UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Iskra

IZDELAVA SPLETNE APLIKACIJE ZA SPREJEM IN VODENJE STRANK V ESTETSKEM STUDIU Z UPORABO ORACLE APPLICATION EXPRESS

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

MENTOR: viš. pred. dr. Damjan Vavpotič

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00114/2011
Datum: 05.04.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARKO ISKRA**

Naslov: **IZDELAVA SPLETNE APLIKACIJE ZA SPREJEM IN VODENJE STRANK V ESTETSKEM STUDIU Z UPORABO ORACLE APPLICATION EXPRESS**

DEVELOPMENT OF A WEB APPLICATION FOR CUSTOMER MANAGEMENT OF AN AESTHETIC STUDIO WITH ORACLE APPLICATION EXPRESS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V prvem delu diplomske naloge predstavite razvojno okolje za razvoj spletnih aplikacij Oracle Application Express (Oracle Apex). Predstavite tehnično arhitekturo, ki je predvidena pri razvoju s tem orodjem ter prednosti in slabosti razvoja v okolju Oracle Apex. V drugem delu diplomske naloge se posvetite problemu aplikacije za sprejem in vodenje strank v estetskem studiu. Najprej problem analizirajte in načrtujte z uporabo ustreznih diagramskih in drugih tehnik, nato pa predstavite, kako je potekala implementacija aplikacije v okolju Oracle Apex.

Mentor:

viš. pred. dr. Damjan Vavpotič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Marko Iskra,

z vpisno številko 63040451,

sem avtor/-ica diplomskega dela z naslovom:

IZDELAVA SPLETNE APLIKACIJE ZA SPREJEM IN VODENJE STRANK V ESTETSKEM STUDIU Z UPORABO ORACLE APPLICATION EXPRESS

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)
  viš. pred. dr. Damjan Vavpotič

  in somentorstvom (naziv, ime in priimek)

  _____

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se viš. pred. dr. Damjanu Vavpotiču za mentorstvo, posvečen čas ter informacije, ki so pripomogle k nastajanju diplomskega dela.

Za potrpeţljivost in vzpodbudo se zahvaljujem svojim staršem, ki so mi omogočili študij v Ljubljani in me vsa leta podpirali.

POVZETEK

Diplomsko delo opisuje izdelavo spletne aplikacije za sprejem in vodenje strank v estetskem studiu z uporabo orodja oracle APEX. Ideja se je porodila iz potreb družinskega estetskega studia, ker se je z večanjem ponudbe in prometa večala tudi potreba po uvedbi učinkovitega sistema. Implementacija spletne aplikacije bi tako prinesla ekonomske in časovne izboljšave ter posledično finančne prihranke.

V diplomski nalogi so še na kratko opisani metodologija RAD (angl. Rapid Application Development), orodje APEX ter njegove prednosti in slabosti.

Za doseg cilja smo poleg APEX uporabili tehnologije in orodja, kot so HTML (angl. Hyper Text Markup Language), Oraclovo podatkovno bazo, javascript, Toad for Oracle in Power Designer.

ABSTRACT

The diploma thesis describes how to make a web application that will receive and lead customers in the aestethic studio using the oracle APEX tool. The idea was born thanks to a family aestethic studio that needed a new and better system due to the increase of their offer and traffic circulation.
The implementation of the web application would therefore bring economic and time improvements as well as financial savings.

In the diploma thesis are, among others, shortly described the methodology RAD (Rapid Application Development), tool APEX and its advantages and disadvantages.
To reach the goal, we have also used technologies and tools such as HTML (Hyper Text Markup Language), Oracle data base, javascript, Toad for Oracle and Power Design.

