorizacijo uporabnika znotraj APEX-ove aplikacije mora biti aplikacija ustvarjena s funkcijo “Access Control Feature”. Pooblaščeni uporabnik (Administrator) lahko doda druge uporabnike z vlogami Administrator, Contributor in Reader.



20. Materialna kosovnica in kalkulacija stroškov

R. LESKOVAR, U. RAJKOVIČ, A.BAGGIA; PREVOD R. LESKOVAR

20.1 Poslovni vidik primera

Opazovano majhno podjetje, ki se bo imenovalo **OSC** (10-20 zaposlenih) proizvaja kabelske sestave, kabelske snope, vodnike in signalne svetilke. Nudijo tudi montažo elektromehanskih delov in mehanskih polizdelkov. Njihovi kupci so večja podjetja, ki ponujajo hišne aparate, merilno opremo, krmilne sisteme, števce električne energije, medicinske pripomočke in podobno. Naročila strank sprejemajo po telefonu in elektronski pošti. Za enostavne izdelke in operacije ter za ponavljajoča se naročila **OSC** zelo hitro pošlje potrditev kupcu. V zadnjem času naraščajo naročila večnivojsko sestavljenih izdelkov in kompleksnejših polizdelkov. Poleg tega je v enem naročilu več deset artiklov in vsak je lahko sestavljen iz več polizdelkov in materialov. Polizdelki so lahko sestavljeni iz polizdelkov in materialov. V praksi se prikazu sestave izdelka reče tudi materialna kosovnica (ang. bill of material, kratko BOM). Stranka želi hitro prejeti ponudbo. Izdelava kalkulacije cene materiala in polizdelkov je za večnivojsko sestavljene izdelke in polizdelke zahtevna, podvržena napakam in dolgotrajna. Zato lahko traja nekaj dni, da stranki odgovorijo s ponudbo. Obračun stroškov dela za posamezno naročilo je še en izračun, ki je zaradi preglednosti v poglavju izpuščen. Obseg tega poglavja bo torej samo izračun kosovnice. Direktor podjetja je izrazil naslednje potrebe:

- poročilo o izdelkih, polizdelkih, materialu in morebitnih povpraševanjih strank
- poročilo o listnici materiala
- obrazec za urejanje izdelkov, polizdelkov, materiala in morebitnih povpraševanj strank
- obrazec za urejanje kosovnice
- izračun in drevesni prikaz kosovnice

Aplikacija mora biti uporabna na telefonu, tablici ali namiznem računalniku.

20.2 Definicija problema

Direktor se sooča z več težavami, povezanimi z agilnim odzivom:

- izračun je zamuden, podvržen napakam in zahteven, vendar ga je treba pripraviti zelo hitro
- zamude in računske napake lahko vplivajo na poslovanje na več načinov, kot so izguba potencialnih kupcev, nižji finančni rezultati, zmanjševanje poslovnega ugleda, zniževanje prodaje in zaposlenih in podobno.

Zgornja tveganja je treba učinkovito zmanjšati s spletno aplikacijo.

Tabela 20.1: Opis primera uporabe: poročanje in vzdrževanje osnovnih podatkov.

Ključni pojem	**Vrednost**
ID:	*Ch20-01*
Naslov:	*Poročanje in vzdrževanje osnovnih podatkov*
Opis:	*Direktor uporablja aplikacijo APEX za poročanje in vzdrževanje (izbiranje, ažuriranje) osnovnih podatkov ne glede na to, ali gre za izdelek, polizdelek, material ali povpraševanje. Materiali in agregati so opisani z istim nizom atributov. Ažuriranje cen materialov ne bo sprožilo ponovnega izračuna cen agregatov, v katerih je material uporabljen. Povpraševanje kupca je lahko sestavljeno iz več izdelkov in polizdelkov. Merska enota v tem primeru bo kos.*
Glavni igralec:	*Direktor*
Predhodni pogoji:	*Direktor ima uporabniški račun v APEX-ovi aplikaciji.*
Pogoji po zaključku:	*Po poročanju ali vzdrževanju podatkov je mogoče izvesti drugo dejanje, vključno z izračunom kosovnice.*
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v aplikacijo.* 2. *V meniju ali navigacijski strani izberite "Prod./Semi/Material".* 3. *Za urejanje se pomaknite na strani ali filtrirajte podatke, da dobite želeni izdelek, polizdelek ali material.* 4. *Kliknite ikono peresa pri določenem izdelku, polizdelku ali materialu in spremenite podatke.* 5. *Potrdite spremembe s klikom na gumb "Apply changes".*
Razširitve:	-
Pogostost uporabe:	*V poprečju 5 krat dnevno*
Status:	*Zaključen*
Lastnik:	*Direktor*
Prioriteta:	*Visoka*

20.3 Primeri uporabe

20.3.1 Opis primera uporabe

Določimo lahko osnovne naloge:

- Poročanje in vzdrževanje podatkov o izdelkih, polizdelkih in materialih. Porabo obravnavamo kot poročanje v tabelarnih in drevesnih oblikah, vzdrževanje pa kot dodajanje, posodabljanje in brisanje.
- Izračunavanje materialne klosovnice za specifično povpraševanje, izdelek ali polizdelek.

Nazivi primerov uporabe so torej "Sporočilo in vzdrževanje osnovnih podatkov", "Poročanje in vzdrževanje strukturnih podatkov" in "Izračun BOM".

20.3.2 Delno strukturiran opis

Zgornja zgodba je prikazana na diagramu primerov uporabe.

Tabela 20.2: Opis primera uporabe: poročanje in vzdrževanje podatkov o sestavi.

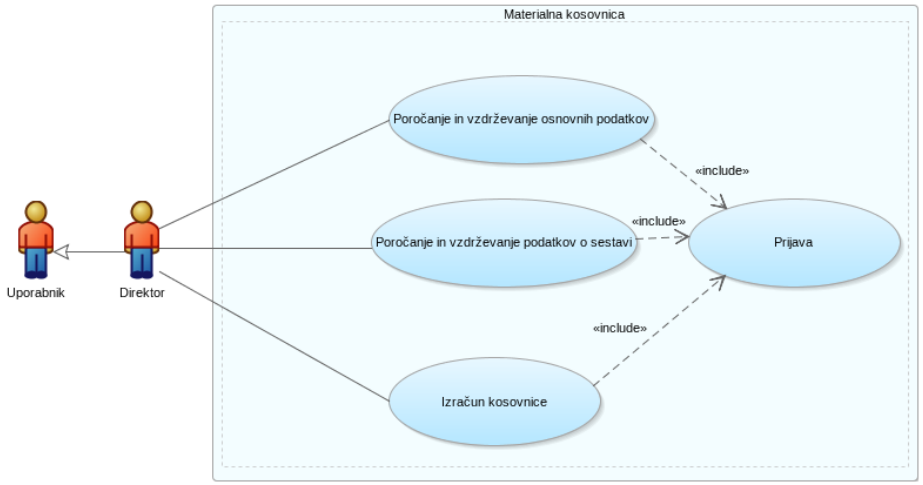
Ključni pojem	**Vrednost**
ID:	*Ch20-02*
Naslov:	*Poročanje in vzdrževanje podatkov o sestavi*
Opis:	*Direktor uporablja aplikacijo APEX za poročanje (pregled izbranih podatkov) in vzdrževanje (vnos in brisanje) strukture izdelkov, polizdelkov in materiala. Povpraševanje kupca je lahko sestavljeno iz več izdelkov in polizdelkov. Vstavljanje novih ali brisanje obstoječih artiklov ne bo sprožilo (ponovnega) izračuna cen agregata.*
Primary Actor:	*Manager*
Predhodni pogoji:	*Direktor ima uporabniški račun v APEX-ovi aplikaciji.*
Pogoji po zaključku:	*Po poročanju ali vzdrževanju podatkov je mogoče izvesti drugo dejanje, vključno z izračunom kosovnice.*
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v aplikacijo.* 2. *V meniju ali navigacijski strani izberite "Bill-of-material".* 3. *Za dodajanje kliknite gumb "Create".* 4. *Za brisanje postavke v strukturi kliknite "Delete".*
Razširitve:	-
Pogostost uporabe:	*V poprečju 30 na mesec*
Status:	*Zaključen*
Lastnik:	*Direktor*
Prioriteta:	*Visoka*

Tabela 20.3: Opis primera uporabe: izračun kosovnice.

Ključni pojem	**Vrednost**
ID:	*Ch20-03*
Naslov:	*Izračun kosovnice*
Opis:	*Direktor z aplikacijo v APEX-u izračuna ceno strukture, ki jo opisuje kosovnica za določen artikel, ne glede na to, ali gre za izdelek, polizdelek ali povpraševanje. Dobljene cene se ažurirajo za vse postavke, agregirane v strukturi, ne glede na njihovo kompleksnost in globino, vključno z zgornjim agregatom.*
Glavni igralec:	*Direktor*
Predhodni pogoji:	*Direktor ima uporabniški račun v APEX-ovi aplikaciji.*
Pogoji po zaključku:	*Po izračunu je mogoče izvesti katero koli drugo dejanje, vključno z novim izračunom kosovnice.*
lavni del	*Scenariji*
Uspešen scenarij:	1. *Odprite spletni brskalnik in se prijavite v aplikacijo.* 2. *V meniju ali navigacijski strani izberite "Tree view and calculation".* 3. *Izberite agregat, ki ga želite izračunati.* 4. *Kliknite gumb "Calculate". Prikaže se sporočilo o statusu izračuna in izračunani ceni.* 5. *Razširite ali strnite drevesno strukturo izbranega elementa, da prikažete podatke o sestavinah.*
Razširitve:	-
Pogostost uporabe:	*V poprečju 30 na mesec*
Status:	*Zaključen*
Lastnik:	*Direktor*
Prioriteta:	*Visoka*

20.3.3 Diagram primera uporabe

Diagram primera uporabe prikazuje opisano zgodbo (Slika 20.1).



Slika 20.1: Diagram primera uporabe.

20.4 Model podatkov

20.4.1 Opis modela podatkov

V tem primeru sta v podatkovnem modelu samo dve entiteti. Entiteta **ch20 psm** (kasneje tabela) opisuje izdelke, polizdelke, materiale in povpraševanja strank s samo tremi atributi, ki so skupni vsem: ID, opis, enota mere in cena na enoto mere. Zahtevane so vrednosti za vse atribute. Zakaj? Nesmiselno je hraniti primerek, ne da bi vedeli, kako se imenuje, kako se meri in kakšna je cena na enoto. ID je edinstvena številka brez decimalnih delov. Opis mora vsebovati vsaj en natisljiv znak. Domena merskih enot je omejena na milimeter, centimeter, meter in kos. Zaradi majhnega števila različnih vrednosti merskih enot ne bomo uvedli nobene dodatne entitete, ki bi predstavila dejstvo, da ima ena merska enota veliko pojavov v **ch20 psm**, medtem ko se en primerek **ch20 psm** nanaša samo na en primerek merske enote.

Za predstavitev poljubne hierarhične strukture (seznam materiala poljubne širine in globine) ima entiteta **ch20 bom** samo tri atribute: identifikacijo sestavine, identifikacijo agregata in količino sestavine, ki ga vsebuje agregat. Prisotne morajo biti vrednosti za vse atribute. Par (**identifikacija sestavine, identifikacija agregata**) je edinstven, vendar bomo dodali primarni ključ, ustvarjen kot zaporedje celih vrednosti, da izkoristimo prednosti malokodnega okolja APEX. En primerek **ch20 bom** ima lahko dvojno povezavo z **ch20 psm**: sestavina se nanaša na en primerek **ch20 psm** in agregat se nanaša na en primerek **ch20 psm**. V nasprotni smeri imamo: en primerek **ch20 psm** lahko kaže na veliko sestavin in veliko agregatov v entiteti **ch20 bom**. Zato imamo dve relaciji 1 proti mnogo med **ch20 psm** in **ch20 bom**. Opozoriti je treba še na eno stvar glede logičnega podatkovnega modela: edinstvenost para **(identifikacija sestavine, identifikacija agregata)**. Logični model v Oracle Data Modeler nima "sintakse", ki bi to edinstvenost izrazila kot poseben objekt v bazi podatkov, imenovan indeks. Toda na relacijskem podatkovnem modelu je mogoče uvesti edinstvenost določenega para vrednosti npr. indeks. V relacijskem podatkovnem modelu je edinstveni indeks sestavljen iz dveh polj v tabeli. Ta edinstveni indeks zagotavlja, da v bazi podatkov ne bo shranjen noben dvojnik vrednosti parov - mehanizem baze podatkov bo zaščitil razvijalca in končnega uporabnika pred napakami, ki bi prizadele celovitost podatkov.

20.4.2 Vpeljava poslovnih pravil v model podatkov

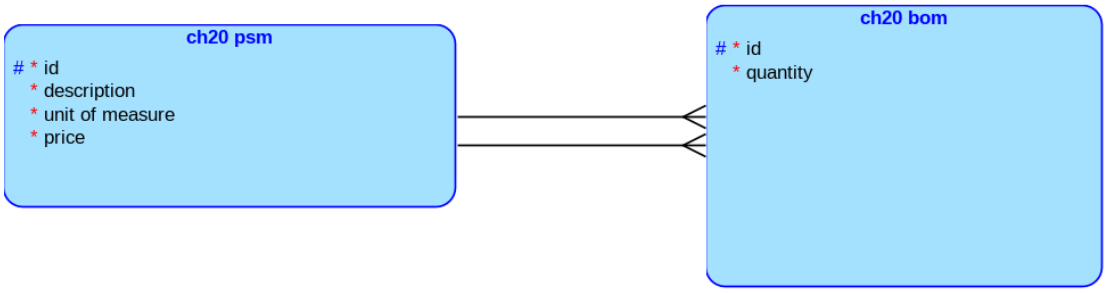
Logični in relacijski podatkovni model je preprost. V matematiki je kosovnica posebna vrsta grafa, kjer je vsako vozlišče povezano samo z enim nadrejenim vozliščem, razen korensko vozlišče, ki nima nadrejenega vozlišča. Zelo enostavno je razbiti hierarhijo z uvedbo dodatne povezave med vozlišči ali brisanjem povezave. V kontekstu baze podatkov spremembe v povezavah vplivajo na celovitost podatkov. Shranjevanje ciklične strukture v **ch20 bom** bi povzročilo, da se izračun kosovnice ne bi nikoli končal. V kontekstu poslovnih pravil je nemogoče in prepovedano, da je avto (izdelek) sestavljen iz koles (polizdelek) in hkrati to kolo sestavljeno iz avtomobilov. Tudi avto ne more biti sestavljen iz samega sebe. Da bi preprečili kršitev poslovnih pravil, bomo uvedli sprožilni mehanizem (drug objekt baze podatkov, ang. trigger), ki je običajno predstavljen z nekaj vrsticami kode v jeziku PL/SQL. Prožilec se bo izvedel pred pisanjem ali posodabljanjem **ch20 bom**. Takšna koda bi preprečila dodajanje v **ch20 bom**, če par v obratnem vrstnem redu že obstaja ali če imata nov sestav in agregat enako vrednost. Implementacije tovrstnih pravil ni mogoče predstaviti na logičnem ali relacijskem podatkovnem modelu.

Prav tako ni mogoče predstaviti, kako bo izračun kosovnice implementiran v podatkovnem modelu. To poglavje bo zagotovilo vpogled v kodo, ki obrne hierarhijo na glavo (od spodaj navzgor) in začne seštevati skupne stroške od spodaj - tj. najprej izračuna stroške materiala v polizdelkih in njihove cene polizdelkov v izdelku. Ne pozabite, da je podatkovni model z dvema entitetama ali dvema tabelama sposoben predstaviti kakršno koli kompleksnost (širino in globino hierarhije) strukture kosovnice.

Zato najprej začnimo s podatkovnimi modeli.

20.4.3 Logični model podatkov

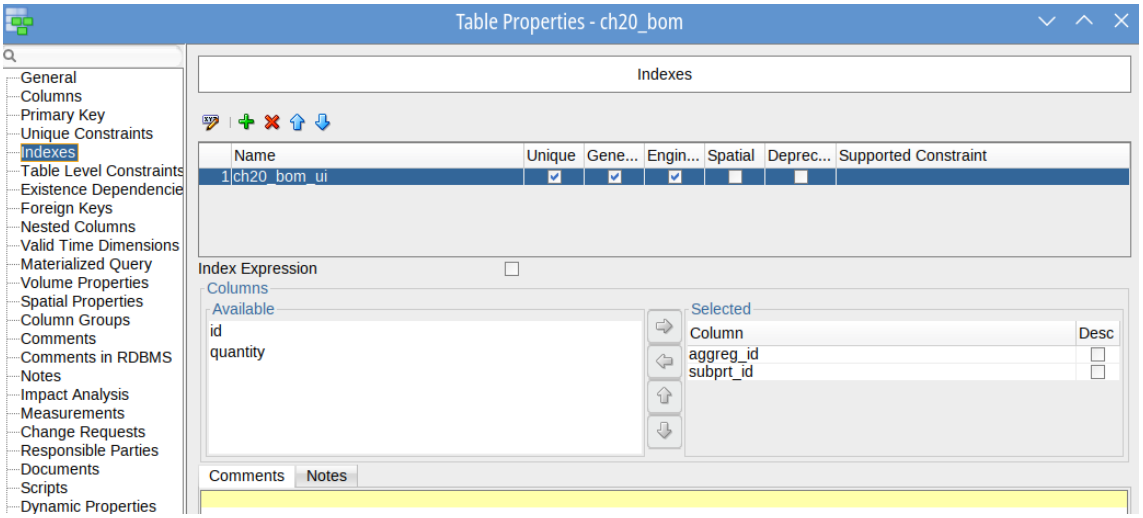
Logični podatkovni model je predstavljen na Sliki 20.2.



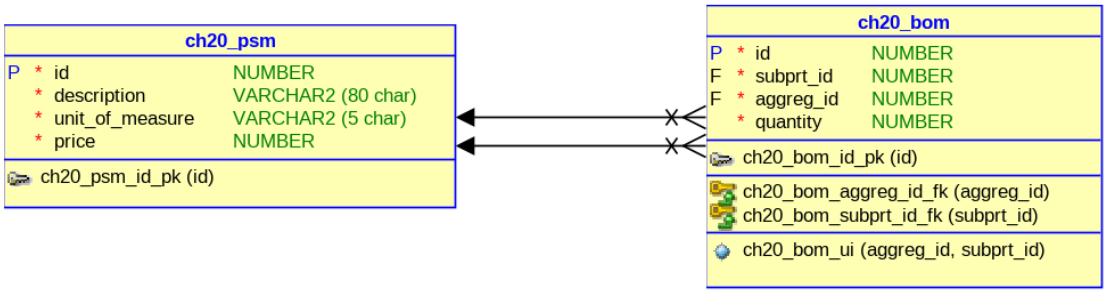
Slika 20.2: Logični model podatkov.

20.4.4 Relacijski model podatkov

Samodejno pretvorbo iz logičnega podatkovnega modela v relacijski podatkovni model v Oracle SQL Data Modeler zagotavlja funkcija *Engineer to relational*. Nato uvedemo edinstven indeks *ch20_bom_ui*, ki je sestavljen iz dveh polj: *aggreg_id* in *subprt_id* (Slika 20.3). Rezultat pretvorbe je relacijski model podatkov (Slika 20.4), pripravljen za izvoz kot skript SQL.



Slika 20.3: Definiranje unikatnega indeksa v Oracle SQL Data Modeler.



Slika 20.4: Relacijski model podatkov.

Oracle SQL Data Modeler ustvari tudi skript SQL za kreiranje tabel, zaporedij in prožilcev. Izberite vse tabele v relacijskem modelu in uporabite funkcijo *File* > *Export* > *DDL File*, da dobite skript. Shranite skript in preverite vrstni red definicij tabel. Najprej je treba definirati tabelo *ch20_psm* in nato *ch20_bom*. Naslednja izvorna koda definira tabelo *ch20_psm* in nastavi omejitve:

```
CREATE TABLE ch20_psm (
        id NUMBER GENERATED BY DEFAULT ON NULL
        AS IDENTITY ( START WITH 1 NOCACHE ) NOT NULL,
        description      VARCHAR2(80 CHAR) NOT NULL,
        unit_of_measure  VARCHAR2(5 CHAR) NOT NULL,
        price            NUMBER NOT NULL
        ) LOGGING;

ALTER TABLE ch20_psm ADD CONSTRAINT ch20_psm_id_pk PRIMARY KEY ( id );

ALTER TABLE ch20_psm ADD CONSTRAINT ch20_psm_unit_of_measure_ck
        CHECK ( unit_of_measure IN ( 'CM', 'M', 'MM', 'PIECE' ) );
```

Naslednja izvorna koda definira tabelo *ch20_bom*, nastavi omejitve in definira edinstven indeks:

```
CREATE TABLE ch20_bom (
        id         NUMBER GENERATED BY DEFAULT ON NULL
        AS IDENTITY ( START WITH 1 NOCACHE ) NOT NULL,
```

```
        subprt_id NUMBER NOT NULL,
        aggreg_id NUMBER NOT NULL,
        quantity  NUMBER NOT NULL
        ) LOGGING;

ALTER TABLE ch20_bom
          ADD CONSTRAINT ch20_bom_id_pk PRIMARY KEY ( id );

ALTER TABLE ch20_bom ADD CONSTRAINT
          ch20_bom_subprt_id_fk FOREIGN KEY ( subprt_id )
          REFERENCES ch20_psm ( id ) ON DELETE CASCADE NOT DEFERRABLE;

ALTER TABLE ch20_bom ADD CONSTRAINT
          ch20_bom_aggreg_id_fk FOREIGN KEY ( aggreg_id )
          REFERENCES ch20_psm ( id ) ON DELETE CASCADE NOT DEFERRABLE;

CREATE UNIQUE INDEX ch20_bom_ui ON
          ch20_bom (aggreg_id, subprt_id );
```

Shranite skripto z ukazi, ki ustvarijo tabeli s primarnimi in tujimi ključi, preverjajo omejitve in definirajo edinstveni indeks. Zdaj je čas za ustvarjanje tabel v bazi podatkov Oracle. Generirano skripto lahko uvozimo in izvedemo v APEX-u.

20.4.5 Objekti v APEX-u

V brskalniku se pomaknite v vaš delovni prostor APEX-a. V tem razdelku bomo:

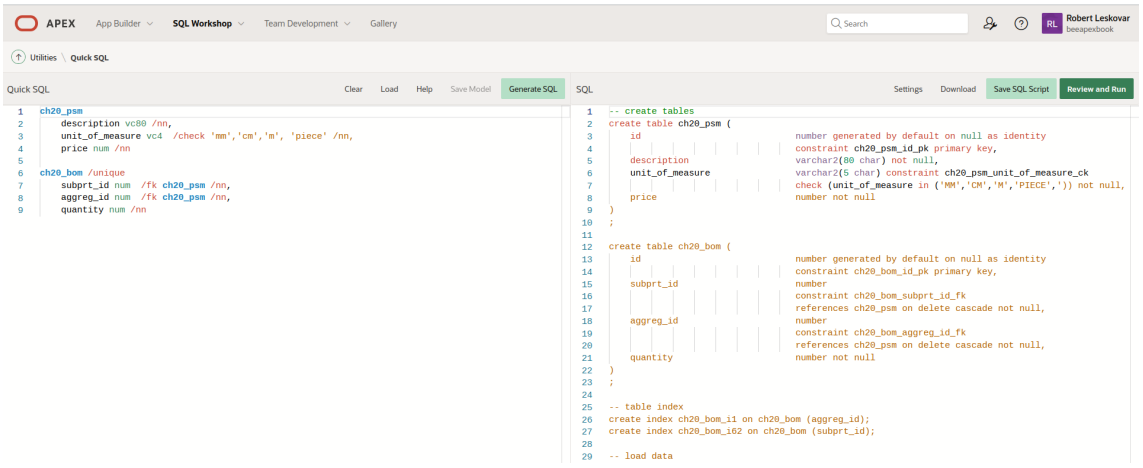
- ustvarili tabele in indekse
- ustvarili funkcijo in proceduro
- ustvarili prožilec

Obstajata dve možnosti za ustvarjanje tabel in indeksa. Če ste za ustvarjanje skripta uporabili Oracle Data Modeler, ga najprej uvozite v delovi prostor APEX-a (SQL Workshop > SQL Scripts > Upload) in ga nato zaženite. Druga možnost je generiranje skripta v SQL z orodjem Quick SQL v primeru, da želite za generiranje takega skripta uporabiti samo APEX. Pomaknite se do SQL Workshop > Utilities > Quick SQL. V polje za besedilo Quick SQL (levi del okna) vnesite naslednje besedilo:

```
ch20_psm
  description vc80 /nn,
  unit_of_measure vc4  /check 'mm','cm','m', 'piece' /nn,
  price num /nn

ch20_bom /unique
  subprt_id num  /fk ch20_psm /nn,
  aggreg_id num  /fk ch20_psm /nn,
  quantity num /nn
```

Kliknite gumb za generiranje SQL. Generirane tabele bodo imele iste lastnosti kot tiste, ki so ustvarjene v Oracle Data Modeler (Slika 20.5).



Slika 20.5: Generiranje skripta SQL s pomočjo orodja Quick SQL.

Poglejte vrstici 26 in 27. Čeprav sta sintaktično pravilni, jih bomo kasneje spremenili. Kliknite "Save SQL Script and Review and Run". Zamenjate vrstici 26 in 27 (kreiranje dveh indeksov) z naslednjim ukazom:

```
create unique index ch20_bom_ui on ch20_bom (aggreg_id, subprt_id);
```

Sedaj izvedite skript. Nastali bosta dve tabeli in unikatni indeks. Izvajanje skripta ne bo uspešno, če ste že izvedli skript, ki ga je generiralo orodje Oracle Data Modeler. Razlog je v tem, da tabeli že obstajata.

Sledi kreiranje funkcije v SQL Workshop > SQL Commands:

```
create or replace FUNCTION  CH20_GET_PRICE (
  xpsm IN   NUMBER
  ) RETURN NUMBER
IS
  y_price number;
BEGIN
  SELECT price INTO y_price FROM ch20_psm  WHERE ID = xpsm;
  return y_price;
EXCEPTION
  WHEN OTHERS THEN
    return(-505);
END CH20_GET_PRICE;
```

Ta funkcija bo vrnila ceno (izdelka, polizdelka, materiala, materiala) za dani ID. Če se z bazo podatkov zgodi nekaj nepričakovanega (izguba povezave, strežnik ne deluje, napaka strojne opreme, ni podatkov za določen ID), bo vrnjena negativna vrednost -505. Izberite celotno besedilo in zaženite. Prejeli boste povratno informacijo, da je funkcija ustvarjena.

Za kreiranje procedure za izračun kosovnice vnesite naslednje besedilo v SQL Workshop > SQL Commands:

```
create or replace PROCEDURE  CH20_CALCULATE_BOM
  (xpsm IN NUMBER,
  ystatus OUT INTEGER,
  yprice OUT NUMBER)
-- creation date: 20.1.2023
-- author: Robert Leskovar
```

```
-- input parameter: xpsm, id of product or semi-product
-- output parameters 1: status of calculation: 0=not calculated, 1=succesfully calculated
-- output parameters 2: price calculated
-- description: calculation of the total costs of given product and all underlaying levels
-- exceptions: system and user defined
-- date of testing: 20.1.2023
-- author of test: swqlab
-- status: APPROVED
is
  v_rezult   NUMBER(38, 2);
  v_psm        NUMBER;
  v_price    NUMBER(38, 2);
  e_negative_price EXCEPTION;
  CURSOR c_ds IS
   WITH STRUK(hier, subprt_id, aggreg_id, quantity) AS
    (SELECT 1 AS hier, subprt_id, aggreg_id, quantity
     FROM CH20_BOM JOIN CH20_PSM ON subprt_id = CH20_PSM.ID WHERE
      aggreg_id = xpsm
   UNION ALL
    SELECT hier+1, C1.subprt_id, C1.aggreg_id, C1.quantity
     FROM CH20_BOM C1 JOIN CH20_PSM P ON C1.subprt_id = P.ID
      JOIN STRUK C2 ON C1.aggreg_id = C2.subprt_id)
   SELECT hier, subprt_id, aggreg_id, quantity
    FROM STRUK ORDER BY hier DESC, aggreg_id, subprt_id;
BEGIN
  ystatus := 0;
  v_rezult := 0.0;
  v_psm := 0;
  SAVEPOINT old_state;
  FOR r_ds IN c_ds LOOP
   v_price:= CH20_GET_PRICE(r_ds.subprt_id);
   IF ( v_price <= 0 ) THEN
    RAISE e_negative_price;
   END IF;
   IF ( r_ds.aggreg_id = v_psm ) THEN
    v_rezult := v_rezult + v_price * r_ds.quantity;
    UPDATE ch20_psm SET price = v_rezult WHERE ID = r_ds.aggreg_id;
   ELSE
    v_psm := r_ds.aggreg_id;
    v_rezult := v_price * r_ds.quantity;
    UPDATE ch20_psm SET price = v_rezult WHERE ID = r_ds.aggreg_id;
   END IF;
  END LOOP;
  COMMIT;
  ystatus := 1;
  yprice := v_rezult;
EXCEPTION
  WHEN e_negative_price THEN
   ROLLBACK TO old_state;
   ystatus := 0;
```

```
   yprice := -505;
  WHEN OTHERS THEN
   ROLLBACK TO old_state;
   ystatus := 0;
   yprice := -100;
end CH20_CALCULATE_BOM;
```

Ta procedura bo vrnila status izračuna in ceno (izdelek, polizdelek, material, poizvedba) za dani ID, razen v primeru, da ima katerakoli sestavina negativno vrednost. Če se z bazo podatkov zgodi kaj nepričakovanega (izguba povezave, strežnik ne deluje, napaka strojne opreme, ni podatkov za podani ID), bo vrnjena negativna vrednost. Če bralec ne pozna PL/SQL, bi bilo dovolj, da prebere komentarje v postopku. Sicer precej zapleten kurzor se ustvari s t.i. CTE (stavek WITH, CTE je kratica za Common Table Expression). Ta kurzor vrne unijo dveh nizov. Zagotavlja popolno strukturo agregata, ki je urejena od spodaj navzgor. Cene vmesnih agregatov so shranjene v ustreznih vrsticah in kumulirane, dokler niso obdelane vse vrstice v kurzorju.

Zadnji predmet, ki se ustvari, je prožilec. Prožilec bo zagotovil, da noben agregat in sestavina v kosovnici ne moreta imeti enake vrednosti (vozilo je sestavljeno iz avtomobila) in da se ustvari relacija prečkanja (če je avtomobil sestavljen iz koles, kolo ne more biti sestavljeno iz avtomobilov). Če želite ustvariti prožilec, ki izvaja navedena pravila, vnesite naslednje besedilo v SQL Workshop > SQL Commands:

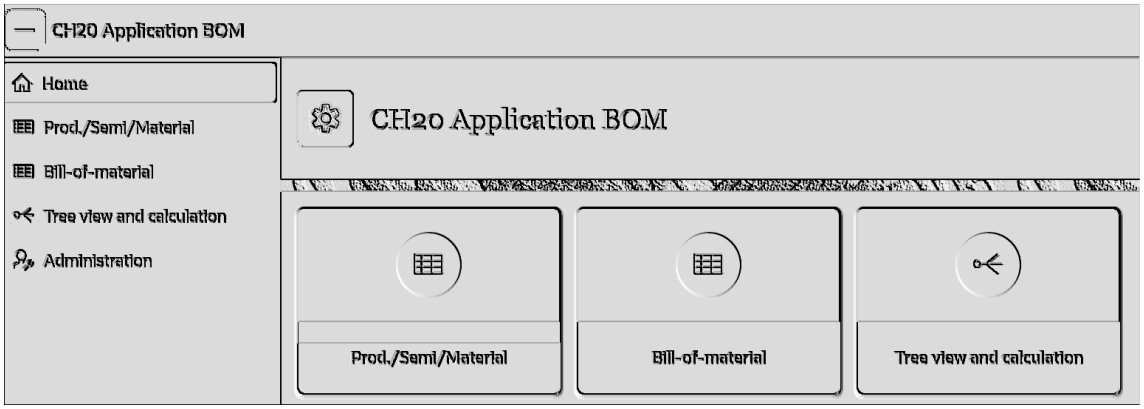
```
create or replace trigger CH20_TRG_BOM_RULES
  before insert or update on ch20_bom
  for each row

  declare
   xsubprt numeric :=:new.subprt_id;
   xaggreg numeric :=:new.aggreg_id;
   xcount numeric;
  begin
   select count(subprt_id) into xcount from ch20_bom
    where subprt_id = xaggreg and aggreg_id = xsubprt;
   if xsubprt = xaggreg or xcount > 0 then
    :new.subprt_id :=null;
   end if;
end;
```

S tem prožilcem bomo preprečili vstavljanje in sicer tako, da vsilimo vrednost NULL v polju tabele, ki mora imeti vrednost. Prednost definicije NOT NULL v stolpcu tabele pred prožilnim mehanizmom bo povzročila, da se podatki ne vstavijo, če je to poslovno pravilo kršeno. Vendar smo zaščitili hierarhijo, da ne postane ciklični graf. Če bi preskočili ustvarjanje tega prožilca, bi morali usposobiti končnega uporabnika, da ne ustvari cikličnega grafa namesto hierarhije. Veliko bolje je preprečiti kot zdraviti, kajne?

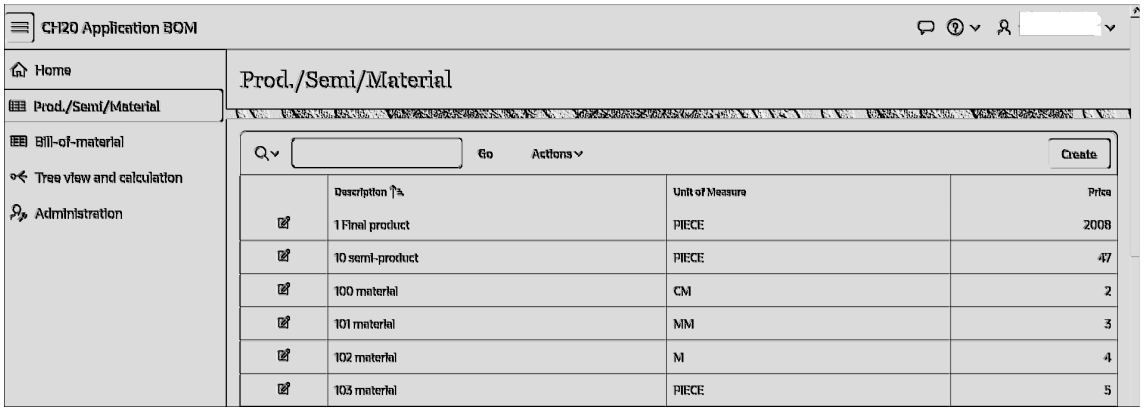
20.5 Uporabniški vmesniki

Na naslednjih Slikah so predstavljeni aplikacijski vmesniki. Najprej predstavimo skico vstopnega okna. Uporabnik se bo pomaknil do upravljanja osnovnih podatkov, upravljanja strukturnih podatkov in izračuna bodisi z levim stranskim menijem ali z navigacijsko stranjo (Slika 20.6).

