Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mihael Podplatnik

Razvoj spletne aplikacije z orodjem Oracle Application Express

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleš Smrdel

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pri razvoju in plasiranju aplikacij na trg je zelo pomembno, da so aplikacije hitro razvite in dostopne uporabnikom. V ta namen je na trgu nekaj orodij, ki omogočajo hiter in strukturiran razvoj. V okviru diplome je vaša naloga, da proučite orodje Oracle Application Express in ga uporabite pri razvoju aplikacije, ki naj bo namenjena naprednemu rokovanju s podatki. Zamislite si svojo aplikacijo. Aplikacija naj omogoča več vlog in rokovanje s podatki. Določite potrebne funkcionalnosti, s pomočjo katerih se bodo testirale možnosti razvojnega orodja. Funkcionalnosti aplikacije naj bodo različne za različne vloge. Za razvoj aplikacije uporabite razvojno metodo, ki omogoča učinkovit razvoj aplikacij. Predstavite potek razvoja in razvito aplikacijo.

Zahvaljujem se družini in vsem, ki so mi stali ob strani v času študija in dokončanju diplome. Posebej se zahvaljujem mentorju za koristne napotke in izredno hiter odziv.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Motivacija in cilj 1

1.2 Vsebina diplomske naloge 2

2 Oracle APEX: Pregled in opis **3**

2.1 Tehnologije 3

2.1.1 Oraclova podatkovna baza 3

2.1.2 HTML 4

2.1.3 CSS 5

2.1.4 JavaScript 5

2.1.5 PL/SQL 6

2.2 Arhitektura 7

2.3 Oblikovalec strani 8

2.3.1 Orodna vrstica 9

2.3.2 Levo podokno 10

2.3.3 Osrednje podokno 10

2.3.4 Desno podokno 11

2.4 Lastnosti 12

2.4.1 Prednosti 12

2.4.2 Slabosti 13

2.4.3 Varnost 13

3 Uporabljena metodologija 15

3.1 Faze hitrega razvoja aplikacij 15

3.2 Lastnosti hitrega razvoja aplikacij 17

3.2.1 Prednosti metodologije RAD 17

3.2.2 Slabosti metodologije RAD 17

4 Opis zahtev, načrtovanje in izvedba spletne aplikacije 19

4.1 Zahteve 19

4.2 Podatkovni model 20

4.3 Konfiguracija 25

4.4 Prevod aplikacije 27

4.5 SMTP strežnik 27

4.6 Prijavna stran z avtentikacijo LDAP 27

4.6.1 Razlaga 27

4.6.2 Implementacija 28

4.7 Avtorizacijske sheme 32

4.8 Podmeni *Računi* 33

4.9 Podmeni *Potrjevanje* 36

4.10 Podmeni *Administracija* 36

4.10.1 *Uporabniki* 37

4.10.2 *Nazivi polj* 38

4.10.3 *Napake* 39

5 Prenos in nastavitev aplikacije v produkcijsko okolje 41

5.1 Izvoz aplikacije 41

5.2 Uvoz aplikacije 42

5.3 Odstranitev orodne vrstice 42

5.4 Prenos podatkov 42

6 Zaključne misli 45

6.1 Možnosti za nadaljnje delo 45

Seznam uporabljenih kratic

kratica	angleško	slovensko
APEX	Application Express	Application Express
CSS	Cascading style sheets	kaskadne stilske predloge
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
Oracle SQL	Oracle structured query language	poizvedbeni jezik za Oracle
RAD	Rapid application development	hitri način razvoja aplikacij
AD	Active Directory	aktivni imenik
LDAP	Lightweight Directory Access Protocol	lahki protokol za dostop do imenikov
SMTP	Simple Mail Transfer Protocol	preprosti protokol za prenos pošte
ACL	Access Control List	seznam za preveritev dostopa

Povzetek

Naslov: Razvoj spletne aplikacije z orodjem Oracle Application Express

Avtor: Mihael Podplatnik

V današnjem času je ključen izziv, kako čim prej priti do uporabne aplikacije, ki rešuje uporabniške zahteve. Ena od možnosti je orodje Oracle APEX. V diplomski nalogi predstavljamo postopek razvoja spletne aplikacije z orodjem Oracle APEX. Na začetku se osredotočamo na opis, kaj Oracle APEX pravzaprav je, kako deluje, in tudi predstavimo ključne sklope, ki jih uporabljamo pri razvoju aplikacij s tem orodjem. V nadaljevanju opisujemo uporabljeno metodologijo, imenovano hiter razvoj aplikacij, ki se je razvoj aplikacij z uporabo Oracle APEXa poslužuje. Rezultat naloge je delujoča spletna aplikacija, ki je povezana na Oraclovo podatkovno bazo. Z aplikacijo lahko pregledujemo in potrjujemo račune v več fazah. Na koncu podamo zaključne misli, ki vključujejo oceno orodja APEX in možnosti za izboljšave.

Ključne besede: APEX, Oraclova podatkovna baza, hitri način razvoja aplikacij, spletna aplikacija.

Abstract

Title: Developing a web application using Oracle Application Express

Author: Mihael Podplatnik

In the present time the key challenge is how to obtain a working application for solving user requirements as soon as possible. One option is to use Oracle APEX tool. In this thesis we introduce the process of developing a web application using Oracle APEX. In the beginning, we explain what Oracle APEX is, how it works, and also introduce the key concepts of developing a web application with this tool. This is followed by the description of the software development approach used by Oracle APEX called the rapid-application development (RAD). The result of the thesis is a working web application connected to the Oracle database. The application can be used for reviewing and multi-phase validation of accounts. At the end, we present our thoughts which include evaluation of the APEX tool and possible improvements.

Keywords: APEX, Oracle database, RAD, web application.

Poglavje 1

Uvod

Hitra in točna priprava ter potrditev računov je izredno pomembna za uspešno poslovanje podjetij in državnih ustanov. Informacijska tehnologija omogoča pohitritev in izboljšanje točnosti ter zagotovitev sledljivosti poslovanja z računi. Ob tem se povpraševanje po novih aplikacijah oz. nadgradnjah obstoječih iz leta v leto povečuje, razvijalci programske opreme pa so primorani tem zahtevam hitro ustreči [3]. V Sloveniji in tudi drugod po svetu mnoga podjetja in tudi državne ustanove uporabljajo Oraclovo podatkovno bazo. Posledično je razvoj aplikacij z Oraclovimi orodji izjemno aktualen.

1.1 Motivacija in cilj

Podjetje Oracle Corporation je v zadnjih desetletjih postalo vodilno med proizvajalci programske opreme, ki nudijo sisteme za upravljanje podatkovnih baz [6]. Prav tako visoko kotira med tehnološkimi podjetji, ki nudijo in prodajajo svoja programska orodja za razvoj novih aplikacij [21]. Zato smo posebno pozornost posvetili Oraclovim orodjem kot so Application Development Framework (okr. ADF) in Oracle Forms and Reports. Odločili smo se za orodje Oracle Application Express (okr. APEX), ki omogoča razvoj spletnih aplikacij in ki temelji na Oraclovi podatkovni bazi. Orodje Oracle APEX je brezplačno in je zanimivo tudi zato, ker ni posebej znano orodje za

razvoj spletnih aplikacij. Motivacija za to diplomsko delo je torej prikazati uporabnost omenjenega orodja pri izdelavi spletne aplikacije.

Naloga, ki smo si jo zadali je, da opišemo postopek razvoja spletne aplikacije z orodjem Oracle APEX skozi praktičen primer. Končni cilj je torej delujoča spletna aplikacija, ki omogoča napredno upravljanje s podatki, kar je zelo uporabno pri velikem številu aplikacij.

1.2 Vsebina diplomske naloge

V nadaljevanju sledi opis orodja Oracle APEX in vsega kar spada zraven za uspešen razvoj aplikacije.

V 2. poglavju je pregled omenjenega orodja, od uporabljenih tehnologij in opisa arhitekture do opisa lastnosti.

V 3. poglavju je predstavitev metodologije razvoja aplikacij v Oracle APEXu, imenovane hiter razvoj aplikacij.

V 4. poglavju je opis zahtev, načrtovanje in izvedba spletne aplikacije aplikacije za potrjevanje računov.

5. poglavje se osredotoča na prenos in nastavitev aplikacije iz razvojnega okolja v produkcijsko okolje.

Na koncu so v 6. poglavju predstavljene možnosti za izboljšave in nadaljnje delo ter zaključne misli.

Poglavje 2

Oracle APEX: Pregled in opis

Oracle APEX je spletno razvojno okolje namenjeno razvoju kompleksnih spletnih aplikacij na Oraclovi podatkovni bazi. Za uporabo in razvoj potrebujemo nekaj programerskega znanja in čim novejšo verzijo spletnega brskalnika, saj Oracle APEX temelji na standardih HTML5, CSS3 ter JavaScript. Oracle APEX se je pojavil na začetku tega tisočletja, do sedaj pa je platforma že večkrat spremenila ime in prešla več različič. Za razvoj spletne aplikacije bomo uporabili Oracle APEX različice 5.1, ki je del namestitve brezplačne Oraclove podatkovne baze verzije 11g XE (ang. eXpress Edition). Omenjeno podatkovno bazo je možno prenesti iz Oraclove uradne spletne strani, ob tem pa prav tako brezplačno naložimo Oracle APEX.

2.1 Tehnologije

V tem razdelku sledi pregled in kratek opis tehnologij, ki se uporabljajo v povezavi z razvojem aplikacij z orodjem Oracle APEX. Opisane so le tiste, ki so uporabljene pri izdelavi te naloge.

2.1.1 Oraclova podatkovna baza

Pod splošnim nazivom Oraclova podatkovna baza [8] se skriva relacijski sistem za upravljanje podatkovnih baz, katerega je razvilo podjetje Oracle Cor-

poration. Gre za enega izmed najbolj zaupanja vrednih in široko uporabljenih sistemov za upravljanje podatkovnih baz nasploh. Namen podatkovne baze je zbiranje, shranjevanje in ponovna uporaba podatkov s strani uporabnikov ali aplikacij. Uporabniki in aplikacije lahko dostopajo do podatkov v relacijskih podatkovnih bazah z uporabo poizvedovalnega jezika SQL.

Oracle SQL Developer

Oracle SQL Developer [9] je brezplačno integrirano razvojno okolje, ki se uporablja za delo in poizvedovanje z jezikom SQL na Oraclovi podatkovni bazi. Omogoča tudi pregled in izdelavo entitetno-relacijskih modelov.

2.1.2 HTML

HTML [10] je označevalni jezik, ki se uporablja za prikaz teksta, slik in ostalih virov na spletnih straneh preko spletnega brskalnika. HTML je uporaben pri strukturiranju dokumentov in ima veliko omejitev pri oblikovanju spletnih strani. Z dodajanjem HTMLja preko Oracle APEXa lahko spreminjamo izgled aplikacije, tako da se spremenjen HTML posreduje odjemalcu.