Kazalo

Seznam uporabljenih kratic in simbolov

Apache - spletni strežnik
APEX - Oracle Application Express, orodje za hiter razvoj aplikacij
CGI - Common Gateway Interface, je standardni protokol oziroma vmesnik ter označuje skupek pravil, katera določajo kako nek informacijski strežnik komunicira z lokalno programsko opremo.
DBA - administrator podatkovne baze
EPG - Embedded PL/SQL Gateway - Vgrajeni PL/SQL prehod
HTML - HyperText Markup Language - standarden jezik za razvoj spletnih strani. Z jezikom HTML torej označujemo in določamo lastnosti besedila.
HTMLDB - starejša verzija APEX
HTTP - HyperText Transfer Protocol - komunikacijski protokol med odjemalci in strežniki. Protokol je namenjen objavljanju in prejemanju informacij na spletu preko HTML strani.
JavaScript - skriptni programski jezik za razvoj interaktivnih spletnih strani
OC4J - Oracle9iAS Containers for J2EE - Oraclov aplikacijski strežnik
Oracle Database - Oraclova podatkovna baza
OWA - Oracle Web Agent - Oracle spletni zastopnik
PERL - Practical Extraction and Report Language tolmačeni programski jezik
PHP - Hypertext Preprocessor, izvirno Personal Home Page tools - strežniški skriptni jezik.
PL/SQL - Procedural Language/Structured Query Language - proceduralni jezik, ki je podaljšek strukturiranega poizvedovalnega jezika
RAD - Rapid Application development - model hitrega razvoja programskih rešitev
UML - Unified Modeling Language - je splošen vizualni jezik za modeliranje.
URL - Uniform Resource Locator - naslov lokacije objekta, običajno internetne strani.
World Wide Web - svetovni splet ali samo splet

1 Uvod

V uvodnem delu bomo opredelili idejo, namen in cilje diplomske naloge. Opisana je tudi struktura diplomske naloge, ki je razdeljena na 6 poglavij.

1.1 Ideja

V današnjem času si življenja brez informacijske in komunikacijske tehnologije ne moremo več predstavljati. Računalniki, informacijski sistemi in komunikacije so dobesedno postali hrbtenica vsake sodobne družbe. Tako večina podjetnikov, ki še nima aplikacije za vodenje svojega podjetja, zaostaja za svojo konkurenco, kar seveda v končni fazi prinaša izgubo strank in zmanjševanje dobička. Kar je danes splet tako razširjen in praktično dostopen kjer koli, so spletne aplikacije najboljši način za interakcijo, dostopnost do uporabnika in reklamo podjetja.

1.2 Namen

Namen diplomske naloge je:

- na splošno predstaviti spletne aplikacije,
- opisati tehnologije in rešitve za razvoj spletne aplikacije,
- definirati uporabnikove zahteve za spletno aplikacijo,
- sistematično razviti rešitev spletne aplikacije za sprejem in vodenje uporabnikov.

1.3 Cilji

Primarni cilj te diplomske naloge je narediti delujočo spletno aplikacijo. Za dosego tega cilja so potrebni manjši cilji, kot so:

- spoznati ves postopek razvoja spletna aplikacije,
- izbrati orodje in tehnologije za razvoj,
- ugotoviti prednosti in slabosti predstavljenih tehnologij.

1.4 Struktura diplomske naloge

V uvodnem poglavju diplomske naloge so predstavljeni ideja, namen in cilji ter opis strukture po poglavjih.

V drugem poglavju so na kratko predstavljene spletne aplikacije, metodologija RAD (hiter razvoj aplikacij) ter prednosti in slabosti hitrega razvoja aplikacij. Na kratko so opisani jezik PL/SQL, zgodovina orodja APEX ter nove funkcionalnosti verzije APEX, v kateri je narejena izdelana aplikacija.

V tretjem poglavju je podrobneje predstavljeno orodje APEX. Na kratko so opisane arhitektura, komponente in varnost, ki jo omogoča orodje. V nadaljevanju so naštete prednosti, slabosti in priporočila kdaj in kako uporabljati oracle APEX.

Četrto poglavje predstavlja jedro diplomske naloge. Opisani so obstoječe stanje in problemi, definirane so zahteve za izdelavo aplikacije. V nadaljevanju so opisana orodja in tehnologije, ki smo jih (poleg APEX) uporabljali za načrtovanje in razvoj spletne aplikacije, ter postopek izvedbe aplikacije in način uporabe končnega izdelka. Za lažjo predstavitev so dodane slike aplikacije.

V petem poglavju je poudarek na sklepnih ugotovitvah diplomske naloge.

Šesto poglavje so priloge, kje so podane nekatere funkcije in procedure, ki so uporabljene pri izdelavi spletne aplikacije.