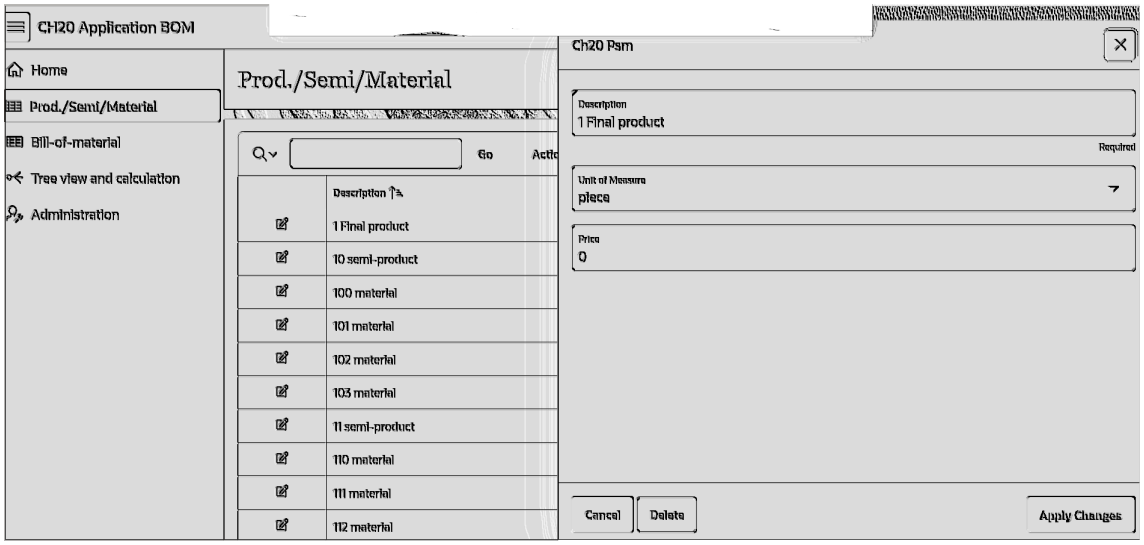


Slika 20.6: Domača stran aplikacije.

Naslednji dve Sliki (20.7 in 20.8) prikazujeta skico poročila in obrazca za izdelek, polizdelek, material in povpraševanje. Omogočeno je dodajanje, spreminjanje in brisanje.



Slika 20.7: Upravljanje osnovnih podatkov - poročilo.



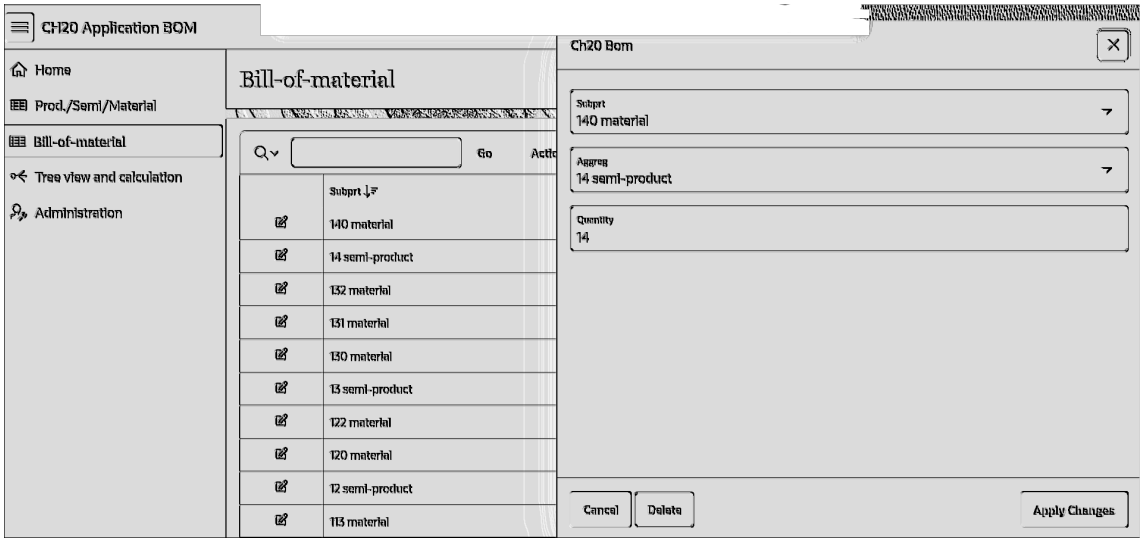
Slika 20.8: Upravljanje osnovnih podatkov - obrazec.

Naslednji dve Sliki (20.9 in 20.10) prikazujeta skico poročila in obrazca za urejanje strukture

kosovnice. Omogočeno je dodajanje, spreminjanje in brisanje.

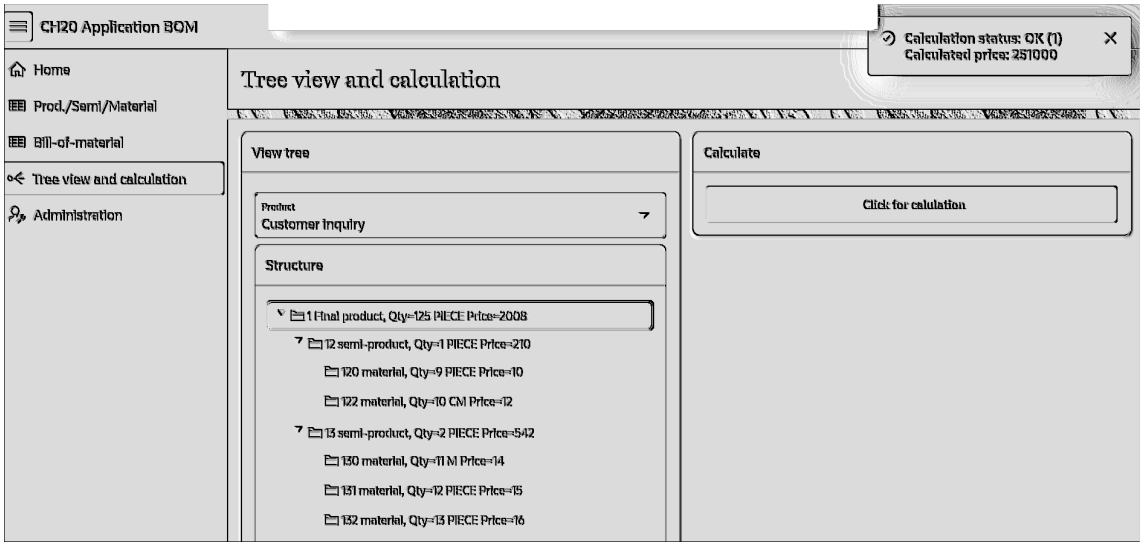


Slika 20.9: Upravljanje strukturnih podatkov - poročilo.



Slika 20.10: Upravljanje strukturnih podatkov - obrazec.

Skica na Sliki 20.11 prikazuje vmesnik za izvedbo kalkulacije za katerikoli agregat (izdelek, polizdelek ali povpraševanje). Uporabnik najprej izbere agregat in kliokne na gumb Calculate. Status izračuna in izračunana cena enote agregata se pojavi kot prikazno okno skupaj z drevesno strukturo. Uporabnik lahko razširi ali skrči hierarhijo.



Slika 20.11: Stran za izračun materialen kosovnice.

20.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija
- video vodiči

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter20 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

20.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

20.6.2 Video vodiči

Video vodiči natančno prikazujejo razvoj aplikacije.

20.7 Vprašanja

1. Kako bi spremenili logični podatkovni model za implementacijo nove entitete za merske enote, ker želimo vključiti več deset enot?
2. Kako bi spremenili skript v Quick SQL, da bi odražal zgornjo spremembo?
3. Kakšne bodo posledice, če brišemo edinstveni indeks ch20_bom_ui?

20.8 Odgovori

1. Uporabimo novo entiteto *ch20_uom* in nastavimo vsaj dva atributa (ID in opis merske enote). Potem nastavimo razmerje 1 proti mnogo med *ch20_psm* in *ch20_uom*.
2. V Quick SQL definiramo tabelo *ch20_uom*:

```
ch20_uom
  description vc80
```

in spremenimo definicijo *ch20_psm*:

```
ch20_psm
  description vc80 /nn,
  unit_of_measure num /fk ch20_psm
  price num
```

3. Brisanje edinstvenega indeksa ch20_bom_ui bi omogočila shranjevanje več parov (sestavina, agregat) z enakimi vrednostmi. Tabela bi porabila več prostora na disku. To ne bi vplivalo na funkcionalnost izračuna, vendar bi obdelava trajala dlje in program bi porabil več RAM-a.



21. Vodenje prehrane in diete

A. ANGEIOPLASTIS, G. MYLLIS, A. TSIMPIRIS, D. VARSAMIS; PREVOD R. LESKOVAR

21.1 Poslovni vidik primera

Naš primer je najbolj priljubljena grška medijska blagovna znamka za hrano z dosegom več kot 500.000 uporabnikov. Podjetje je ponosni lastnik vodilne grške revije o hrani na trgu in digitalne izdaje, spletnega mesta številka ena v državi o hrani, vse večjega števila dogodkov v živo, vključno z največjim dogodkom o hrani v Grčiji – Greek Food Show – in izjemno uspešne serije kuharskih knjig.

Podjetje se je odločilo ustvariti in promovirati spletno orodje, ki uporabnikom omogoča dostop do več tisoč receptov prek njihove spletne strani. Poleg tega lahko uporabniki prilagodijo svoje najljubše recepte z dodajanjem ali odstranjevanjem sestavin, storitev pa bo zagotovila podatke o hranilni vrednosti novih receptov, kar uporabnikom omogoča ustvarjanje lastne prilagojene kuharske knjige najljubših receptov.

21.2 Definicija problema

Uporabniki, ki uživajo v kuhanju, se pogosto soočajo z omejitvami, ko gre za to, kaj lahko jedo, bodisi zaradi zdravstvenih težav, kot je sladkorna bolezen ali visok holesterol, ali osebnih prehranskih preferenc, kot je veganska ali brezglutenska. Ena od težav z obstoječimi spletnimi viri receptov je, da pogosto ne nudijo celovitih alternativ za prilagoditev tem omejitvam. Posledično lahko postanejo uporabniki nezadovoljni z omejenimi informacijami in popolnoma opustijo iskanje receptov. Ta pa povzroči slabo sodelovanje s podjetjem.

21.3 Primeri uporabe

21.3.1 Opis primera uporabe

Izdelali bomo aplikacijo z Oracle APEX, ki bo uporabnikom omogočala ogled in dodajanje različnih zdravih receptov. Aplikacija bo zagotovila podroben opis vsakega recepta, vključno s seznamom sestavin in podatki o hranilni vrednosti vsake sestavine. Uporabniki lahko z obrazcem dodajo tudi lastne recepte in ustrezne sestavine. Hranilne vrednosti vsakega recepta, kot so število kalorij, skupne beljakovine, število porcij in čas priprave, bodo predstavljene v diagramih za lažjo berljivost in ne v preprosti tabeli.

Tabela 21.1: Opis primera uporabe: vodenje receptov in sestavin ter izračun hranilnih vrednosti

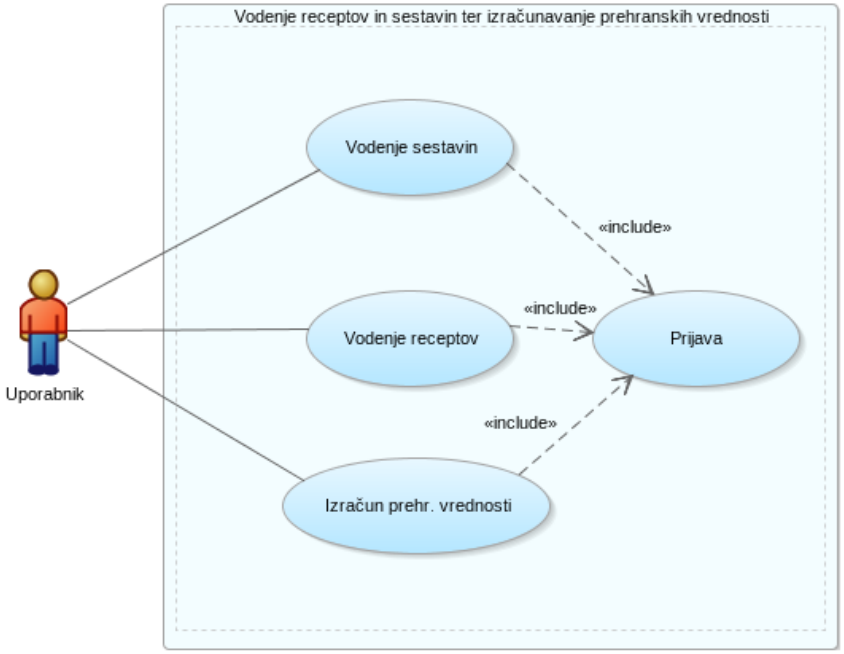
Keyword	**Value**
ID:	*Ch21-01*
Title:	*Vodenje receptov in sestavin ter izračun hranilnih vrednosti*
Description:	*Uporabnik z aplikacijo lahko brska po obstoječih receptih in sestavinah, svoje ureja in izračunava prehranske vrednosti.*
Glavni igralec:	*Uporabnik*
Predhodni pogoji:	Uporabnik ima poverilnice za aplikacijo, spletna stran aplikacije je dostopna.
Pogoji po zaključku:	Vse spremembe receptov in sestavin (dodajanje, spreminjanje, brisanje), ki jih izvede uporabnik so shranjene v bazi podatkov.
Glavni del	Scenariji
Uspešen scenarij:	1. *Uporabnik se prijavi v aplikacijo s svojimimi poverilnicami.* 2. *Iz menuja ali navigacijske strani izbere recepte ali sestavine.* 3. *Za dodajanje receptov klikne gumb **Add a recipe**. Za dodajanje sestavine klikne gumb **Add Ingredient**.* 4. *Za urejanje recepta ali sestavine klikne ikono peresa v prvem stolpcu.* 5. *Po izvedeni spremembi klikne gumb **Apply Changes** za shranjevanje. Za brisanje klikne gumb **Delete**.* 6. *Za izračun prehranske vrednosti v menujju ali na navigacijski strani izbere **Recipe total nutrition facts**.* 7. *Prikaže se poročilo, ki vsebuje velikost porcije glede na podane količine sestavin, težo proteinov, število kalorij in skupni čas za pripravo recepta.*
Razširitve:	-
Pogostost uporabe:	*Uporabniki dnevno objavijo okoli 5 receptov, letno okoli 1000. Število vpogledov je okoli 10 tisoč dnevno.*
Status:	*Zaključen*
Lastnik:	*Uporabnik*
Prioriteta:	*Zmerna*

21.3.2 Delno strukturiran opis

Delno strukturiran opis primera uporabe je predstavljen v tabeli 21.1.

21.3.3 Diagram primera uporabe

Na Sliki 21.1 je prikazan diagram primera uporabe.



Slika 21.1: Diagram primera uporabe.

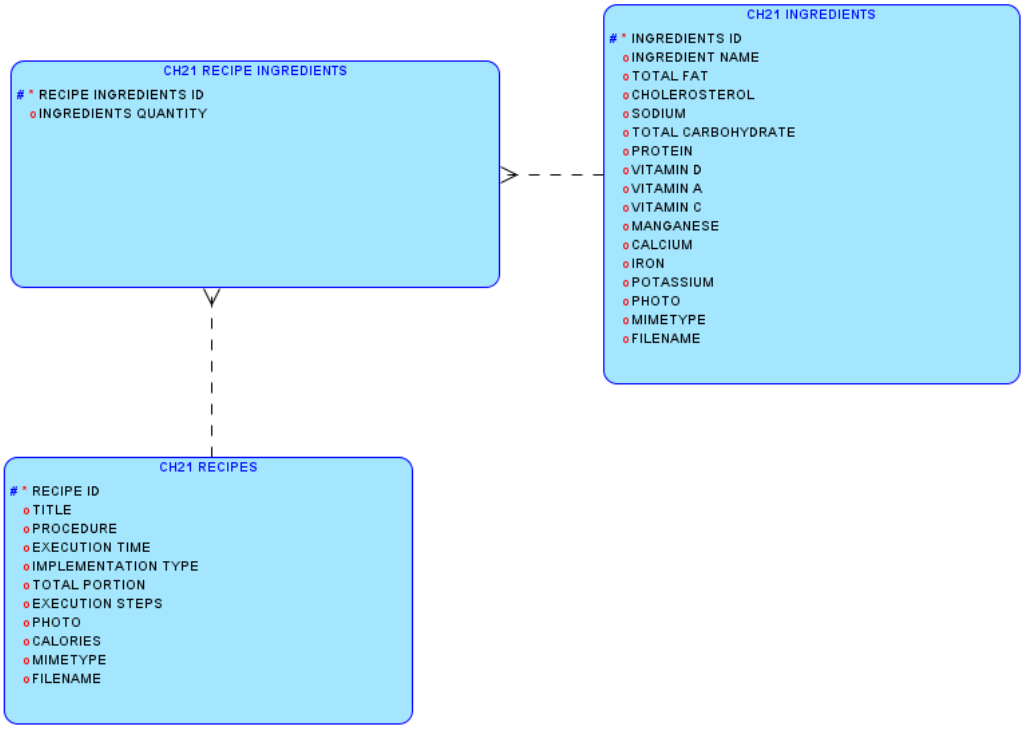
21.4 Model podatkov

21.4.1 Opis modela podatkov

Model podatkov vsebuje tri entitete: recept, sestavina in sestavina recepta.

21.4.2 Logični model podatkov

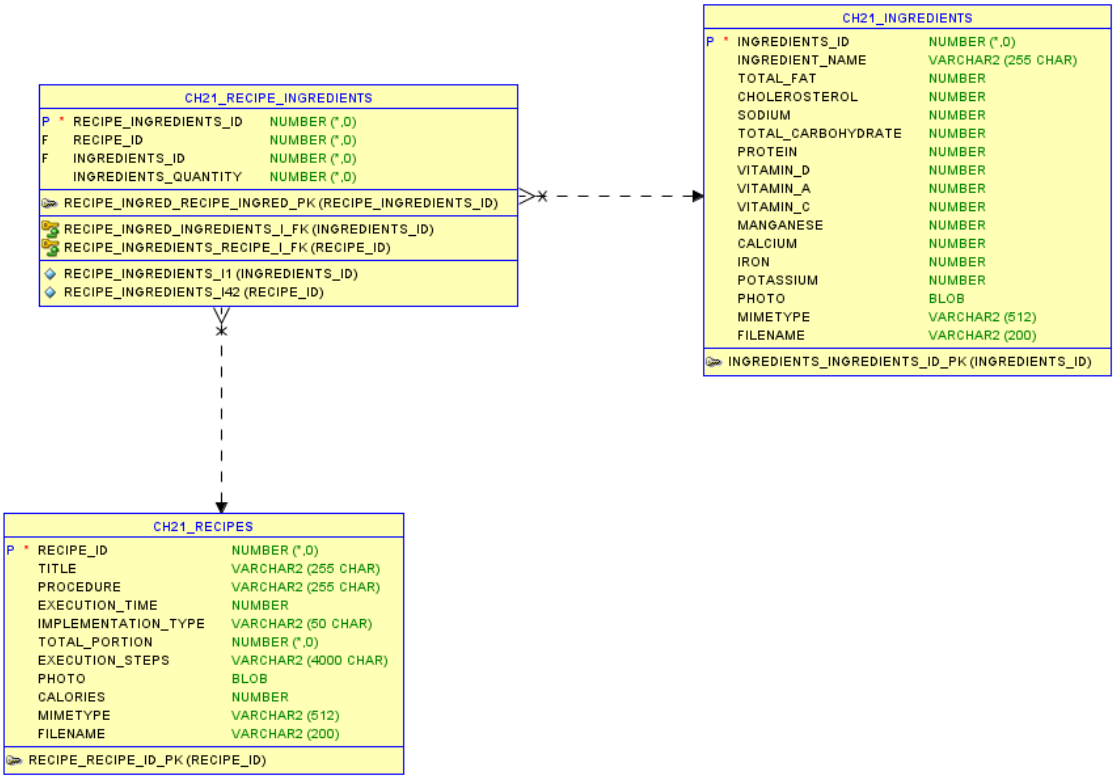
Logični model podatkov je prikazan na Sliki 21.2.



Slika 21.2: Logični model podatkov.

21.4.3 Relacijski model podatkov

Relacijski model podatkov je prikazan na Sliki 21.3.



Slika 21.3: Relacijski model podatkov.

21.4.4 Skript Quick SQL za generiranje skripta v SQL

V nadaljevanju prikazujemo skript v Quick SQL. S pomočjo skripta Quick SQL APEX generira skript v SQL. Tako bodo nastale tabele CH21_RECIPES, CH21_RECIPE_INGREDIENTS in CH21_INGREDIENTS.

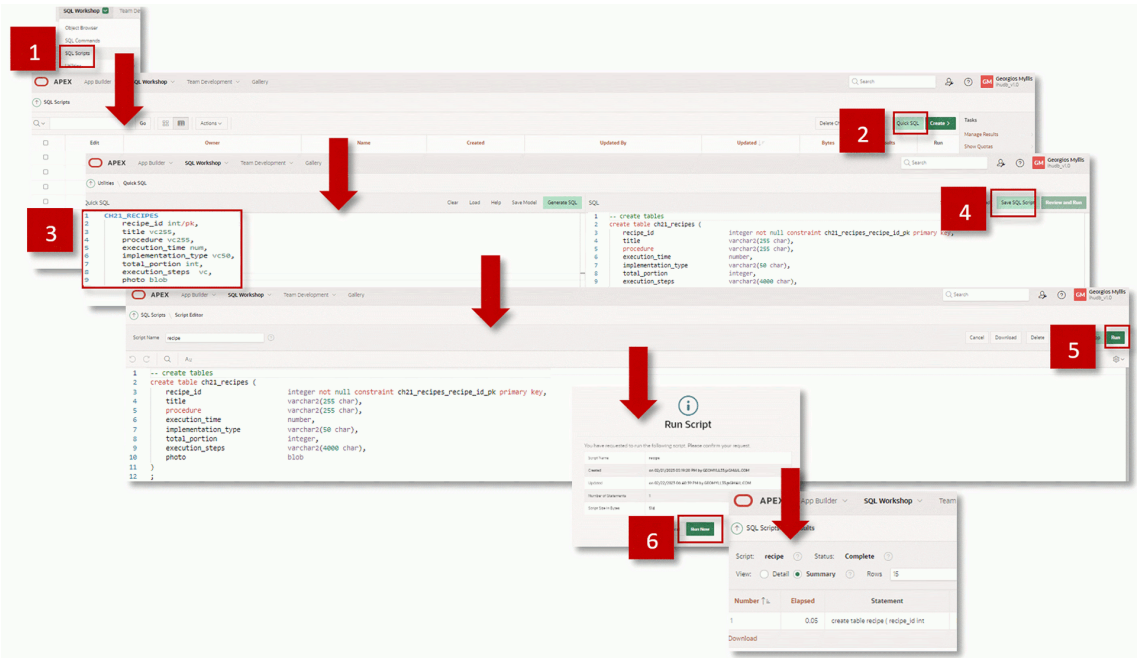
```
CH21_RECIPES
        RECIPE_ID int/pk,
        TITLE VC255,
        PROCEDURE VC255,
        EXECUTION_TIME NUM,
        IMPLEMENTATION_TYPE VC50,
        TOTAL_PORTION int,
        EXECUTION_STEPS  VC,
        PHOTO BLOB

CH21_INGREDIENTS
        INGREDIENTS_ID int/pk,
        INGREDIENT_NAME VC255,
        TOTAL_FAT NUM,
        CHOLEROSTEROL NUM,
        SODIUM NUM,
        TOTAL_CARBOHYDRATE NUM,
        PROTEIN NUM,
        VITAMIN_D NUM,
        VITAMIN_A NUM,
        VITAMIN_C NUM,
```

```
        MANGANESE NUM,
        CALCIUM NUM,
        IRON NUM,
        POTASSIUM NUM,
        PHOTO BLOB,
        MIMETYPE VC512,
        FILENAME VC200

CH21_RECIPE_INGREDIENTS
        RECIPE_INGREDIENTS_ID INT/PK,
        RECIPE_ID INT/FK RECIPE,
        INGREDIENTS_ID INT/FK INGREDIENTS,
        INGREDIENTS_QUANTITY INT
```

Za zagon kode Quick SQL moramo najprej klikniti **SQL Workshop** in izbrati **SQL Scripts**. Od tam lahko izberemo **Quick SQL** za ustvarjanje tabel. Postopek je preprost. Določiti moramo samo ime tabele in njena polja ter vse podrejene tabele, povezane z našo tabelo. Slika 21.4 prikazuje, kako hitro ustvariti tabele s to funkcijo.



Slika 21.4: Quick SQL in kreiranje skriptav v SQL.

Da bo aplikacija pravilno delovala bomo definirali še tri sekvence in prožilce, ki bodo samodejno povečevali vrednosti primarnih ključev ob dodajanju zapisov.

Sekvenca in prožilec za ID recepta:

```
CREATE SEQUENCE seq_RECIPE
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE OR REPLACE TRIGGER RECIPE_trg
```

```
BEFORE INSERT
ON RECIPE
FOR EACH ROW
BEGIN
:new.RECIPE_ID:=seq_RECIPE.nextval;
END;
```

Sekvenca in prožilec za ID sestavine:

```
CREATE SEQUENCE seq_INGREDIENTS
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;

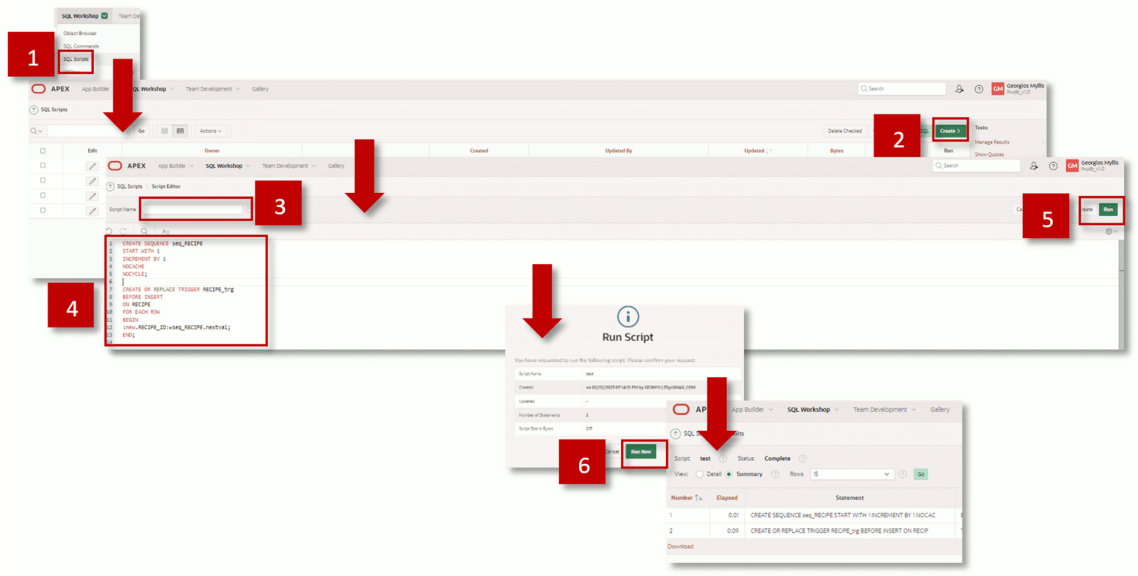
CREATE OR REPLACE TRIGGER INGREDIENTS_trg
BEFORE INSERT
ON INGREDIENTS
FOR EACH ROW
BEGIN
:new.INGREDIENTS_ID:=seq_INGREDIENTS.nextval;
END;
```

Sekvenca in prožilec za ID sestavine v receptu:

```
CREATE SEQUENCE seq_RECIPE_INGREDIENTS
START WITH 31
INCREMENT BY 1
NOCACHE
NOCYCLE;

CREATE OR REPLACE TRIGGER RECIPE_INGREDIENTS_trg
BEFORE INSERT
ON RECIPE_INGREDIENTS
FOR EACH ROW
BEGIN
:new.RECIPE_INGREDIENTS_ID:=seq_RECIPE_INGREDIENTS.nextval;
END;
```

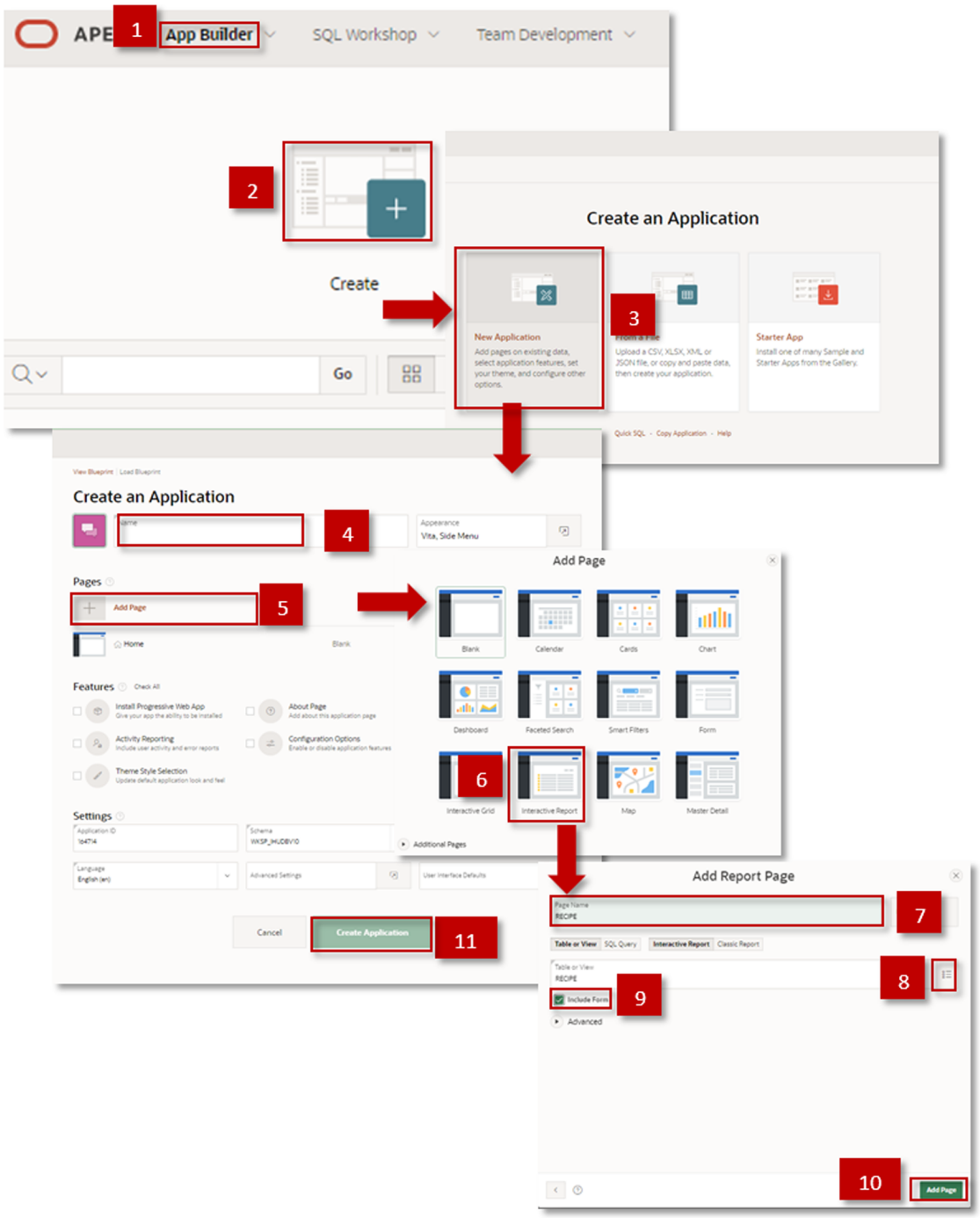
Pomaknite se v **SQL Workshop** in **SQL Scripts**. Nato lahko kliknite na gumb **Create** ter prilepite zgornjo kodo. Ko skriptu damo ime kliknemo gumb **Run**, da ga izvedemo. V bazi podatkov sekcenca in prožilec zagotovita, da bo vsak zapis v obravavanih tabela imel unikatno identifikacijsko oznako. Uporabniku torej ni potrebno vpisovati teh vrednosti, temveč se generirajo v bazi. Na Sliki 21.5 je prikazan primer izdelave in izvedbe skripta.



Slika 21.5: Primer izdelave in izvedbe skripta.

21.5 Uporabniški vmesniki

Generiranje nove aplikacije s časovnikom v APEX-u je prikazano na Sliki 21.6. Kliknite **App Builder** in **Create**. V obrazec vpišite naziv aplikacije in izberite funkcije, ki jih želite vključiti. Poročila in obrazce, ki se nanašajo na recepte, sestavine in izračun hranilnih vrednosti bomo prikazali v nadaljevanju.



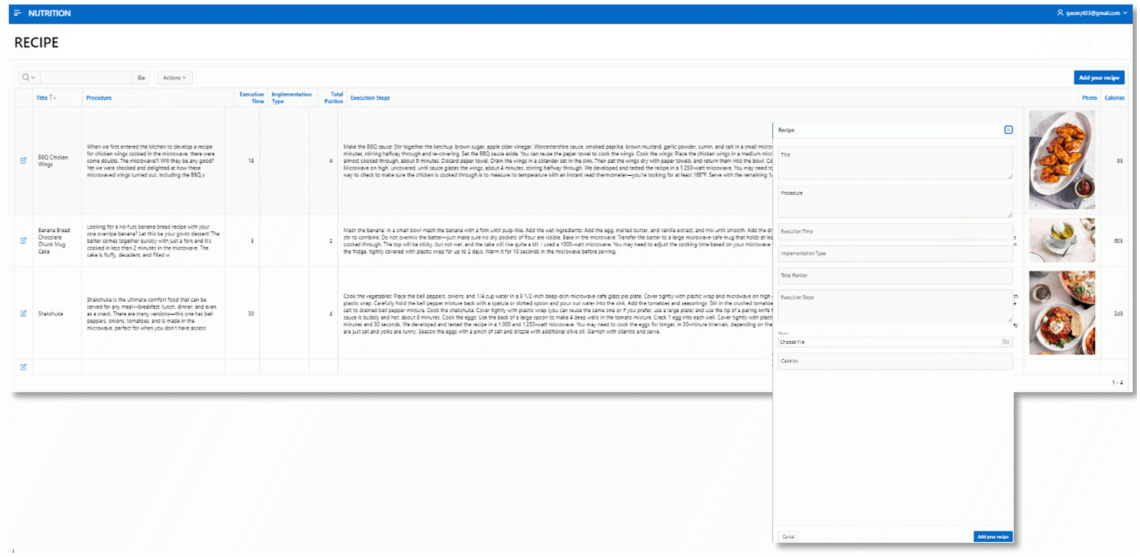
Slika 21.6: Čarovnik za kreiranje nove aplikacije.

Na Sliki 21.7 je prikazan rezultat generiranja ogrodja aplikacije. Aplikacija je že izvedljiva. Prijavite s poverilnicami, ki ste jih uporabili za generiranje.



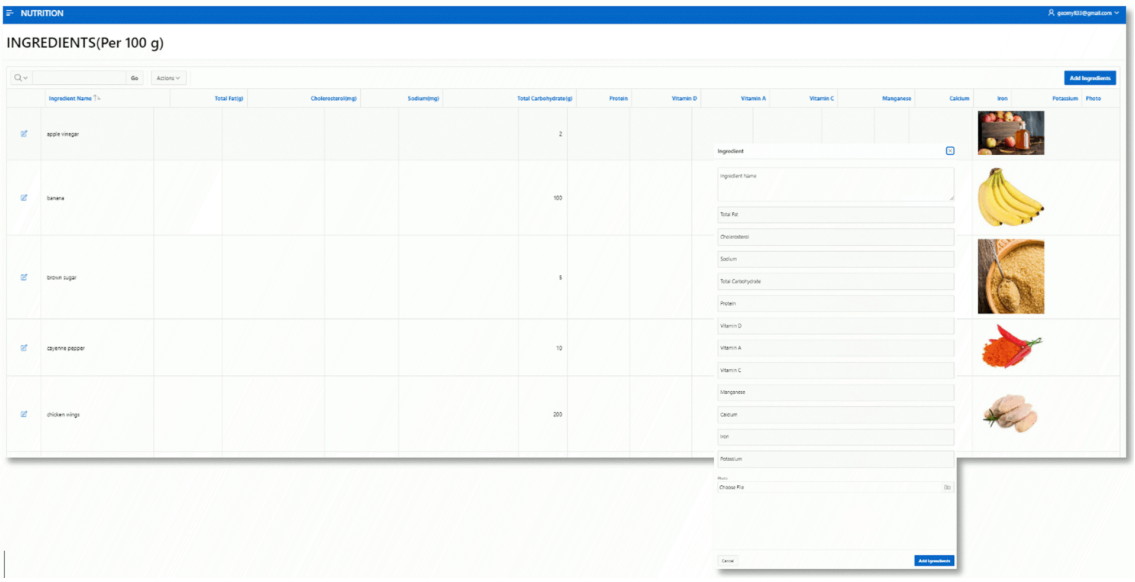
Slika 21.7: Aplikacija z nazivom “Nutrition application”.