Sintaksa

HTML uporablja oznake, sestavljene iz znakov `<`, `>` in `/`. HTML element je sestavljen iz začetne in končne oznake, med njima pa se nahaja vsebina. HTML element ima takšen izgled:

```
<začetna oznaka>vsebina</končna oznaka>
```

V začetne oznake lahko opcijsko dodamo atribute z vrednostjo.

```
<začetna oznaka atribut="vrednost">
```

Tako se lahko na določene elemente sklicujemo, npr. z uporabo sintakse za določanja izgleda, CSS.

2.1.3 CSS

CSS (Cascading Style Sheets) [7] oz. kaskadne stilske predloge ponujajo možnost za oblikovanje spletnih strani brez spreminjanja HTML strukture. CSS se uporablja za definiranje stilov besedila, izgled tabel in drugih vidikov spletnih strani, ki so bile prej definirane samo s HTML oznakami. Datoteko, ki vsebuje CSSje lahko naložimo v Oracle APEX. Na te CSSje se lahko nato sklicujemo, omogočeno pa je tudi vključevanje CSS na vsaki strani posebej. CSSji se izvajajo na strani odjemalca.

Sintaksa

Primer CSS stila, s katerim spremenimo barvo in velikost pisave HTML odstavkov (oznaka p):

```
p {
  color: red;
  font-size: 15px;
}
```

CSS sintaksa najprej zahteva selektor, s katerim se sklicujemo na eno ali več HTML oznak. Nato v zavitih oklepajih sledi deklaracijski blok. Deklaracija je sestavljena iz lastnosti, dvopičja in vrednosti. Deklaracije v istem bloku ločujemo s podpičjem.

2.1.4 JavaScript

JavaScript [13] je programski jezik, ki ga je razvilo podjetje Netscape in se ga večinoma uporablja pri razvoju spletnih strani. Poseduje sintakso podobno programskemu jeziku C in omogoča dodajanje dinamičnih in interaktivnih komponent spletnim stranem. JavaScript kodo lahko dodajamo aplikacijam preko Oracle APEXa za dodajanje funkcionalnosti na strani odjemalca.

2.1.5 PL/SQL

PL/SQL (Procedural Language/Structured Query Language) [16, 3] je Oraclov programski jezik, ki preko zaporednih ukazov v kombinaciji s SQL ukazi obdeluje podatke v bazi. PL/SQL omogoča uporabo zank ter vključuje pogojne stavke in druge proceduralne funkcionalnosti tipičnih programskih jezikov. Struktura PL/SQL programskega bloka je naslednja:

```
DECLARE
    -- deklaracija spremenljivk
BEGIN
    -- programski stavki
EXCEPTION
    -- stavki za obravnavo napak
END;
```

PL/SQL programske spremenljivke se določajo v deklaracijskem delu programskega bloka, označenega z rezervirano besedo DECLARE. Telo programskega bloka se nahaja med rezerviranimi besedami BEGIN in EXCEPTION. Telo sestavljajo prireditveni stavki, pogojni stavki, zanke itd. Na koncu programskega bloka je del za obravnavo napak. Program se konča z rezervirano besedo END s podpičjem. Dela za deklaracijo spremenljivk in obravnavo napak sta opcijska. Komentarje se označuje z dvojnim pomišljajem. Oracle APEX omogoča dodajanje SQL in PL/SQL kode, ki se izvaja na strani strežnika.

Programske enote in sekvence

Programska enota je PL/SQL koda, ki je razvita, prevedena, shranjena in izvajana na podatkovni bazi. SQL Developer omogoča pregled, dodajanje in upravljanje programskih enot. Tipični predstavniki so prožilci (ang. triggers), funkcije (ang. functions) in paketi (ang. packages). Prožilec je program, ki se izvede ob spremembi na podatkovni bazi. Funkcija je koda, katero

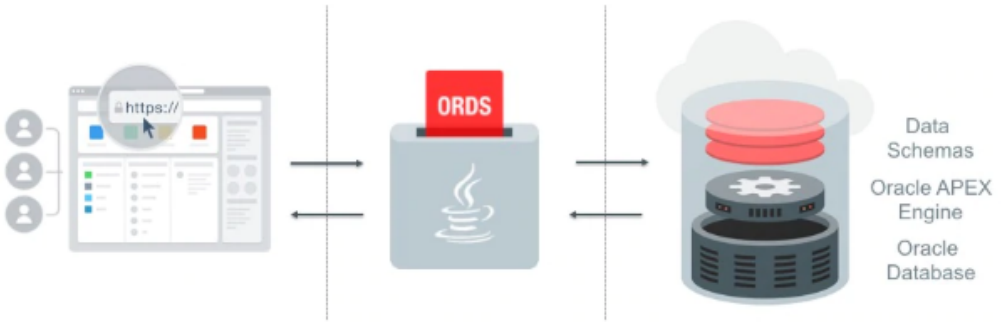
kličemo in nam vrne vrednost. Tesno povezane funkcije in ostale programske enote lahko zapišemo in shranimo v pakete.

Sekvenca (ang. sequence) je objekt podatkovne baze, s pomočjo katere lahko avtomatično generiramo edinstvena cela števila za primarne ključe tabel. Spodnja koda ustvari sekvenco, ki se prične s številko 1, povečuje za 1, vnaprej ne alocira vrednosti in se ne prične znova pri 1, ko doseže maksimalno vrednost.

```
CREATE SEQUENCE racuni_seq
 MINVALUE 1
 MAXVALUE 999999999999999999999999999
 START WITH     1
 INCREMENT BY   1
 NOCACHE
 NOORDER
 NOCYCLE;
```

2.2 Arhitektura

Pri razvoju programske opreme imamo pogosto opravka z večnivojskimi arhitekturami, kjer so predstavitev, procesiranje in podatkovna struktura fizično ločeni. Oracle APEX sestavlja klasična trinivojska arhitektura [4, 5], ki je prikazana na sliki 2.1. Arhitektura Oracle APEXa je zasnovana tako, da na srednjem nivoju zahteva strežnik, ki obdela zahteve med spletnim brskalnikom in pogonom APEX (ang. APEX engine). Srednji sloj je tako lahko Oracle Embedded PL/SQL Gateway (EPG), ki je del podatkovne baze, Oracle HTTP Server (OHS) ali Oracle REST Data Services (ORDS), ki teče na Javanskem strežniku (npr. Web Logic Server WLS). Priporočljiva izbira je ORDS, saj zagotavlja RESTful storitve in povečuje varnostne zmogljivosti. Takšna arhitektura je imenovana Oracle RAD Stack in je sestavljena iz naslednjih komponent:



Slika 2.1: Diagram poteka procesov, ko uporabnik na aplikaciji sproži zahtevo [4].

Oracle REST Data Services (ORDS) je podatkovna storitev, ki temelji na tehnologiji Java EE (Enterprise Edition).

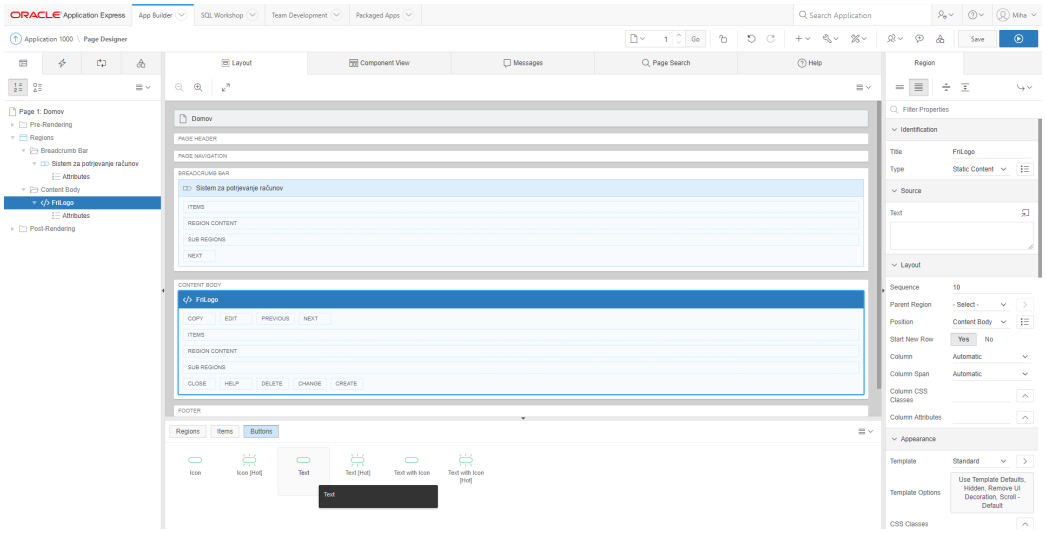
APEX je orodje za razvoj spletnih aplikacij, katerega opisujemo.

Database oz. Oraclova podatkovna baza je temelj, ki zagotavlja varnost podatkov in nad katerim izdelujemo spletne aplikacije.

2.3 Oblikovalec strani

Ta razdelek je namenjen pregledu in opisu oblikovalca strani (ang. Page Designer) [12, 15]. Oblikovalec strani (slika 2.2) je najpomembnejši del APEXa. Uporablja se za gradnjo oz. razvoj posameznih strani aplikacije. Sestavljajo ga štiri glavne komponente:

- Orodna vrstica;
- Levo podokno oblikovalca strani oz. pregled komponent;
- Osrednje podokno oblikovalca strani oz. postavitev;
- Desno podokno oblikovalca strani oz. urejevalnik lastnosti.



Slika 2.2: Izgled oblikovalca strani.

2.3.1 Orodna vrstica

Orodna vrstica se nahaja na vrhu strani in vsebuje oz. ponuja naslednje izbire:

izbirnik strani prikazuje trenutno stran in omogoča iskanje strani,

ključavnica strani označuje, ali je možno stran urejati ali ne,

gumba razveljavi oz. povrni omogočata razveljavitev oz. povrnitev zadnje spremembe,

meni ustvari ponuja opcije kot so kreiranje novih strani, kopiranje trenutne strani, pomoč pri kreiranju/urejanju strani, čarovnike za ustvarjanje novih komponent kot so forme in poročila, dostop do deljenih komponent in dodajanje komentarjev,

meni storitve vsebuje možnosti, kot so brisanje strani, pregled zgodovine in omogočanja predpomnenja za povečano učinkovitost,

meni nastavitve omogoča določitev števila prikazanih podoken,

meni za skupinski razvoj, ki se uporablja kadar več razvijalcev sodeluje na isti aplikaciji,

meni pripombe za dodajanje komentarjev, prijavo hroščev in kaj je še potrebno dokončati,

gumb za deljene komponente, ki preusmeri na stran, kjer urejamo skupne lastnosti vseh strani v aplikaciji,

gumba shrani in zaženi, s katerima najprej shranimo spremembe, nato pa stran zaženemo in vidimo spremembe na dejanski aplikaciji.