Na koncu so navedeni uporabljeni viri.

2 Splet in spletne aplikacije

Svetovni splet (World wide web) je bil predstavljen na začetku 90. let s ciljem omogočiti dostop do informacij iz vseh virov na skladen in enostaven način. Razvit je bil v CERN-u v Ženevi v Švici, namenjen pa je bil fizikom in drugim znanstvenikom, ki so ustvarjali ogromne količine podatkov in dokumentov ter čutili potrebo, da jih delijo z drugimi znanstveniki. Hipertekst (Hypertext) je bil sprejet kot enostaven način za omogočanje dostopa do dokumentov in njihovega povezovanja. Protokol HTTP je bil zasnovan z namenom dopuščanja enemu računalniku – računalniku odjemalca –, da povprašuje po podatkih in dokumentih iz drugega računalnika – strežnika –, tako da je dokument lahko na voljo uporabnikom na računalniku odjemalca. Na tak način je bil svetovni splet viden kot obsežno odlagališče informacij, ki je omogočalo dostop velikemu številu uporabnikom. Ta pogled na splet je bil precej statičen in se je s časom zelo spremenil. Navkljub velikemu napredku v zadnjem desetletju so ostali temeljni principi, na katerih je bil osnovan svetovni splet, konstantni. Svetovni splet strukturalno bazira na računalništvu odjemalec-strežnik, kjer strežniki shranjujejo dokumente, do katerih dostopajo odjemalci. Isti računalnik je lahko odjemalec in strežnik ob različnem času. Svetovni splet je poleg računalništva odjemalec-strežnik predstavil tri temeljne koncepte: metodo poimenovanja in posredovanja dokumentov (URL), jezik za pisanje dokumentov, ki lahko vsebuje podatke in povezave do drugih dokumentov (HTML), protokol za naprave odjemalca in strežnika za medsebojno komuniciranje (HTTP). Splet je bil v svojih začetkih sestavljen iz serije dokumentov, ki so bili lahko med sabo prosto povezani. To so bile preproste besedilne datoteke, ki so vsebovale statično vsebino in statične povezave do strani. Programerji so že zelo zgodaj spoznali, da arhitektura spleta odjemalec-strežnik zagotavlja zmogljivo platformo, v kateri je brskalnik lahko univerzalen uporabniški vmesnik do aplikacij, ki lahko tečejo lokalno ali oddaljeno na strežniku. Za ohranitev razmerij brskalnik-strežnik je spletna stran vedno vrnjena iz strežnika k brskalniku, vendar je ta spletna stran lahko programsko generirana kot rezultat procesiranja na strežniku. Strežnik bi lahko na primer prejel podatke iz podatkovne baze, jih formatiral v stran HTML in to stran poslal odjemalcu. Ideja procesiranja na strežniku in dinamično generiranih strani je privedla do aplikacij CGI (common gateway interface), kjer je bil URL s strani odjemalca posredovan "skripti" na strežniku. Skripta je bila lahko napisana v specializiranem jeziku, kot so PERL ali celo večji kompajlirani programi, ki so tekli na strežniku za dinamično generiranje spletnih strani. Ta realizacija je privedla do neke vrste hibridnega modela, kjer je bil jezik PHP zgodnji vpliv. S PHP je tipična struktura ustvarjanje

spletnih strani, ki imajo majhne dele kode direktno v tekstu strani. V času zahteve bi bila koda izvršena in njeni rezultati vstavljeni v trenutni dokument. To je privedlo do veliko večje fleksibilnosti in lažje ponovne uporabe komponent strani, s časom pa so ljudje spoznali, da je pogosto težko spremeniti kodo, ko je ta razširjena po vsej spletni strani. Zaradi tega je bil model še dodatno izboljšan v to, kar je tipično za današnje spletne aplikacije, kjer osrednja komponenta v aplikaciji upravlja dostop do podatkov, koda, razpršena po spletnih straneh, pa je omejena samo na to, kar je prikazano uporabniku. Tipična struktura spletne aplikacije sestoji iz podatkov HTML, prikazanih uporabniku, skript odjemalca, ki tečejo na odjemalcu in lahko komunicirajo z uporabnikom, ter skript strežnika, ki opravljajo procesiranje na strežniku in običajno komunicirajo s podatkovnimi bazami. Fraternali je v raziskavi, ki je zajemala zadnje stanje razvoja spletnih aplikacij v letu 1999, opisal spletno aplikacijo kot hibrid med »hipermedijo« in informacijskim sistemom. Posledično je za spletne aplikacije določil naslednje zahteve [2]:

- nujnost upravljanja tako strukturiranih (npr. podatkovni zapisi) kot nestrukturiranih podatkov (npr. večpredstavnostni predmeti),
- podpora raziskovalnemu dostopu z navigacijskimi vmesniki,
- visoka raven grafične kvalitete,
- prilagodljivost in morebitna dinamična prilagoditev strukture vsebine, primitivnost navigacije in stili predstavitev,
- podpora proaktivnemu obnašanju, tj. za priporočila in filtriranje.

2.1 Tipi in prednosti spletnih aplikacij

V prejšnjem poglavju smo definirali spletno aplikacijo kot hibrid med »hipermedijo« in informacijskim sistemom. Seveda spletne aplikacije niso omejene na en tip aplikacij. Zajemajo vse od enostavnih statičnih spletnih strani do zapletenih aplikacij. Različne kategorije spletnih aplikacij so razvrščene glede na podatke in zapletenost nadzora, kot je prikazano na sliki 1:

Slika 1: Taksonomija spletnih aplikacij

1. »Brošurne« spletne aplikacije: To je prva generacija spletnih aplikacij. Sestavljene so iz statičnih spletnih strani in načeloma ne vsebujejo veliko programerske logike. Pri njihovem razvoju je poudarek na razvoju vsebine, grafičnega načrtovanja in besedila. Primer so osebne spletne strani, ki vsebujejo rezime in osebne podatke ali spletne strani o produktu družbe.
2. Storitveno usmerjene aplikacije: Te strani ponujajo storitev uporabnikom spleta. Storitveno usmerjene aplikacije vsebujejo programersko logiko, potrebno za implementacijo storitve. Postavitev podatkov pogosto ni prvotnega pomena. Med vzdrževanjem razvijalci potrebujejo dobro razumevanje nadzorne logike. Primer so storitve spletne pošte ali sistemi spletnega procesiranja besed.
3. Podatkovno intenzivne aplikacije: Te strani zagotavljajo vmesnik za brskanje in povpraševanje po velikih količinah podatkov. Poudarek pri teh aplikacijah je na podatkih, vključena je minimalna količina programerske logike. Med vzdrževanjem morajo razvijalci dobro razumeti podatkovni tok. Primer tega aplikacijskega tipa je katalog spletne knjižnice.
4. Aplikacije informacijskih sistemov: Te aplikacije združujejo storitveno usmerjene aplikacije in podatkovno intenzivne aplikacije. Razvijalci aplikacij informacijskih sistemov se ukvarjajo s podatkovnim tokom (za brskanje in pridobivanje podatkov) in nadzornim tokom (za različne faze, vključene v upravljanje s podatki). Elektronske knjigarne in spletno bančništvo sta primera tega tipa aplikacij.

Glavne prednosti spletnih aplikacij

- Neodvisnost od operacijskega sistema. Spletna aplikacija teče na centralnem strežniku, zato je popolnoma neodvisna od uporabnikovega računalnika, njegovega operacijskega sistema in nastavitev.

- Ne potrebujejo namestitve. To omogoča vzdrževalcem veliko fleksibilnost in odzivnost v primeru okvare računalnikov.

- Ne potrebuje dodatne programske opreme. Protokol HTTP, uporabljen v spletnih aplikacijah, je standardni protokol, ki se lahko giblje prek požarnega zidu podjetij. Edina programska oprema odjemalca je spletni brskalnik. Spletne aplikacije so prav tako na voljo na več platformah, spletni brskalniki pa so danes vključeni v vse operacijske sisteme.