Na Sliki 21.8 je prikazana stran za recepte, kjer so podane informacije o receptu, postopku, času priprave, številu porcij in fotografija jedi. Če ima uporabnik ustrezne pravice, lahko doda nov recept ali popravi obstoječega.



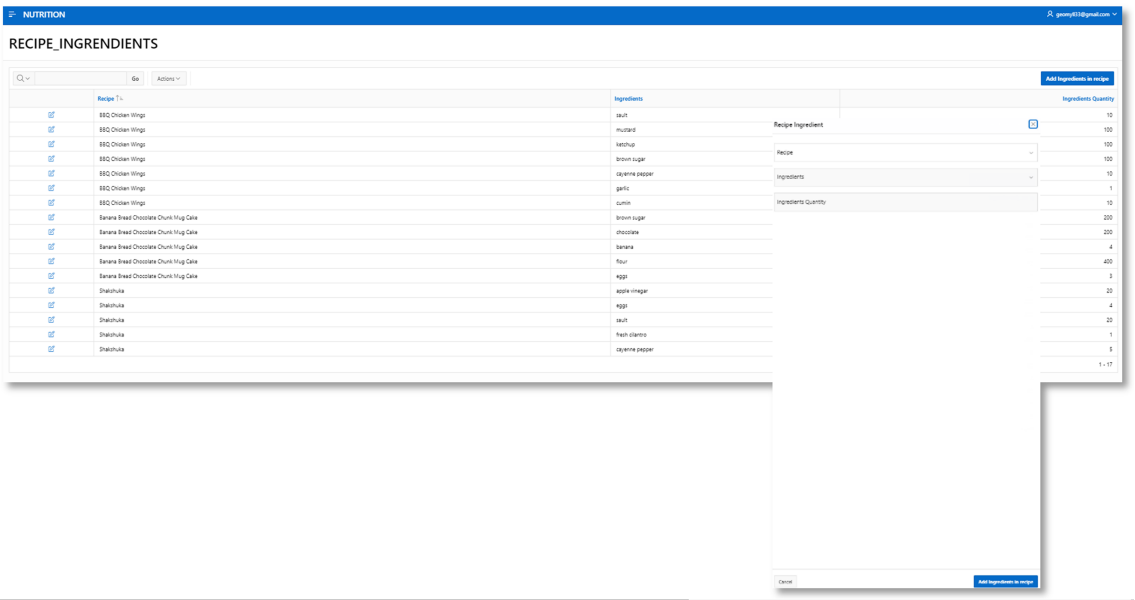
Slika 21.8: Stran z recepti.

Slika 21.9 prikazuje prehranske vrednosti sestavin kot so maščobe, holesterol, natrij, skupni ogljikovi hidrati, proteini, vitamini in podobno. S pomočjo teh vrednosti je možno izračunati prehranske vrednosti za pripravljeno jed, ki je ponavadi sestavljena iz več sestavin. Če ima uporabnik ustrezne pravice, lahko doda novo sestavino ali popravi obstoječo.



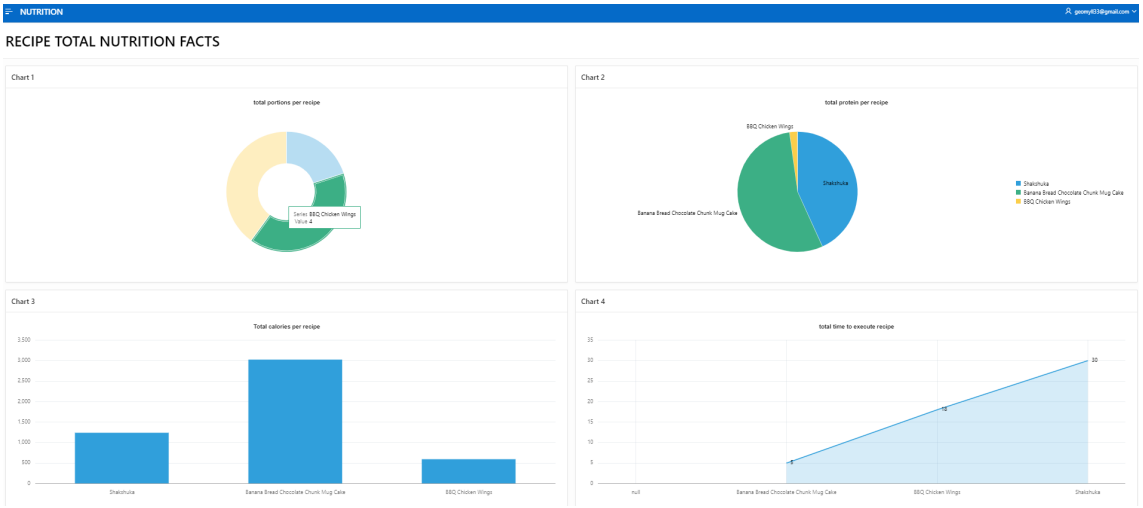
Slika 21.9: Stran s sestavinami.

Slika 21.10 prikazuje sestavljanje recepta. Uporabniki izberejo sestavine ter določijo recept. Če želijo preverijo tudi prehranske vrednosti jedi v receptu. Če ima uporabnik ustrezne pravice, lahko doda v recept novo sestavino ali popravi obstoječo.



Slika 21.10: Stran s sestavinami recepta.

Poročilo na sliki 21.11 prikazuje prehranske vrednosti izbranega recepta. Poleg količine proteinov, kalorične vrednosti in števila porcij je prikazan tudi čas priprave.



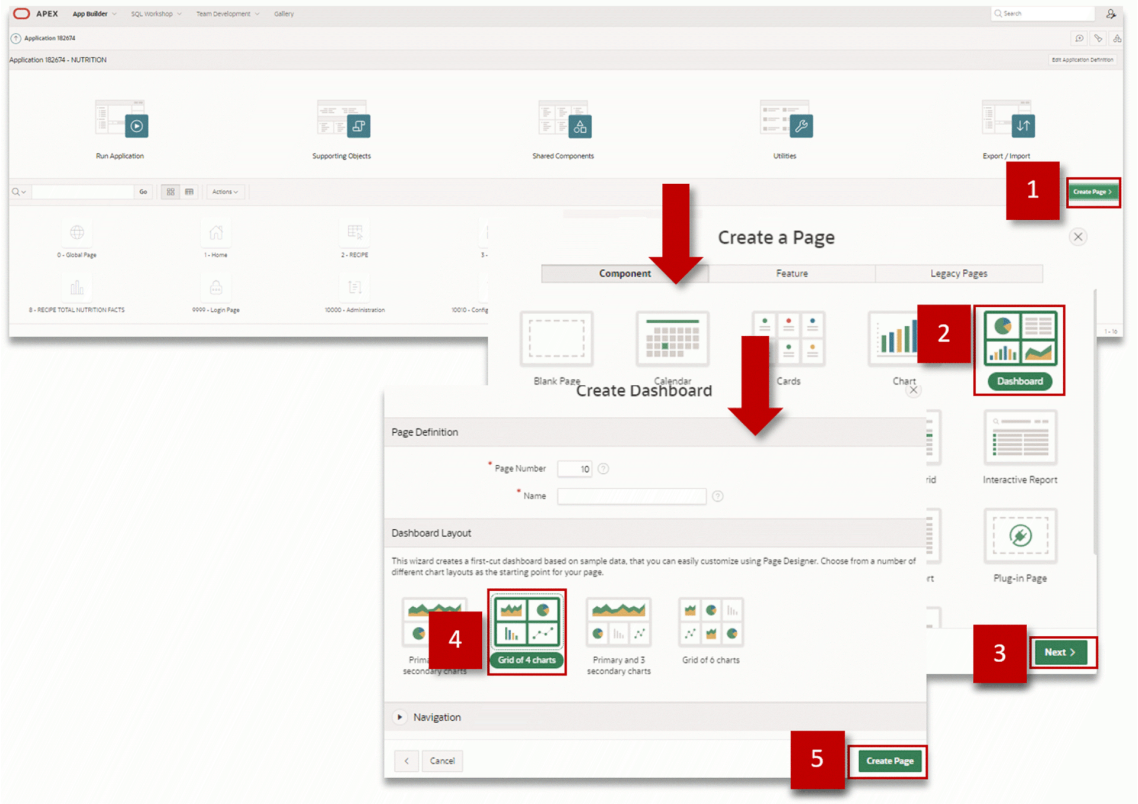
Slika 21.11: Stran z izračunom prehranskih vrednosti v receptu.

21.5.1 Izdelava nadzorne plošče

Koraki za idelavo nadzorne plošče so prikazani na Sliki 21.12.

- Vstopite v razvojno okolje aplikacije.
- Kliknite gumb **Create Page**.
- Izberite **Dashboard** iz seznama predlog strani.
- Kliknite gumb **Next**.
- Izberite želeni izgled nadzorne plošče.
- Kliknite gumb **Create Page**.

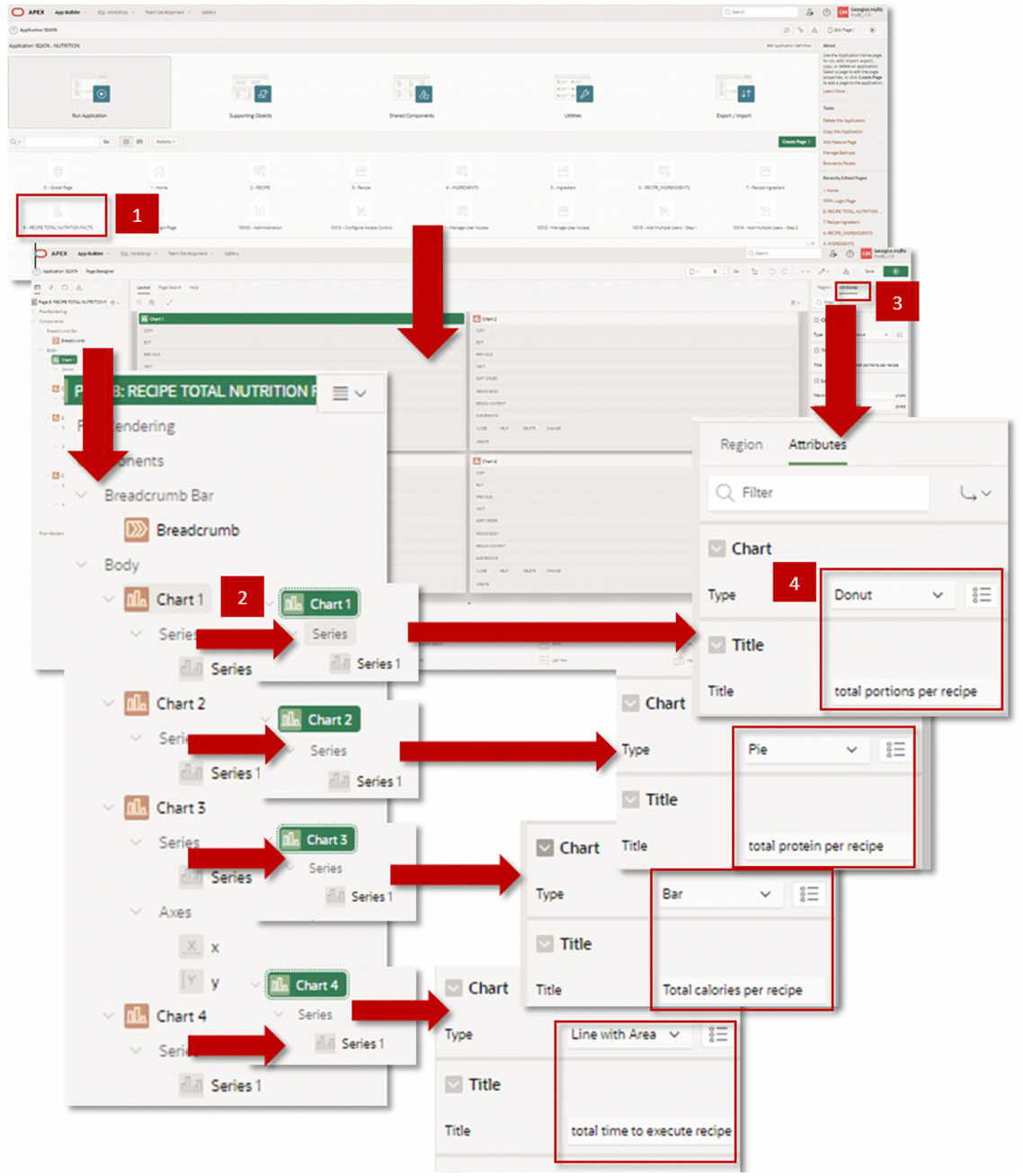
Ko je stran - nadzorna plošča ustvarjena, jo bomo prilagodili. Določiti moramo vsebino in izgled te vsebine.



Slika 21.12: Ustvarjanje nadzorne plošče.

Uredimo sedaj nadzorno ploščo v urejevalniku strani. Čarovnik je ustvaril štiri regije z grafikoni (Slika 21.12), pri čemer seveda ni mogel vedeti, kaj mi želimo predstaviti.

Grafikon 1 preimenujte v “Total Portion per Recipe”. Izberite **Attribute** ter določite tip “Donut Chart”. Vsebina grafikona 1 bo torej število porcij. Grafikon 2 preimenujte v “Total Protein per Recipe” (količina proteinov). Določite mu tip “Pie Chart”. Grafikon 3 preimenujte v “Total Calories per Recipe” (skupno kalorij v receptu). Določite mu tip “Bar Chart type”. Četrti grafikon preimenujte v “Total Time to Execute Recipe” ter mu določite tip “Line with Area” (glejte Sliko 21.13).



Slika 21.13: Prilagoditev vsebine in izgleda nadzorne plošče.

Za prvi grafikon določite **Series 1** poizvedbo za polje Source:

```
SELECT DISTINCT RECIPE.TITLE,
RECIPE.TOTAL PORTION
FROM RECIPE
```

Za drugi grafikon določite **Series 2** poizvedbo za polje Source:

```
SELECT DISTINCT RECIPE.TITLE, SUM(protein) AS protein
FROM RECIPE
INNER JOIN RECIPE_INGREDIENTS ON
RECIPE.RECIPE_ID=RECIPE_INGREDIENTS.RECIPE_ID
```

```
INNER JOIN INGREDIENTS ON
INGREDIENTS.INGREDIENTS_ID=RECIPE_INGREDIENTS.INGREDIENTS_ID
GROUP BY RECIPE.TITLE
```

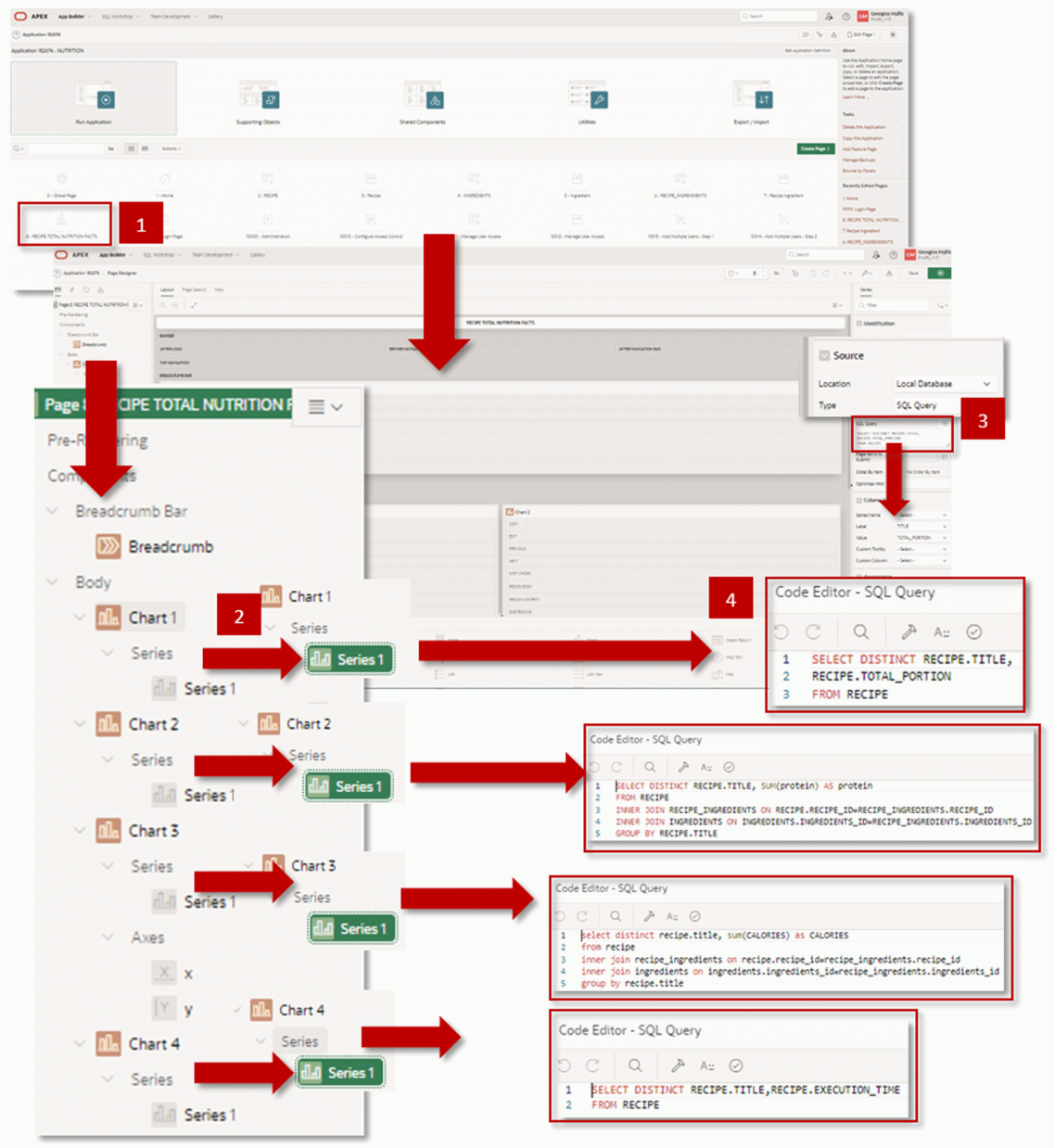
Za tretji grafikon določite **Series 3** poizvedbo za polje Source:

```
select distinct recipe.title, sum(CALORIES) as CALORIES
from recipe
inner join recipe_ingredients on
recipe.recipe_id=recipe_ingredients.recipe_id
inner join ingredients on
ingredients.ingredients_id=recipe_ingredients.ingredients_id
group by recipe.title
```

Za četrti grafikon določite **Series 4** poizvedbo za polje Source:

```
SELECT DISTINCT RECIPE.TITLE,RECIPE.EXECUTION_TIME
FROM RECIPE
```

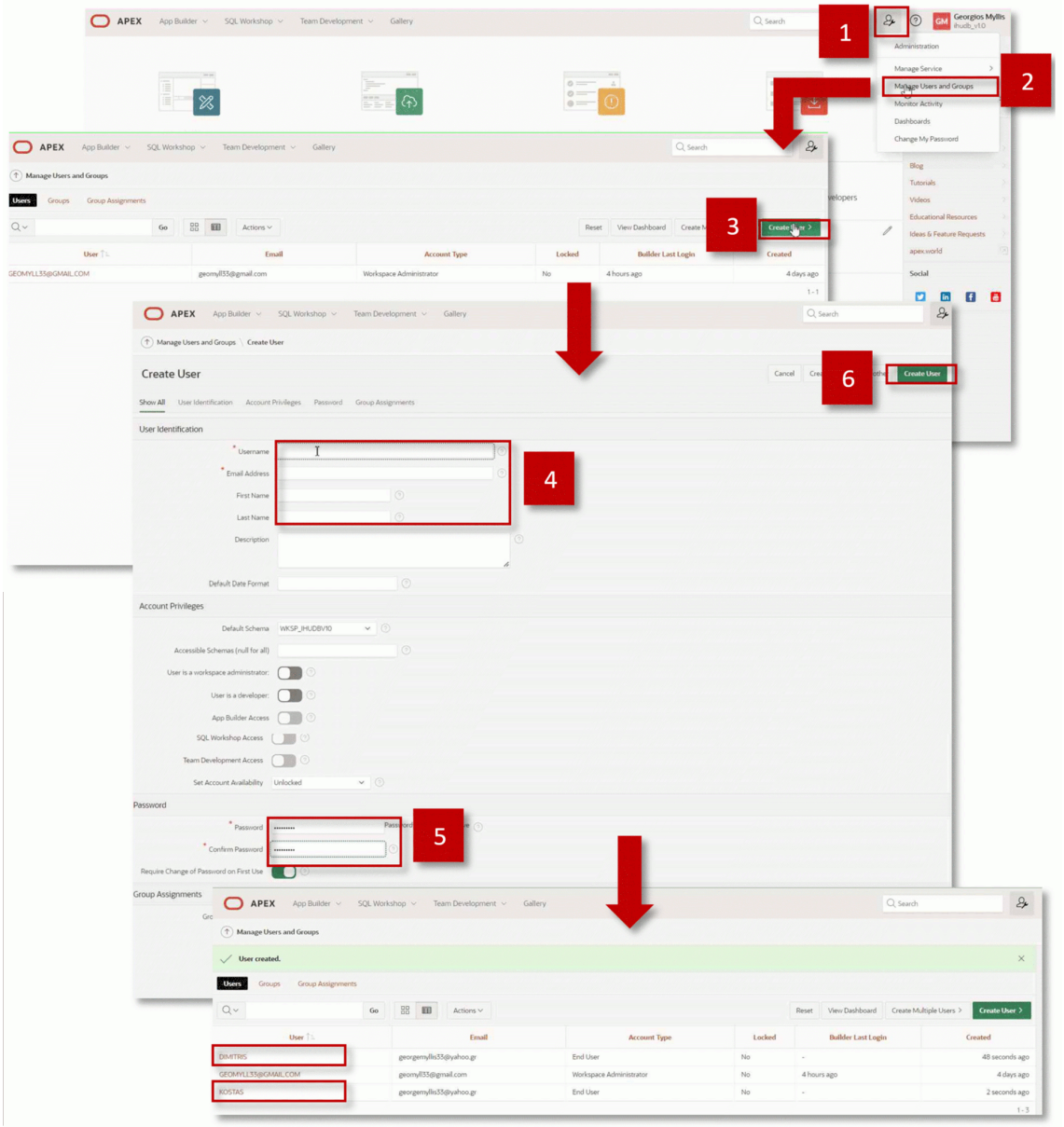
Na Sliki 21.14 so prikazane se smernice za prilagoditve.



Slika 21.14: Določitev poizvedbe v SQL za polje Source.

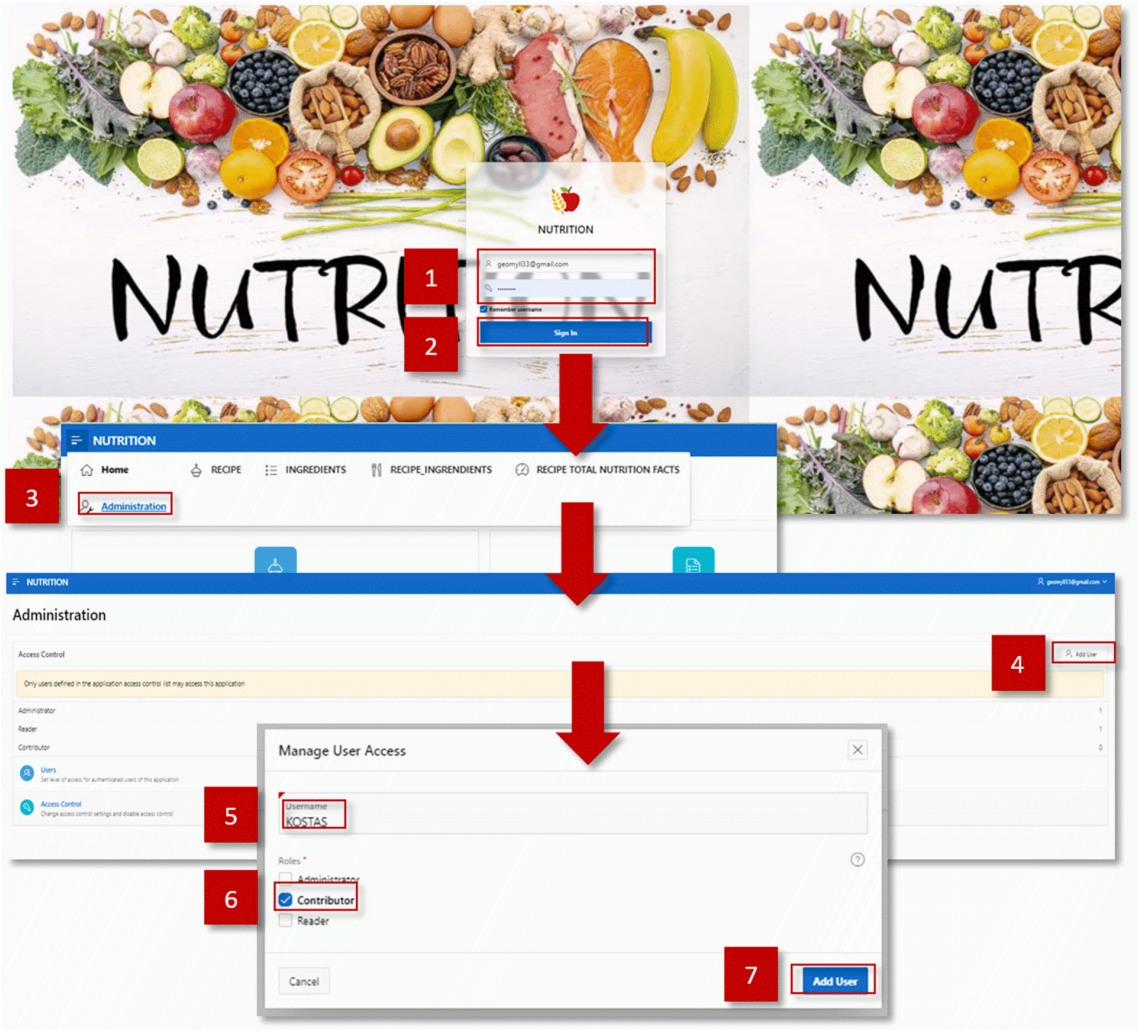
21.5.2 Razvijalci in uporabniki aplikacije

V APEXU morajo imeti uporabniški račun tako razvijalci kot končni uporabniki. Moramo pa razlikovati med privilegiji, ki so povezani z delovnim prostorom, aplikacijo in posamično stranjo aplikacije. Pravice v delovnem prostoru so nanašajo na administracijo le tega, pravico razvoja, uporabo čarovnika za generiranje aplikacije ter orodij SQL Workshop in Team development. Uporabniki aplikacije imaje ugasnjene vse navedene pravice. Nov uporabniški račun ustvarite v **Manage Users and Groups**. Izberite **Create User**, vnesite podatke o uporabniku, njegovo začetno geslo ter mu določite vlogo pri razvoju. Kliknite **Create User** (glejte sliko 21.15).



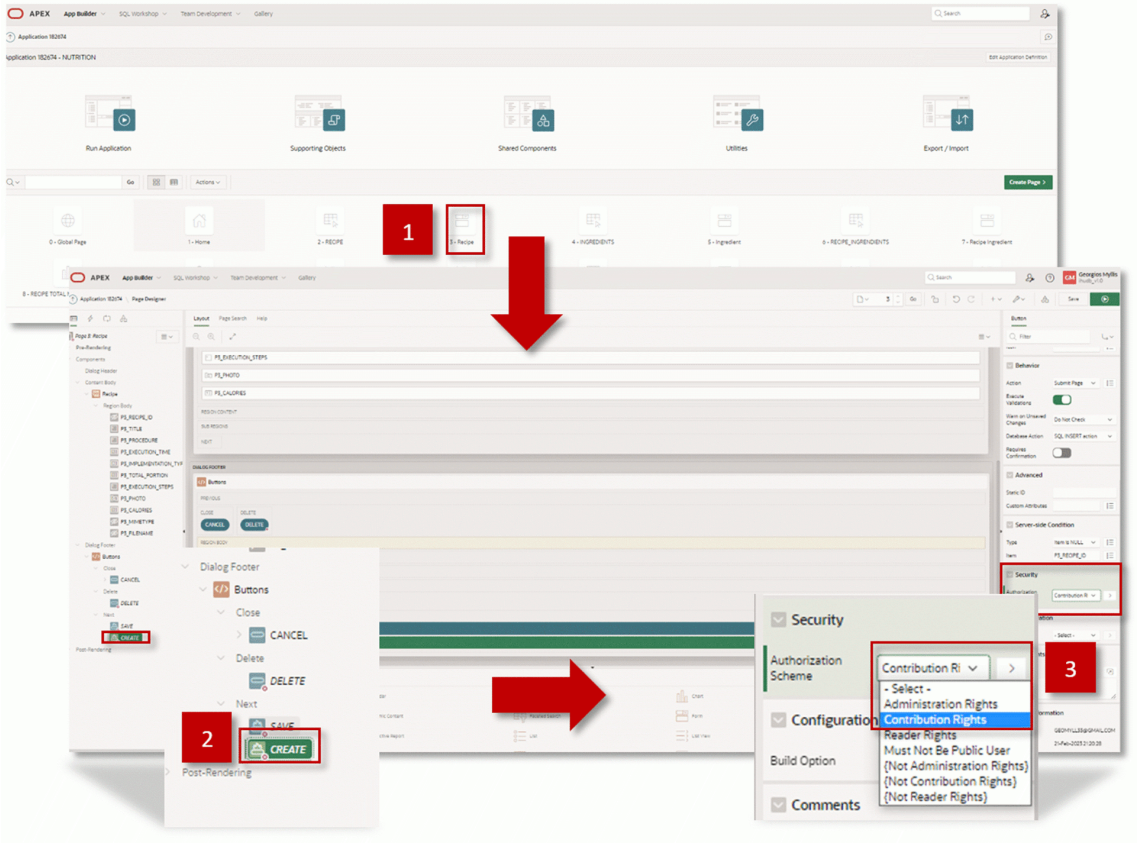
Slika 21.15: Ustvarjanje novega uporabniškega računa.

Uporabnike aplikacij in njihove pravice lahko definiramo tudi na strani “Administration” v sami aplikaciji. Pogoj za to je, da smo pri ustvarjanju aplikacije s čarovnikom označili funkcijo (Feature) pri “Access Control”. V tej konkretni aplikaciji je stran za administracijo dostopna v mega menuju. Kliknite **Administration**. Poiščite **Add User**, vnesite uporabniško ime in vlogo. Potrdite s klikom **Add User**. Slika 21.16 prikazuje primer kreiranja uporabnika in njegovih vlog v aplikaciji sami.



Slika 21.16: Kreiranje uporabnškega računa in določitev vlog v sami aplikaciji.

Dostop do posamezne strani lahko dovolite uporabniku ali vlogi. V urejevalniku strani (glejte Sliko 21.17 za stran z recepti) poiščite varnostne opcije (security options). Izberite **Contribution Rights** kot avtorizacijso shemo. Kliknite **Save**. Samo uporabniki z vlogo “Contributor” bodo lahko dodajali recepte.



Slika 21.17: Dodelitev dostopa do strani samo določeni skupini (vlogi) uporabnikov.

21.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija
- video vodič

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter21 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

21.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele in drugi objekti ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

21.6.2 Video vodiči

Video vodič natančno prikazujejo razvoj aplikacije.

21.7 Vprašanja

1. Kako spremenimo ozadje na prijavnem oknu aplikacije?
2. Kje še lahko določite vloge uporabnikov aplikacije (če niste ustvarili strani za administracijo v sami aplikaciji?
3. Kako spremenite privzeto besedilo v gumbu za dodajanje (Create)?

4. Kako spremenite ime strani po tem, ko ste jo že ustvarili?

21.8 Odgovori

1. V razvojnem okolju aplikacije izberite **Shared Components**, nato **Static Application Files**, kliknite **Create File** in v prikaznem oknu izberite želeno grafiko. Kopirajte referenco te datoteke v odložišče, se postavite na urejanje strani **9999-Login Page**, postavite se v polje css in vpišite besedilo v inline:

```
body {
background-image:
url("#APP_FILES#\textit{the reference name of the image}");
background-size:1700px 900 px:
}
```

2. V razvojnem okolju aplikacije izberite **Shared Components** in nato **Application Access control**.
3. V urejevalniku želene strani poiščite gumb **Create**. Spremenite besedilo polja Label in shranite spremembo
4. V izvajalni strani kliknite **Quick Edit**. Nato kliknite naslov in izberite puščico v področju zgornjega menija. Kliknite **Edit Component** in v polju **Short Name** spremenite ime. Potrdite spremembo s klikom **Apply Changes**.



22. Razporejanje uradnih ur

J. MAŃKO, M. SOŃTA, R. LESKOVAR; PREVOD R. LESKOVAR

22.1 Poslovni vidik primera

Aplikacijo za rezervacijo uradnih ur (govorilnih ur) na univerzi je s poslovnega vidika mogoče razumeti kot orodje za racionalizacijo in optimizacijo postopka načrtovanja urnikov za univerzitetno osebje. Pomaga lahko izboljšati učinkovitost razporejanja, zmanjšati administrativno obremenitev, povečati zadovoljstvo študentov in zagotoviti informacije o razpoložljivosti v realnem času. Aplikacija lahko ustvari tudi podatke in vpoglede v vzorce sestankov in uporabo, ki lahko prispevajo k odločanju in dodeljevanju virov. Poleg tega lahko aplikacija ponudi priročno in dostopno platformo za rezervacije za študente, ki pomaga spodbujati bolj proaktivno in angažirano študentsko skupnost. Vsak teden je akademsko osebje dolžno dati študentom na razpolago 60 minut svojega časa, ki se interno imenuje “uradne ure”. Ko se bliža konec semestra se pogosto zgodi, da se povpraševanja študentov povečajo tako po številu kot po obsegu. V takem primeru je 60 minut na teden premalo, saj so študenti odločeni, da čim prej dobijo nujno želene informacije od akademskega osebja. Čeprav obstaja nekaj standardiziranih tem za razpravo, kot so: disertacija, izpiti, zaključne naloge, pa so nekateri primeri povezani z drugimi zadevami, o katerih bi morda morali razpravljati, vendar ne nujno med uradnimi urami z učiteljem (na primer: administrativne zahteve ali težave, ki jih je mogoče enostavno rešiti brez posredovanja akademskega učitelja). V zadnjih letih, predvsem zaradi pandemije COVID-19, so univerze doživele radikalen tehnološki prehod, ki je omogočil različne oblike poučevanja na daljavo z uporabo razpoložljivih platform in komunikatorjev, ki bi olajšali spletno poučevanje. Pravzaprav so bile skoraj vse akademske dejavnosti prenesene na splet, vključno seveda z uradnimi urami na daljavo. Čeprav obstaja soglasje med akademsko skupnostjo, da poučevanje na daljavo ne bo nikoli v celoti nadomestilo poučevanja v predavalnici, se poučevanje na daljavo in celo delo še naprej do neke mere izvaja po vsem svetu. Še pomembneje je, da so nekatere univerzitetne dejavnosti in procesi že prešli ali bodo v prihodnje v celoti prešli na spletni način, na primer za zagotovitev učinkovitega brezpapirnega kroženja dokumentov. Te okoliščine še bolj krepijo nujno poslovno potrebo po izvedljivem digitalnem orodju, ki bi omogočalo spletno rezervacijo uradnih ur in bi lahko potekala prek spleta. Jasno je, da bi celotna akademska skupnost imela veliko koristi od takšne aplikacije, ki naj bi kmalu postala vsakodnevno orodje. Vaskodnevno bi ga uporabljali vsi člani akademske skupnosti, tako kot e-pošto in ne nekakšna digitalno modno muho.