2.3.2 Levo podokno

Levo podokno vsebuje štiri zavihke, ki prikazujejo vse komponente na strani v drevesni obliki. Ti zavihki so:

upodabljanje (ang. Rendering), ki prikazuje vse možnosti na strani, kot so poročila, koledar, drevesne strukture in drugo, vključno z gumbi in ostalimi elementi,

dinamične komponente (ang. Dynamic Actions) za deklarativno dodajanje dinamike na trenutni strani ali z dodajanjem JavaScript kode, ki se bo izvedla na strani odjemalca,

zavihek za obdelave (ang. Processing Tab), ki prikazuje seznam procesov, ki se bodo izvedli ob predložitvi strani (page submit),

deljene komponente strani (ang. Page Shared Components), ki prikazuje deljene komponente ali lastnosti na trenutni strani.

2.3.3 Osrednje podokno

Osrednje podokno sestavlja naslednjih pet zavihkov:

postavitev (ang. Layout), je vizualna predstavitev, kako so komponente postavljene na strani, ter omogoča povleci in spusti funkcionalnost za urejanje izgleda strani,

zavihek za pregled komponent (ang. Component View Tab) združuje elemente uporabniškega vmesnika in logiko aplikacije po tipu komponent,

zavihek s sporočilom (ang. Messages Tab) javlja morebitne napake, ko nepravilno spreminjamo aplikacijo z dodajanem komponent ali s spreminjanjem logike,

iskalnik (ang. Page Search Tab) za iskanja katerekoli lastnosti npr. dinamične akcije ali gumba, ki se nahaja na trenutni strani,

zavihek za pomoč (ang. Help Tab) ob kliku izpiše navodila za uporabo kateregakoli atributa na desnem podoknu.

2.3.4 Desno podokno

Desno podokno ali urejevalnik lastnosti prikazuje štiri možnosti prikaza atributov, katerih uporaba je samoumevna. Vsebuje vse atribute za trenutno izbrano komponento, ki jih je mogoče urejati. Kadar je potrebno vključiti daljšo kodo uporabimo vgrajen urejevalnik kode, ki vključuje naslednje funkcionalnosti:

- preverjanje pravilnosti sintakse,
- razveljavljanje,
- povrnitev,
- iskanje,
- menjanje,
- poudarjanje sintakse,

- zamik bloka,
- samodejno dokončanje.

2.4 Lastnosti

Oracle APEX ima prednosti in slabosti, vključno z varnostnimi zadržki z APEXom razvitih aplikacij. Ti zadržki so podobni, kot pri aplikacijah, ki temeljijo na bolj neposrednih tehnologijah kot npr. PHP ali Java in se nanašajo predvsem na vrinjenje SQL stavkov (ang. SQL injection). V tem razdelku sledi pregled teh lastnosti.

2.4.1 Prednosti

Prednosti Oracle APEXa so:

- Centralni nadzor nad podatki in aplikacijami;
- Edina zahteva je ustrezen spletni brskalnik;
- Omogoča deljen razvoj in dostop do aplikacij;
- Hitra izdelava prototipov z uporabo vgrajenih vpogledov (ang. Built-in View) in gradnikov;
- Ne zahteva naprednega programerskega znanja za izdelavo osnovnih aplikacij;
- Aplikacije so lahko vezane na brezplačno različico Oraclove podatkovne baze XE;
- Enostavna namestitev aplikacij. Končni uporabniki do aplikacije dostopajo preko URL naslova.

2.4.2 Slabosti

Slabosti Oracle APEXa so:

- Aplikacije izdelane z uporabo Oracle APEXa nujno potrebujejo Oraclovo bazo za delovanje. To lahko vodi do pojava imenovanega priklenitev na prodajalca, saj za nadaljnjo uporabo podatkovne baze in delovanje aplikacije stranka postane odvisna od Oracla;
- APEX ne nudi dobre podpore za skupinski razvoj. Razvijalci morajo drug drugega obvestiti o svojih namerah kadar urejajo isto stran;
- Le redki spletni gostitelji nudijo Oraclovo podatkovno bazo. Posledično imajo APEX aplikacije omejen izbor spletnih gostiteljev.

2.4.3 Varnost

Proizvajalci trdijo, da so aplikacije izdelane z uporabo Oracle APEX v sami osnovi že varne [19]. Seje so definirane v spletnem piškotku in kot naključno dolgo število v URL naslovu, kar je nemogoče ugotoviti. Seje so prav tako časovno omejene. APEX uporablja kontrolne vsote na način, da prepreči uporabnikom manipulacijo parametrov v URL naslovih in pri shranjevanju sprememb. Tako napadalci ne morejo žrtvam pošiljati ponarejenih URL naslovov, ki bi lahko sprožili izvedbo nezaželene akcije. Razvijalci imajo možnost definirati avtentikacijske sheme in pravice dostopa do določenih aplikacij, strani in drugih komponent na strani.

Glavne grožnje predstavljata prekdomenski napad (ang. cross-site scripting) in vrinjenje SQL stavkov, kjer napadalec vnese škodljive skripte, ki se izvedejo na strani odjemalca. V ta namen ima APEX definirane funkcije, kot je na primer APEX_ESCAPE_HTML [3], s katero se zamenjajo znaki s posebnim pomenom v HTMLju.

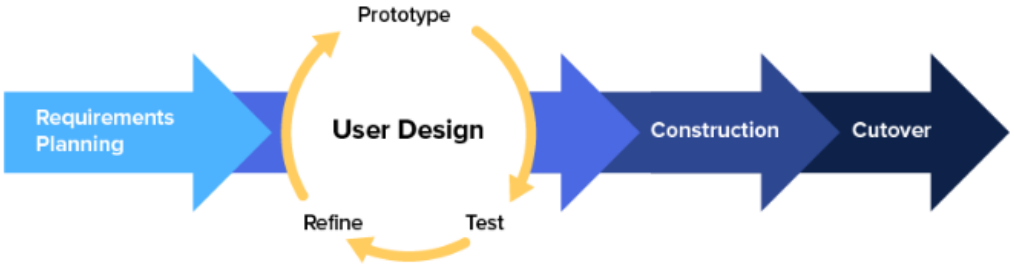
Poglavje 3

Uporabljena metodologija

Programska oprema ne zahteva vnaprej določene fiksne strukture, temveč mora biti prilagodljiva na zahteve uporabnika. Klasične načrtno usmerjene metode planiranja razvoja aplikacij, kot je slapovni model, niso dovzetne za spremembe v kasnejših fazah razvoja, saj že na začetku zahtevajo strogo opredelitev zahtev. Pri razvoju programske opreme lahko v vsakem koraku pride do sprememb, katerim je potrebno čim prej ustreči. Znanje pridobljeno med samim razvojem lahko veliko pripomore h končni rešitvi. Potreben je model, ki dopušča testiranje, brisanje in dodajanje funkcionalnosti v vseh fazah razvoja brez posledic pri končni rešitvi. Metodologija, ki se je pojavila zaradi prej navedenih zahtev, kot odgovor slapovnemu modelu se imenuje hiter razvoj aplikacij (ang. Rapid Application Development, okr. RAD) [17, 18]. Pri tej metodologiji je večji poudarek dan prilagodljivemu razvoju kot pa samemu načrtovanju, v nasprotju s slapovnim modelom.

3.1 Faze hitrega razvoja aplikacij

RAD je razvil James Martin v osemdesetih letih prejšnjega stoletja. Od prvih realizacij se je RAD naprej razvijal in dopolnjeval, pojavile pa so se tudi druge metode za hiter razvoj aplikacij. Med bolj znane spadajo agilne metode razvoja programske opreme. Toda glavna ideja Martinove metodolo-



Slika 3.1: Faze razvoja programske opreme z metodo RAD [17].

gije je ostala ista. Njegov pristop vsebuje štiri osrednje faze hitrega razvoja aplikacij, prikazane v diagramu 3.1. Te faze so naslednje:

Faza načrtovanja zahtev (ang. Requirements Planning) Uporabniki in razvijalci se morajo najprej sestati in dogovoriti kakšne so zahteve in funkcionalnosti končnega produkta. Ta faza naj ne bi trajala preveč časa.

Faza uporabniškega načrtovanja (ang. User Design) Večina razvoja poteka v tej fazi. Razvijalci nenehno izdelujejo prototipe dogovorjenih zahtev v čim krajšem možnem času. Nato jih predstavijo končnim strankam, ki podajo povratne informacije vključno z novimi zahtevami, lepotnimi popravki ali dopolnjenimi funkcionalnostmi. Po dogovorjenih popravkih se postopek ponovi.

Faza izdelave (ang. Construction) Ko se faza načrtovanja konča, se prične izdelava končnega produkta. Posebnost te faze je, da se stranke še vedno lahko odločijo spremeniti določene funkcionalnosti.

Prehod na nov sistem (ang. Cutover) Končni produkt je potrebno testirati, uporabnike seznaniti z uporabo in zagotoviti delovanje aplikacije. Nato se lahko prične uporabljati aplikacijo na produkciji.

3.2 Lastnosti hitrega razvoja aplikacij

Iz prejšnjega razdelka je možno razbrati določene pozitivne lastnosti, ki jih prinaša metoda RAD pri izdelavi programske opreme. Toda takšna zasnova modela zraven pozitivnih obrodi tudi nekaj negativnih lastnosti. V nadaljevanju so naštete nekatere prednosti in slabosti.

3.2.1 Prednosti metodologije RAD

Glavne prednosti hitrega razvoja aplikacij so:

- Zahteve in funkcionalnosti se lahko kadarkoli spremenijo, saj uporabniki nenehno sodelujejo med razvojem aplikacije.
- Preko prototipiranja se vzpodbuja in upošteva uporabniške zahteve. Posledično je končni produkt kakovostnejši in funkcionalno bolj izpolnjuje zahteve v primerjavi s slapovnim modelom.
- Zaradi visoke prilagodljivosti so možnosti, da bi se znašli v slepi ulici manjše, kot v primerjavi s slapovnim modelom, kjer je v primeru ene napake celoten proces potrebno začeti znova.
- Čas razvoja se zaradi kratkega razvojnega cikla zmanjša, cena pa pogosteje ostane znotraj načrtovanih okvirov.
- Delo je bolj produktivno, saj na projektih sodeluje manj ljudi.