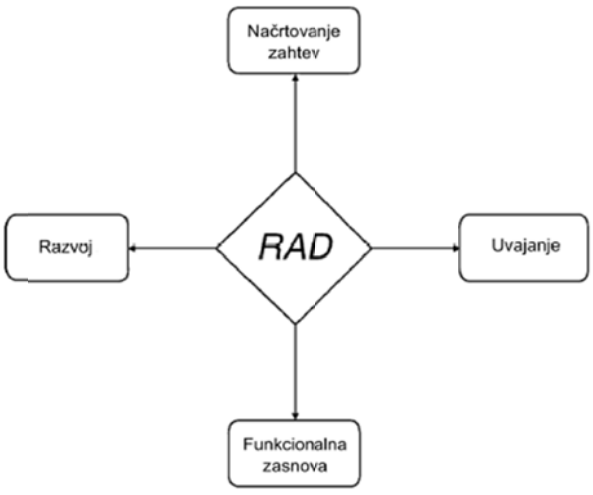
- Hitra in temeljita izvedba nadgradenj. Spletne aplikacije se lahko vzdržuje s prilagoditvijo kode, ki leži na strežniku. To zmanjša čas in stroške nadgradnje ter razvitje spletnih aplikacij v primerjavi s klasičnimi aplikacijami odjemalec-strežnik.

- Centraliziran nadzor in ustvarjanje varnostnih kopij.

2.2 Model hitrega razvoja aplikacij

Beseda RAD ali Rapid application development je termin, ki zaznamuje ves nabor orodij, tehnik, metod in celo stilov razvijanja informacijskih sistemov [1]. James Martin je leta 1991 definiral RAD in njegove cilje: »RAD pomeni poslovne potrebe, izdelati programske rešitve v čim krajšem času in s čim nižjimi stroški.« [8]. Model hitrega razvoja programskih rešitev (RAD) uporablja minimalno načrtovanje in iterativni razvoj v korist hitre izdelave prototipov. Razvoj prototipa pomaga analitikom in uporabnikom preverjati poslovne zahteve in formalno izboljšati postopke razvoja programskih rešitev. Za hitro izdelavo prototipa se pogosto uporabljajo tako imenovana orodja CASE (ang. Computer Aided Software Engineering Tools), ki omogočajo pretvorbo uporabnikovih zahtev v podatkovni model in naprej v bazo podatkov. Nekatera orodja CASE omogočajo celo generiranje kode in s tem celo izgradnjo prototipa in ne zgolj njegove zgradbe (ang. designa). Model RAD je združitev različnih strukturiranih tehnik, še posebno na osnovi informacijskih podatkov in prototipnih tehnik za pospešen razvoj programskih rešitev [11]. Tako lahko podjetje z modelom RAD uspešno razvija programsko rešitev v nestabilnem okolju in z nejasnimi poslovnimi zahtevami. Model RAD je bil odgovor na zaporedno metodologijo razvoja programskih rešitev, pri katerih so se lahko zaradi dolgotrajnega razvoja programskih rešitev zahteve spremenile, preden je bila

programska rešitev v celoti razvita, kar pogosto povzroči neuporabno programsko rešitev [9]. Zato se model RAD pogosto uporablja v primerih časovne omejitve, v katerih ima hitrejši razvoj programskih rešitev prednost pred popolno funkcionalnostjo in učinkovitostjo. Model RAD lahko omogoči kompromis pri funkcionalnosti in zmogljivosti programske rešitve v zameno za hitrejši razvoj in lažje vzdrževanje programske rešitve [8]. Model RAD razdeli razvoj programske rešitve na različne zahteve razvoja programske rešitve z različnimi prioritetami. To omogoči, da se v primeru pomanjkanja časa za razvoj programske rešitve zahteve z manjšo prioriteto premaknejo v naslednji časovni okvir razvoja programske rešitve [5]. Struktura življenjskega cikla metode RAD je zasnovana tako, da zagotovi razvijalcem gradnjo programske rešitve, ki jo uporabniki resnično potrebujejo. Model RAD vsebuje štiri faze [5], ki vključujejo vse dejavnosti in naloge, potrebne za opredelitev obsega in poslovnih zahtev programske rešitve, ter razvoj, izvajanje in uporabo programske rešitve (Slika 2). Načrtovanje zahtev – ta faza opredeli poslovne funkcije in zahteve, ki jih bo podpirala programska rešitev. Funkcionalna zasnova – v tej fazi se razvije delujoči prototip programske rešitve, ki vsebuje modele procesov in zahtev končne programske rešitve. Razvoj – ta faza zaključi razvoj programske rešitve s pomočjo razvoja in izboljšav prototipov. Prototip programske rešitve uporabniki preizkušajo in vračajo razvojni ekipi za popravilo ali predelavo programske rešitve toliko časa, dokler se prototip ne razvije v končno programsko rešitev. Razvoj prototipa se tako uporablja za vizualno pomoč uporabnikom in zahtevam programskim rešitvam ter omogoča ponavljajoči razvoj programskih rešitev. Uvajanje – ta faza vključuje preizkušanje in usposabljanje končnega uporabnika za pravilno uporabo končne programske rešitve.