22.2 Definicija problema

Rezervacija (prijava) termina in vnaprejšnje vsebinsko načrtovanje srečanja zaenkrat nista mogoča. Pravzaprav so študenti, če se pojavijo v poljubnem številu, postreženi po principu, kdor prvi pride, prvi melje. Ne samo, da to povečuje frustracije študentov, ki morajo čakati v vrsti, ne da bi bili prepričani, da bo njihov sestanek sploh izveden, ampak tudi vznemirja akademsko osebje, ki mora izbirati med tem, ali pusti nekaterim študentom, da čakajo zaman, ali prestavijo njihov sestanek na kasnejši termin. Posledično prihaja do situacij, v katerih akademsko osebje učinkovito izvaja pro bono svetovanje za študente ali celo na račun njihovega zasebnega življenja, samo, da bi se izognili temu, da bi jih študentje zaznali kot nedostopne ali neprijazne. Poleg tega ta status quo pomeni neučinkovito razporeditev časa za študente in akademsko osebje ter nepotrebno delovno obremenitev za slednje.

Ta administrativna sistemska neučinkovitost sili akademsko osebje, da prevzame dodatne, pogosto neplačane odgovornosti, hkrati pa ohranja študente negotove glede rezultatov in najverjetneje komajda zadovoljne s celotnim procesom. Je izjemen primer prelaganja posledic nedelujočega sistema naročanja sestankov na skupine, ki so nižje v akademski hierarhiji (študentje in akademsko osebje), namesto da bi našli rešitev od zgoraj navzdol, ki bi jo sprožili univerzitetni organi. Tehnologija torej ponuja izvedljivo, opolnomočeno in izvedljivo rešitev, ki bi močno razbremenila akademsko osebje, hkrati pa zagotovila večje zadovoljstvo in učinkovitost uporabe za študente.

Študenti se pogosto obračajo na študentski referat z vprašanji o datumih in krajih uradnih ur. Ker ni zanesljivega vira, se zanašajo na izkušnje in študente pogosto usmerjajo na napačna mesta ob napačnem času. V skladu s pravili akademske ustanove mora vsako spremembo uradnih ur odobriti prodekan. Soglasja so podeljena v neposredni komunikaciji med učiteljem in prodekanom, študentski referat pa o tem nima informacij. Prodekan si zato želi, da bi študentski referat vstopil v postopek prestavljanja uradnih ur.

Vodstvo akademskih ustanov nima pravega vpogleda v razporede in obremenitve na uradnih urah pedagoških delavcev. Govorice in druge nezanesljive informacije niso dobra podlaga za racionalne odločitve o povečanju ali zmanjšanju števila delovnih ur. To je lahko povezano tudi z nagrajevanjem učiteljev za njihove dodatni napor pri delu s študenti.

22.3 Primeri uporabe

Aplikacija za rezervacijo uradnih ur omogoča dostop vodstva, osebja študentske pisarne, študentov in akademskega osebja (učiteljev), pri čemer ima vsaka skupina različne privilegije:

- Vodstvo dobi povzetek informacij, zbranih na nadzorni plošči za upravljanje (število uradnih ur po letu in mesecu, 10 najboljših učiteljev glede na število študentov, 5 najboljših učiteljev s prezasedenostjo in 5 učiteljev z najmanj študenti med uradnimi urami).
- Osebje študentske pisarne ima dostop do interaktivnega poročila o uradnih urah in prestavi morebitne uradne ure.
- Študent ima dostop do koledarja svojih terminov, koledarja vseh uradnih ur, ki jih ponujajo učitelji študentovega študijskega programa, prijave preko koledarja in podrobnega interaktivnega poročilo o vpisih študentov na uradne ure.
- Akademsko osebje oz. učitelji lahko dobijo podrobno interaktivno poročilo o uradnih urah učiteljev, koledar z uradnimi urami učiteljev in lahko prerazporedijo govorilne ure brez vpisov.

22.3.1 Opis primera uporabe

Zaradi jedrnatosti bomo predstavili samo dva primera uporabe:

- Prerazporeditev uradnih ur učitelja, kjer ni vpisanega študenta, zahteva, da se učitelj prijavi, pregleda razpored uradnih ur brez vpisanih in enega za drugim “premika” na druge datume.
- Prijava študenta na uradne ure je sestavljena iz pregleda koledarja (vseh govorilnih ur, ki

Tabela 22.1: Opis primera uporabe: prerazporeditev uradnih ur učitelja.

Ključni pojem	**Vrednost**
ID:	*Ch22-01*
Naslov:	*Prerazporeditev uradnih ur učitelja*
Opis:	*Učitelj uporabi aplikacijo v APEX-u za prerayporeditev uradnih ur, na katere se ni prijavil noben študent.*
Glavni igralec:	*Učitelj*
Predhodni pogoji:	Brskalnik na osebnem računalniku ali telefonu, učitelj ima poverilnice za dostop in privilegije, aplikacija je dostopna na strežniku
Pogoji po zaključku:	Podatki so shranjeni v tabeli ch22_duty_hours
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Učitelj odpre brskalnik in se prijavi v aplikacijo* 2. *Izbere uradne ure z nič prijavljenimi* 3. *Klikne na oznako na koledarju za termin z nič prijavljenimi* 4. *Odpre se obrazec - učitelj v obrazec vnese nove podatke (locacijo, začetni in končni čas)* 5. *Pregleda vnesene podatke in potrdi z gumbom*
Razširitve:	-
Pogostost uporabe:	*Približno 4 od skupaj 30 obveznih uradnih ur v akademskem letu za posameznega učitelja.*
Status:	[Zaključen]
Lastnik:	*Uporabnik s privilegiji učitelja*
Prioriteta:	*Zmerna*

jih ponujajo vsi učitelji na študijskem programu študenta), izbire želenih uradnih ur, vnosa namena in potrditve vpisa. Vpis ni dovoljen več kot enkrat na razpisane uradne ure.

22.3.2 Delno strukturiran opis

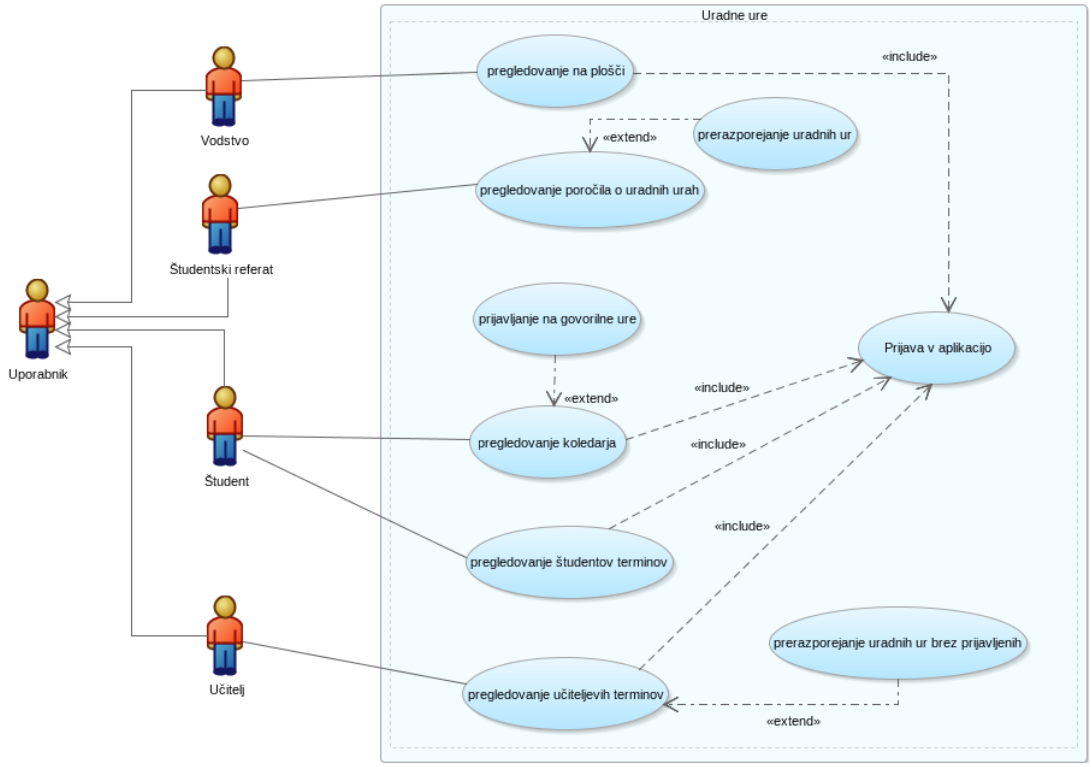
Prvi primer uporabe je prerazporeditev uradnih ur učitelja, ko se ne vpiše noben študent (tabela 22.1). Prerazporeditev omejujemo na to vrsto, ker ni prikrajšan noben študent. Drugi primer uporabe cilja na vpis študentov za uradne ure (tabela 22.2). Aplikacija naj bi preprečevala tudi večkratno vpisovanje v en termin uradnih ur.

22.3.3 Diagram primera uporabe

Slika 22.1 predstavlja identificirane primere uporabe. Upoštevajte, da se “prijava” primera uporabe izvede samo enkrat na uporabniško sejo in da nobenega od drugih primerov uporabe ni mogoče izvesti pred uspešnim zaključkom “prijave”. Trije primeri uporabe, in sicer “prerazporeditev uradnih ur”, “prijava na uradne ure” in “prerazporeditev uradnih ur brez vpisanih”, so odvisni in označeni s povezavo “extend”. Ti primeri uporabe se izvedejo na zahtevo po zaključku nadrejenega primera uporabe.

Tabela 22.2: Opis primera uporabe: prijava študenta na uradne ure.

Ključni pojem	**Vrednost**
ID:	*Ch22-02*
Naslov:	*Prijava študenta na uradne ure*
Opis:	*Študent uporabi aplikacijo v APEX-u za prijavo na uradne ure. V koledarju so razpisane vse uradne ure za vse učitelje, ki predavajo v študentovem študijskem programu. Običajno je študent vpisan v en študijski program vpisan, izjemoma pa se lahko dovoli študij dveh ali več. Aplikacija bo preprečila večkratne vpise enega študenta na določene uradne ure.*
Glavni igralec:	*Študent*
Predhodni pogoji:	Brskalnik na osebnem računalniku ali telefonu, študent ima poverilnice za dostop in privilegije, aplikacija je dostopna na strežniku
Pogoji po zaključku:	Podatki so shranjeni v tabeli ch22_dh_slot
Glavni del	*Scenariji*
Uspešen scenarij:	1. *Študent odpre brskalnik in se prijavi v aplikacijo* 2. *Prijava poteka s pomočjo koledarja* 3. *Študent klikne na želeni termin razpisanih uradnih ur (termini so obarvani in prikazujejo zasedenost)* 4. *Odpre se obrazec in študent s klikom na gumb potrdi prijavo na označeni termin*
Razširitve:	-
Pogostost uporabe:	*Preko 1000 študentov se prijavi na uradne ure v poprečju 2 krat na semester pri vsakem od 10 do 15 učiteljev na študentovem programu.*
Status:	[Zaključen]
Lastnik:	*Študent*
Prioriteta:	*Visoka*



Slika 22.1: Diagram primerov uporabe.

22.4 Model podatkov

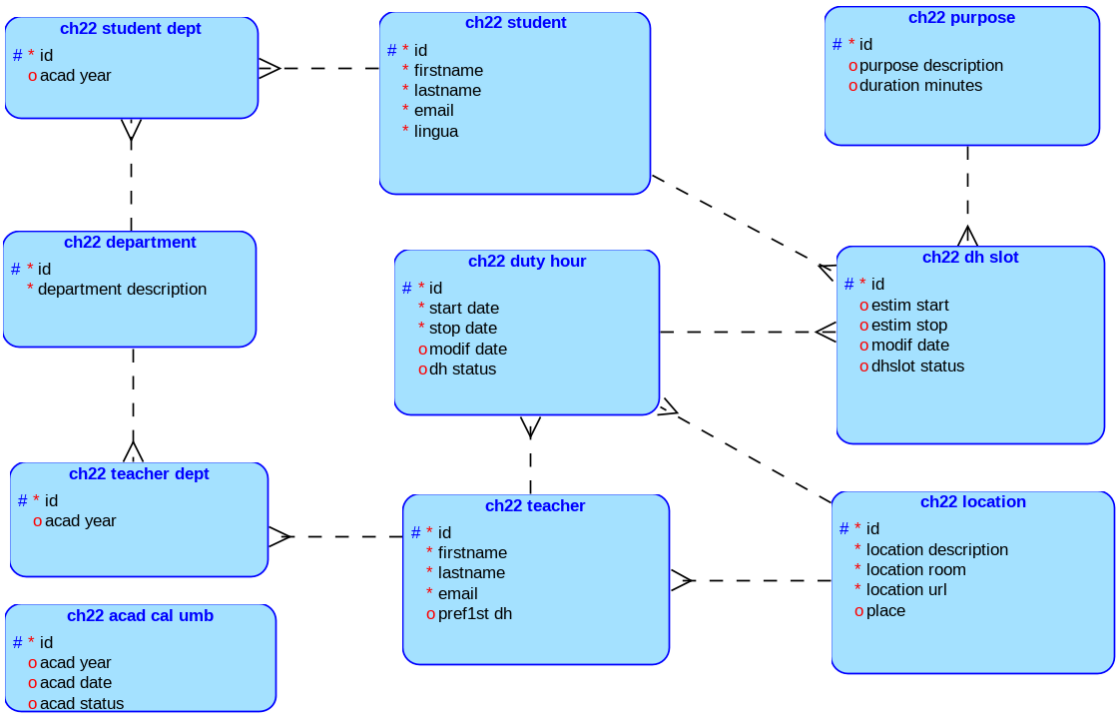
22.4.1 Opis modela podatkov

V logičnem modelu podatkov je deset entitet:

- **ch22_dh_slot** predstavi eno srečanje študent-učitelj;
- **ch22_duty_hour** predstavlja vnaprej dodeljeni časovni odsek učitelja (uradne ure), ki vsebuje več terminov sestankov študent-učitelj;
- **ch22_purpose** vsebuje namene in povprečno trajanje sestanka študent-učitelj za ta namen;
- **ch22_student_dept** kaže na en študijski program enega študenta. Izjemoma je dovoljen vpis v več kot en program (sinonim za oddelek) v enem študijskem letu;
- **ch22_student** je oseba, ki študira na univerzi in je vključena v študijski program (sinonim za oddelek);
- **ch22_teacher_dept** kaže na en študijski program (ali oddelek) enega učitelja. En učitelj je običajno zaposlen v več kot enem študijskem programu;
- **ch22_teacher** je oseba, ki izvaja akademske dejavnosti (predavanja, laboratorijske vaje itd.)
- **ch22_location** lokacija je fizična (R) ali virtualna (I) ali oboje (B) - to je kraj, kjer potekajo uradne ure;
- **ch22_department** je sinonim za študijski program;
- **ch22_acad_cal_umb** določa delovne/nedelovne dni. Ta entiteta je povezana z vsemi entitetami, ki vsebujejo atribute datuma, vendar teh relacij ne bomo modelirali eksplicitno, ker je implicitna relacija dovolj za popolnost modela.

22.4.2 Logični model podatkov

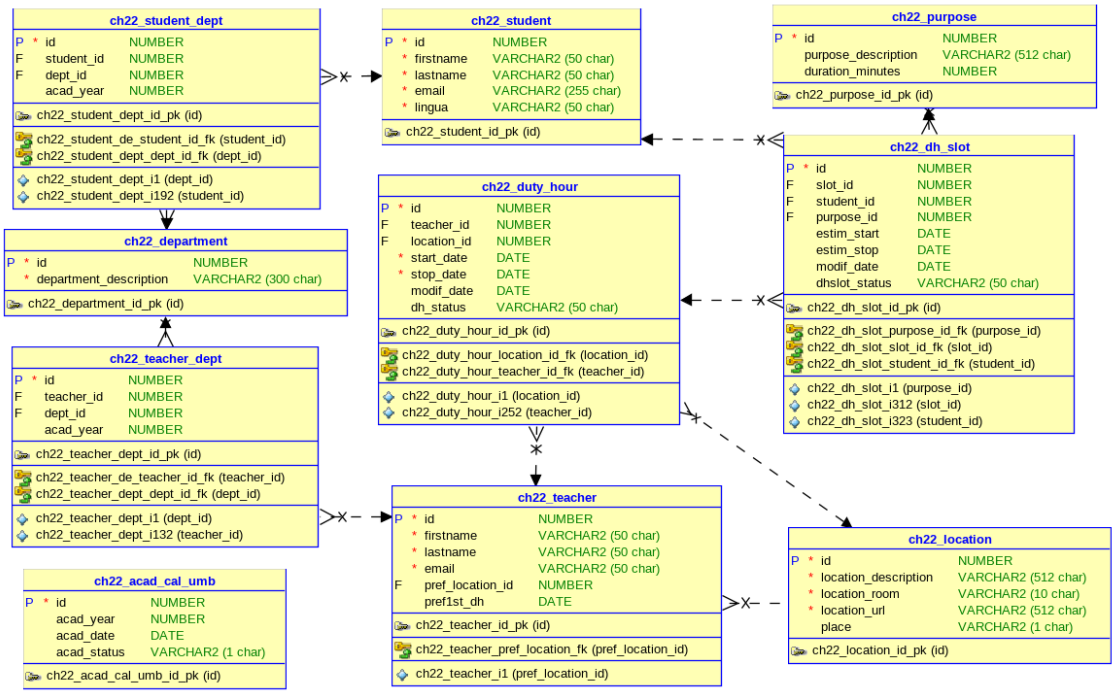
Logični model podatkov je prikazan na sliki 22.2.



Slika 22.2: Logični model podatkov.

22.4.3 Relacijski model podatkov

Samodejno pretvorbo iz logičnega podatkovnega modela v relacijski podatkovni model v Oracle SQL Data Modeler zagotavlja funkcija *Engineer to relational*. Rezultat pretvorbe je relacijski model podatkov (slika 22.3), pripravljen za izvoz kot skript SQL.



Slika 22.3: Relacijski model podatkov.

Relacijski podatkovni model je mogoče razviti tudi brez SQL Data Developer Data Modeler. V APEX-u lahko uporabljamo Quick SQL.

22.4.4 Skript v Quick SQL

Prikazani Quick SQL (slika 22.4) vsebuje vse potrebne napotke za generiranje skripta SQL.

```
ch22_acad_cal_umb
    acad_year num
    acad_date date
    acad_status vc1 /check y n
ch22_department
    department_description vc300 /nn
ch22_purpose
    purpose_description vc512
    duration_minutes num /check 10 15 20 25 30 45 60 /default 15
ch22_location
    location_description vc512 /nn
    location_room vc10 /nn /default R1
    location_url vc512 /nn /default https://beeapex.eu/bbb
    place vc1 /check R,I,B /default I
ch22_teacher
    firstname vc50 /nn
    lastname vc50 /nn
    email vc50 /nn
    pref_location_id num /fk ch22_location
    pref1st_dh date
ch22_teacher_dept
    teacher_id num /fk ch22_teacher
    dept_id num /fk ch22_department
    acad_year num
ch22_student_dept
    student_id num /fk ch22_student
    dept_id num /fk ch22_department
    acad_year num
ch22_duty_hour
    teacher_id /fk ch22_teacher
    location_id /fk ch22_location
    start_date date /nn
    stop_date date /nn
    modif_date date
    dh_status vc50
ch22_dh_slot
    slot_id /fk ch22_duty_hour
    student_id /fk ch22_student
    purpose_id /fk ch22_purpose
    estim_start date
    estim_stop date
    modif_date date
    dhslot_status vc50
ch22_student
    firstname vc50 /nn
    lastname vc50 /nn
    email vc255 /nn
    lingua vc50 /nn
```

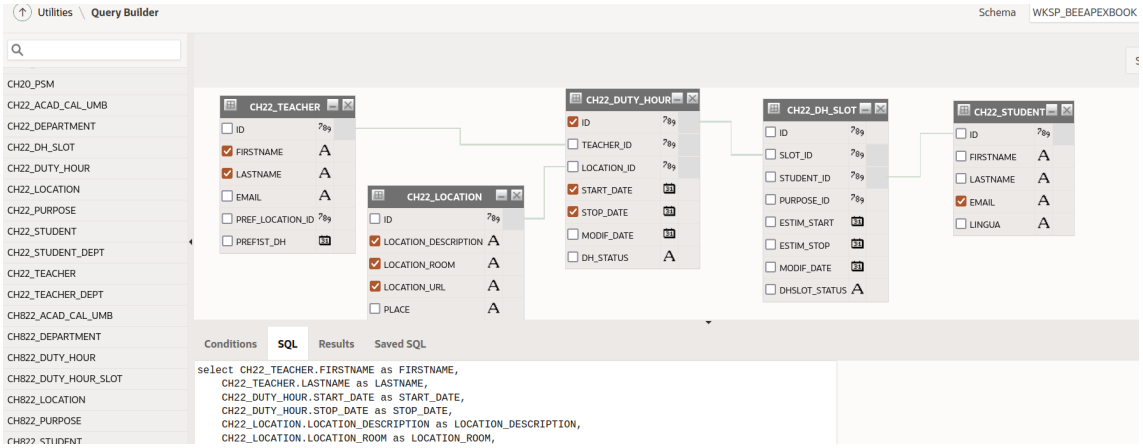
Slika 22.4: Model podatkov, ki je opisan s skriptom Quick SQL.

22.4.5 Skript v SQL za kreiranje tabel

Skript SQL za ustvarjanje tabel je na voljo v učnem gradivu kot datoteka CH22CREATE.sql. Skript za izdelavo tabel je na voljo tudi v pakirani aplikaciji v učnih gradivih.

22.4.6 Pomoč pri pisanju poizvedb - Query builder

Obravnavana aplikacija vsebuje nekaj zelo dolgih poizvedb, ki bi lahko začetnika v APEX prestrašile. Toda skrbi so odveč. V APEX-u je na voljo Query Builder (SQL Workshop > Utilities > Query Builder). Ta funkcija omogoča, da vsakdo napiše sintaktično pravilno poizvedbo zgolj s klikanjem in vlečenjem. Slika 22.5 prikazuje primer poizvedbe za študente, ki so vpisani v določeni termin uradnih ur.



Slika 22.5: Primer uporabe funkcije Query Builder.

Generana poizvedba v SQL ima naslednjo vsebino:

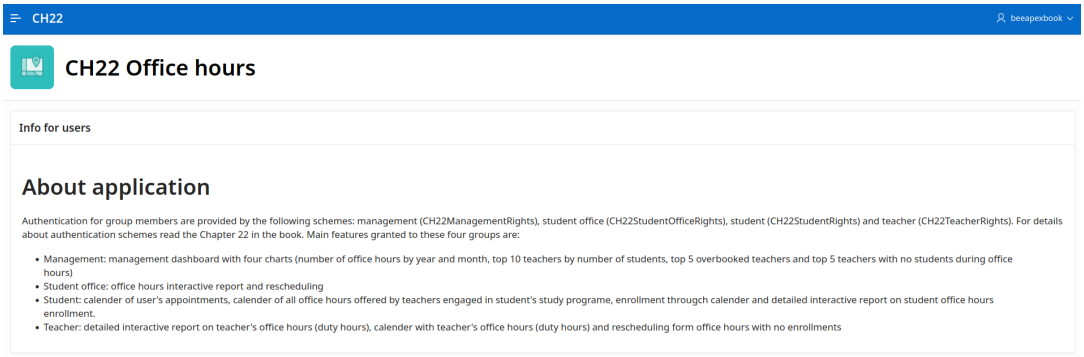
```
select CH22_TEACHER.FIRSTNAME as FIRSTNAME,
       CH22_TEACHER.LASTNAME as LASTNAME,
       CH22_DUTY_HOUR.START_DATE as START_DATE,
       CH22_DUTY_HOUR.STOP_DATE as STOP_DATE,
       CH22_LOCATION.LOCATION_DESCRIPTION as
       LOCATION_DESCRIPTION,
       CH22_LOCATION.LOCATION_ROOM as LOCATION_ROOM,
       CH22_LOCATION.LOCATION_URL as LOCATION_URL,
       CH22_DUTY_HOUR.ID as ID,
       CH22_STUDENT.EMAIL as EMAIL
from CH22_DH_SLOT CH22_DH_SLOT,
       CH22_DUTY_HOUR CH22_DUTY_HOUR,
       CH22_STUDENT CH22_STUDENT,
       CH22_TEACHER CH22_TEACHER,
       CH22_LOCATION CH22_LOCATION
where CH22_DH_SLOT.SLOT_ID=CH22_DUTY_HOUR.ID
       and CH22_DUTY_HOUR.LOCATION_ID=CH22_LOCATION.ID
       and CH22_DUTY_HOUR.TEACHER_ID=CH22_TEACHER.ID
       and CH22_STUDENT.ID=CH22_DH_SLOT.STUDENT_ID
```

Z dodatnim pogojem na koncu “and CH22_TEACHER.EMAIL = :APP_USER” bo poizvedba vrnila prijavljene študente pri določenem učitelju. Query Builder je nujen za popolne začetnike, izkušenejši razvijalci pa bodo hitro odkrili, da s to funkcijo prihranijo čas pri pisanju dolgih poizvedb.

Poizvedbe v tej aplikaciji uporabljajo operator združevanja (||), funkcijo EXTRACT, CTE, pridobitev prvih 10 in 5 vrstic ter uporabniško razviro funkcijo CH22_dh_utilization. Če boste najprej namestili pakirano aplikacijo v dodatnem gradivu ter si ogledali poizvedbe na koledarjih, preglednih ploščah in poročilih, se boste hitreje naučili pisati poizvedbe z SQL.

22.5 Uporabniški vmesniki

Aplikacija ima eno skupno, domačo stran za vse uporabnike - Home page. Ta stran vsebuje informacije o aplikaciji za vse uporabnike (slika 22.6).



Slika 22.6: Domača stran aplikacije.

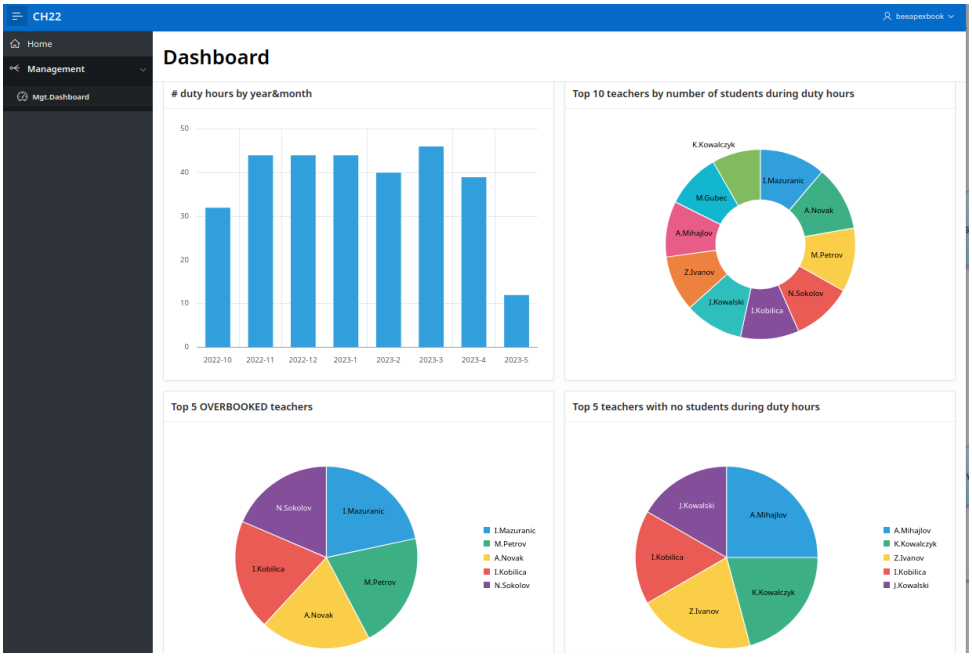
Vodstvo, študentski referat, študenti in učitelji po uspešni prijavi dobijo dostop do vmesnikov tistih funkcij aplikacije, ki jih smejo uporabljati.

22.5.1 Uporabniški vmesniki za vodstvo

Uporabniki s pravicami vodstva lahko dostopajo do nadzorne plošče (slika 22.7). Nadzorna plošča nudi podporo odločanju v zvezi z uradnimi urami in učitelji. Prikazuje:

- število uradnih ur po letih in mesecih
- 10 najboljših učiteljev po številu študentov
- top 5 prezasedenih učiteljev in
- top 5 učiteljev brez študentov med uradnimi urami

V času razvoja aplikacije vodstvo ni podalo nobenih dodatnih zahtev za poročila.

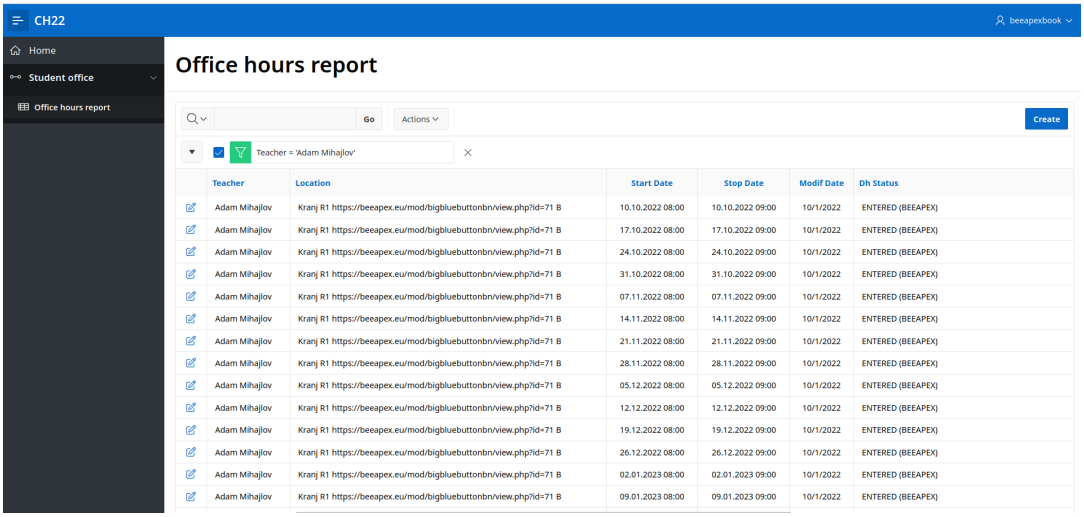


Slika 22.7: Nadzorna plošča za vodstvo.

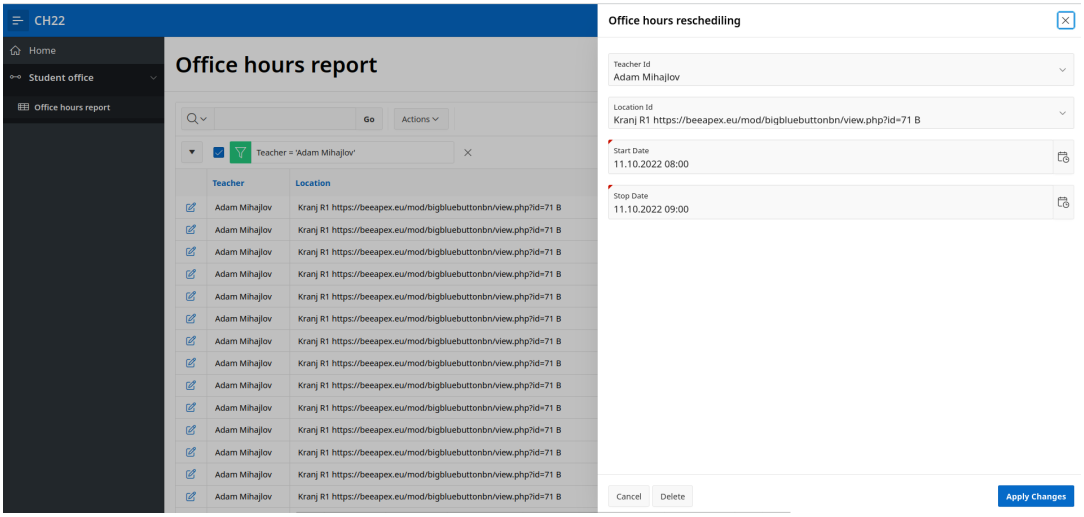
22.5.2 Uporabniški vmesniki za študentsko pisarno

Uporabniki s pravicami študentske pisarne lahko dostopajo do interaktivnih poročil ter smejo prerazporejati uradne ure. Slika 22.8 prikazuje interaktivno poročilo z aktiviranim filtrom (zato je prikazan samo en učitelj). Slika 22.9 prikazuje obrazec za prerazporejanje uradnih ur poljubnega

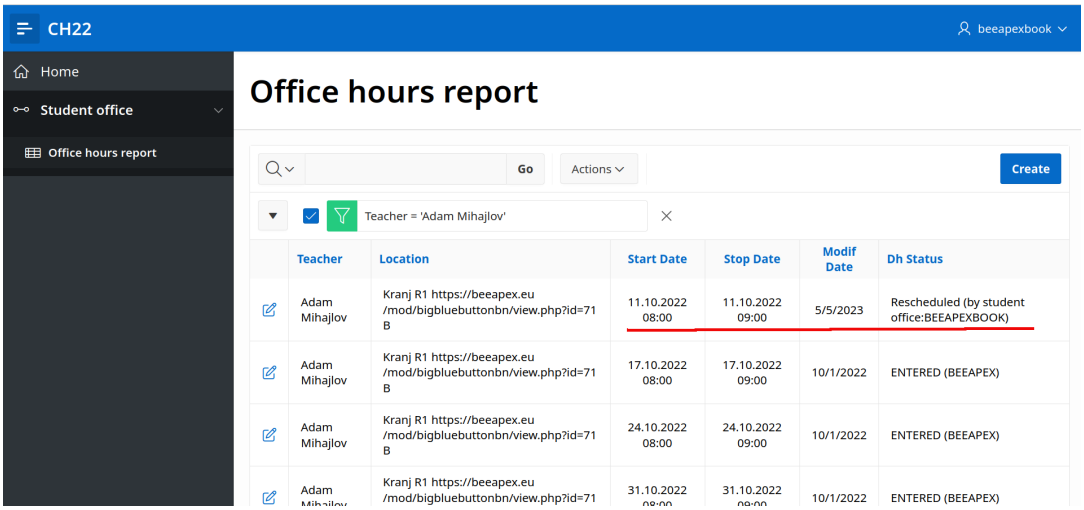
učitelja. Uporabnik lahko spremeni učitelja, lokacijo, začetni in končni čas uradnih ur. ki so že razpisane ali pa doda popolnoma nov zapis. Slika 22.10 prikazuje poročilo, ki se prikaže po prerazporeditvi. Avtomatsko sta spremenjena dva podatka: čas zadnje spremembe in status uradnih ur. Status vsebuje tudi ime uporabnika, ki je izvedel spremembo.



Slika 22.8: Interaktivno poročilo o uradnih urah za študentsko pisarno.



Slika 22.9: Obrazec za študentsko pisarno - prerazporejanje uradnih ur.

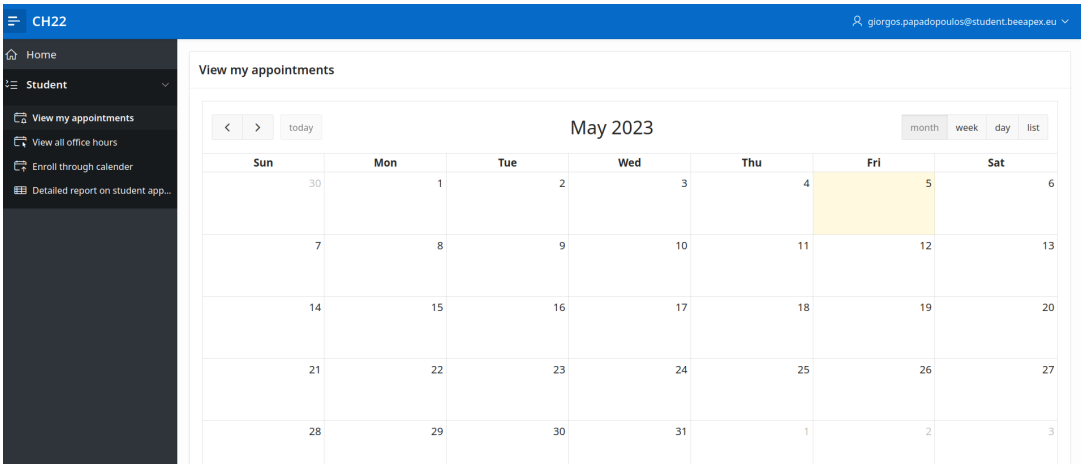


Slika 22.10: Interaktivno poročilo o uradnih urah za študentsko pisarno po prerazporeditvi.