3.2.2 Slabosti metodologije RAD

Glavne slabosti hitrega razvoja aplikacij so:

- Veliko časa se porabi za interakcijo med uporabnikom in razvijalcem, ki bi bil lahko uporabljen za razvoj. V slapovnem modelu so vse zahteve bile določene na začetku, tako da so razvijalci lahko več časa namenili razvoju aplikacij.

- Zahteva po visoko usposobljenemu kadru.
- Pri pristopih kot je Martinov se lahko razvijalec zaradi nenehnega prototipiranja preveč osredotoči na določene komponente namesto na kakovost celostne zasnove aplikacije.
- Metodologija ni primerna v primerih, ko je kontrola projekta pomembnejša od končnega roka dokončanja. Zaradi fleksibilnosti in velikega števila prototipiranja je razvoj težko dobro nadzirati oz. je vodenje zahtevnejše.
- Metodologija se zaradi težjega nadzora ne obnese v večjih ekipah.
- Potrebno je pridobiti najboljše uporabnike, kar ni zmeraj mogoče, ker se morajo le-ti posvečati tekočemu delu. Njihovi nadrejeni jim pogosto ne dopuščajo dovolj časa za potrebe razvoja aplikacije, ker so najboljši in nezamenljivi v opravljanju svojega običajnega dela.

Poglavje 4

Opis zahtev, načrtovanje in izvedba spletne aplikacije

V tem poglavju bomo predstavili zahteve, načrtovanje in izvedbo spletne aplikacije, katere glavni namen je priprava, pregled in več-fazno potrjevanje računov.

4.1 Zahteve

Pred začetkom razvoja katerekoli aplikacije je treba najprej skupaj z uporabniki doreči ključne zahteve, ki jih naša aplikacija mora omogočati. Pri razvoju spletne aplikacije smo uporabili metodologijo RAD ter upoštevali njena načela. Ker po tej metodologiji faza načrtovanja zahtev traja krajši čas, na tem mestu naštejemo ključne zahteve spletne aplikacije. Ključne zahteve funkcionalnosti spletne aplikacije so:

- možnost kreiranja računov,
- možnost kreiranja postavk računa,
- beleženje, kateri uporabnik je kreiral ali spremenil račun in kdaj,
- pregled računov in pripadajočih postavk,

- več-fazno potrjevanje računov v določenem vrstnem redu,
- obveščanje o potrebi potrditve računa preko elektronske pošte,
- pregled računov in uporabnikov.

Določili smo tudi dve posebni zahtevi. Prva je, da se uporabniki v spletno aplikacijo lahko prijavijo z AD uporabniškim imenom in računom. Druga posebna zahteva spletne aplikacije je, da ob spremembah zakonodaje oz. zahtev glede poimenovanja atributov računa ali postavk, ni treba spreminjati podatkovnega modela. Za uporabo aplikacije smo določili tri različne uporabniške vloge:

Izdajatelj računa ima pravico ustvarjati in pregledovati račune in njegove postavke, določiti potrjevalce in račun poslati v potrjevanje.

Potrjevalec računov ima omogočen pregled nad računi in postavkami ter ima možnost potrditi račun, za katerega je pravočasno obveščen preko elektronske pošte.

Administrator ima omogočen pregled nad računi in postavkami ter ima pregled nad uporabniki. Lahko ustvarja nove izdajatelje računov in potrjevalce.

4.2 Podatkovni model

Pred začetkom razvoja spletne aplikacije je najprej treba izdelati podatkovni model, na katerem temelji spletna aplikacija. Za izdelavo podatkovnega modela smo uporabili program Oracle SQL Developer. Primer SQL ukaza, s katerim smo izdelali glavno entiteto:

```
CREATE TABLE "RACUN"
   ("ID_RACUN" NUMBER,
    "POLJE_1" VARCHAR2(500 CHAR),
    "POLJE_2" VARCHAR2(500 CHAR),
```

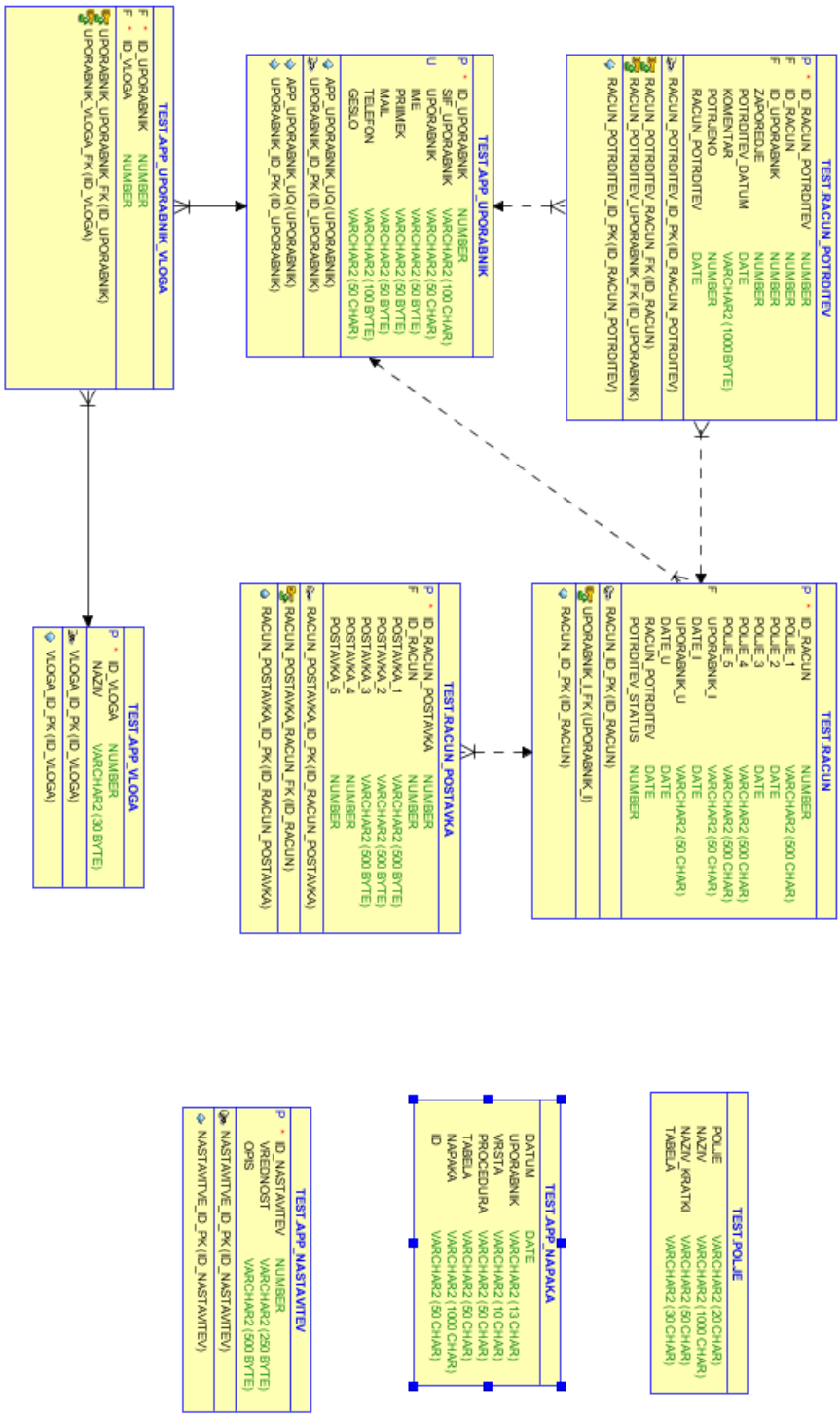
```
"POLJE_3" VARCHAR2(500 CHAR),
"POLJE_4" VARCHAR2(500 CHAR),
"POLJE_5" VARCHAR2(500 CHAR),
"UPORABNIK_I" VARCHAR2(50 CHAR),
"DATE_I" DATE,
"UPORABNIK_U" VARCHAR2(50 CHAR),
"DATE_U" DATE,
"RACUN_POTRDITEV" DATE,
"POTRDITEV_STATUS" NUMBER,
CONSTRAINT "RACUN_ID_PK" PRIMARY KEY ("ID_RACUN")
```

Slika 4.1 prikazuje shemo podatkovnega modela, katero smo ustvarili z grafičnim orodjem Data Modeler, ki je del programa Oracle SQL Developer. V shemi so prikazane entite, vključno s svojimi stolpci, primarnimi ključi, tujimi ključi in kot posebnost, indeksi [11]. Z uporabo indeksov se ustvari B-drevo iz vseh zapisov v indeksiranih stolpcih. S tem povečamo učinkovitost pri pridobivanju zapisov iz podatkovne baze. V nadaljevanju sledi podrobnejši opis glavnih atributov posameznih entitet.

Racun

Entiteta *racun* predstavlja osrednjo entiteto podatkovnega modela. Zraven edinstvenega identifikacijskega stolpca vsebuje še naslednje pomembne podatke:

- polja, ki opisujejo račun,
- kateri uporabnik je kreiral ali urejal račun,
- datum kreiranja ali urejanja računa,
- datum potrditve računa,
- status potrditve računa.



Slika 4.1: Shema podatkovnega modela.

Racun_potrditev

Ta entiteta zraven edinstvene identifikacijske številke beleži naslednje podatke:

- številka računa,
- kateri uporabnik je potrdil račun,
- zaporedje potrjevalca,
- datum potrditve računa,
- morebiten komentar ob potrditvi računa,
- status potrditve računa.

Racun_postavka

Entiteta *racun_postavka* hrani postavke računa. Ostali pomembni atributi so:

- edinstvena identifikacijska številka,
- številka računa,
- pet polj s postavkami.

App_uporabnik

V tej entiteti so shranjeni podatki o uporabnikih. Struktura entitete je naslednja:

- edinstvena identifikacijska številka uporabnika,
- uporabniško ime,
- ime uporabnika,
- priimek uporabnika,

- elektronski naslov,
- telefon,
- geslo.

App_vloga

V tej entiteti so določene možne vloge, ki jih ima lahko uporabnik. Vsebuje

- edinstveno identifikacijsko številko vloge,
- naziv oz. opis vloge.

App_uporabnik_vloga

V tej entiteti je določeno, katero vlogo ima uporabnik. Vsebuje naslednja podatka:

- edinstvena identifikacijska številka uporabnika,
- identifikacijska številka vloge.

App_napaka

Ta entiteta se uporablja za beleženje morebitnih napak, ki se lahko pojavijo med uporabo aplikacije. Zabeležijo se naslednji podatki:

- datum napake,
- kateremu uporabniku se je napaka pojavila,
- vrsta napake,
- tabela,
- naziv napake.

App_nastavitev

Ta entiteta vsebuje podatke o nastavitvah, ki jih uporablja aplikacija, kot je na primer naziv domene omrežja. Vsebuje:

- edinstveno vrednost,
- opis, kaj ta vrednost pomeni.