Slika 2: Model RAD

2.3 Orodja za hiter razvoj aplikacij

Ob koncu devetdesetih let prejšnjega stoletja so se v želji po čim hitrejšem, cenejšem in hkrati še vedno uspešnem razvoju informacijskih rešitev pojavila tako imenovana orodja za hiter razvoj programskih rešitev (Rapid Application Development Tools – orodja RAD). Orodja so si začela utirati pot v različnih programerskih hišah in različnih programskih jezikih. Od takrat so se orodja precej spremenila, osnovni koncepti pa so ostali zelo podobni. Razvili so se v različne agilne metodologije, ki pa so v svojem bistvu zelo podobne izvorni metodologiji RAD. Začetki RAD so sicer stari več kot 20 let, a dejansko še vedno predstavljajo metodološko osnovo praktično vsem modernim stilom agilnega razvijanja informacijskih rešitev [6]. Glavne prednosti RAD so informacijske rešitve visoke kakovosti, ki se razvijejo hitro in z nizkimi stroški. Moderna orodja za hiter razvoj programskih rešitev (ang. Tools for Rapid Application Development) običajno danes predstavljajo kombinacijo programskega jezika četrte generacije in grafičnega vmesnika za izdelavo gradnikov programskih rešitev, kot so okna, seznami, gumbi [7]… Najbolj znani primeri orodij za hiter razvoj informacijskih rešitev so MS Visual Studio (Visual Basic, Visual C#), Oracle Application Express – APEX, Powerbuilder, Jbuilder itn.

2.4 Prednosti in slabosti hitrega razvoja aplikacij

Glavni prednosti hitrega razvoja informacijskih rešitev sta krajši čas izdelave in izboljšana kakovost informacijskih rešitev [6].

Krajši čas izdelave

Ta prednost hitrega razvoja informacijskih rešitev je posledica dveh lastnosti: uporabe orodij CASE, s katerimi se definirane uporabniške zahteve zelo hitro pretvorijo v programsko kodo, in uporabe časovne omejitve, zaradi katere se nove verzije prototipa razvijajo zelo hitro, čeprav (še) ne podpirajo vseh dogovorjenih funkcionalnosti.

Izboljšana kakovost

V skladu metodologije RAD [8] kakovost pomeni tako stopnjo ustreznosti informacijske rešitve za končnega uporabnika (to zajema tudi morebitne napake in ne zgolj ustreznosti načrta informacijske rešitve) kot tudi nizke stroške vzdrževanja ter izdelave aplikacije. Obe merili sta izpolnjeni, saj so stroški izgradnje nizki zaradi velike avtomatizacije postopkov,

stroški vzdrževanja pa zaradi razširjenosti programskih okolij, v katerih so rešitve izdelane. To seveda pomeni lažje popravke oziroma možnost nakupa novih komponent. Hiter razvoj informacijskih rešitev ima seveda tudi svoje pomanjkljivosti. Glavni sta predvsem dve: stopnjevana zmogljivost (zmanjšana zmogljivost ob povečanju števila uporabnikov) in zmanjšana funkcionalnost [6].