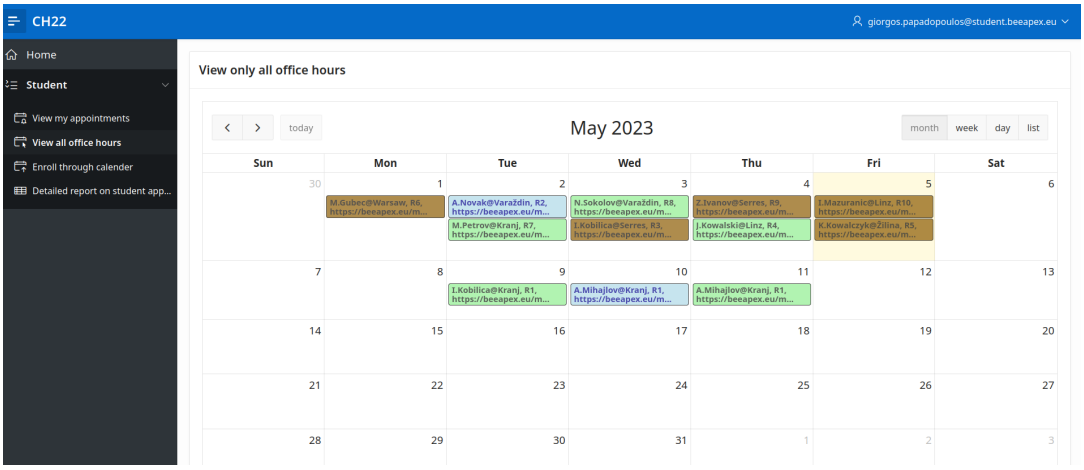
22.5.3 Uporabniški vmesniki za študente

Barve na koledarju so odvisne od zasedenosti posamezne uradne ure. Ko ni prijavljenih (NONE), je barva zelena. Če ni izpolnjena minutaža uradnih ur (AVAILABLE), je termin prikazan modro. Kadar je prijavljenih preveč (glede na vsoto planiranih minut namena, OVEBOOKED), je uporabljena rjava barva. V trenutni verziji aplikacije je možna prijava tudi v primeru prezasedenosti. Uporabniki s pravico študenta lahko dostopajo do:

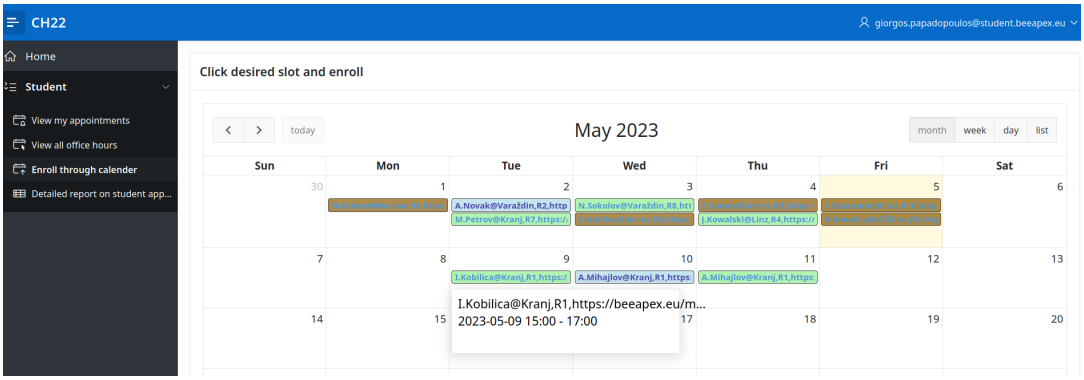
- Koledarja lastnih sestankov (slika 22.11).
- Koledarja vseh uradnih ur učiteljev študentovega študijskega programa (slika 22.12).
- Prijavnega obrazca na govorilne ure preko koledarja. Študent najprej izbere želene uradne ure (slika 22.13). V prijavni obrazed vnese le namen (slika 22.14) in klikne gumb “Enroll”.Če je prijava uspešna, se na ekranu izpiše odziv, ki je prikazan na sliki 22.15. Študent lahko preveri prijavo tudi s pregledom “View my appointments” na menujski izbiri (slika 22.16).
- Detajlnega poročila o vseh študentov udeležbah na uradnih urah (slika 22.17).



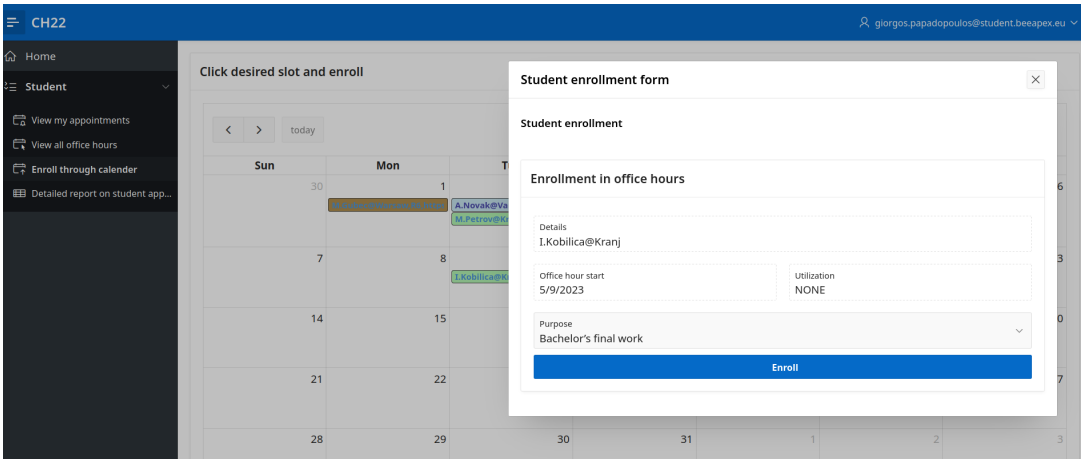
Slika 22.11: Koledar študentovih prijav na uradne ure.



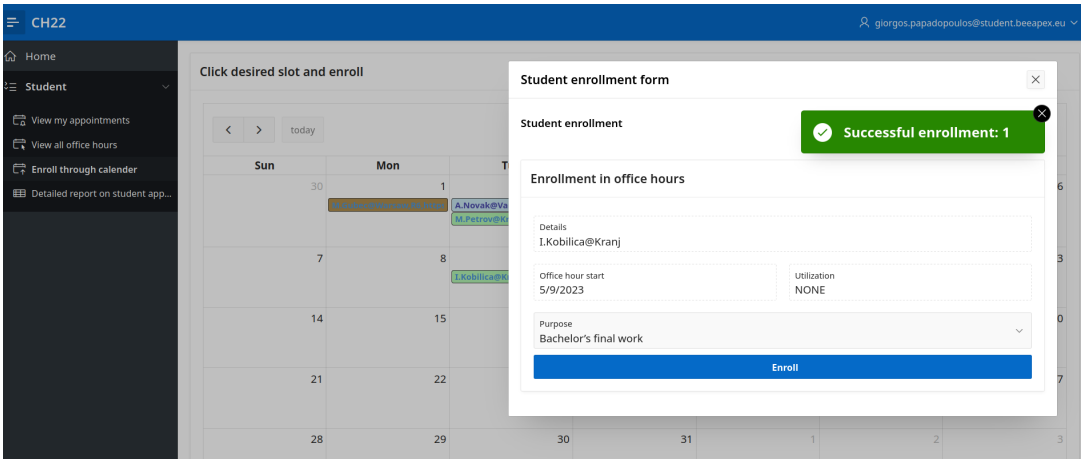
Slika 22.12: Koledar ponujenih uradnih ur vseh učiteljev vpisanega študijskega programa.



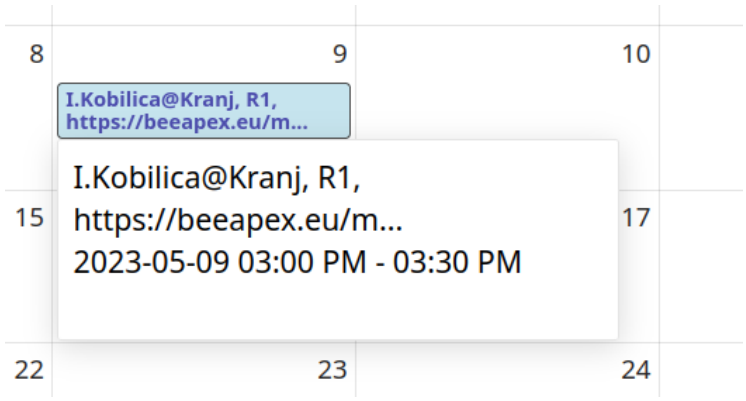
Slika 22.13: Prijava na uradne ure preko koledarja - izbira koledarskega termina.



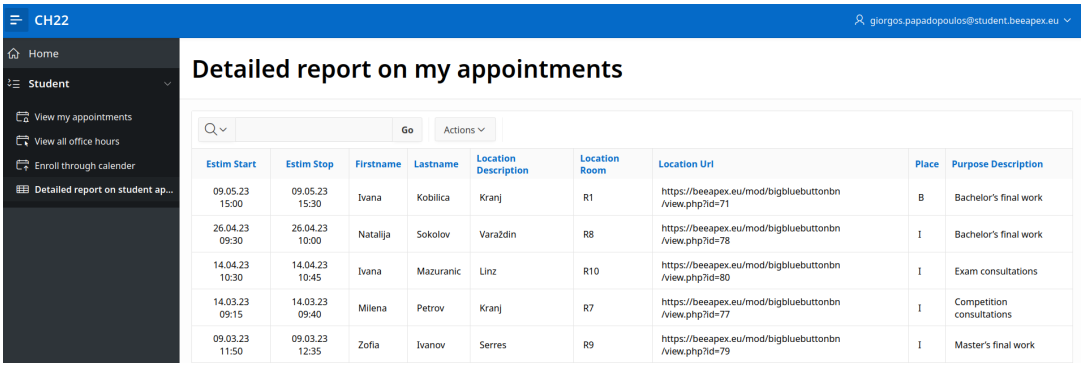
Slika 22.14: Prijava na uradne ure preko koledarja - izbira namena.



Slika 22.15: Prijava na uradne ure preko koledarja - prikaz uspešne prijave.



Slika 22.16: Preverjanje prijave z menujsko izbiro “View my appointments”.



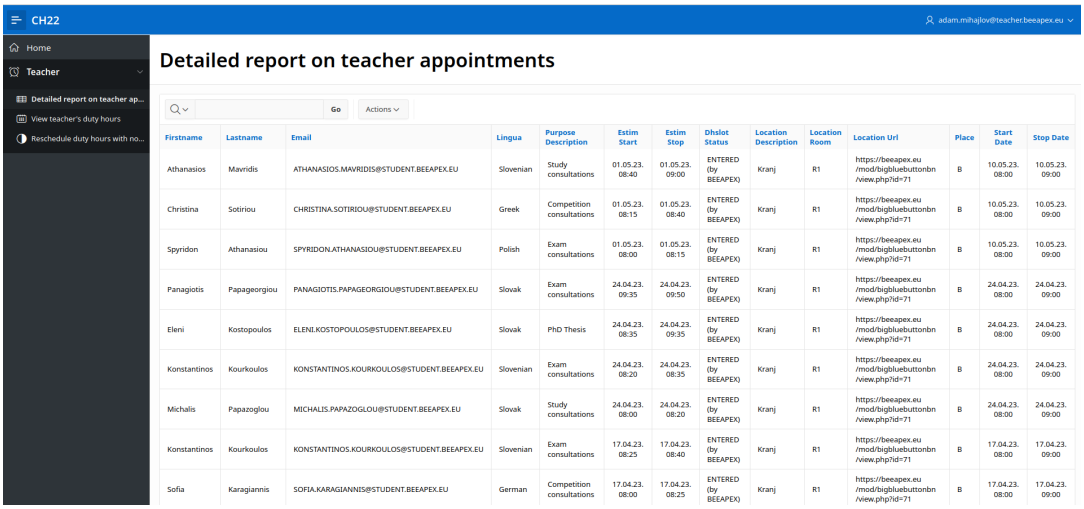
Slika 22.17: Detajlno interaktivno poročilo vseh študentovih uradnih ur.

22.5.4 Uporabniški vmesnik za predavatelje

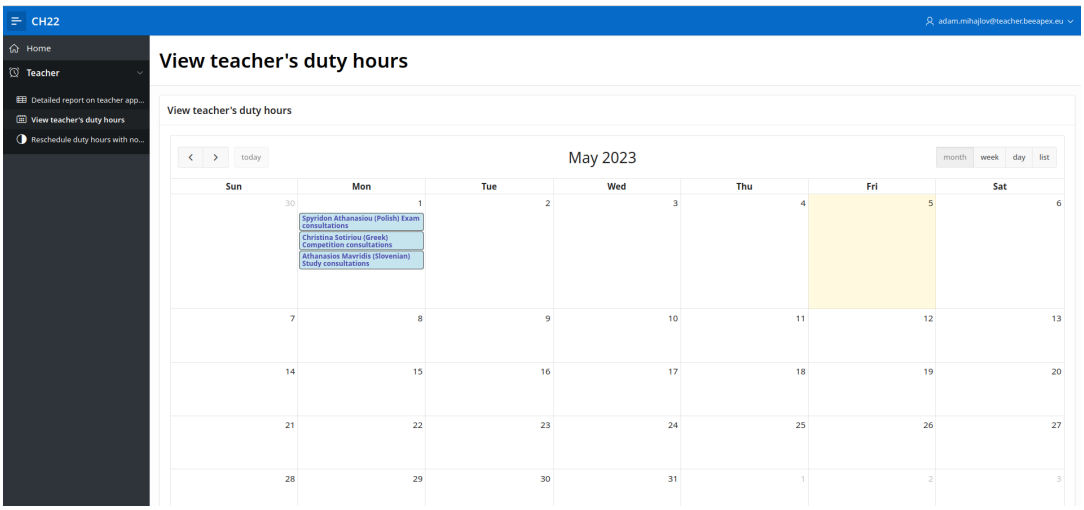
Barve na koledarju so odvisne od zasedenosti posamezne uradne ure. Ko ni prijavljenih (NONE), je barva zelena. Če ni izpolnjena minutaža uradnih ur (AVAILABLE), je termin prikazan modro. Kadar je prijavljenih preveč (glede na vsoto planiranih minut namena, OVEBOOKED), je uporabljena rjava barva. Uporabnik s pravicami učitelja laho dostopa do:

- Detajnega pregleda lastnih uradnih ur (slika 22.18).

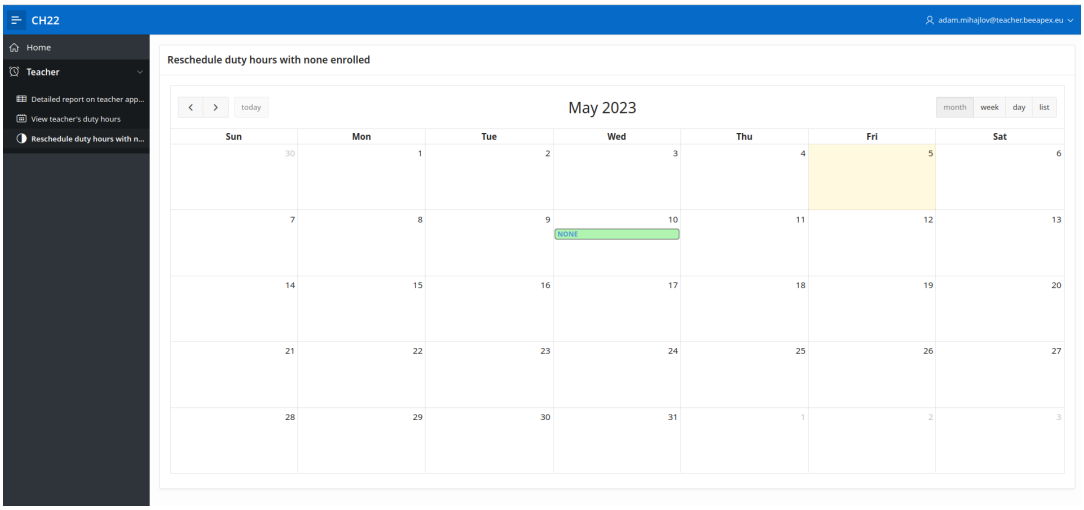
- Koledarja s podatki o prijavljenih študentih. Učitelj za vsako svojo prijavo vidi ime študenta, jezik študenta in namen (slika 22.19).
- Funkcije prerazporeditve uradnih ur, na katere ni nihče prijavljen. Najprej se prikažejo na koledarju termini s statusom NONE (slika 22.20). Klik na tak termin odpre modalni obrazec (slika 22.21) z možnostjo popravka začetka in konca (datum, ura, minuta). S klikom na gumb "Apply changes" se uradne ure premaknejo na koledarju na želeni termin (slika 22.22).



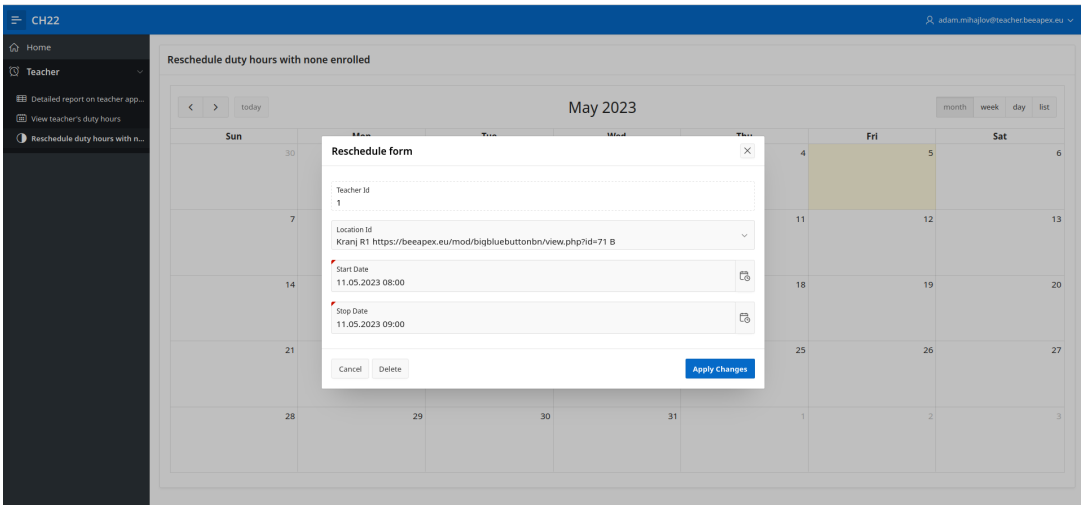
Slika 22.18: Detajno poročilo o učiteljevih uradnih urah.



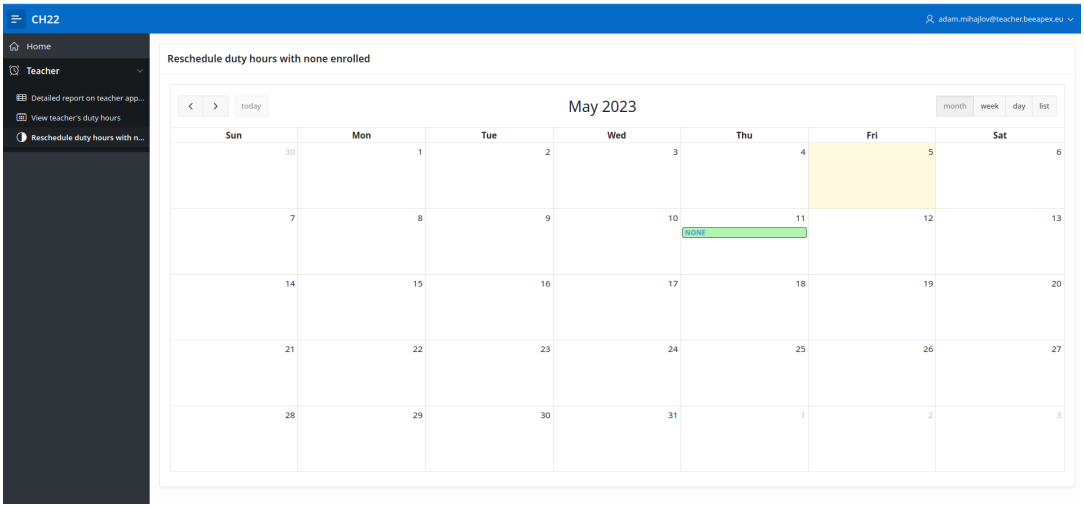
Slika 22.19: Učiteljev koledar uradnih ur z vidnimu imeni študentov, njihovimi jeziki in nameni prijave.



Slika 22.20: Prerazporeditev uradnih ur z nič prijavljenimi - pogled na koledarju.



Slika 22.21: Prerazporeditev uradnih ur z nič prijavljenimi - vnos novih datumov.



Slika 22.22: Rezultat prerazporeditve uradnih ur z nič prijavljenimi - pogled na koledarju.

22.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija, skripti za kreiranje tabel, dodajanje testnih podatkov, kreiranje funkcij v PL/SQL, brisanje tabel
- video vodič za: a) namestitev pakirane aplikacije in kreiranje študentov ter učiteljev) in b) razvoj aplikacije od začetka do konca

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (gesla ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter22 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

22.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele, indeksi, funkcije ter napolnijo testni podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo. Pakirana aplikacija uporablja datoteko za stil CSS (ch22_cal.css), ki se aktivira na večini koledarskih straneh kot #APP_FILES#ch22_cal#MIN#.css:

```
.fc-event .fc-content div.fc-time { display: none;}

.fc-event.my-cal-blue {
background-color: lightblue;
border: 0.5pt solid black;
opacity: 0.7;
}

.fc-event.my-cal-blue .fc-event-title {
color: darkblue;
font-weight: bold;
}

.fc-event.my-cal-orange {
background-color: orange;
border: 0.5pt solid black;
opacity: 0.7;
}

.fc-event.my-cal-orange .fc-title {
color: darkred;
font-weight: bold;
}

.fc-event.my-cal-dark-orange {
background-color: #8B5A00;
border: 0.5pt solid black;
opacity: 0.7;
}

.fc-event.my-cal-dark-orange .fc-title {
color: white;
```

```
font-weight: bold;
}

.fc-event.my-cal-white {
background-color: white;
border: 0.5pt solid black;
opacity: 0.7;
}

.fc-event.my-cal-white .fc-title {
color: black;
font-weight: bold;
}

.fc-event.my-cal-green {
background-color: lightgreen;
border: 0.5pt solid black;
opacity: 0.7;
}

.fc-event.my-cal-green .fc-title {
color: darkgreen;
font-weight: bold;
}
```

Poizvedba za nadzorno ploščo vodstva uporablja naslednje poizvedbe:

```
SELECT
        EXTRACT(YEAR FROM start_date) || '-' ||
        EXTRACT(MONTH FROM  start_date) AS "Year␣and␣Month",
        COUNT(id) AS count
FROM ch22_duty_hour
        GROUP BY EXTRACT(YEAR FROM start_date),
        EXTRACT(MONTH FROM start_date)
        order by 1;

with teacher_appointmets (appointment, teacher) as
        (select CH22_DH_SLOT.ID,
        substr(CH22_TEACHER.FIRSTNAME,1,1) || '.' ||
        CH22_TEACHER.LASTNAME
from CH22_DH_SLOT CH22_DH_SLOT,
        CH22_DUTY_HOUR CH22_DUTY_HOUR,
        CH22_TEACHER CH22_TEACHER
where CH22_DUTY_HOUR.ID=CH22_DH_SLOT.SLOT_ID
        and CH22_TEACHER.ID=CH22_DUTY_HOUR.TEACHER_ID)
        select count(appointment), teacher from teacher_appointmets
        group by teacher
        order by 1 desc
        fetch  first 10 rows only;

with teacher_dh_statuses (duty_hour_id, teacher,dh_utilization) as
        (select CH22_DUTY_HOUR.ID,
```

```
        substr(CH22_TEACHER.FIRSTNAME,1,1) || '.' ||
        CH22_TEACHER.LASTNAME,
        CH22_dh_utilization(CH22_DUTY_HOUR.ID)
        from CH22_DUTY_HOUR CH22_DUTY_HOUR,
        CH22_TEACHER CH22_TEACHER
        where CH22_DUTY_HOUR.TEACHER_ID=CH22_TEACHER.ID and
        CH22_dh_utilization(CH22_DUTY_HOUR.ID) = 'OVERBOOKED')
        select count(duty_hour_id), teacher from teacher_dh_statuses
        group by teacher
        order by 1 DESC
        fetch  first 5 rows only;

with teacher_dh_statuses (duty_hour_id, teacher,dh_utilization) as
        (select CH22_DUTY_HOUR.ID,
        substr(CH22_TEACHER.FIRSTNAME,1,1) || '.' || CH22_TEACHER.LASTNAME,
        CH22_dh_utilization(CH22_DUTY_HOUR.ID)
        from CH22_DUTY_HOUR CH22_DUTY_HOUR,
        CH22_TEACHER CH22_TEACHER
        where CH22_DUTY_HOUR.TEACHER_ID=CH22_TEACHER.ID and
        CH22_dh_utilization(CH22_DUTY_HOUR.ID) = 'NONE')
        select count(duty_hour_id), teacher from teacher_dh_statuses
        group by teacher
        order by 1 DESC
        fetch  first 5 rows only;
```

Ostale poizvedbe si oglejte v pakirani aplikaciji v dopolnilnem študijskem gradivu.

22.6.2 Video vodiči

Na voljo sta dva video vodiča. Prvi kaže namestitev pakirane aplikacije in kreiranje študentov ter učiteljev, drugi pa razvoj aplikacije od začetka do konca. Zaradi velikosti video zapisov je razvoj od začetka do konca prikazan v treh ločenih datotekah.

22.7 Vprašanja

1. Kako dodati uporabnika z učiteljskimi pravicami?
2. Kako bi izboljšali aplikacijo tako, da bi študentu omogočili pisanje zapiskov za določen sestanek?
3. Kako bi omejili rezultate poizvedbe na vrstice, ki imajo datumske stolpce enake ali večje od trenutnega časa?

22.8 Odgovori

1. Uporabite “Manage Users and Groups”. Dodajte novega uporabnika. Uporabite SQL Workshop > Object Browser in izberite tabelo CH22_TEACHER. Nato izberite zavihek “Data” in gumb “Insert row”. Naslov epošte mora biti enak tistemu, ki ste ga uporabili pri kreiranju novega uporabnika - učitelja.
2. Kreirajte novo tabelo, npr. CH22_NOTES s polji: ID, note in DH_SLOT_ID kot tujim ključem z referenco na tabelo CH22_DH_SLOT. Kreirajte novo poročilo in obrazec za to tabelo ter dovolite uporabo le tistim uporabnikom, ki imajo študentske pravice. Če institucionalna pravila dovoljujejo, lahko učiteljem omogočite vpogled v zapiske.
3. Da bi omejili rezultate poizvedb na datume, ki so enaki ali večji od trenutnega časa, dodajte pogoj “WHERE date_column >= SYSDATE”.



23. Primer telekomunikacijskih storitev

VERONIKA ŠALGOVÁ, JOZEF KOSTOLNÝ, MICHAL MRENA, MICHAL KVET, MIROSLAV POTOČÁR; PREVOD R. LESKOVAR

23.1 Poslovni vidik primera

Za učinkovit prikaz postopnega postopka razvoja prototipne aplikacije si za primer oglejmo hipotetično podjetje VEYOMI. VEYOMI je majhno podjetje, ki se osredotoča na prodajo telekomunikacijskih storitev svojim strankam. Ker želi VEYOMI razširiti svoj portfelj izdelkov in poskrbeti za rastočo bazo strank, postane nujno vzpostaviti in vzdrževati robustno aplikacijo, ki omogoča brezhibno upravljanje različnih vidikov, vključno z inventarjem storitev, informacijami o strankah in avtomatiziranimi sistemi zaračunavanja. V vedno bolj konkurenčnem tržnem okolju se VEYOMI zaveda izjemnega pomena zagotavljanja hitrega in varnega dostopa do ključnih informacij za svoje cenjene stranke. Z razvojem intuitivne aplikacije lahko VEYOMI poveča svojo odzivnost in zagotovi, da so pomembne podrobnosti o izdelkih, storitvah, cenah in zaračunavanju zlahka na voljo na dosegu roke zahtevnim strankam.

23.2 Definicija problema

V poslovnem svetu je izredno pomembno, da podjetje vodi evidenco o svojih strankah, izdelkih ali storitvah, ki so jih stranke kupile in o vseh storitvah, ki jih prejemajo. To omogoča podjetju, da ima vse potrebne informacije na dosegu roke, kar olajša učinkovito oskrbo strank. Poleg tega je ključnega pomena, da strankam zagotovi enostaven in dostopen način za preverjanje statusa njihovih storitev. S ponudbo uporabniku prijazne platforme ali aplikacije lahko svojim strankam omogoči, da preprosto izvedo, kaj se dogaja z njihovimi storitvami. Z vzdrževanjem natančnih evidenc in zagotavljanjem enostavnega dostopa do posodobitev storitev lahko podjetje poveča zadovoljstvo strank, zgradi zaupanje in izboljša rezultate poslovanja.

23.3 Primeri uporabe

Pri izdelavi aplikacije je treba začeti s specifikacijo storitev, ki naj jih aplikacija zagotavlja in kaj naj zajema. Zato je priprava seznama posameznih funkcionalnosti in storitev ključnega pomena. Najprej pa se je treba osredotočiti na opredelitev vrst uporabnikov. Za našo uporabo lahko upoštevamo tri temeljne vloge – stranko, upravitelja in skrbnika. Vsaka vloga ima posebne storitve, ki zagotavljajo celotno pokritost funkcionalnosti aplikacije.

Tabela 23.1: Opis primera uporabe: Dodaj storitev.

Ključni pojem	**Vrednost**
ID:	*Ch23-01*
Naslov:	*Dodaj storitev*
Opis:	*Stranka uporabi APEX-ovo aplikacijo, da med ponujenimi storitvami doda novo.*
Glavni igralec:	*Stranka*
Predhodni pogoji:	Dostop do spletne aplikacije, veljavne poverilnice, spletna stran aplikacije je dostopna
Pogoji po zaključku:	Podatki o dodani storitvi so zapisani v bazi podatkov
Glavni del:	*Scenariji*
Uspešen scenarij:	1. *Stranka se prijavi v spletno aplikaciji* 2. *Stranka poišče regijo Buy extra na domači strani* 3. *Stranka vpiše "Amounth"* 4. *Stranka s pomočjo padajočega menija izbere novo storitev (buy type)* 5. *Stranka s klikom na gumb "Buy" potrdi aktiviranje nove storitve*
Razširitve:	-
Pogostost uporabe:	*Stranka v poprečju vsakih 6 mesecev spremeni nabor storitev.*
Status:	[zaključen]
Lastnik:	*Stranka*
Prioriteta:	*Visoka*

23.3.1 Opis primera uporabe

Zaradi jedrnatosti bomo predstavili samo dva primera uporabe :

- Dodaj storitev - Stranke lahko preprosto zahtevajo dodajanje novih storitev svoji obstoječi naročnini. Sistem aktivira novo storitev in ustrezno posodobi obračun stranke. Stranka prejme potrditev in informacijo o dodani storitvi.
- Prikaži stanje storitev - Možno je enostavno videti trenutno stanje in podrobnosti strankinih naročenih storitev. Sistem pridobi in prikaže pregled naročnikovih naročenih storitev. To vključuje informacije, kot so vrsta storitve, stanje aktivacije, preostala uporaba ali obdobje veljavnosti in vse povezane funkcije ali omejitve. Znotraj pregleda storitev lahko stranke dostopajo do podrobnejših informacij o uporabi svojih storitev. To lahko vključuje porabo podatkov, minute klicev, poslana besedilna sporočila ali druge ustrezne meritve uporabe.

23.3.2 Delno strukturiran opis

Navedena primera uporabe sta opisana v tabelah 23.1 in 23.2.

23.3.3 Diagram primerov uporabe

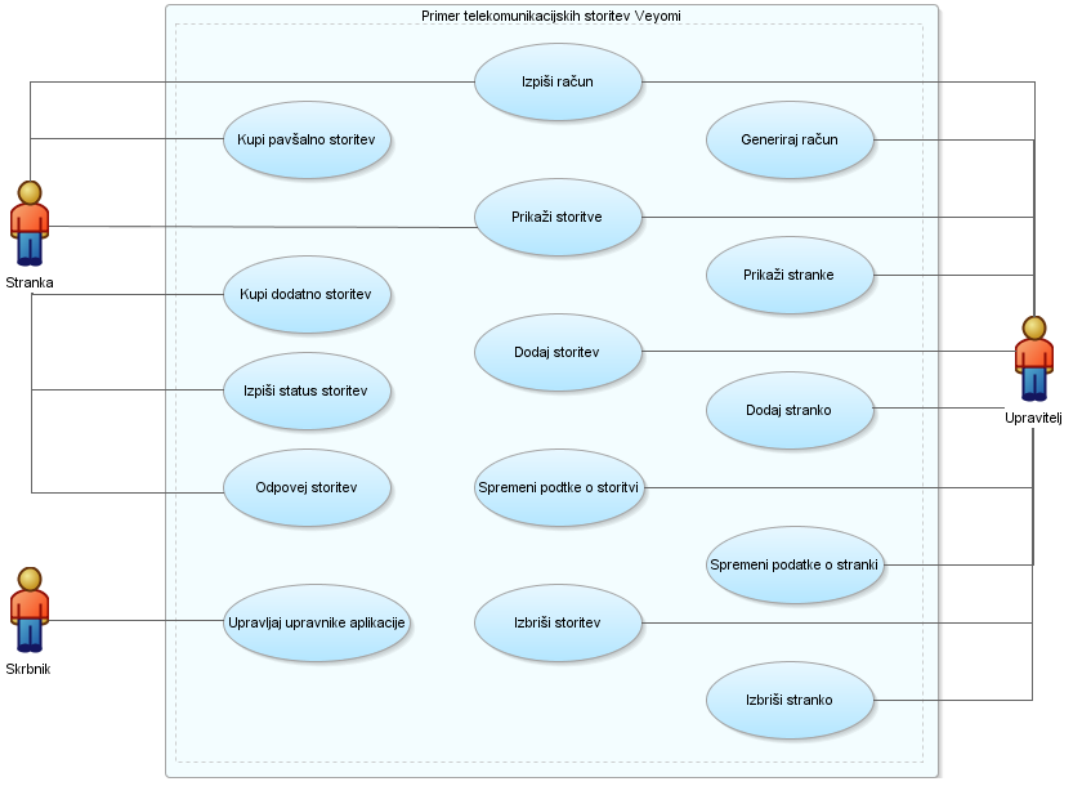
Sistem mora strankam zagotoviti nakup pavšalnih in dodatnih storitev, izpis statusa storitev, odpoved storitev in izpis računa

Upravitelj lahko dostopa do pregleda strank ter njihovega dodajanja, urejanja in brisanja. Poleg tega lahko ta skrbnik generira tudi račune.

Zadnja uporabniška vloga je skrbnik, ki ima eno osnovno funkcijo: upravljanje upravnikov aplikacije. Diagrami primerov uporabe so prikazani na Sliki 23.1.

Tabela 23.2: Opis primera uporabe: Prikaži stanje storitev.