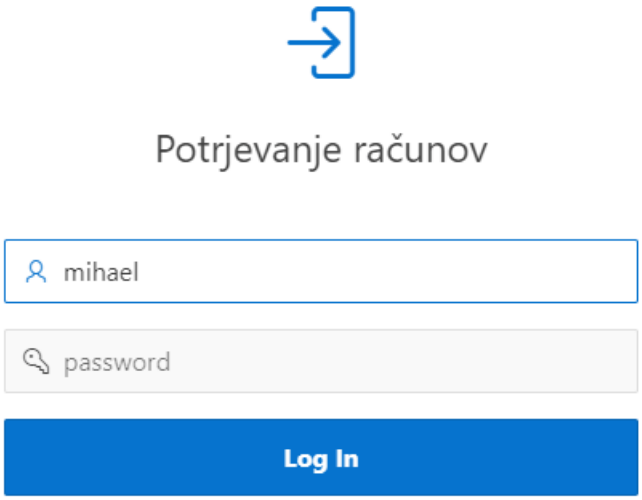
Polje

Entiteta *polje* vsebuje opise oz. razlage polj, ki se pojavljajo v aplikaciji. Struktura entitete je naslednja:

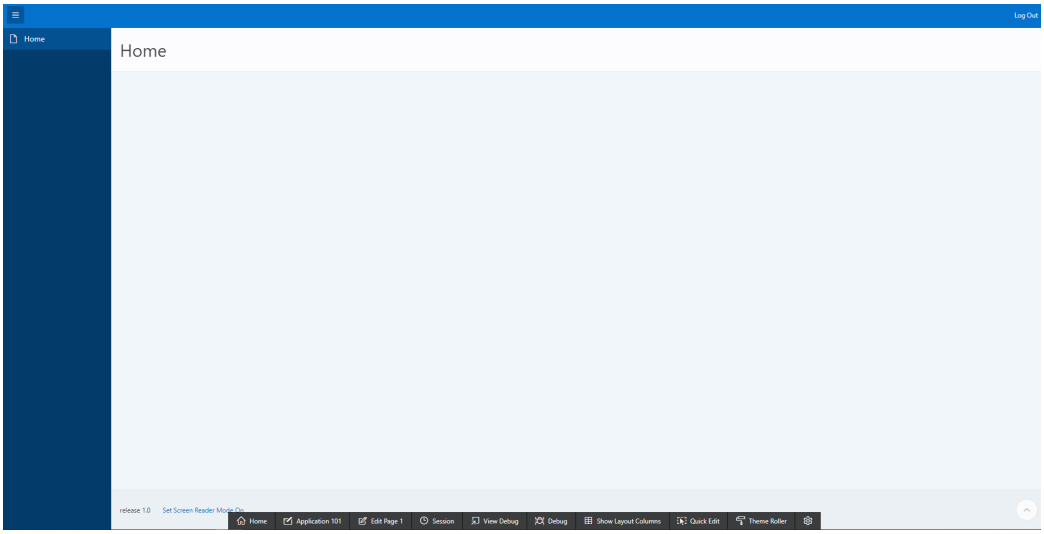
- ime polja,
- razlaga oz. naziv polja,
- tabela, v kateri se polje nahaja.

4.3 Konfiguracija

Preden lahko pričnemo z izdelavo spletne aplikacije je najprej treba ustvariti delovno okolje (ang. Workspace), znotraj katerega lahko nato ustvarimo in razvijamo našo spletno aplikacijo. To storimo tako, da se vpišemo v administrativne storitve (ang. Administration Services) z administratorskim uporabniškim računom, ki smo ga določili ob namestitvi APEXa. Nato izberemo možnost za kreiranje novega delovnega okolja, kjer preko čarovnika določimo na katero podatkovno shemo se želimo povezati in določimo ostale lastnosti delovnega okolja, kot je ime. Tukaj prav tako ustvarimo razvijalca oz. razvijalce, ki bodo imeli pravico dostopati in razvijati v prej ustvarjenem delovnenm okolju. Ko se prijavimo v delovno okolje ustvarimo aplikacijo. Sliki 4.2 in 4.3 prikazujeta vpisno in domačo stran na novo ustvarjene aplikacije.



Slika 4.2: Vpisno okno na novo ustvarjene aplikacije.



Slika 4.3: Domača stran na novo ustvarjene aplikacije.

4.4 Prevod aplikacije

Predvideni končni uporabniki naše spletne aplikacije se nahajajo v slovenskem prostoru, zato je smiselno našo spletno aplikacijo prevesti v slovenščino. Na spletni strani `http://translate-apex.com/apex/f?p=800:TRANSLATIONS:::NO:::` (Dostopano: 29.1.2019) poiščemo in prenesemo skripto s slovenskimi prevodi ter jo poženemo v programu SQL Developer.

4.5 SMTP strežnik

Za obveščanje preko elektronske pošte smo najprej morali namestiti strežnik, ki bo skrbel za pošiljanje pošte v našem omrežju. Uporabili smo SMTP oz. Simple Mail Transport Protocol (slov. preprosti protokol za prenos pošte) strežnik, katerega mora po navadi namestiti sistemski administrator. SMTP strežnik ni nič drugega kot računalnik, ki uporablja SMTP za pošiljanje in izmenjavo elektronske pošte [20].

4.6 Prijavna stran z avtentikacijo LDAP

Ob kreiranju aplikacije se nastavi privzeta overitev dostopa za uporabnike, ki so kreirani znotraj Oracle APEXa. Izberemo lahko katero drugo že vgrajeno overitev ali pa jo ustvarimo popolnoma sami. Ker bodo vsi končni uporabniki aplikacije za potrjevanje računov do nje dostopali preko deljenega omrežja, je smotrno dovoliti dostop do aplikacije le uporabnikom znotraj tega omrežja.

4.6.1 Razlaga

Za lažjo razlago tega razdelka je potrebno najprej na kratko definirati nekaj pojmov:

Directory service (slov. imeniška storitev) je programski sistem, ki shranjuje, organizira in zagotavlja dostop do informacij v imeniku omrežja. V računalništvu je imenik preslikava med imeni in vrednostmi.

Active Directory okr. AD [2] (slov. aktivni imenik) je Microsoftova imeniška storitev za domensko omrežje Windows, ki omogoča administracijo računalnikov in uporabnikov.

Lightweight Directory Access Protocol okr. LDAP (slov. lahki protokol za dostop do imenikov) [14] je programski protokol, ki ga uporablja AD, za poizvedovanje in spreminjanje imeniških storitev, ki teče preko TCP/IP.

Access Control List okr. ACL (slov. seznam za preveritev dostopa) [12, 1] je seznam dovoljenj (ang. Access control entry, okr. ACE), ki opredeli uporabnika in mu določa dostopne pravice do določenih računalniških objektov.

4.6.2 Implementacija

Za uporabo LDAP dostopa mora najprej sistemski administrator dodati ustrezne pravice ACL preko podatkovne baze. Po tem ustvarimo lastno overitev, kot je prikazano na sliki 4.4. Funkcija authenticate_user za overitev je shranjena v paketu v podatkovni bazi, saj tako zmanjšamo omrežni promet med aplikacijo in podatkovno bazo ter povečamo učinkovitost [16]. Sedaj se lahko v aplikacijo prijavi z istim uporabniškim imenom in geslom kot v računalnik. Aplikacija je tako dobro zavarovana pred zunanjimi grožnjami.

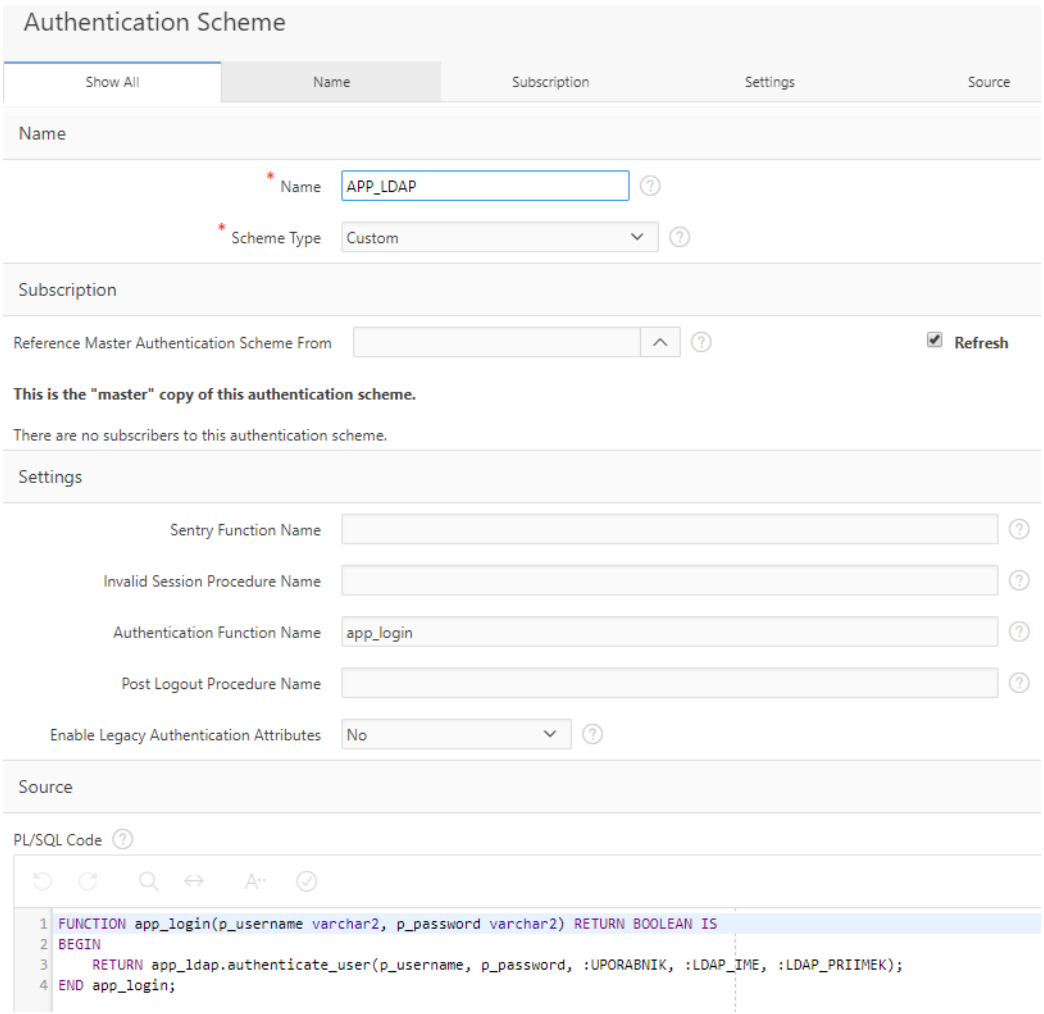
Sedaj, ko imamo implementiran željen način avtentikacije, lahko z uporabo tehnologij HTML, CSS in JavaScript uredimo prijavno stran. Temo smo prilagodili barvi Univerze v Ljubljani. Dodelana prijavna stran je prikazana na sliki 4.5. Primer uporabljenega HTMLja:

```
<i> Fakulteta za računalništvo in informatiko</i>
```