Ključni pojem	**Vrednost**
ID:	*Ch23-02*
Title:	*Prikaži stanje storitev*
Opis:	*Stranka uporabi APEX-ovo aplikacijo, da pregleda stanje in porabo storitev.*
Glavni igralec:	*Stranka*
Predhodni pogoji:	Dostop do spletne aplikacije, veljavne poverilnice, spletna stran aplikacije je dostopna
Pogoji po zaključku:	-
Glavni del:	*Scenariji*
Uspešen scenarij:	1. *Stranka se prijavi v spletno aplikaciji* 2. *Stranka izbere zavihek "Stats".* 3. *S pomočjo padajočega menija izbere storitev. Prikaže se stolpčni graf za to storitev.*
Razširitve:	-
Pogostost uporabe:	*Stranka v poprečju enkrat mesečno pogleda stanje storitev.*
Status:	[zaključen]
Lastnik:	*Stranka*
Prioriteta:	*Visoka*



Slika 23.1: Diagram primerov uporabe.

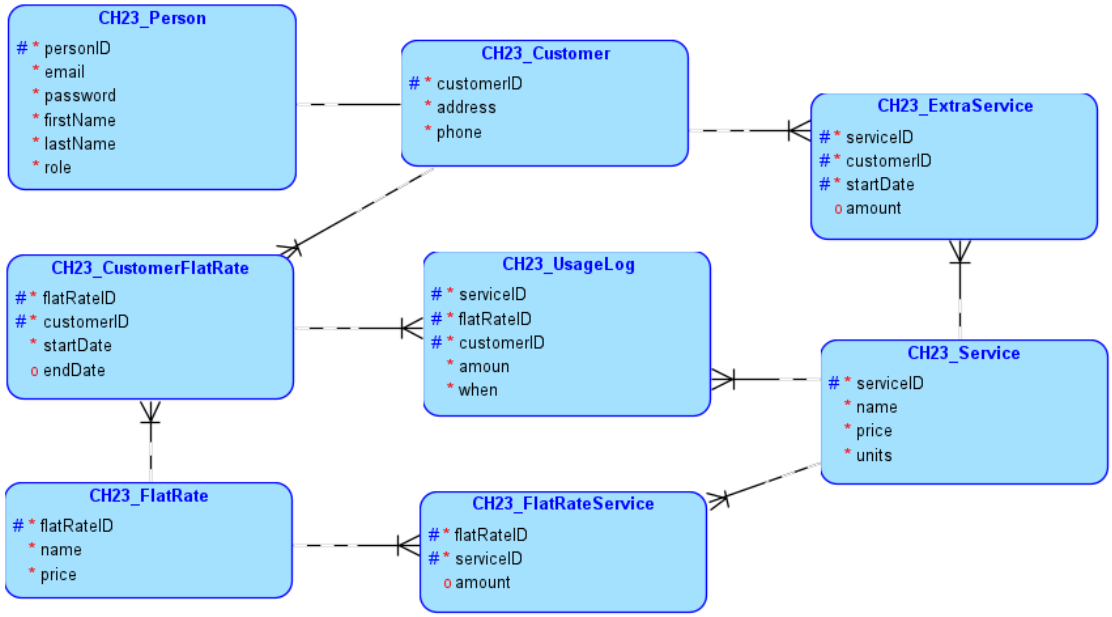
23.4 Model podatkov

23.4.1 Opis modela podatkov

Tabela **CH23_Person** shranjuje osebne podatke in vrsto uporabniške vloge. Ko je oseba stranka, ima kreiran zapis v tabeli **CH23_Customer**, kjer sta poleg ključev shranjena tudi njen naslov in telefonska številka. V tabeli **CH23_FlatRate** so shranjeni podatki o pavšalu, in sicer ime in cena. Podatki o posameznih storitvah, njihovem nazivu, ceni in ponujenih enotah so shranjeni v tabeli **CH23_Service**. Posamezne pavšalne storitve so shranjene v tabeli **CH23_FlatRateService**, kjer poleg ključev shranjujemo dodeljeno količino. Če stranka kupi pavšalno storitev, so informacije o tem shranjene v tabeli **CH23_CustomerFlatRate**, vključno z začetnim in končnim datumom. Porabljene enote iz pavšala se evidentirajo v tabeli **CH23_UsageLog**, kjer vemo, katera stranka je uporabila katero storitev v okviru katerega pavšala, skupaj z datumom in količino. Stranka lahko poleg pavšala dokupi dodatne storitve, ki so zapisane v tabeli **CH23_ExtraService**.

23.4.2 Logični model podatkov

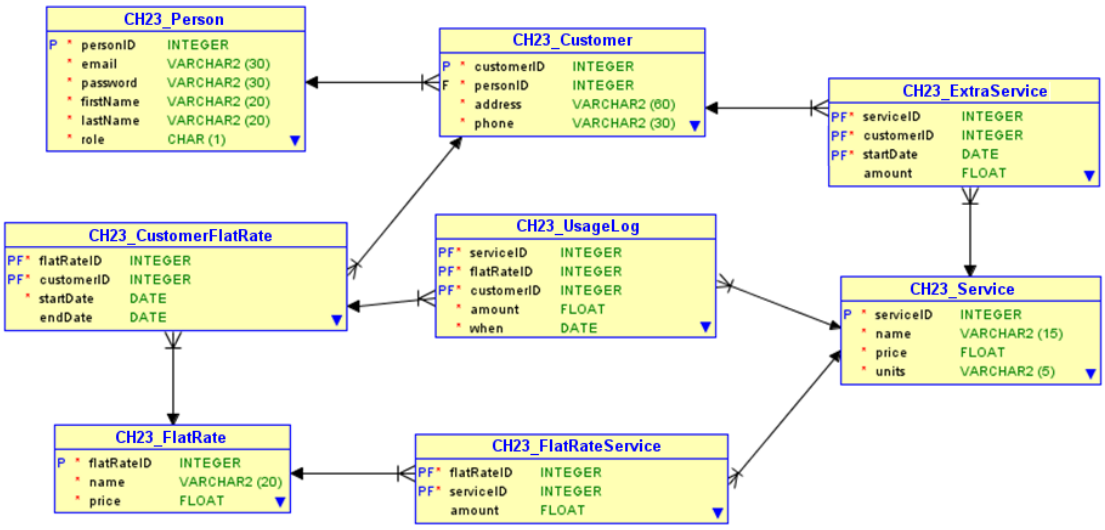
Logični model podatkov je prikazan na Sliki 23.2.



Slika 23.2: Logični model podatkov.

23.4.3 Relacijski model podatkov

Relacijski model podatkov je prikazan na Sliki 23.3.



Slika 23.3: Relacijski model podatkov.

23.5 Preverjanje pristnosti uporabnika in uporabniške vloge

V zgornjem opisu smo identificirali tri uporabniške vloge. Po prijavi uporabniku aplikacija omogoči dostop do vsebine glede na njegovo vlogo. Vendar je sam postopek prijave enak za vse uporabnike. APEX nam ponuja več načinov za upravljanje uporabnikov in njihovo avtentikacijo. Na naslednjem seznamu predstavljamo nekaj izbranih metod:

- račun baze podatkov,
- strežnik LDAP,
- račun Oracle APEX,
- prijava z računom družbenih omrežij,
- po meri.

V orodju Oracle Apex izberemo način avtentikacije - **Authentication Scheme** - na podstrani **Shared Components \ Authentication Schemes**. Na tej podstrani najdemo seznam obstoječih prijavnih shem, prav tako pa lahko tukaj ustvarimo novo prijavno shemo. Pri ustvarjanju nove sheme lahko izberemo enega od zgoraj navedenih načinov avtentikacije. Za potrebe naše aplikacije smo izbrali najbolj splošno metodo, **Custom**.

Glavni del avtentikacije **Custom** je tako imenovana funkcija avtentikacije. Vhod v to funkcijo je uporabniško ime in geslo, ki ju uporabnik vnese na strani za prijavo, izhod pa je vrednost **boolean**. Po pričakovanjih funkcija vrne **true**, če se dano ime in geslo ujemata z obstoječim uporabnikom, v nasprotnem primeru pa **false**. V izpisu programa (izvorni kodi) 23.1 lahko vidimo glavo takšne funkcije. Zato moramo pri ustvarjanju prijavne sheme **Custom** podati ime take funkcije. To pomeni, da mora taka funkcija že obstajati.

```
CREATE OR REPLACE FUNCTION authenticate_user
(p_username IN varchar2,
p_password IN varchar2)
RETURN boolean;
```

Izvorni program 23.1: Glava funkcije za preverjanje pristnosti.

Telo avtentikacijske funkcije realiziramo z uporabo jezika PL/SQL, kar pomeni, da se sami odločimo, kako bomo avtenticirali uporabnika. To je razlog, zakaj je metoda **Custom** najbolj splošna. Lahko na primer stopimo v stik z zunanjo avtentikacijsko storitvijo z REST API-jem, poiščemo uporabnika v lokalni bazi podatkov ali avtentikacijo prepustimo drugemu lokalnemu

procesu. V naši aplikaciji smo izbrali pogosto uporabljen postopek iskanja in preverjanja uporabnika v lokalni bazi podatkov.

V podatkovnem modelu, ki smo ga predstavili zgoraj, lahko vidimo tabelo CH23_Person. Stolpca **email** in **password** sta pomembna za preverjanje pristnosti. **Email** uporabnika služi tudi kot edinstveno prijavno ime. Stolpec **password** vsebuje uporabniško geslo, shranjeno v varni zgoščeni obliki. Opis varnega shranjevanja gesel presega obseg tega poglavja. Več o tem lahko bralec izve iz literature. Kljub temu je pred vnosom podatkov ali urejanjem stolpca **password** v tabeli **CH23_Person** potrebno geslo zgostiti. V ta namen smo ustvarili pomožno funkcijo **hash_password**, katere glavo lahko vidimo v izpisu programa (izvorni kodi) 23.2. Ta funkcija vzame uporabniško ime in geslo v obliki navadnega besedila in ustvari geslo v zaščiteni zgoščeni obliki, ki se lahko shrani v zbirko podatkov.

```
CREATE OR REPLACE FUNCTION hash_password
(p_username IN varchar2,
p_password IN varchar2)
RETURN varchar2;
```

Izvorni program 23.2: Glava pomožne funkcije hash_password.

Samodejno zgoščevanje gesel pri delu s tabelo **CH23_Person** smo rešili tako, da smo ustvarili prožilce za tabelo CH23_Person. Prožilci zgostijo geslo v navadnem besedilu z uporabo zgoraj opisane funkcije **hash_password**, preden vstavijo ali posodobijo stolpec **password**. Zagotovil bo, da bodo vsa gesla v tabeli **CH23_Person** varno shranjena – niti skrbnik baze podatkov nima dostopa do dejanskega gesla. V tabeli 23.3 lahko vidimo vzorec podatkov, shranjenih v tabeli **CH23_Person**. Zanimivo je, da imata lahko dva uporabnika, navedena v tabeli, enako geslo. Tega dejstva pa ne moremo ugotoviti na podlagi varno shranjenih gesel.

PERSONID	EMAIL	PASSWORD	ROLE
5	DUIS.A@OUTLOOK.CA	$2a$12$EfSjb2zSiKjqWCoZ47mQeOY6IPoHP/6LvECZxGEWjFZIZcXTF3UgGasd	'a'
11	LOBORTIS@OUTLOOK.EDU	$2a$12$IUqALTj3reLAoTdtC5SNde2.C5A.DOm..7pFw9UBnIsaf7gSNcZyapql	'c'
14	AC.FERMENTUM@AOL.NET	$2a$12$GuaO7Tjp.65d7NYLiEjSzeO6VWQfDz8eXja7dWubcjy77aZDAKRTutyq	'm'

Tabela 23.3: Podatki, shranjeni v tabeli CH23_Person.

Implementacija funkcije avtentikacije je zdaj preprosta. Pri tej funkciji najprej poiščemo v tabeli **CH23_Person** geslo uporabnika z danim uporabniškim imenom. Nato zgostimo geslo, ki ga je uporabnik vnesel s funkcijo **hash _password**. Če se obe zgoščeni vrednosti ujemata, je avtentikacija uporabnika uspešna in funkcija vrne **true**. Po drugi strani pa, če se razpršitve ne ujemajo ali v tabeli CH23_Person ni uporabnika z danim imenom, vrnemo false.

Slabost metode **Custom**, ki smo jo izbrali, je večja začetna težava pri izdelavi baze, znotraj katere je potrebno razviti pomožne funkcije in prožilce. Po drugi strani pa je prednost v tem, da mi, razvijalci aplikacij, nadziramo celoten postopek avtentikacije. Druga prednost je, da je uporabljena izvedba enostavno nadgradljiva. Če bi na primer želeli v prihodnosti za avtentikacijo uporabiti zunanjo storitev, bi morali spremeniti implementacijo avtentikacijske funkcije, tako da se bo povezala z zunanjo storitvijo namesto z lokalno bazo podatkov.

Druga pomembna naloga, povezana z upravljanjem uporabnikov, je upravljanje vlog uporabnikov. Medtem, ko je bil postopek prijave enak za vse uporabnike, smo morali omogočiti dostop do

različnih delov aplikacije glede na vlogo uporabnika pri upravljanju vlog. Upravljanje vlog je tesno povezano z upravljanjem uporabnikov. Najprej je treba v okolju Oracle APEX določiti vse možne uporabniške vloge. Upravljanje vlog najdemo na podstrani **Shared Components \ Application Access Control**. Na tej podstrani vidimo seznam obstoječih vlog, tu pa lahko ustvarimo tudi novo vlogo. Za vsako novo vlogo je potrebno določiti njen statični identifikator, ki ga bomo kasneje uporabili pri dodeljevanju vloge posameznim uporabnikom. V naši aplikaciji uporabljamo vloge z naslednjimi statičnimi identifikatorji:

- ADMIN,
- CUSTOMER,
- MANAGER.

Uporabniku dodelimo vlogo s kodo PL/SQL. V seznamu 23.3 lahko vidimo vzorec kode za dodelitev vloge **CUSTOMER** (statični identifikator vloge) uporabniku **LOBORTIS@OUTLOOK.EDU**. Za avtomatizacijo tega procesa smo uporabili drug nabor prožilcev za tabelo Person. Stolpec **Role** v tej tabeli lahko sprejme vrednosti iz množice (a, m, c), ki se ujemajo z začetnimi črkami zgoraj definiranih vlog. V telesu prožilca pred vstavljanjem ali posodabljanjem podatkov nastavimo ustrezno vlogo za danega uporabnika glede na vrednost stolpca Vloga z uporabo kode, podobne izvorni kodi 23.3.

```
APEX_ACL.REPLACE_USER_ROLES (
 p_application_id    => 151905,
p_user_name          => 'LOBORTIS@OUTLOOK.EDU',
 p_role_static_ids   => wwv_flow_t_varchar2('CUSTOMER') );
RETURN boolean;
```

Izvorni program 23.3: PL/SQL ukazi, ki povežejo uporabnika z določeno vlogo.

Zadnji korak implementacije uporabniških vlog v aplikaciji je prikaz različnih podstrani glede na vlogo prijavljenega uporabnika. Za vsako podstran, ki je namenjena le določeni uporabniški vlogi, je potrebno v nastavitvah **Page Designer** strani v razdelku **Security \ Authorization Scheme** izbrati uporabniško vlogo, ki ji je določena podstran namenjena. Ta korak zagotavlja, da uporabniki z drugimi vlogami ne morejo dostopati do te podstrani. Domača stran naše aplikacije je na primer specifična po tem, da za ogled ni potrebna prijava. To vedenje nastavimo tudi v razdelku **Security** z elementom **Authentication**, kjer izberemo možnost **Page Is Public**.

Zgoraj opisane nastavitve bodo zagotovile, da bo imel uporabnik dostop le do podstrani svoje vloge. Vendar bodo vsi uporabniki še vedno videli vse podstrani v navigacijskem meniju – tudi tiste, do katerih nimajo dostopa. Takšno obnašanje vsekakor ni zaželeno. Da aplikacija uporabnikom prikazuje samo tiste podstrani, do katerih imajo dostop, je treba prilagoditi nastavitve navigacijskega menija. To lahko nastavimo na podstrani **Shared Components \ Navigation and Search \ Navigation Menu**. Tukaj je treba v stolpcu **Authorization Scheme** za vsako postavko v meniju nastaviti uporabniško vlogo, ki ji bo prikazana določena postavka.

23.6 Uporabniški vmesniki

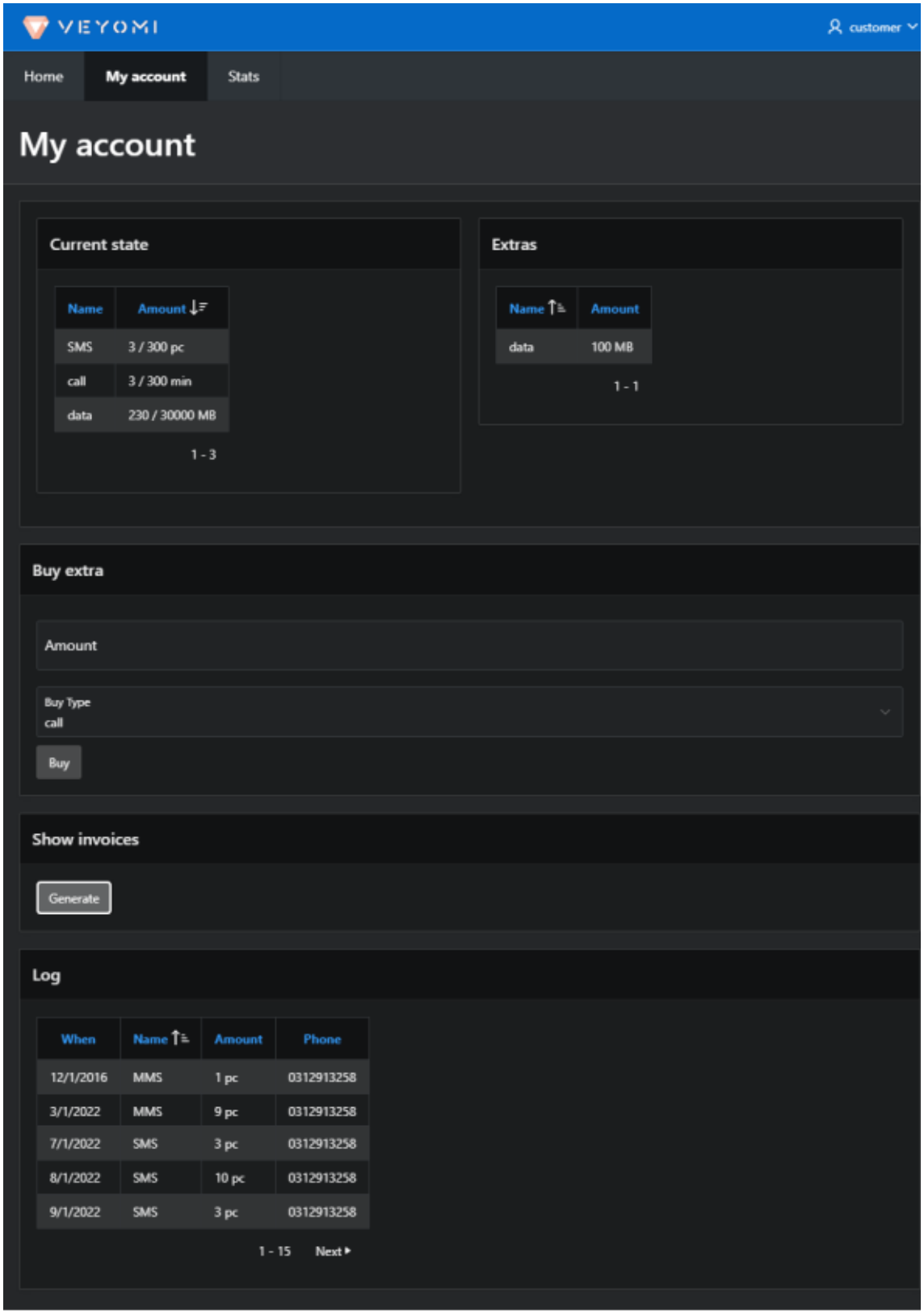
Zasnova aplikacije obsega več podstrani, ki bodo zagotavljale predlagane funlkcije. Primarna stran je domača stran, ki omogoča pregled ponujenih storitev in portfelja pavšalnih storitev. Na tej strani je možno predstaviti podjetje in storitve, ki jih stranka lahko kupi. Ta stran je dostopna brez prijave za vse obiskovalce. Glavna stran ponuja še eno pomembno funkcionalnost: dostop do administracije po prijavi. Z administracijo razumemo nabor podstrani, razdeljenih glede na vrsto računa, v katerega se prijavljate – stranka, upravitelj ali skrbnik. Za vsako od njih je torej potrebno ustvariti ločene podstrani, ki pokrivajo predlagane funkcionalnosti storitev posamezne vloge.

Stran za stranke omogoča pregled trenutnega stanja naročenih storitev in dodatnih storitev po posameznih poljih. Dodaten zavihek omogoča nakup posameznih storitev z izbiro vrste storitve v kombiniranem polju. Druga funkcionalnost omogoča prikazovanje in generiranje računov, kar

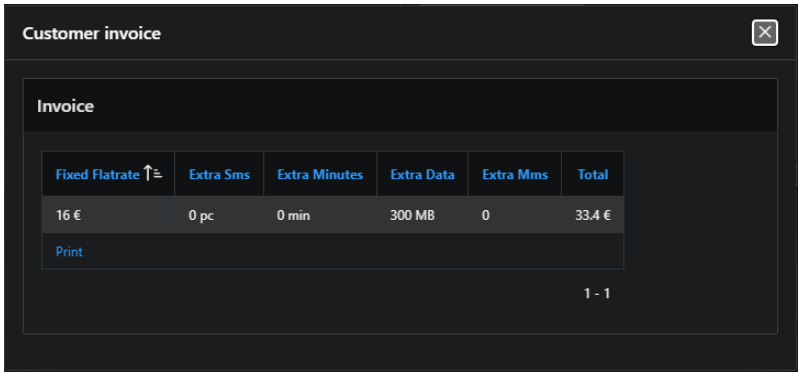
je mogoče rešiti s pojavnim oknom, v katerem se po odprtju prikaže račun v obliki poročila, ki omogoča konfiguracijo shranjevanja in tiskanja v PDF formatu. Poročilo za račun prikliče posebna skripta, ki povzema podatke za opravljeno storitev v določenem mesecu. Drug pomemben del strani za stranke je pregled uporabe storitve s podrobnostmi, kot so količina, vrsta storitve in čas. Ta tabela omogoča filtriranje in razvrščanje tega pregleda z uporabo specifikacije filtra in razvrščanje po posameznih stolpcih. Takšna funkcionalnost je lahko koristna za naročnika, ko želi na primer dobiti pregled nad številom poslanih SMS sporočil v določenem obdobju. Nazadnje je stranki zagotovljena funkcionalnost prikaza statistike v grafih v določeni mapi. Ta funkcionalnost se izvaja na ločeni strani, ki je z zavihkom pritrjena na strankino stran in je možno preklapljati med posameznimi funkcionalnostmi. Stran s statistiko vsebuje predstavitev grafične predstavitve logotipov. Taka predstavitev je primerna predstavitev za prikaz sprememb skozi čas. Enostavno lahko na primer prikažemo priklicane minute v posameznem mesecu, število poslanih SMS-ov ali količino prenesenih podatkov v naročenem programu.

23.6.1 Zasnova aplikacije

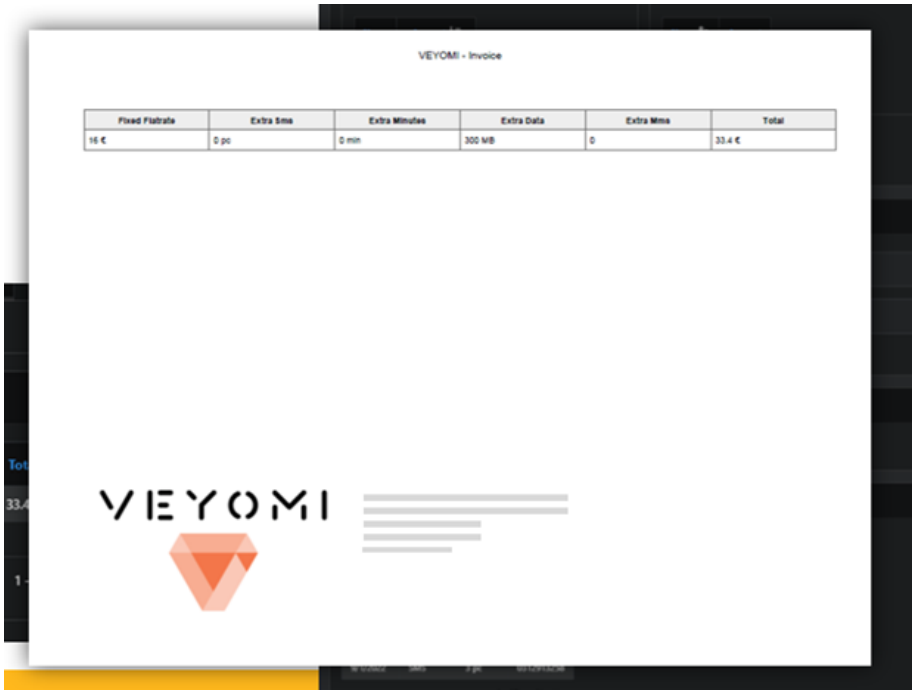
V tem delu bomo prikazali vmesnike za različne vloge, ki jih imajo uporabniki.



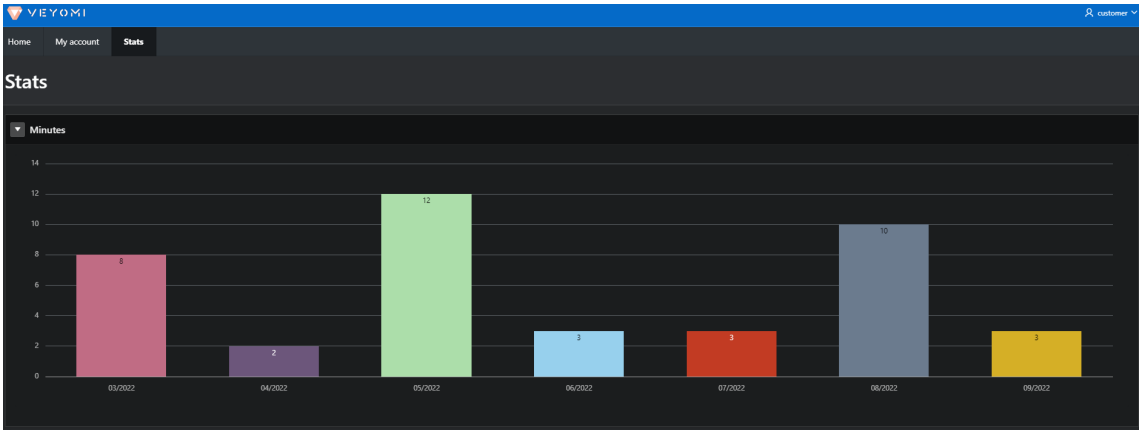
Slika 23.4: Stranka - pregledna plošča.



Slika 23.5: Stranka - pregledna plošča – modalno okno računa.



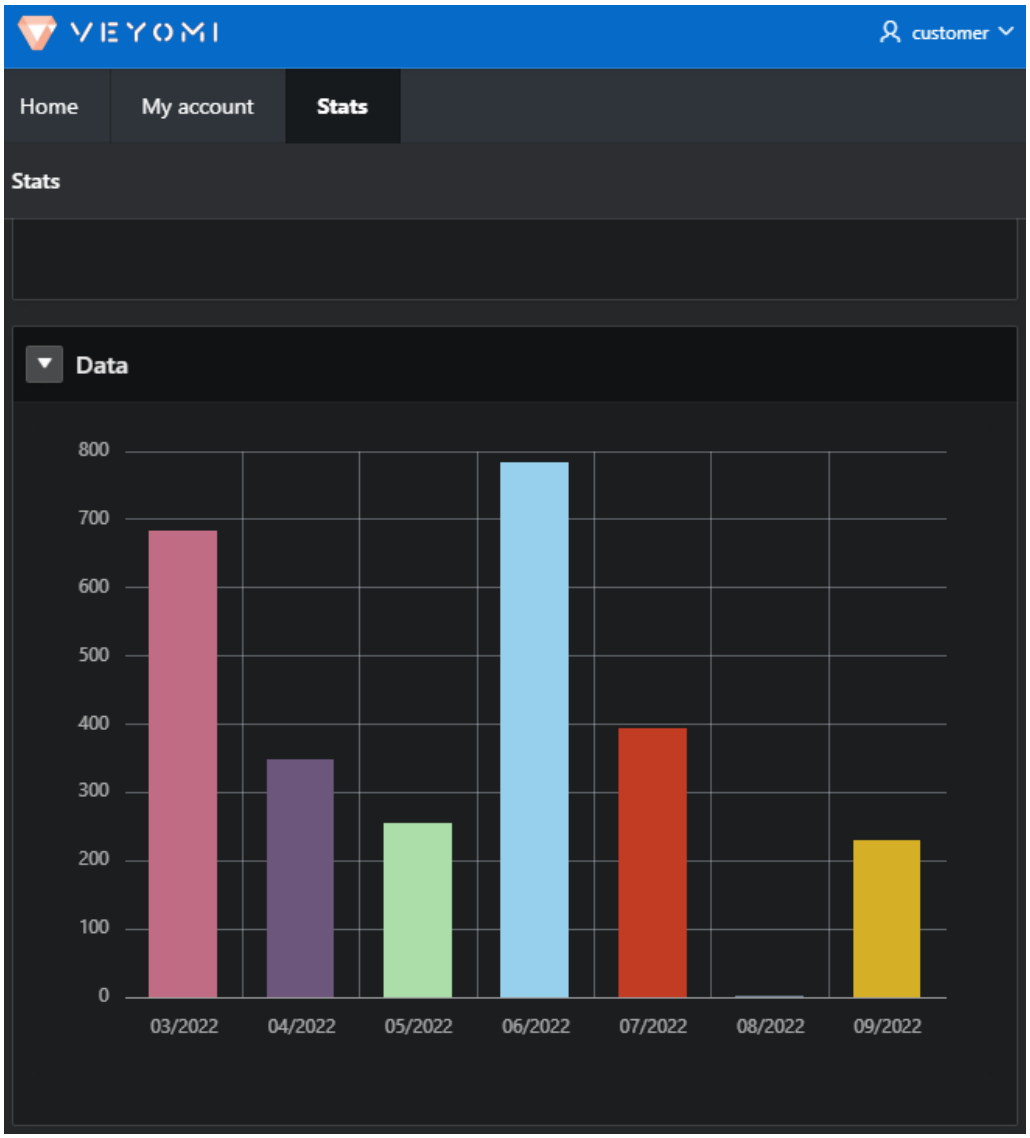
Slika 23.6: Stranka - pregledna plošča - račun v obliki PDF.



Slika 23.7: Stranka - pregledna plošča - statistika porabljenih minut.



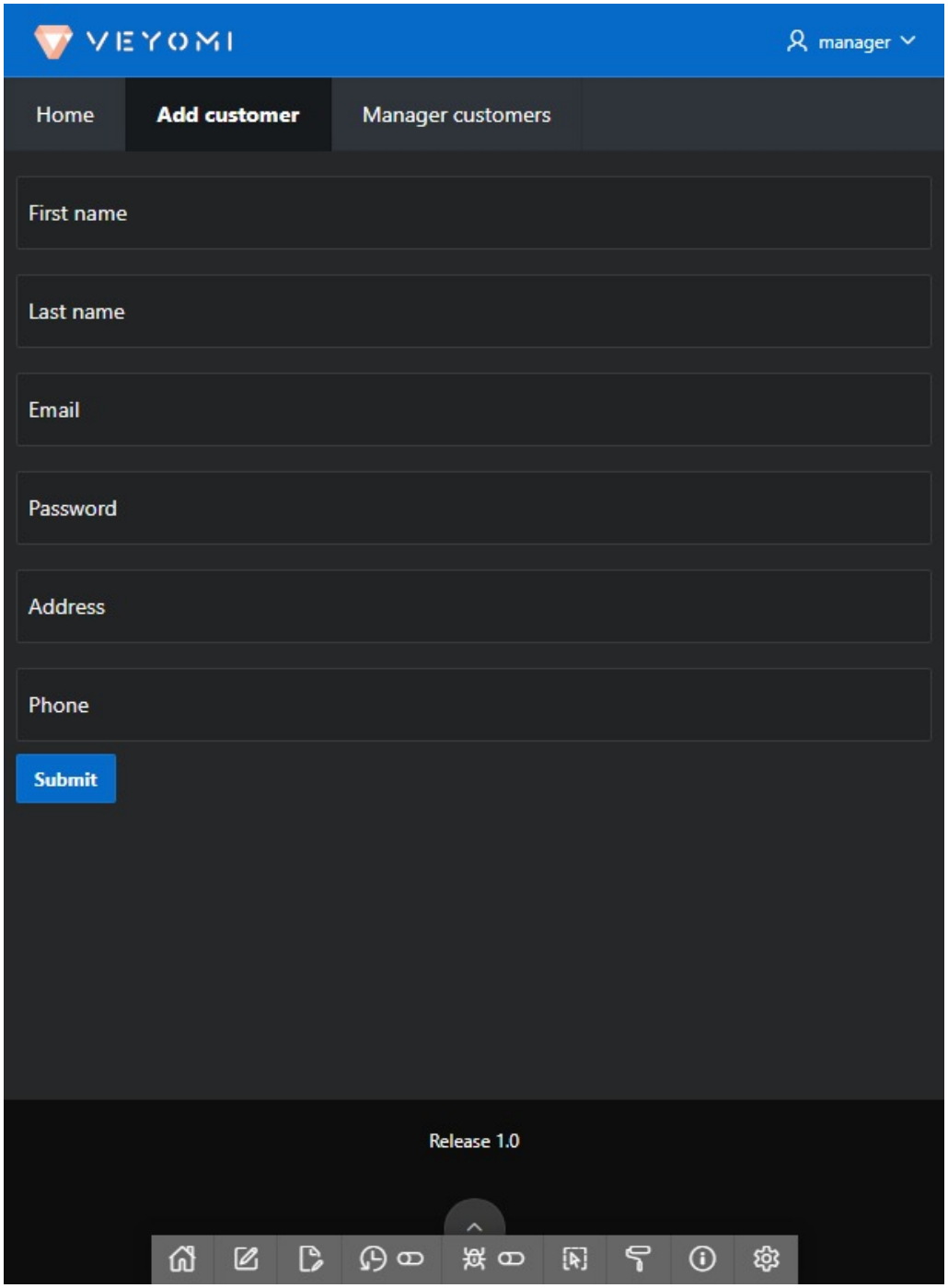
Slika 23.8: Stranka - pregledna plošča - statistika SMS.



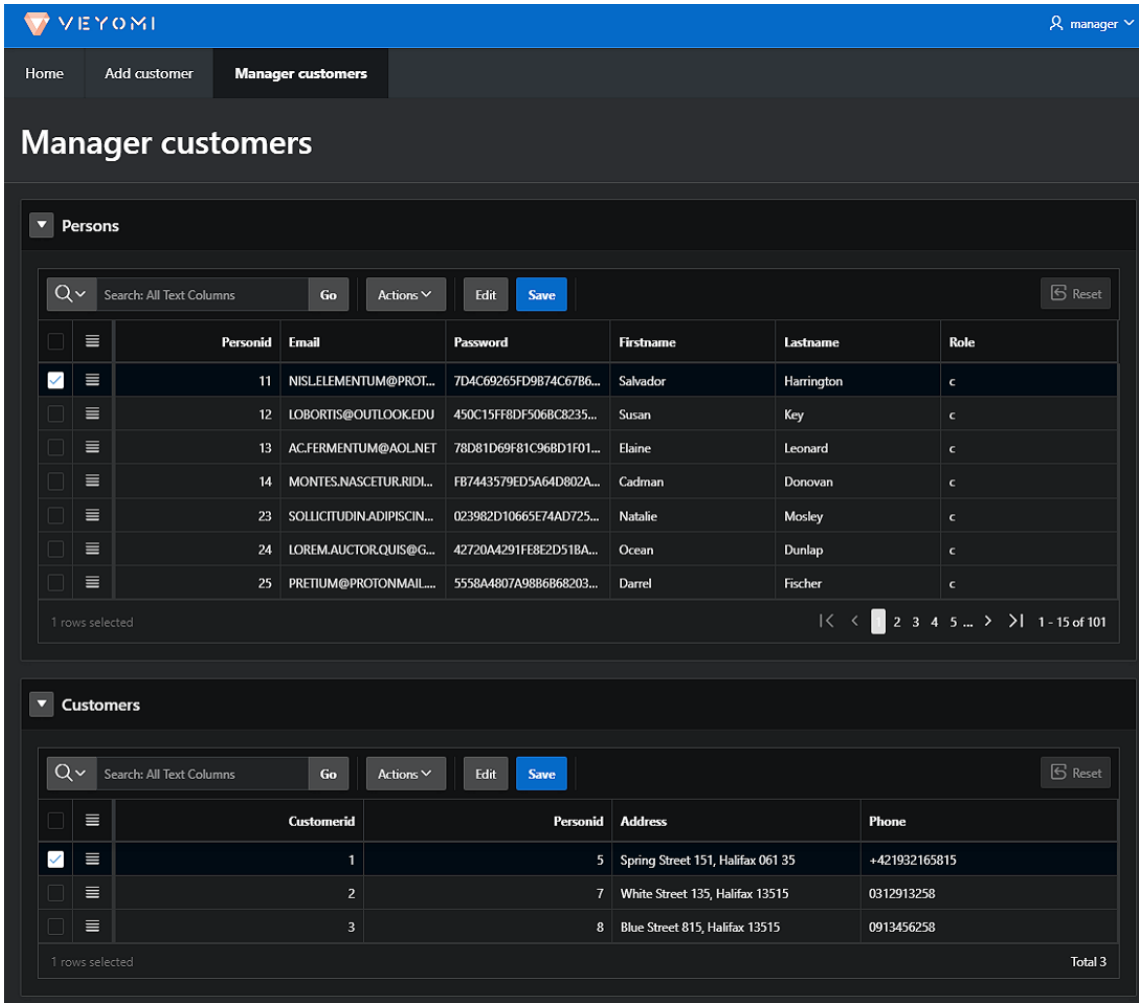
Slika 23.9: Stranka - pregledna plošča - statistika porabe prenosa podatkov.

Po prijavi upravitelja je funkcionalnost vodenja strank zagotovljena na ločenih zavihkih, ki so del strani upravitelja. Prva funkcionalnost je dodajanje nove stranke, kjer lahko upravitelj

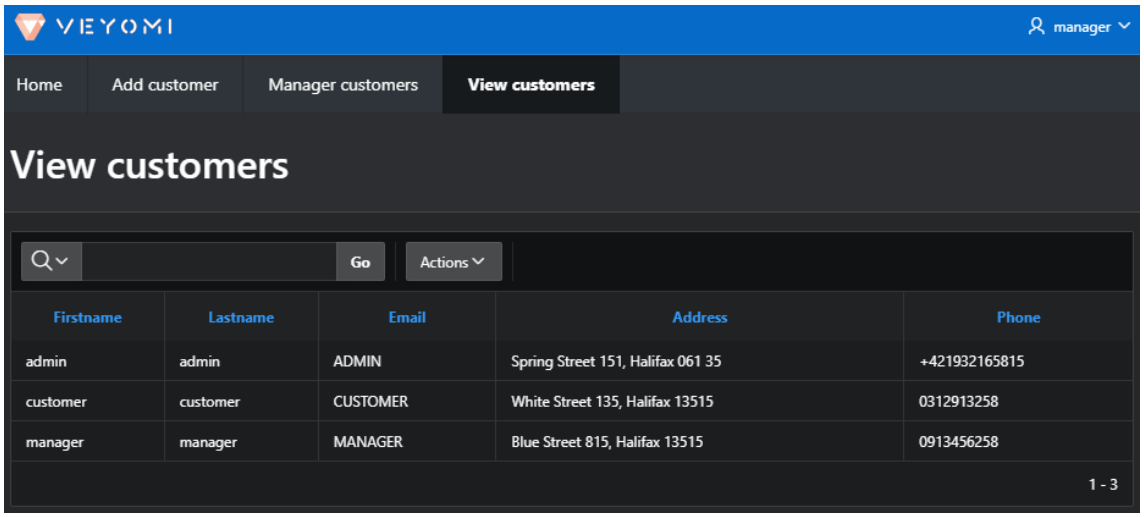
doda in ustvari nov račun stranke. Nato uporabimo komponento za upravljanje strank za urejanje informacij o strankah, ki so oblikovane v podatkovni mreži. Ta komponenta nudi tudi funkcionalnost preprostega poročanja ali izvoza podatkov o strankah v druge aplikacije tretjih oseb, če je potrebno. Nazadnje zavihek Manage customers prikazuje poenostavljeno obliko prikaza pregleda podatkov o strankah. Ta mreža je konfigurirana tako, da ne omogoča urejanja posameznih postavk, ima pa možnost ustvarjanja poročil in izvozov iz tega seznama.



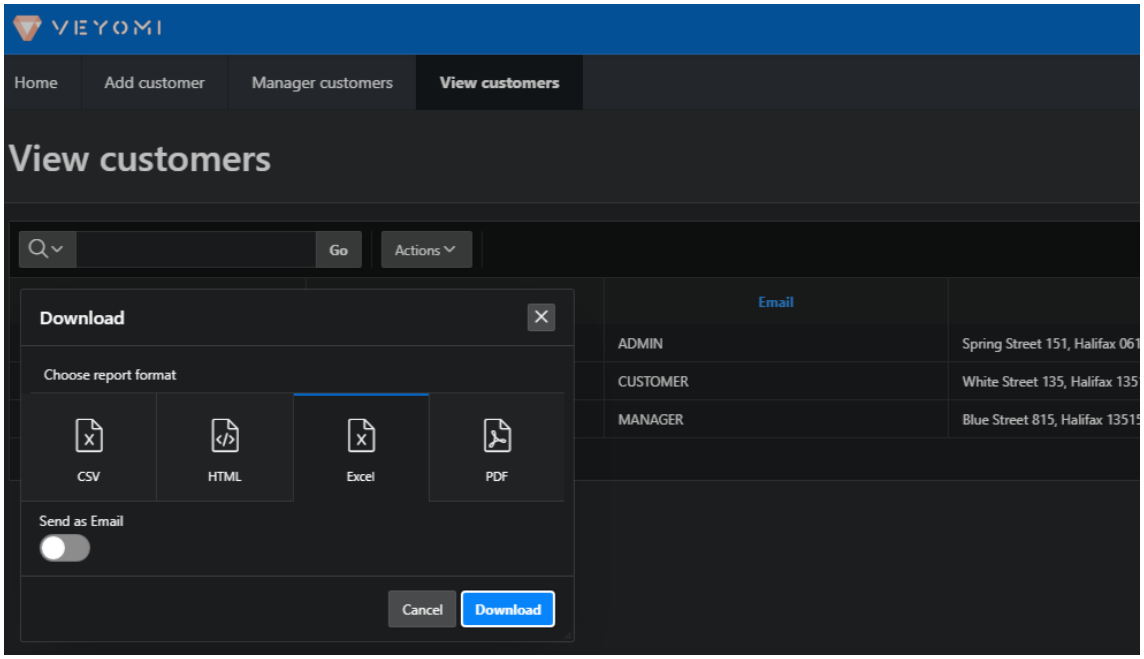
Slika 23.10: Upravitelj - pregledna plošča – dodajanje stranke.



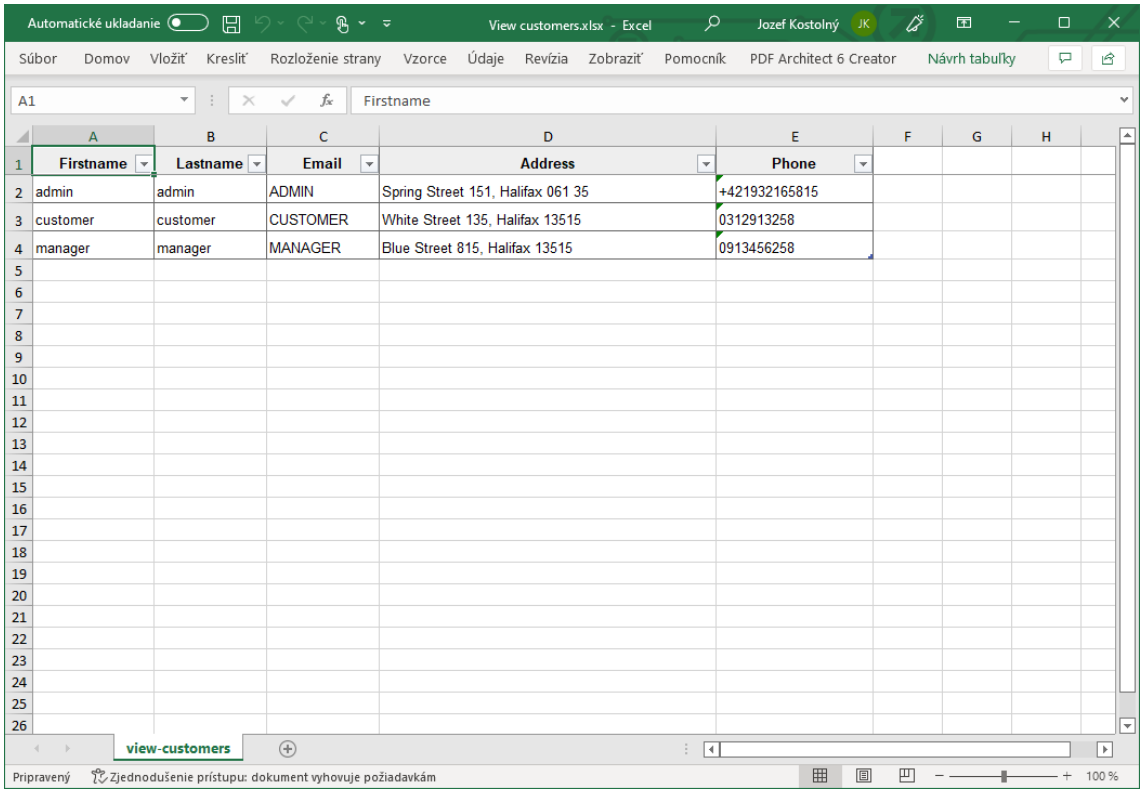
Slika 23.11: Upravitelj - pregledna plošča – upravljanje strank.



Slika 23.12: Upravitelj - pregledna plošča – pregled strank.

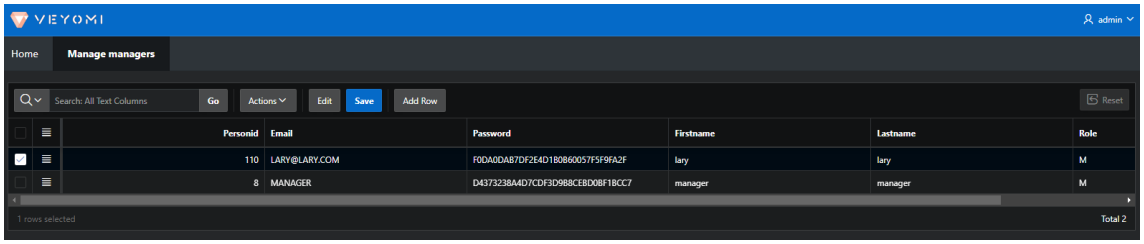


Slika 23.13: Upravitelj - pregledna plošča – izvoz podatkov o strankah.



Slika 23.14: Upravitelj - pregledna plošča – izvoz podatkov o strankah v preglednico.

Zadnja stran, ki jo je treba ustvariti, je stran za skrbnika aplikacije. Na tej strani nudimo storitev upravljanja upraviteljev v podatkovni mreži, ki jo je mogoče urejati, kar omogoča izvoz poročil in funkcijo filtra.

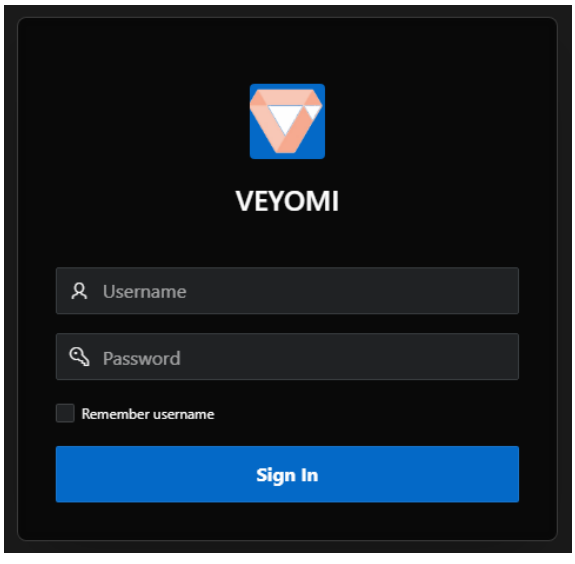


Slika 23.15: Skrbnik - pregledna plošča.

Pomemben del je izdelava prijavnega obrazca. V naši aplikaciji razmišljamo, da bi ta obrazec postavili na določeno stran, ki se vedno prikliče ob vstopu na varne strani, kjer se je potrebno čim prej avtentikirati.



Slika 23.16: Pristajalna stran.



Slika 23.17: Prijavna stran.

23.7 Skripti

```
CREATE SEQUENCE "CUSTOMER_SEQ"
MINVALUE 4 MAXVALUE 9999999999999999999999999999
INCREMENT BY 1
START WITH 4
NOCACHE NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL
/

CREATE OR REPLACE EDITIONABLE TRIGGER "BI_CUSTOMER"
BEFORE INSERT ON CH23_CUSTOMER
FOR EACH ROW
BEGIN
SELECT CUSTOMER_SEQ.nextval INTO :new.customerid
FROM dual;
END;
/

ALTER TRIGGER "BI_CUSTOMER" ENABLE;
/
```

Izvorni program 23.4: Sekvence in prožilci.

```
CREATE SEQUENCE "PERSON_SEQ"
MINVALUE 1 MAXVALUE 9999999999999999999999999999
INCREMENT BY 1
START WITH 1
NOCACHE NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL
/
CREATE OR REPLACE EDITIONABLE TRIGGER "BI_PERSON"
BEFORE INSERT ON CH23_PERSON
FOR EACH ROW
BEGIN
SELECT PERSON_SEQ.nextval INTO :new.personid
FROM dual;
:new.email := UPPER(:new.email);
:new.password := hash_password(UPPER(:new.email), :new.password);
END;
/

ALTER TRIGGER "BI_PERSON" ENABLE;
/

CREATE OR REPLACE EDITIONABLE TRIGGER "BU_PERSON"
BEFORE UPDATE ON CH23_PERSON
FOR EACH ROW
BEGIN
:new.email := UPPER(:new.email);
IF :new.password IS NOT NULL THEN
:new.password := hash_password(UPPER(:new.email), :new.password);
ELSE
```

```
:new.password := :old.password;
END IF;
END;
/
ALTER TRIGGER "BU_PERSON" ENABLE;
/
```

Izvorni program 23.5: Prožilec ob dodajanju osebe.

```
CREATE OR REPLACE FUNCTION AUTHENTICATE_USER
(p_username in varchar2,
p_password in varchar2)
 return boolean
IS
l_user_name          CH23_PERSON.email%type := UPPER(p_username);
l_password           CH23_PERSON.password%type;
l_hashed_password varchar2(1000);
l_count              number;
BEGIN
SELECT COUNT(*) INTO l_count
FROM CH23_PERSON
WHERE email = l_user_name;

IF l_count > 0 THEN
l_hashed_password := hash_password(l_user_name, p_password);
SELECT password INTO l_password
FROM CH23_PERSON
WHERE email = l_user_name;

IF l_hashed_password = l_password THEN
APEX_UTIL.SET_AUTHENTICATION_RESULT(0);
 return true;
ELSE
APEX_UTIL.SET_AUTHENTICATION_RESULT(4);
 return false;
END IF;

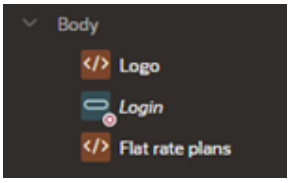
ELSE
APEX_UTIL.SET_AUTHENTICATION_RESULT(1);
 return false;
END IF;
APEX_UTIL.SET_AUTHENTICATION_RESULT(7);
 return false;
EXCEPTION
WHEN OTHERS THEN
APEX_UTIL.SET_AUTHENTICATION_RESULT(7);
APEX_UTIL.SET_CUSTOM_AUTH_STATUS(sqlerrm);
 return false;
END AUTHENTICATE_USER;
/
```

Izvorni program 23.6: Funkcija AUTHENTICATE_USER.

23.8 Ustvarjanje domače strani

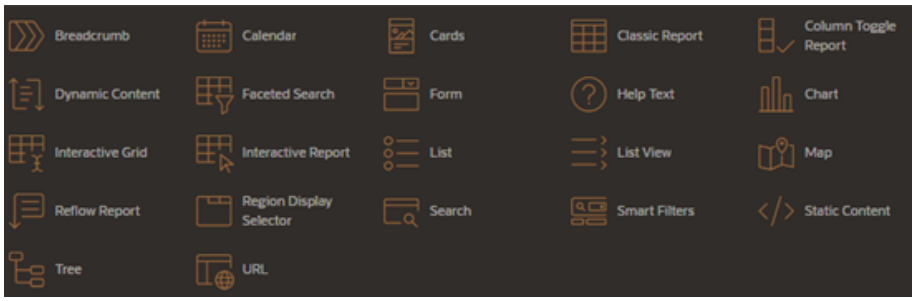
V APEXU-u lahko pri kreiranju strani izbirate med različnimi predlogami kot so prazna stran ali stran s funkcijami, kot so prijava, stran o strani in možnosti konfiguracije. Najprej morate vnesti osnovne podatke, kot sta številka in ime strani ter izbrati model strani, kot je normalno, modalno pogovorno okno ali predal. Izpolnite lahko tudi dodatne informacije, kot je nastavitev navigacijskega vnosa za stran (ali bo vnos na vrhu ali ugnezden v obstoječem vnosu) in izbira ikone, prikazane v meniju nadzorne plošče.

Po kliku na Ustvari stran se prikaže nadzorna plošča za urejanje informacij in vsebine strani. Glavna vsebina strani (Telo) je razdeljena na regije.



Slika 23.18: Regije v telesu strani.

Vsaka stran lahko vsebuje statični HTML ali vnaprej definirano vsebino.



Slika 23.19: Seznam možnih vsebin v regiji.

Naša regija **Login** vsebuje gumb za preusmeritev strani v obrazec za prijavo uporabnika. V lastnosti Behaviour smo definirali Action, ki je bilo v tej aplikaciji nastavljeno kot Redirect to Page in this Application. Nato nastavimo Target na vnaprej določeno stran za prijavo. Ker tega gumba ni treba prikazati, ko je uporabnik že prijavljen, moramo gumb skriti tako, da nastavimo Type as User is the Public User (uporabnik ni overjen). Končno smo ustvarili **Flat rate plans** kot območje statične vsebine, ki vsebuje kodo HTML, za katero lahko definiramo CSS.

```
11
12  </head>
13  <body>
14  <div id="container">
15      <div class="whole">
16      <div class="type">
17          <p>VEYOMI 100+</p>
18          </div>
19      <div class="plan">
20
21          <div class="header">
22              11<span>€</span>
23              <br>
24          </div>
25          <div class="content">
26              <ul>
27                  <li>100 minutes calls</li>
28                  <li>100 SMS/MMS</li>
29                  <li>10 GB data</li>
30              </ul>
31
32          </div>
33
34          <div class="price">
```

Slika 23.20: HTML v statični regiji strani.

23.9 Ustvarjanje strani stranke

To stran smo ustvarili po podobnih korakih kot pri ustvarjanju domače strani. Ko je stran ustvarjena, nadaljujemo z razvojem postavitve strankine strani, ki je razdeljena na več regij, na primer za prikaz trenutnega stanja storitev, nakup dodatkov, prikaz računov in prikaz dnevnikov.

Regijo **Current state** sestavljata dve podregiji, ki prikazujeta numerične podatke – trenutno ceno porabljenih enot pavšala in dodatkov. Za pridobitev teh podatkov se uporablja poizvedba SQL.

```
SELECT name, sum(u.amount) || ' ' || ' / ' || fs.amount || ' ' || s.units AS
    amount
FROM CH23_USAGELOG u
JOIN CH23_CUSTOMERFLATRATE cf ON (u.customerid = cf.customerid AND u.
    flatrateid = cf.flatrateid)
JOIN CH23_CUSTOMER c ON (cf.customerid = c.customerid)
JOIN CH23_PERSON p ON (c.personid = p.personid)
JOIN CH23_SERVICE s ON (u.serviceid = s.serviceid)
JOIN CH23_FLATRATESERVICE fs ON (fs.flatrateid = cf.flatrateid AND fs.serviceid = s
    .serviceid )
WHERE email = :APP_USER
AND EXTRACT(month from when) = EXTRACT(month from sysdate)
GROUP BY s.serviceid, s.name, s.units, fs.amount;
```

Izvorni program 23.7: Poizvedba z SQL - porabljene pavšalne storitve.

```
SELECT name, SUM(amount) || ' ' || units AS amount
FROM CH23_EXTRASERVICE JOIN CH23_CUSTOMER USING(customerid)
JOIN CH23_PERSON USING (personid)
JOIN CH23_SERVICE USING (serviceid)
WHERE email = :APP_USER
AND EXTRACT(month from startDate) = EXTRACT(month from sysdate)
GROUP BY serviceid, name, units;
```

Izvorni program 23.8: Poizvedba z SQL - Porabljene dodatne storitve.

Področje **Buy extra** vsebuje številsko polje, izbirni seznam in gumb. Možno je določiti znesek in izbrati, katero dodatno storitev želimo kupiti. Upravlja se z gumbom, ki izvaja postopek s kodo PL/SQL:

```
INSERT INTO CH23_EXTRASERVICE VALUES (
:BUY_TYPE,
(SELECT customerid FROM CH23_CUSTOMER JOIN CH23_PERSON USING (personid)
WHERE email = :APP_USER),sysdate,:BUY_AMOUNT);
```

Izvorni program 23.9: PL/SQL za dodajanje dodatnih storitev.

Področje **Show invoices** je sestavljeno iz gumba, ki se uporablja za generiranje računov. V lastnosti gumba *Behaviour* mo definirali *Action*, ki je bilo nastavljeno kot *Redirect to Page in this Application*. Nato smo nastavili *Target* na vnaprej določeno stran računa stranke, ki je bila nastavljena kot modalna. Na tej strani se s pomočjo poizvedbe SQL prikaže klasično poročilo, ki ga je možno tudi natisniti:

```
SELECT f.price || ' €' as "Fixed Flatrate ",
SUM(CASE e.serviceid WHEN 1 THEN NVL(e.amount, 0) ELSE 0 END)
|| ' ' || listagg (CASE s.serviceid WHEN 1 THEN s.units ELSE '' END, '') AS "Extra
    SMS",
SUM(CASE e.serviceid WHEN 2 THEN NVL(e.amount, 0) ELSE 0 END)
|| ' ' || listagg (CASE s.serviceid WHEN 2 THEN s.units ELSE '' END, '') AS "Extra
    Minutes",
SUM(CASE e.serviceid WHEN 3 THEN NVL(e.amount, 0) ELSE 0 END)
|| ' ' || listagg (CASE s.serviceid WHEN 3 THEN s.units ELSE '' END, '') AS "Extra
    Data",
SUM(CASE e.serviceid WHEN 4 THEN NVL(e.amount, 0) ELSE 0 END)
|| ' ' || listagg (CASE s.serviceid WHEN 4 THEN s.units ELSE '' END, '') AS "Extra
    MMS",
(f.price
+ SUM(CASE e.serviceid WHEN 1 THEN NVL(e.amount, 0) * s.price ELSE 0 END)
+ SUM(CASE e.serviceid WHEN 2 THEN NVL(e.amount, 0) * s.price ELSE 0 END)
+ SUM(CASE e.serviceid WHEN 3 THEN NVL(e.amount, 0) * s.price ELSE 0 END)
+ SUM(CASE e.serviceid WHEN 4 THEN NVL(e.amount, 0) * s.price ELSE 0 END)
|| ' €') AS "Total"
FROM CH23_CUSTOMER c
JOIN CH23_PERSON p ON (p.personid = c.personid)
JOIN CH23_CUSTOMERFLATRATE cf ON (cf.customerid = c.customerid)
JOIN CH23_FLATRATE f ON (f.flatrateid = cf. flatrateid )
JOIN CH23_FLATRATESERVICE fs ON (fs.flatrateid = f.flatrateid)
JOIN CH23_SERVICE s ON (s.serviceid = fs.serviceid)
LEFT JOIN CH23_EXTRASERVICE e ON (e.customerid = cf.customerid)
WHERE p.email = :APP_USER
AND (e. startdate IS NULL OR
(EXTRACT(month from e.startdate) = EXTRACT(month from sysdate)
AND EXTRACT(year from e.startdate) = EXTRACT(year from sysdate)))
GROUP BY f.price;
```

Izvorni program 23.10: Poizvedba v SQL za poročilo - račun.

Regija **Log** vsebuje interaktivno poročilo za prikaz dnevnika uporabe z uporabo poizvedbe SQL:

```
SELECT name, phone, amount || ' ' || units AS amount, when
FROM CH23_USAGELOG JOIN CH23_CUSTOMER USING (customerid)
JOIN CH23_PERSON USING (personid)
JOIN CH23_SERVICE USING (serviceid)
WHERE email = :APP_USER
ORDER BY when DESC;
```

Izvorni program 23.11: Poizvedba z SQL – Dnevnik porabe.

Zavihek **Stats** je nova stran, sestavljena iz treh območij, kot so minute, SMS in podatki. Tip regije je nastavljen na *Static content*, vključno s komponentami *Chart*, ki prikazujejo grafikone. Njegov izvor je definiran s poizvedbo SQL. Privzeto je barva stolpcev grafikona nastavljena na enotno. Lahko se prilagodi v poizvedbi SQL:

```
SELECT TO_CHAR(when, 'MM/YYYY') AS "Month",
SUM(amount) AS Amount,
CASE WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '01'
THEN '#F8B195'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '02'
THEN '#F67280'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '03'
THEN '#C06C84'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '04'
THEN '#6C567B'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '05'
THEN '#ACDEAA'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '06'
THEN '#97D0ED'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '07'
THEN '#C23B23'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '08'
THEN '#6B7B8E'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '09'
THEN '#D5AF26'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '10'
THEN '#9AB5C1'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '11'
THEN '#74698C'
WHEN SUBSTR(TO_CHAR(when, 'MM/YYYY'), 0, 2) = '12'
THEN '#C1867B'
ELSE 'black'
END AS MY_COLOR
FROM CH23_USAGELOG
JOIN CH23_CUSTOMER USING (customerid)
JOIN CH23_PERSON USING (personid)
JOIN CH23_SERVICE USING (serviceid)
WHERE email = :APP_USER AND serviceid = 2
GROUP BY TO_CHAR(when, 'MM/YYYY')
ORDER BY 1;
```

Izvorni program 23.12: Poizvedba z SQL - Statistika.

23.10 Ustvarjanje upraviteljske strani

Na strani upravitelja smo uporabili podobne metode in komponente za ustvarjanje podstrani. Vsaka podstran je ustvarjena kot nova stran, na primer Add customer, Manage customers in View customers.

Stran *Add customer* vsebuje preprost obrazec, ki je ustvarjen iz komponent. Postopek dodajanja nove stranke se izvede s kodo PL/SQL:

```
INSERT INTO CH23_PERSON (EMAIL, PASSWORD, FIRSTNAME, LASTNAME,
    ROLE)
VALUES (:CUSTOMER_EMAIL, :CUSTOMER_PASSWORD, :
    CUSTOMER_FIRSTNAME,
```

```
:CUSTOMER_LASTNAME, 'C');
```

Izvorni program 23.13: PL/SQL code for inserting a person.

Stran *Manage customers* je kreirana z dvema interaktivnima poročiloma z možnostjo urejanja: Manage persons in Manage current customers. Interaktivni poročili sta povezani z urejanjem in shranjevanjem sprememb. Tip procesa je nastavljen na *Interactive Grid – Automatic Row Processing (DML)*.

Stran *View customers* vsebuje interaktivno poročilo brez možnosti urejanje podatkov. Poročilo temelji na poizvedbi z SQL:

```
SELECT firstname, lastname, email, address, phone
FROM CH23_PERSON
JOIN CH23_CUSTOMER USING (personid);
```

Izvorni program 23.14: Poizvedba z SQL - Stranke (Customers).

23.11 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- skripti za kreiranje, brisanje in polnenje tabel
- skripti za brisanje tabel
- izvožena pakirana aplikacija
- podatki v preglednici
- video vodič, ki kaže postopek generiranja aplikacije

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter23 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

23.11.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo. Pakirana aplikacija je testirana in se bo izvajala v novem delovnem prostoru, če so izpolnjene naslednje zahteve:

- dodajte uporabnika APEX pred zagonom aplikacije. V delovnem prostoru za razvoj in testiranje se pomaknite do Shared Components > Application Access Control > Add User Role Assignment; vnesite uporabnika APEX in temu uporabniku nastavite vloge Administrator, Contributer in Reader. V produkcijskem okolju je obvezno posvetovanje z usposobljenim osebjem pred uvedbo aplikacije.

Če katerakoli od navedenih zahtev ni izpolnjena, se bo aplikacija sesula. Takrat je potrebno počistiti piškotke v spletnem brskalniku (primer za Firefox: Settings > Cookies and Site Data > Manage Data).

23.11.2 Video vodiči

Video vodič natančno prikazuje razvoj aplikacije.

23.12 Vprašanja

1. Katera podstran se uporablja za ustvarjanje nove prijavne sheme?
2. Poimenujte vsaj dve metodi preverjanja pristnosti za upravljanje uporabnikov.
3. Skozi celotno izvedbo so se v stavkih Select sklicevali na različna polja na straneh. Kako lahko prepoznate te predmete?

23.13 Odgovori

1. V orodju Oracle APEX izberemo način avtentikacije - Authentication Scheme - na podstrani Shared Components > Authentication Schemes.
2. Obstaja več metod, ki jih je mogoče uporabiti za preverjanje pristnosti uporabnika, kot so strežnik LDAP, račun baze podatkov, račun Oracle APEX itd.
3. Postavke v stavkih Select so navedene z uporabo dvopičja na začetku imena postavke.



24. Najem vozila

A. ANGEIOPLASTIS, G. MYLLIS, A. TSIMPIRIS, D. VARSAMIS; PREVOD R. LESKOVAR

24.1 Poslovni vidik primera

Najem vozila (ang. rent-a-car) je lahko priročen način za raziskovanje novih destinacij, vendar pri tem obstaja nekaj potencialnih pasti, ki se jih morata ponudnik najemna in najemnik zavedati. Najem vozila je bolj podoben rezervaciji hotelske sobe kot rezervaciji letalske vozovnice. Pri najemu avtomobila obstajata dve vrsti plačil oz. stroškov: ob prevzemu vozila in predplačilo.

Za najemnike, ki to počno prvič, je običajno bolje rezervirati vozilo tako, da izberejo plačilo najema po vrnitvi. To najemniku omogoča večjo prilagodljivost (v primeru, da mora spremeniti svoje potovalne načrte). Za najemodajalca pa to pomeni, da na nek način kreditira najemnika. Pri tej vrsti najema običajno ni dodatnih plačil oz. stroškov v primeru preklica rezervacije zaradi nepredvidenih okoliščin.

Da bi se najmemniki izognili dodatnim stroškom, je pomembno, da skrbno pregledajp najemno pogodbo in morebitne dodatne pristojbine, ki lahko nastanejo (npr. zavarovanje ali dodatne voznine do mesta prevzema). Preden z vozilom najemnik zapusti prevzemno mesto ga prodajno opozori, da skrbno pregleda vozilo in takoj sporoči kakršno koli poškodbo. Takp se lahko najemodajalec in najemnik vnaprej izogneta konflktom ob vrnitvi vozila.

Z razumevanjem procesa najema in morebitnih pristojbin si najemnik lahko zagotovi prijetno izkušnjo brez stresa in resnično uživa v svobodi in prilagodljivosti, najemodajalec pa si zagotovi, ki jo lahko najem avtomobila nudi med potovanjem.

24.2 Definicija problema

Najem vozila se morda zdi zastrašujoč zaradi množice pravil in pogojev. Vendar ti obstajajo z utemeljenimi razlogi. Novo vozilo trenutno stane približno 36,000 EUR. Pri najemu pa za časovno omejeno uporabo najemnik plačuje le majhen del skupne vrednosti vozila. Podjetje, ki nudi najem, mora v življenjski dobi vozila v splošnem doseči ustrezno stopnjo dobička.

Ključnega pomena pri najemu vozila je ustrezno zavarovanje za primera nesreče ali nepredvidenih okoliščin. Vendar pa je velika večina najemov zaključena brez kaakršnihkoli zavarovalnih dogodkov. Kljub temu je zelo pomembno, da natančno preberete najemno pogodbo, razumete pogoje poslovanja ter vozilo pregledate za kakršno koli škodo, preden jo sprejmete, da se izognete morebitnim težavam.

Z upoštevanjem teh smernic ima najemnik lahko varno in prijetno izkušnjo najema brez

nepričakovanih presenečenj. In ne pozabite, pravila in pogoji so vzpostavljeni, da zaščitijo vas in podjetje za najem avtomobilov, zato je vedno najbolje, da jih upoštevate.

24.3 Primeri uporabe

24.3.1 Opis primera uporabe

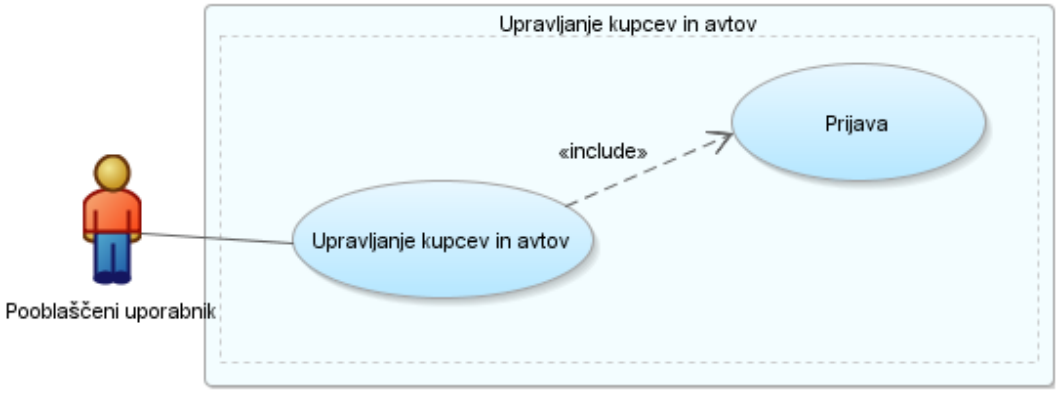
Cilj tega projekta je razviti funkcionalno aplikacijo, ki simulira delovanje rent-a-car poslovalnice. Aplikacija bo uporabljala platformo Oracle APEX za upravljanje in shranjevanje podrobnih informacij o strankah, avtomobilih in najemu. Aplikacija je zasnovana tako, da je uporabniku prijazna za lastnika izposojevalnice avtomobilov, saj mu omogoča upravljanje vseh potrebnih informacij, vključno z najemi, razpoložljivimi avtomobili za najem in podrobnostmi o strankah. Uporabniki bodo lahko najeli tako, da bodo izbrali avtomobil na podlagi meril, kot so znamka, model, letnik in vrsta goriva, prikazane pa bodo vse potrebne podrobnosti za dokončanje najema. Poleg tega bo aplikacija prikazala podatke, kot so trenutni najemi, razpoložljiva vozila za najem in podrobne podatke o strankah. Trenutni najemi bodo uporabniku prikazani v tabelah, odvisno od tega, v katerem delu aplikacije se uporabnik nahaja. Na splošno bo ta aplikacija zagotovila učinkovit in organiziran sistem za upravljanje poslovalnice za najem vozil, s podrobnimi informacijami, ki so na voljo s klikom na gumb. Uporabniku prijazen vmesnik in celovita orodja za upravljanje podatkov bodo poskrbela za preprostejše in učinkovitejše vodenje rent-a-car poslovalnice.