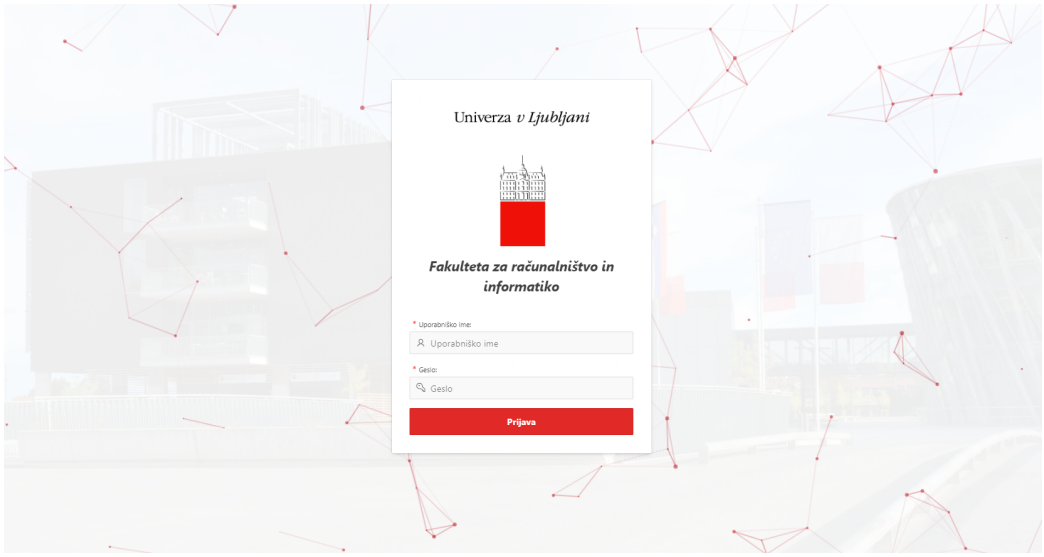
Spodaj je prikazan del CSSja, ki smo ga uporabili pri vpisni strani za prikaz fakultete v ozadju:

```
.t-Body {
    position: relative;
    z-index: 1;
}
.t-Body:before {
    content: "";
    position: absolute;
    width: 100%;
    height: 100%;
    opacity: .08;
    z-index: -1;
    background: url(#APP_IMAGES#fri1.jpg);
    background-size: cover;
}
span.t-Login-logo {
    background-image: url(#APP_IMAGES#UL_logo.gif);
    backgroung-size: cover;
    width: 240px;
    height: 239px;
}
```

Datoteko, ki vsebuje JavaScript kodo za dodajanje dinamike v obliki premikajočih delcev [22], smo dodali v deljene komponente aplikacije in jo z uporabo HTMLja in CSSa ustrezno vključili.



Slika 4.4: Ustvarjanje lastne avtentikacijske sheme LDAP.



Slika 4.5: Končna prijavna stran aplikacije.

4.7 Avtorizacijske sheme

V skladu z zahtevami po treh različnih uporabniških vlogah smo ustvarili tri različne avtorizacijske sheme:

- izdajatelj računov,
- potrjevalec računov in
- administrator.

S pomočjo avtorizacijskih shem smo preko APEXa določili, kateri uporabniki imajo dostop do določenih strani in do določenih funkcionalnosti. Poleg prijavne strani je domača stran edina druga stran znotraj aplikacije, do katere imajo vsi uporabniki enakovredno pravico dostopanja. Domača stran (slika 4.6) je prva stran, do katere pridemo po uspešni prijavi.

Slika 4.6: Po uspešni prijavi se vsi uporabniki znajdejo na domači strani.

Na sliki 4.6 so bile onemogočene vse avtorizacijske sheme z namenom, da prikažemo vse možne opcije na levem navigacijskem meniju, ki jih omogoča spletna aplikacija. V nadaljevanju sledi podrobnejši opis vseh podmenijev, kjer so upoštevane tudi avtorizacijske sheme.

4.8 Podmeni *Računi*

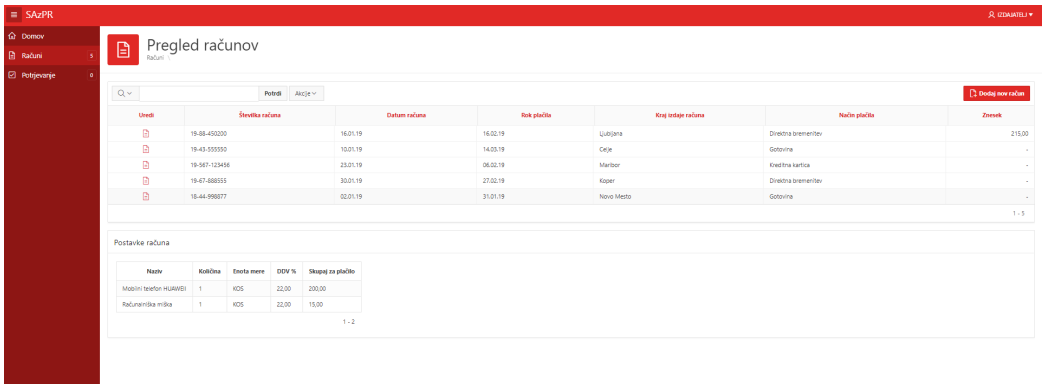
Podmeni *Računi* omogoča vsem uporabnikom pregled vseh računov z njihovimi postavkami. Ob kliku na določen račun, se vrstica obarva v sivo, ob tem pa se spodaj izpišejo vse postavke izbranega računa. Del JavaScript kode, s katero smo dosegli to funkcionalnost:

```
$(item).find('td').each(function () {
    $(this).addClass('izbrana_vrstica');
});
```

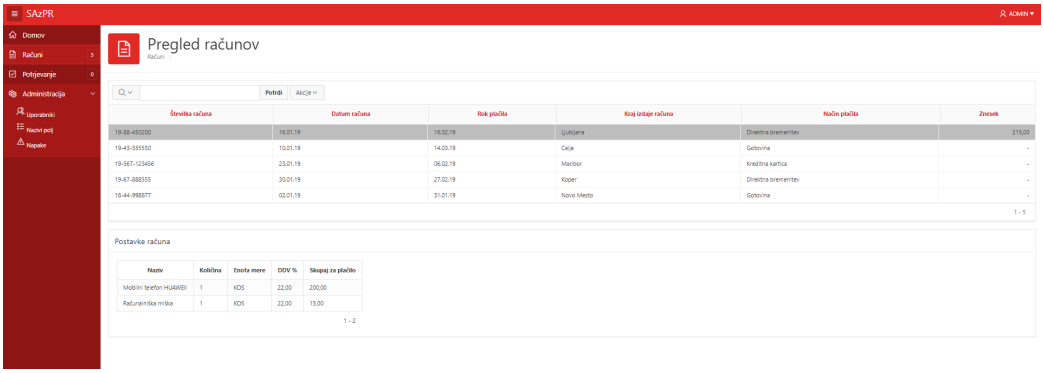
Nato smo na dodan razred dodali še CSS:

```
.izbrana_vrstica{
    background-color: #bdbdbd !important;
}
```

Le uporabniki z vlogo izdajatelja računa lahko urejajo obstoječe račune. To naredijo s klikom na ikono ustreznega računa ali pa ustvarijo novega, s pritiskom na gumb za dodajanje računov. Sliki 4.7 in 4.8 prikazujeta razlike prikaza v vlogi izdajatelja oziroma administratorja. Kot je razvidno iz obeh slik, izdajatelj nima vpogleda v meni *Administracija*, administrator pa nima pravic urejati ali dodajati novih računov.

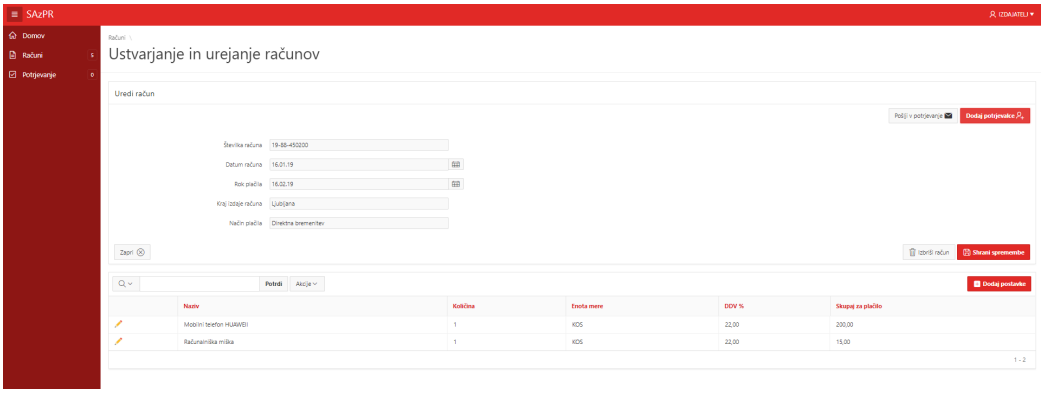


Slika 4.7: Prikaz pregleda računov v vlogi izdajatelja.



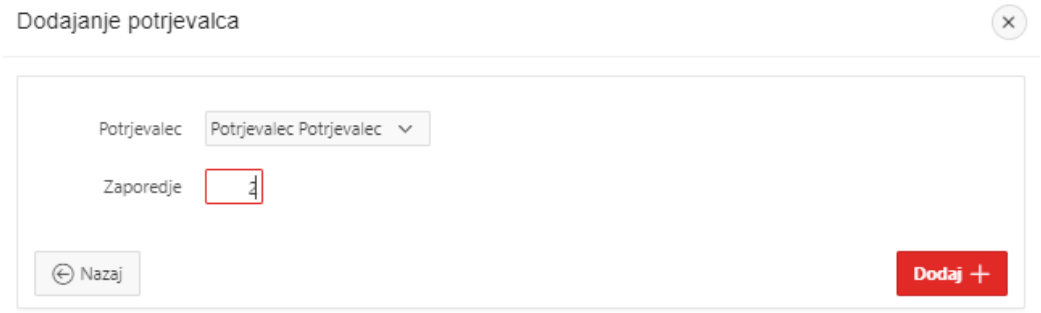
Slika 4.8: Prikaz pregleda računov v vlogi administratorja.

Ob kliku na gumb **Dodaj račun** se nam odpre nova stran, kot je prikazano na sliki 4.9. Na tem mestu lahko na različne načine urejamo račun in njegove postavke, vključno z brisanjem. Zgoraj desno imamo možnosti dodajanja potrjevalcev in pošiljanja računa v potrjevanje. Ob kliku na gumb **Dodaj potrjevalce** se nam najprej prikaže okno, kjer imamo pregled nad že določenimi potrjevalci in datumi njihove potrditve računa (slika 4.10). V tem oknu imamo možnost dodajanja novih potrjevalcev. V tem primeru se nam odpre novo okno (slika 4.11), kjer imamo možnost določiti potrjevalca in zaporedja, kdaj naj bo na vrsti za potrjevanje računa. Že ustvarjene potrjevalce lahko tudi urejamo (slika 4.12).

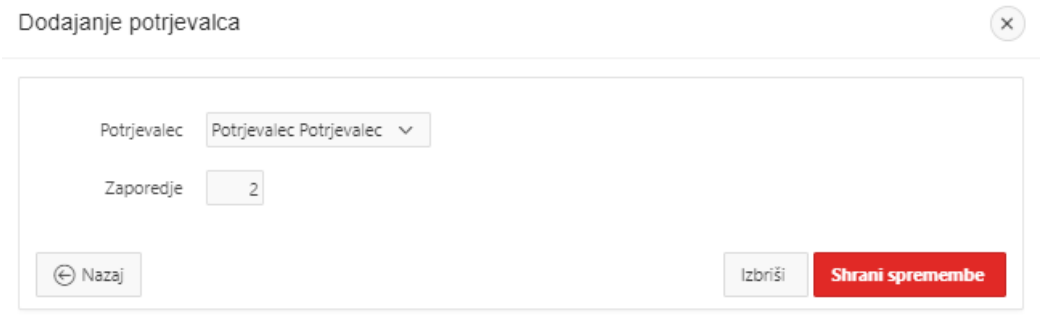


Slika 4.9: Prikaz strani za urejanje računov.

Slika 4.10: Prikaz okna za pregled in urejanje potrjevalcev računa.



Slika 4.11: Prikaz okna za dodajanje potrjevalca in zaporedja potrjevanja ob kliku na gumb za dodajanje potjrevalcev.



Slika 4.12: Prikaz okna za dodajanje potrjevalca in zaporedja potrjevanja ob kliku na ikono za urejanje.

4.9 Podmeni *Potrjevanje*

Ob kliku na gumb **Pošlji v potrjevanje** je potrjevalec z najnižjim zaporedjem prvi na vrsti za potrjevanje računa. O tem je avtomatsko obveščen preko elektronske pošte s sporočilom, da račun čaka na njegovo potrditev. Za avtomatsko obveščanje skrbi prožilec na podatkovni bazi. Ideja tega dela logike obveščanja s pomočjo omenjenga prožilca je naslednja:

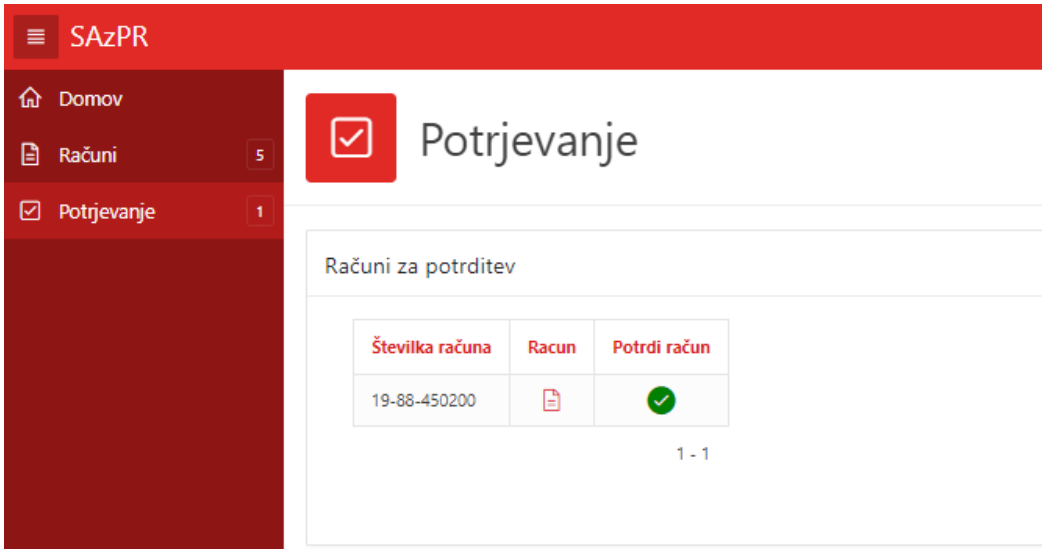
```
TRIGGER RACUN_OBVESTILO
AFTER UPDATE
OF POTRDITEV_STATUS
ON RACUN
DECLARE
  e_podtaki VARCHAR2(1000);
BEGIN
    --če je nova vrednost atributa potrditev_status ustrezna
        --s poizvedbo pridobi elektronski naslov potrjevalca
        --pošlji elektronsko pošto
END;
```

Nato se mora potrjevalec pomakniti na stran za potrjevanje računov preko podmenija *Potrjevanje*. Slika 4.13 prikazuje račun, ki čaka na potrditev. S klikom na ikono računa se premaknemo na stran za pregled računov, s klikom na zeleno kljukico pa se nam odpre novo okno (slika 4.14). Tu lahko, pred potrditvijo računa, še zapišemo komentar. Potrjevanje se zaključi, ko zadnji potrjevalec potrdi račun.

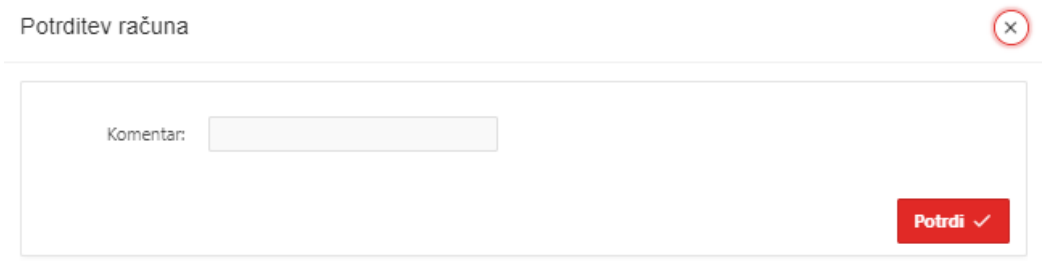
4.10 Podmeni *Administracija*

Podmeni *Administracija* je namenjen za uporabnike z vlogo administratorja in uporabnikom z ostalimi vlogami ni viden. Sestavljajo ga naslednje tri strani:

- Uporabniki,



Slika 4.13: Prikaz računa za potrditev v podmeniju *Potrjevanje*.



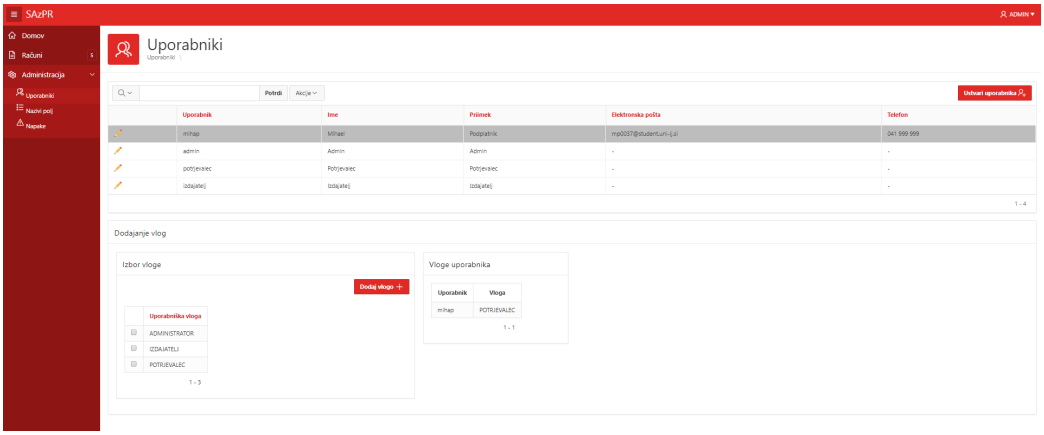
Slika 4.14: Prikaz okna za potrditev računa.

- Nazivi polj,
- Napake.

4.10.1 *Uporabniki*

Na tej strani (slika 4.15) ima administrator pregled nad uporabniki. Čeprav smo implementirali prijavo z uporabo AD uporabniškega imena in gesla, smo zavoljo testiranja aplikacije dodali možnost dodajanja uporabnikov z privzeto APEXovo avtentikacijo. Omogočeno je tudi dodajanje nove vloge novo kreiranim uporabnikom. S klikom na uporabnika se nam vrstica obarva

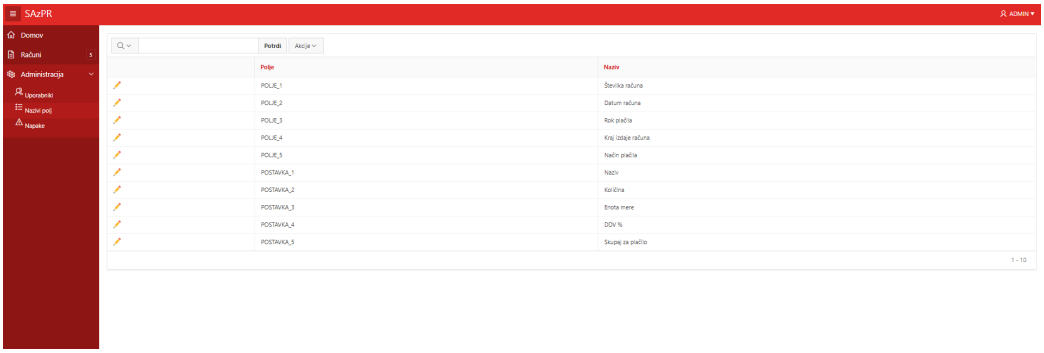
in prikaže se uporabnikova vloga, podobno kot se to izvede v primeru računov in postavk.



Slika 4.15: Stran za pregled, ustvarjanje in urejanje uporabnikov.