24.3.2 Delno strukturiran opis

Namen tega projekta je razviti uporabniku prijazno aplikacijo, ki uporablja Oracle APEX za upravljanje podrobnih informacij o strankah, avtomobilih in najemih za najem avtomobilov. Aplikacija bo uporabnikom omogočala najeme na podlagi posebnih meril, prikaz potrebnih informacij o najemu in zagotavljala celovita orodja za upravljanje podatkov za prikaz trenutnih najemov, razpoložljivih avtomobilov za najem in podrobnosti o strankah. Z zagotavljanjem učinkovitega in organiziranega sistema za upravljanje rent-a-car poslovalnice bo ta aplikacija omogočila preprostejše in učinkovitejše vodenje rent-a-car poslovalnice. Opis polstrukture je podan v tabeli 24.1.

24.3.3 Diagram primera uporabe

Opisani primeri uporabe so prikazani na sliki 24.1.



Slika 24.1: Diagram primera uporabe.

Tabela 24.1: Opis primera uporabe: upravljanje kupcev in avtov

Ključni pojem	**Vrednost**
ID:	*Ch24-01*
Naslov:	*Primer uporabe prikazuje dostopanje, dodajanje in spreminjanje podatkov o avtomobilih, strankah in rezervacijah. Pri rezervacijiah je možno izbrati datum najema in izračunati stroške najema.*
Opis:	*Uporabnik bo uporabil aplikacijo v APEX-u za ogled avtomobilov, ki pripadajo podjetju ter vseh strank, ki so predhodno opravile rezervacijo. Tako avtomobile kot kupce opisuje niz atributov. Poleg tega bo imel uporabnik možnost ustvarjanja, posodabljanja in brisanja podatkov za obe kategoriji, kar omogoča učinkovito upravljanje sistema za najem avtomobila. Poleg tega lahko uporabnik ustvari novo rezervacijo za najem avtomobila z dodajanjem avtomobila, stranke in datumov. Sistem na podlagi izbranih datumov izračuna celotno ceno rezervacije in ustrezno posodobi rezultate. Končno poročilo predstavlja podrobnosti rezervacije, vključno z avtomobilom, stranko, datumi najema in skupnimi stroški.*
Glavni igralec:	*Skrbnik sistema*
Predhodni pogoji:	Uporabnik ima vlogo skrbnika aplikacije.
Pogoji po zaključku:	Po posodobitvi avtomobila in stranke uporabnik lahko kreira rezervacijo. Ko je zaključena rezervacija se izvede izračun skupnih strroškov najema.
Glavni del	Scenarij
Uspešen scenarij:	*1. Odprite brskalnik in se prijavite v aplikacijo za izposojo avtomobilov* *2. Na meniju ali navigacijski strani izberite **Cars** ali **costumers*** *3. Za dodajanje avtomobila kliknite gumb "Create" in podobno za dodajanje stranke* *4. Za spreminjanje avtomobila ali stranke kliknite ikono peresa v želeni vrstici* *5. Za dodajanje ali spreminjanje podatkov kliknite gumb "Apply Changes", za brisanje pa gumb "Delete"* *6. Za izposojo na meniju ali navigacijski strani kliknite **rentcar**.* *7. Za novo izposojo kliknit izberite **create**.* *8. Za rezerviranje avtomobila kliknite na **carid** ain izberite razpoložljivi avto* *9. Izberite najemnika tako, da kliknete na **custid*** *10. Določite začetni datum rezervacije s klikom na **rent date*** *11. Določite končni datum rezervacije s klikom na **return date*** *12. Ko najemnik vrne avto, se izračunajo skupni stroški najema glede na trajanje in morebitne poškodbe. Če najemnik vrne poškodovan avto, se obračuna popravilo in to predstavlja za stranko dodatne stroške.* *13. Končni stroški so predloženi stranki, prav tako tudi detajna specifikacija v poročilu o najemu*
Razširitve:	-
Pogostost uporabe:	*število najemov je odvisno od sezone, letno poprečje pa je 5 dnevno*
Status:	Zaključen
Lastnik:	Uporabnik
Prioriteta:	*Zmerna*

24.4 Model podatkov

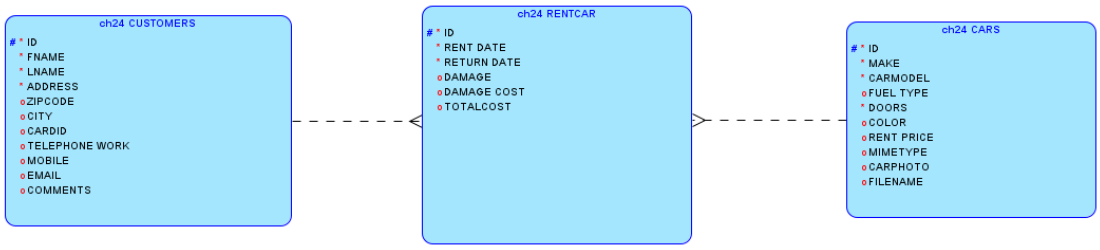
24.4.1 Opis modela podatkov

Šolski projekt uporablja tri entitete: avtomobil, stranka in najem avtomobila. Entitete vsebujejo vse atribute, ki so potrebni za vodenje procesa najema avtomobila z aplikacijo v APEX-u.

1. Entiteta Cars (avtomobili)
   - **id** identifikacijska številka avta in primarni ključ
   - **make** proizvajalec avtomobila
   - **carmodel** odel,
   - **fuel** vrsta goriva (bencin, dizel, LNG ali elektrika)
   - **doors** število vrat
   - **color** barva avta
   - **rent** cena za dan najema
   - **carphoto** fotografija
2. Entiteta Customers (stranke)
   - **id** identifikacijska številka in primarni ključ
   - **fname** ime
   - **lname** priimek
   - **address** naslov
   - **zipcode** poštna številka
   - **city** mesto
   - **cardid** številka vozniškega dovoljenja
   - **telephone-work** službena telefonska številka
   - **mobile** mobilna številka
   - **email** elektronska pošta
   - **comments** komentarji glede na prejšnje najeme
3. Entiteta CarRental (najemi)
   - **id** identifikacijska številka in primarni ključ
   - **carid id** identifikacijska številka avtomobila
   - **custid id** identifikacijska številka stranke
   - **rent date** datum prevzema
   - **return date** datuma vračila
   - **damage** opis škode ob vračilu
   - **damage cost** stroški škode
   - **total cost** skupni stroški najema

24.4.2 Logični model podatkov

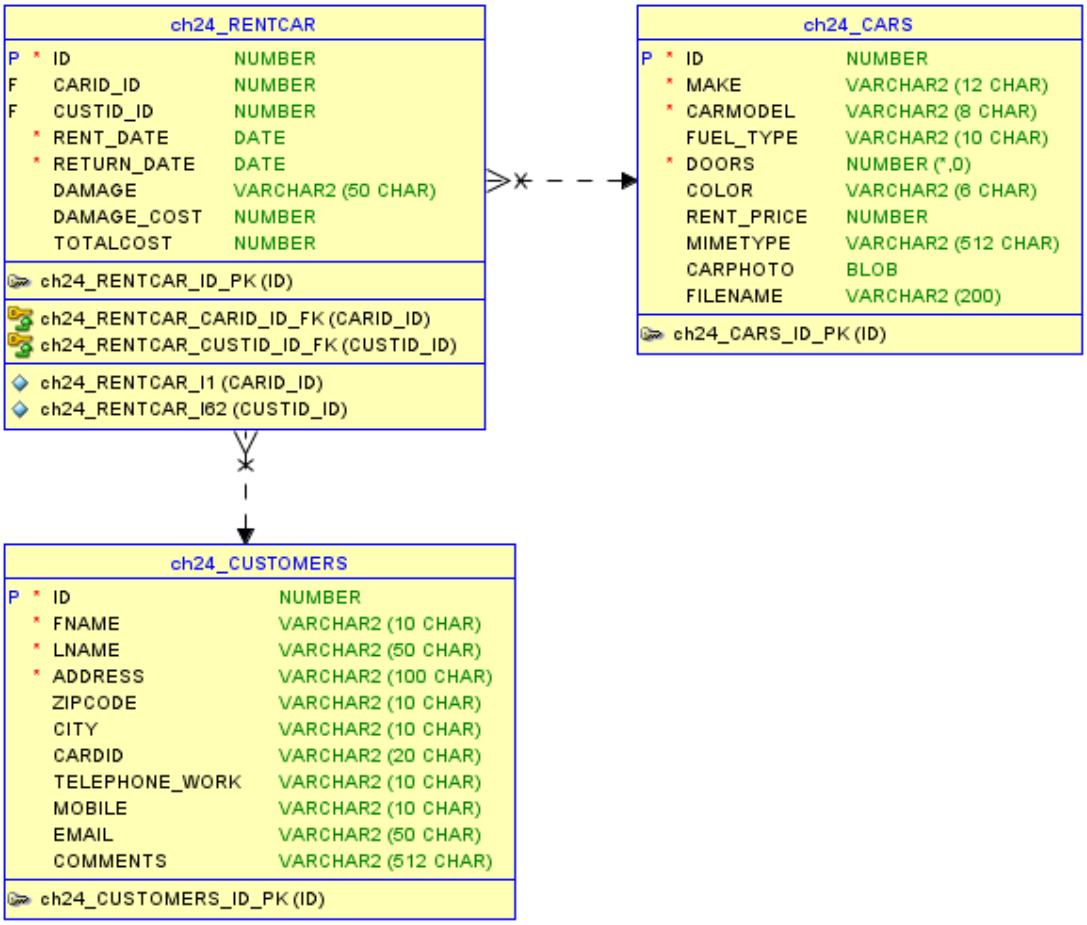
Logični model podatkov je prikazan na sliki 24.2.



Slika 24.2: Logični model podatkov za projekt “Car Rental Project”

24.4.3 Relacijski model podatkov

Relacijski model podatkov je prikazan na Sliki 24.3.



Slika 24.3: Relacijski model podatkov za projekt “Car Rental Project”

24.4.4 Skript v SQL

Tabele CH24_CARS, CH24_CUSTOMERS in CH24_RENTCAR za to aplikacijo kreiramo takole::

```
drop table  ch24_cars;
drop table  ch24_customers;
drop table  ch24_rentcar;
-- create tables
create table ch24_cars (
    id                  number generated by default on null as identity
                        constraint ch24_cars_id_pk primary key,
    make                varchar2(12 char) not null,
    carmodel            varchar2(8 char) not null,
    fuel_type           varchar2(10 char),
    doors               integer not null,
    color               varchar2(6 char),
    rent_price          number default '100',
    carphoto            varchar2(512 char)
```

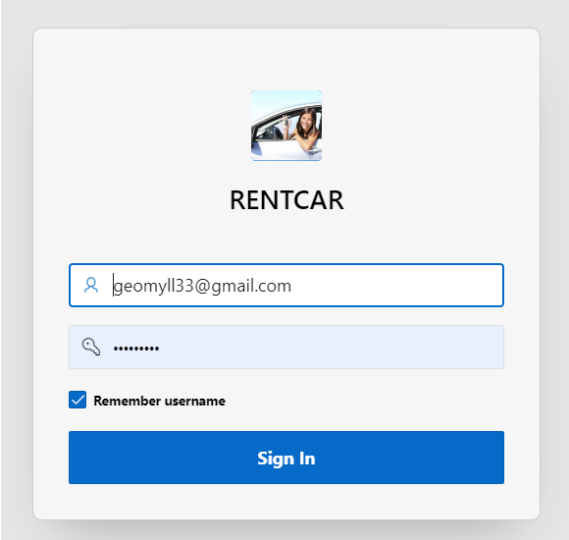
```
)
;

create table ch24_customers (
    id                  number generated by default on null as identity
                        constraint ch24_customers_id_pk primary key,
    fname               varchar2(10 char) not null,
    lname               varchar2(50 char) not null,
    address             varchar2(100 char) not null,
    zipcode             varchar2(10 char),
    city                varchar2(10 char) default 'SERRES',
    cardid              varchar2(20 char),
    telephone_work      varchar2(10 char),
    mobile              varchar2(10 char),
    email               varchar2(50 char),
    comments            varchar2(512 char)
)
;
create table ch24_rentcar (
    id                  number generated by default on null as identity
                        constraint ch24_rentcar_id_pk primary key,
    carid_id            number
                        constraint ch24_rentcar_carid_id_fk
                        references ch24_cars on delete cascade,
    custid_id           number
                        constraint ch24_rentcar_custid_id_fk
                        references ch24_customers on delete cascade,
    rent_date           date not null,
    return_date         date not null,
    damage              varchar2(50 char),
    damage_cost         number,
    totalcost           number
)
;

-- table index
create index ch24_rentcar_i1 on ch24_rentcar (carid_id);
create index ch24_rentcar_i62 on ch24_rentcar (custid_id);
```

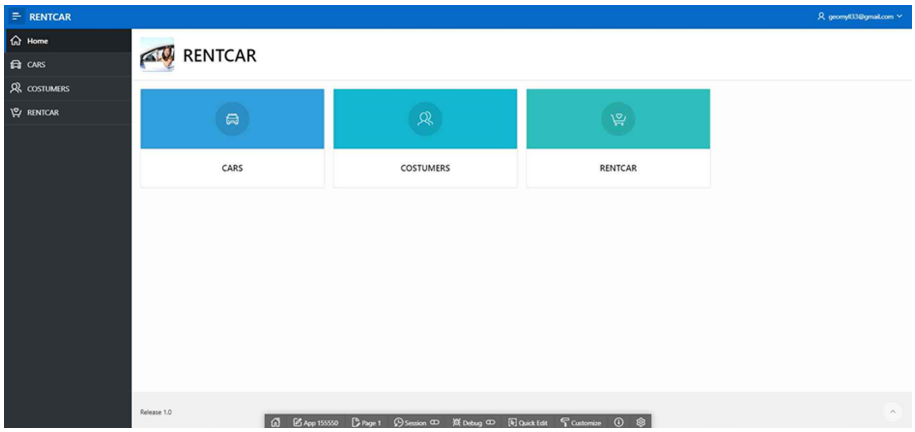
24.5 Uporabniški vmesniki

Uporabnik, ki je lastnik podjetja za najem avtomobilov, mora za dostop do aplikacije vnesti svoje poverilnice. Aplikacija uporablja preverjanje pristnosti računov APEX. Poverilnice lastnika so enake poverilnicam delovnega prostora, prikazanih na sliki 24.4.



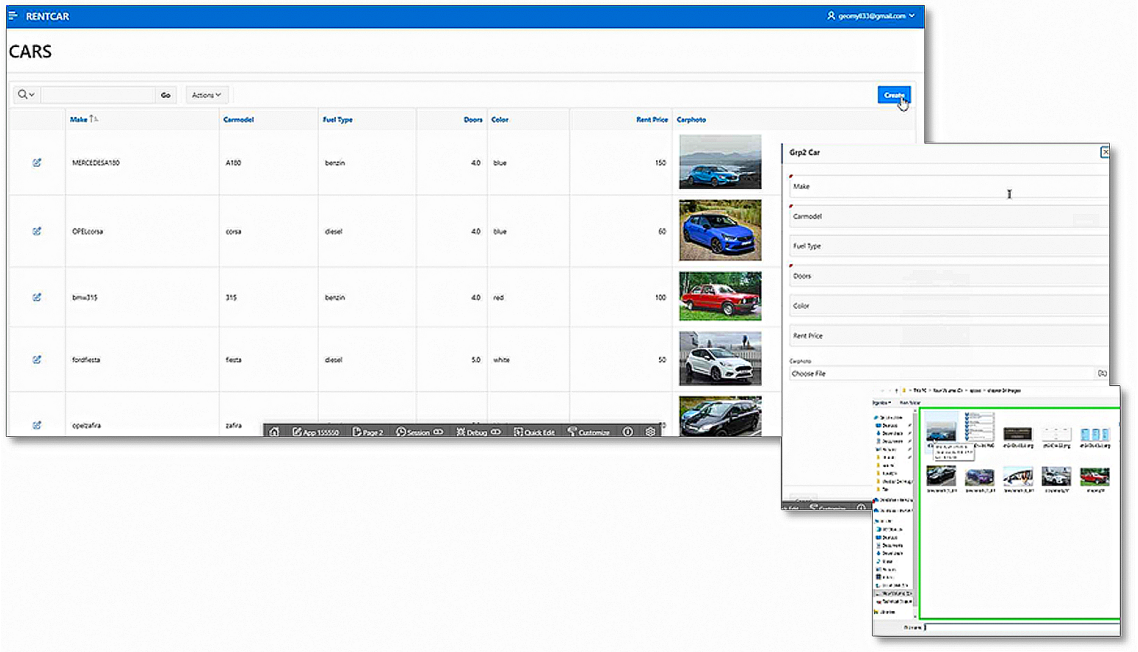
Slika 24.4: Prijava v aplikacijo.

Aplikacija simulira delovanje rent-a-car poslovalnice in vključuje upravljanje strank, avtomobilov in najema. Zagotavlja podrobne informacije o teh entitetah z uporabo prijazne spletne aplikacije, ki temelji na platformi Oracle APEX, kot je prikazano na sliki 24.5.



Slika 24.5: Domača stran aplikacije.

Obrazec na sliki 24.6 je namenjen upravljanju podatkov o avtomobilih v poslovalnici za najem vozil. Uporabnik, ki je lastnik poslovalnice, lahko shrani in upravlja vse potrebne podatke o avtomobilih, kot so znamka, model, letnik, vrsta goriva, ter naloži fotografijo avtomobila, ki prikazuje njegovo stanje.

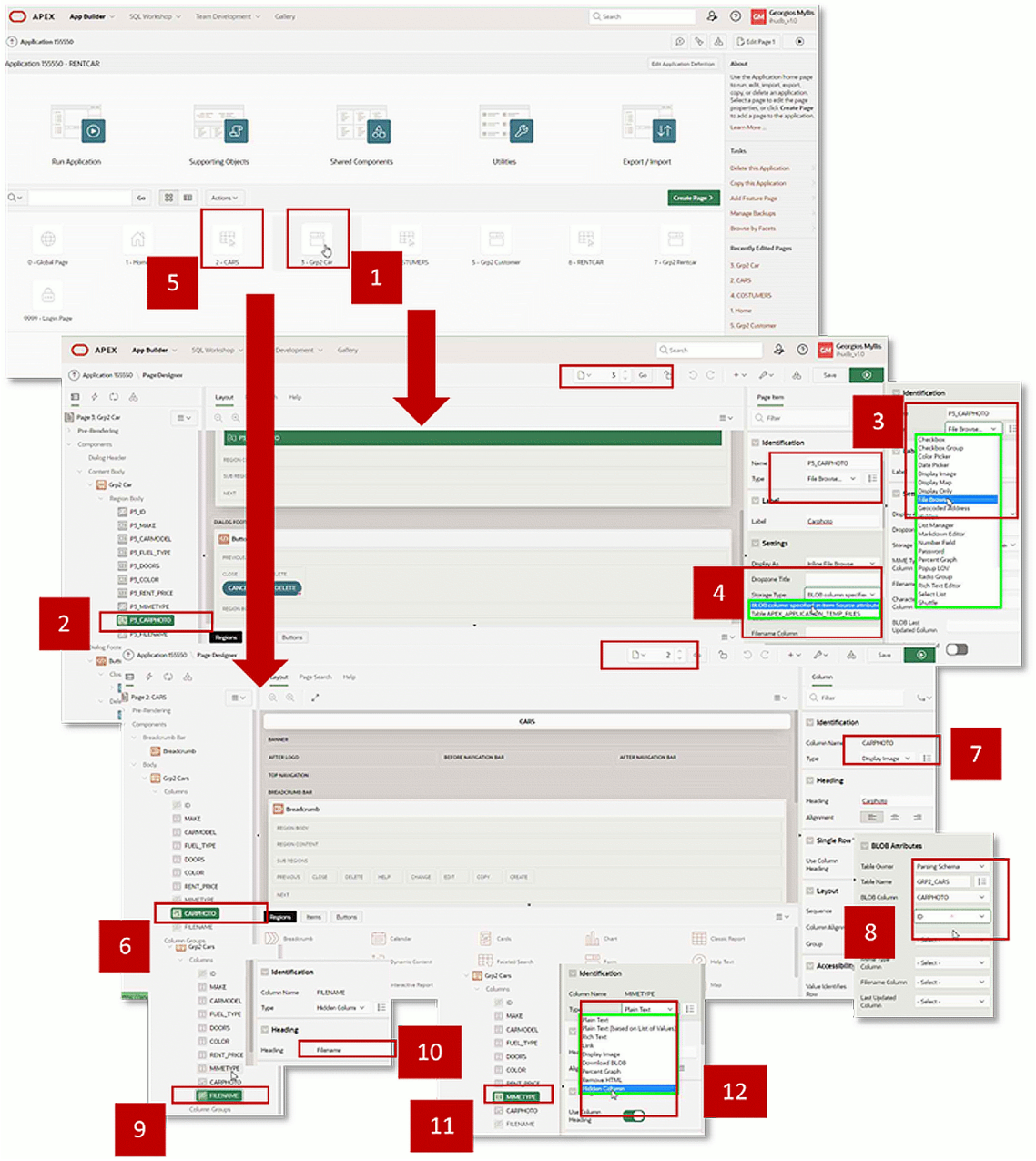


Slika 24.6: Obrazec za avtomobile (Cars)

Na sliki 24.7 je razloženo je, kako dodamo fotografijo. Da bi to dosegli, potrebujemo dva dodatna stolpca za shranjevanje informacij v tabeli: enega za tip MIME in enega za ime datoteke. Stolpec, v katerem bo shranjena fotografija mora imeti tip podatka BLOB. V našem primeru je ime stolpca **CARPHOTO**.

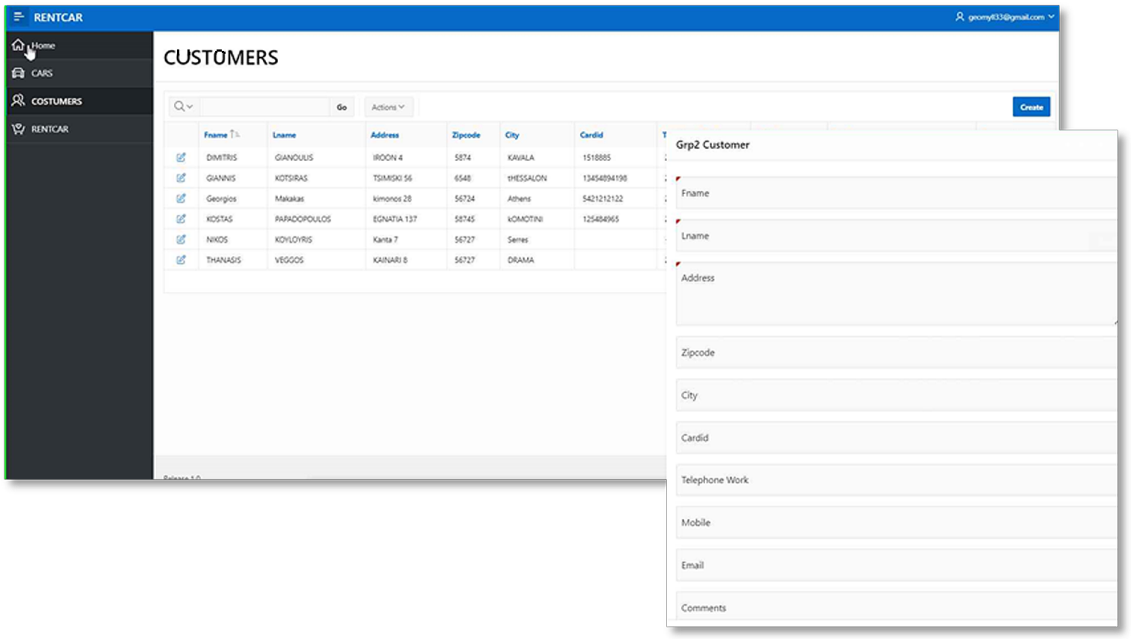
V oblikovalcu strani aplikacije je potrebno generirati poročilo za tabelo CH24_CAR in konfigurirati stolpec **CARPHOTO** kot brskalnik datotek. Nato spremenite nastavitve vrste pomnilnika na “BLOB column specifier in item source attributes”.

Nato odprite obrazec za tabelo ch24_cars in nastavite stolpec **CARPHOTO** kot Display image. Konfigurirajte BLOB tako, da povežete tabelo **ch24_cars**, stolpec **CARPHOTO** in primarni ključ **ID**. Stolpca **MIMETYPE** in **FILENAME** morata biti skrita.



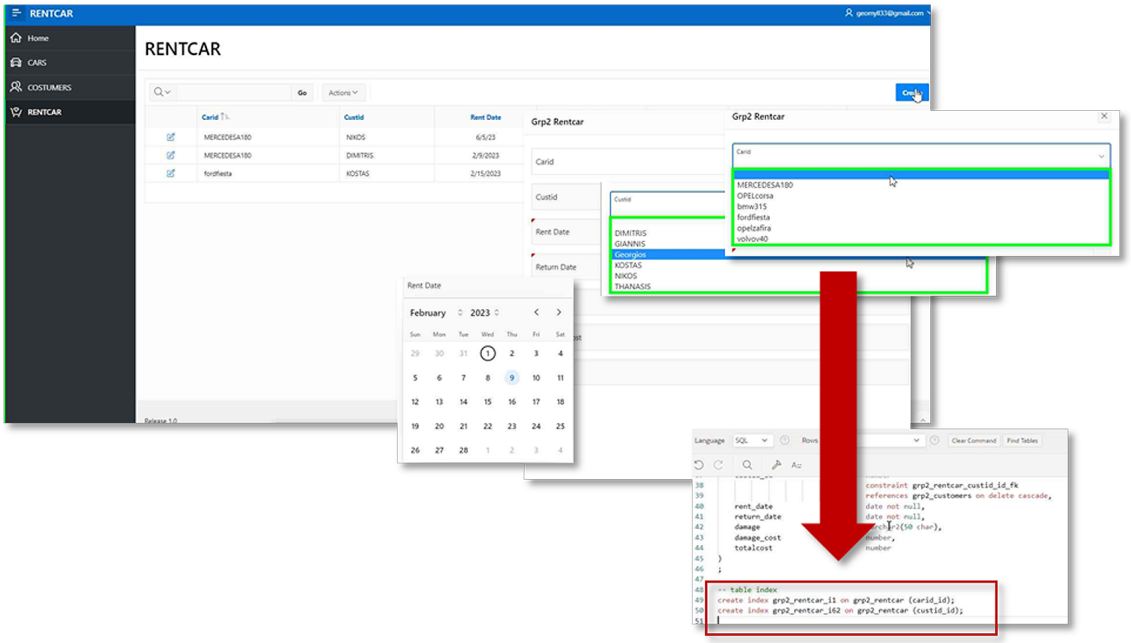
Slika 24.7: Nastavitev stolpca s fotografijo.

Podobno upravljamo tudi s podatki o strankah (slika 24.8).



Slika 24.8: Tabela s podatki o strankah.

V tabeli ch24_rentcar (Slika 24.9) lahko uporabnik izbere avto na podlagi kriterijev, kot so znamka, model in letnik. Najem je nato povezan s stranko. Prikazane bodo vse potrebne podrobnosti, potrebne za najem, kot so avto, stranka, začetni in končni datum najema ter skupni stroški. Uporabnik lahko doda morebitno škodo ob vrnitvi. Po zaključku najema bodo skupni stroški izračunani in predstavljeni v končnem poročilu.



Slika 24.9: Podatki o najemu.

24.6 Dopolnilno študijsko gradivo

V dopolnilnem študijskem gradivu so:

- izvožena aplikacija
- skripti za kreiranje in brisanje tabel
- podatki v preglednici
- video vodič

Vsa dopolnilna študijska gradiva so dostopna na javni spletni strani projekta BeeAPEX. Na tej strani se prijavite kot gost (geslo ni potrebno vpisati). Poiščite učbenik v razdelku Books, skripte v mapi Part 2 > Chapter24 razdelka Scripts ter video vodiče v razdelku Collection of video guides. Študijska gradiva za kratke tečaje poiščite v razdelku Short courses.

24.6.1 Izvožena aplikacija

Izvožena aplikacija je pripravljena kot paket. Ob namestitvi aplikacije se kreirajo tabele ter napolnijo s testnimi podatki. Brisanje aplikacije odstrani vse podatke in definicije tabel in drugih objektov, ki so specifični za to konkretno aplikacijo.

24.6.2 Video vodiči

Video vodič natančno prikazuje razvoj aplikacije.

24.7 Vprašanja

1. Kako lahko dodate ikono v logotip aplikacije, ko ste aplikacijo že ustvarili?
2. Kako lahko dodate slike na svoje strani s poročilom?

24.8 Odgovori

1. V okolju vaše aplikacije izberite **Shared Components** in nato izberite **User Interface Attributes**. Kliknite **Edit** v prikaznem oknu in dodajte poljubno fotografijo.
2. Najprej morate nastaviti podatkovni tip vašega stolpca kot BLOB. Nato v oblikovalniku strani aplikacije za predlogo poročila konfigurirajte stolpec kot brskalnik datotek in spremenite vrsto shranjevanja v **BLOB column specifier in item Source Attributes**. Nato pojdite na obrazec te tabele v okolju App Builder in konfigurirajte stolpec kot **Display image**. Nato konfigurirajte atribute BLOB tako, da povežete tabelo, stolpec in ID..

Literatura

Članki

[1] Hill Kim. „Altruistic cooperation during foraging by the Ache, and the evolved human predisposition to cooperate”. V: *Human Nature* 13 (mar. 2002), strani 105–128. DOI: 10.1007/s12110-002-1016-3 (citirano na strani 185).

Knjige

[2] Roy Fielding in Richard N. Taylor. *Principled design of the modern Web architecture*. Uredil Mehdi Jazayeri andAlexander L. Wolf Carlo Ghezzi. Association for Computing Machinery, 2010. ISBN: 978-1-58113-206-9. DOI: 10.1145/337180.337228 (citirano na straneh 104, 105).

[3] Dariusz Jemielniak in Aleksandra Przegalinska. *Collaborative society*. MIT Press, 2020 (citirano na strani 185).

[4] OMG. *Business Process Model and Notation (BPMN), Version 2.0*. 2011. URL: http://www.omg.org/spec/BPMN/2.0 (citirano na straneh 262, 263).

[5] OASIS OPEN. *Universal Business Language Version 2.1, OASIS Standard, 04 November 2013*. 2013. URL: http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.pdf (citirano na strani 261).

[6] A. Osterwalder in Y. Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers*. Zvezek 1. John Wiley in Sons, 2010 (citirano na straneh 37, 38).

Stvarno kazalo

APEX docker, 44
aplikacijska logika, 87
application programming interface, API, 105
atribut, 51
AutoREST, 109

baza podatkov, 50
BP, 50

Create Application Wizard, 84
CRUD operacija, 108

data workshop, 94, 100
DDL, 60
definicija problema, 225, 237, 245, 251, 265, 293, 299, 311, 327, 348, 365, 389
diagram primera uporabe, 226, 238, 246, 252, 268, 294, 300, 315, 328, 349, 390
Diagram primerov uporabe, 366
DML, 68
DQL, 70
druga normalna forma, 2NF, 59

entiteta, 51
entitetno relacijski diagram, ERD, 51
entitetno-relacijski diagram, ERD, 50
ER, 50
ER diagram, 50, 51

generiranje relacijskega modela, 56
generiranje sheme R-BP, 55
grafikon, 87

HTTP odziv, 106
HTTP zahteva, 105, 108
https://apex.oracle.com, 43

interaktivno poročilo, 87
izmenjava podatkov, 93
izračunavanje, 87
izvoz podatkov, 100
izvoz podatkov s čarovnikom, 100

kako izkoristite galerijo aplikacij in vtičnikov, 197
kako izmenjujete podatke v APEX-u, 93
kako navigirate v APEX-u, 81
kako pripravite bazo podatkov, 49
kako sodelujete v timu, 185
kako tabelarična poročila spremenite v grafikone, 165
kako upravljate paketne in večjezične aplikacije, 205
kako urejate menuje, 181
kako urejate obrazce, 157
kako urejate poročila, 135
kako začeti z Oracle APEX, 35
kardialnost, 51
koledar, 87
kreiranje aplikacije, 87

logični model, 50
logični model podatkov, 50, 230, 240, 246, 252, 271, 296, 302, 316, 329, 351, 368, 392

materialna kosovnica, 311
model delovnega toka, 268
model podatkov, 230, 240, 246, 252, 271, 294, 302, 315, 329, 351, 368, 392
modul virov, 108

nadzor dostopa, 86
najem vozila, rent-a-car, 389
namestitev Flows for APEX, 264
nerelacijski model, 50
nivo abstraktnosti, 50
nivo znanja, 42
normalizacija, 59

object browser, 67, 101
on-premise, 43
Oracle Academy, 46
ORACLE APEX, 81
Oracle Cloud Infrastructure, 46
ORACLE Data Modeler, 61
ORACLE object browser, 69
ORACLE query builder, 75
Oracle REST Data Service, ORDS, 105, 106
ORACLE SQL Developer Data Modeler, 50
ORACLE SQL workshop, 67

page designer, urejevalnik strani, 89
paketna aplikacija, 206
podatkovna plast, 78
podatkovne strukture, 50
podatkovni model, 50
podatkovni tip, 51
polje, 87
poročilo, 103
poslovni vidik primera, 225, 245, 251, 261, 293, 299, 311, 327, 347, 365, 389
povezava, 51
povratne informacije uporabnikov, 86
predloga vira, 108
preverjanje, 87
primer uporabe, UC, 39
primeri uporabe, 226, 237, 246, 251, 265, 294, 300, 312, 327, 348, 365, 390
proces razvoja spletne aplikacije, 81
prva aplikacija, 113
prva normalna forma, 1NF, 59

Quick SQL, 63, 69, 120

R-BP, 54
Razporejanje uradnih ur, 347
regija, 87
relacijski model, 54
relacijski model podatkov, 54, 230, 241, 248, 256, 271, 296, 302, 316, 330, 352, 368, 393
representational state transfer, REST, 104
REST, 104
RESTful dostop, 106

seznam, 87
seznam vrednosti, LOV, 209
shema BP, 54
shema R-BP, 54, 58
shema relacijske BP, 54
sistem za upravljanje baz podatkov, 49
skript QuickSQL in SQL, 331
spletna aplikacija, 49, 89
spletna stran, 87
SQL, 74
SQL ALTER TABLE, 67
SQL COUNT, 74
SQL CREATE TABLE, 66
SQL data definition language, 65
SQL data manipulation language, SQL-DML, 68
SQL data query language, SQL-DQL, 70
SQL DELETE, 69
SQL DROP TABLE, 66
SQL INSERT, 68
SQL JOIN, 73
SQL ORDER BY, 71, 74
SQL Script, 121
SQL SELECT, 71
SQL TO_CHAR(), 72
SQL UPDATE, 68
SQL WHERE, 68
SQL WHERE clause, 72
SQL, jezik za definicijo podatkov, SQL-DDL, 60
SQL-DDL, 60, 68
SQL-DQL, 70
stateless REST, 105
stolpec, 54
storitev RESTful, 104
stran, 90
strukturiran poizvedovalni jezik, SQL, 60
SUBP, 49

tabel, 54
telekomunikacijske storitve, 365
tretja normalna forma, 3NF, 60
tuji ključ, 56

uporabniški vmesniki, 231, 242, 248, 257, 274, 297, 305, 321, 334, 354, 371, 394
upravljanje podatkov, 68
upravljanje sheme relacijske BP, 60
upravljavec vira, 108
urejevalnik strani, 89
ustvarjanje aplikacij, 83
ustvarjanje strani, 86
ustvarjanje strani s čarovnikom, 86
uvoz podatkov, 94
uvoz podatkov s čarovnikom, 96

večjezična aplikacija, 216
Virtual Box Appliance, 43
vodenje poslovnih procesov, 261
vrste strani, 87
vtičnik Flows for APEX, 264
vzorčne in zagonske aplikacije, 197

zaloga vrednosti, 51
zbirka relacij, 54