4.10.2 *Nazivi polj*

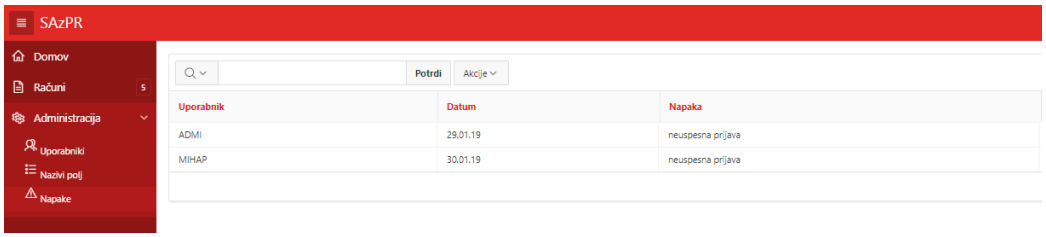
Na tej strani (slika 4.16) izpolnjujemo eno od naših začetnih posebnih zahtev; tj. možnost spremembe nazivov atributov brez poseganja v podatkovni model. S klikom na ikono za urejanje se nam odpre okno, kjer lahko spremenimo nazive in spremembe shranimo v podatkovno bazo, iz katere se nato ti nazivi prikazujejo v ostalih delih spletne aplikacije.



Slika 4.16: Stran za spreminjanje nazivov atributov računov in postavk računov.

4.10.3 *Napake*

V entiteto app_napaka se beležijo določene napake, ki se pojavijo z uporabo spletne aplikacije. Na tej strani (slika 4.17) pa izpisujemo le njeno vsebino. Tako ima administrator boljši pregled nad morebitnimi napakami, ki se lahko pripetijo. Na sliki 4.17 lahko vidimo, da smo se dvakrat neuspešno poskusili prijaviti v aplikacijo.



Slika 4.17: Stran za spremljanje napak.

Poglavje 5

Prenos in nastavitev aplikacije v produkcijsko okolje

V praksi je končni cilj dokončano in preizkušeno aplikacijo prenesti iz razvojnega okolja in jo namestiti v produkcijsko okolje, v primeru, da to okolje ni enako razvojnemu. Z orodjem Oracle APEX je ta postopek enostavnejši, kot če bi to delali ročno, in je sestavljen iz dveh korakov [3]. Najprej je potrebno aplikacijo izvoziti. Oracle APEX generira skriptno tekstovno datoteko s končnico „.SQL“, ki vsebuje celotno aplikacijo vključno s slikami in dodanimi JavaScript ter CSS datotekami. Izvoženo datoteko nato lahko prenesemo na izmenljivi pomnilniški medij, s katerim lahko prenesemo celotno aplikacijo v zasebno okolje uporabnika. V drugem koraku preostane le še uvoz datoteke v produkcijsko okolje uporabnika. Izvoženo aplikacijo lahko namestimo v isto ali katerokoli kasnejšo verzijo Oracle APEXa.

5.1 Izvoz aplikacije

Za izvoz aplikacije je najprej potrebno biti prijavljen v delovno okolje (ang. Workspace) in se pomakniti v zavihek za razvoj aplikacij (ang. App Builder). Nato izberemo aplikacijo, katero želimo izvoziti, in nastavimo polje za format datoteke (ang. File Format) na UNIX oz. na DOS za boljšo pregle-

dnost izvorne kode. Pod izvoznimi nastavitvami (ang. Export Preferences) je priporočljivo vsa polja nastaviti na Da (ang. Yes), da se izvozijo vsi deli aplikacije. Ob kliku na gumb **Izvoz** (ang. Export) se nam shrani datoteka v mapo za prenose.

5.2 Uvoz aplikacije

Pri uvozu aplikacije se v produkcijskem okolju najprej prestavimo v zavihek za razvoj aplikacij, tako kot v prejšnjem razdelku. Pustimo privzete vrednosti za izvoz aplikacije. Nato namestimo aplikacijo s pritiskom na gumb **Namesti aplikacijo** (ang. Install Application).

5.3 Odstranitev orodne vrstice

Med razvojem aplikacije imamo na spodnji strani vidno orodno vrstico, s katero dostopamo do izvora aplikacije in preko katere lahko spreminjamo njen izgled. Te funkcionalnosti niso namenjene za končne uporabnike aplikacije, zato je dobrodošlo, da odstranimo to orodno vrstico. To storimo tako, da pod deljenimi komponentami te aplikacije izberemo opcijo globalizacijski atributi (ang. Globalization Attributes) in se pomaknemo na zavihek definicija (ang. Definition), kjer jo odstranimo.

5.4 Prenos podatkov

Če na produkcijskem okolju nimamo postavljene ustrezne podatkovne baze, je zraven aplikacije potrebno prenesti tudi podatkovne tabele. Na prvi strani najdemo zavihek delavnica SQL (ang. SQL Workshop), kjer pod ikono storitve (ang. Utilities) najdemo storitev imenovano generiraj DDL (ang. Generate DDL), ki nam ustvari še eno skriptno tekstovno datoteko, ki vsebuje SQL kodo za postavitev vseh objektov kot so tabele, prožilci in sekvence. Skripto lahko nato poženemo v programu SQL Developer. Kadar želimo prenesti

tudi podatke, se lahko poslužimo tehnologije imenovane Oracle Data Pump [3] (slov. Oraclova podatkovna črpalka). V ukazni vrstici preko ukaza *expdp* izvozimo podatke, z ukazom *impdp* pa jih uvozimo.

Poglavje 6

Zaključne misli

V sklopu diplomskega dela smo dosegli zastavljen cilj: razviti uporabno spletno aplikacijo za napredno upravljanje s podatki. Pri tem pa smo tudi opisali postopek razvoja z orodjem Oracle APEX in pokazali njegovo uporabnost. Na začetku smo podali motivacijo za razvoj spletne aplikacije z APEXom. Po poglobljeni seznanitvi z orodjem, tehnologijami in po načelih metodologije RAD nam je uspelo v relativno kratkem času razviti predstavljeno spletno aplikacijo za potrjevanje računov, ki omogoča tudi dodatne koristne funkcionalnosti, npr. administracijo. Aplikacijo smo tudi testirali, tako da je pripravljena za uporabo v produkcijskem okolju. Zahvaljujoč orodju APEX in ostalim navedenim tehnologijam, smo uspeli razviti spletno aplikacijo v okviru podanih zahtev in podanega časa.

6.1 Možnosti za nadaljnje delo

Trenutna različica spletne aplikacije za potrjevane računov dopušča mnogo možnosti za izboljšave. Za zaključek jih naštejemo nekaj:

- Mnogo APEX navdušencev nenehno razvija vtičnike za APEX, ki z dinamiko še bolj popestrijo uporabniško izkušnjo. Tako bi lahko uporabili katerega izmed dostopnih vtičnikov, na primer za dodajanje premikajočih gumbov.

- Lahko bi dodali tudi možnost zavrnitve računov.
- APEX omogoča razvoj spletnih aplikacij prilagojenih za pametne telefone, tako da bi lahko pripravili mobilno verzijo te aplikacije.
- Lahko bi onemogočili izdajatelju računov, da samega sebe določi kot potrjevalca.

Možnosti za izboljšave je še več. Trenutna verzija spletne aplikacije pa možnost izboljšav tudi dopušča. S spletno aplikacijo smo dosegli vse zahteve, ki smo si jih zadali ob načrtovanju.

Literatura

[1] Access control lists. Dosegljivo: https://docs.microsoft.com/en-us/windows/desktop/secauthz/access-control-lists. [Dostopano: 27. 1. 2019].

[2] Ad ds getting started. Dosegljivo: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/ad-ds-getting-started. [Dostopano: 13. 1. 2019].

[3] Riaz Ahmed. *Oracle Application Express 5.1 Basics & Beyond*. CreateSpace Independent Publishing Platform, 2017.

[4] Apex architecture. Dosegljivo: https://apex.oracle.com/en/platform/architecture/. [Dostopano: 13. 1. 2019].

[5] What is application express? Dosegljivo: https://www.oracle.com/technetwork/developer-tools/apex/overview/what-is-oracle-apex-3840637.html. [Dostopano: 13. 1. 2019].

[6] Cody Arsenault. The pros and cons of 8 popular databases. Dosegljivo: https://www.keycdn.com/blog/popular-databases. [Dostopano: 13. 1. 2019].

[7] Css. Dosegljivo: https://techterms.com/definition/css. [Dostopano: 13. 1. 2019].

[8] Introduction to oracle database. Dosegljivo: https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm#CNCPT001. [Dostopano: 13. 1. 2019].

[9] Oracle sql developer. Dosegljivo: https://www.oracle.com/database/technologies/appdev/sql-developer.html. [Dostopano: 13. 1. 2019].

[10] Html. Dosegljivo: https://www.w3.org/community/webed/wiki/HTML. [Dostopano: 13. 1. 2019].

[11] Indexes. Dosegljivo: https://www.techonthenet.com/oracle/indexes.php. [Dostopano: 23. 1. 2019].

[12] K. Cannell M. D'Souza D. Gault D. Gielis R. Hartman D. Kubicek R. Mattamal D. McGhan F. Mignault T. Petrus J. Rimblas J. Scott, N. Buytaert and C. Ruepprich. *Expert Oracle Application Express*. Apress, 2015.

[13] Javascript. Dosegljivo: https://techterms.com/definition/javascript. [Dostopano: 13. 1. 2019].

[14] Lightweight directory access protocol. Dosegljivo: https://ldap.com/. [Dostopano: 13. 1. 2019].

[15] Understanding page designer ui elements. Dosegljivo: https://docs.oracle.com/database/apex-5.1/HTMDB/understanding-page-designer-ui-elements.htm#HTMDB29501. [Dostopano: 13. 1. 2019].

[16] Overview of pl/sql. Dosegljivo: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm#i58091. [Dostopano: 13. 1. 2019].

[17] Rapid application development: Changing how developers work. Dosegljivo: https://kissflow.com/rad/rapid-application-development/. [Dostopano: 13. 1. 2019].

[18] Rapid application development. Dosegljivo: https://en.wikipedia.org/wiki/Rapid_application_development. [Dostopano: 13. 1. 2019].

[19] Oracle apex feature. Dosegljivo: https://apex.oracle.com/en/platform/features/. [Dostopano: 13. 1. 2019].

[20] Smtp. Dosegljivo: https://serversmtp.com/what-is-smtp-server/. [Dostopano: 28. 1. 2019].

[21] Kristin Stoller. The world's largest tech companies 2018: Apple, samsung take top spots again. Dosegljivo: https://www.forbes.com/sites/kristinstoller/2018/06/06/worlds-largest-tech-companies-2018-global-2000/#265f93f84de6. [Dostopano: 13. 1. 2019].

[22] Ronny Weiß. Apex particles (1.0). Dosegljivo: https://apex.world/ords/f?p=100:710:4190484818182::::P710_PLG_ID:APEX.PARTICLES. [Dostopano: 4. 12. 2018].