V tabeli 1 je podan krajši seznam prednosti in slabosti hitrega razvoja informacijskih rešitev [10] :

Prednosti in slabosti hitrega razvoja informacijskih rešitev	
PREDNOSTI	**SLABOSTI**
Veliki prihranek časa in denarja.	Zaradi prihranka časa in denarja je lahko kakovost informacijske rešitve nižja kot sicer.
Večja povezava med potrebami končnih uporabnikov in sistemskimi zahtevami.	Nekonsistentna notranja zasnova sistema.
Možnost hitrih nadgraditev.	Posameznih kupljenih modulov včasih ni mogoče hitro nadgraditi.
Možnost nakupa komponent.	Komponente so lahko zelo drage oziroma jih je občasno treba precej spremeniti za lastne potrebe.
Morebitne napake v konceptih se odkrijejo hitro že med samo izgradnjo.	Koda je manj učinkovita, saj je generirana.
Večja fleksibilnost projekta.	Ker se ne uporabljajo formalistične metode, manjka znanstvena natančnost.
Krajši razvojni cikli (veliki poudarek na časovnem okvirju).	Možna manjša funkcionalnost zaradi uporabe že narejenih komponent ali pomembnosti časovnega okvirja.
Standardizirana občutek in videz.	Standardiziran videz lahko povzroči številne zmote in zamenjave.

Tabela 1: Prednosti in slabosti hitrega razvoja aplikacij

2.5 Zgodovina Oracle APEX

APEX obstaja že dolgo časa. Prva javna izdaja APEX oziroma HTML DB, kot se je imenoval takrat, se je zgodila leta 2004, vendar pa njegova zgodovina seže še dlje v preteklost. Izdaja HTMLDB 1.5 je bila javnosti ponujena kot verzija Oracle 10gR1 brez stroškov. Od takrat naprej so bile predstavljene različne nove izdaje, vsaka naslednja je izboljšala lastnosti in funkcionalnost.

Zelo kratek seznam izdaj in njihovih vidnejših lastnosti [13]:

• HTMLDB 1.6 (2004) predstavi teme, grupiranje strani, zaklepanje strani, obrazce master-detail in določene večjezične možnosti.

• HTMLDB 2.0 (2005) predstavi SQL Workshop, grafično izgradnjo poizvedb, iskalec po podatkovni bazi in zaščito stanja seje.

• APEX 2.2 (2006) predstavi arhivirane aplikacije, poglede v slovar APEX in čarovnika za dostop.

• APEX 3.0 (2007) predstavi tiskanje PDF-datotek z BI Publisherjem, možnost prenosa podatkov iz MS Access in pomnjenje strani ter regij.

• APEX 3.1 (2008) predstavi interaktivna poročila, možnost »runtime-only« namestitev in izboljšano varnost.

• APEX 3.2 (2009) predstavi pomočnika za prenos podatkov iz sistemov na osnovi Oracle Forms in precej izboljšav varnosti.

Nato se pojavi različica **Oracle APEX 4.0 (leto 2010).** Ta ponudi uporabniku številne nove funkcije, kot so:

- *Spletne preglednice*

  Spletne preglednice omogočajo vsem uporabnikom hiter razvoj in oblikovanje spletnega vnosa podatkov in aplikacij poročanja. S spletnimi preglednicami je ustvarjanje tabel, sprožilcev in sekvenc avtomatsko. Spletne preglednice ponujajo enostaven, pojasnjevalen pristop k poročanju in postavitvi obrazca in prav tako k seznamu ustvarjanja vrednosti in veljavnosti.

- *Skupinski razvoj*

  »Team Development« je vgrajeno orodje za upravljanje projektov, ki s sledenjem novih funkcij, raznih opravil, hroščev in mejnikov omogoča upravljanje procesa razvoja programske opreme. Uporabniki lahko prispevajo povratne informacije v realnem času. Te so lahko razporejene v funkcije, splošna opravila ali hrošče.

- *Izboljšani diagrami*

Izdaja 4.0 vključuje integracijo AnyChart 5.1, kar poskrbi za izboljšano zmogljivost, zmanjšan čas opravljanja in zmanjšan čas posodabljanja. Poleg dodane podpore za mape in Ganttove diagrame ta izdaja vključuje podporo za drsno premikanje, interaktivne oznake, zaznamke v legendah, številne podatkovne zaznamke in lokalizacijo menija vsebine.

- *Izboljšan graditelj aplikacij*

  Graditelj aplikacij se odlikuje z novim videzom in občutkom. Med izboljšanja uporabniškega vmesnika spadajo izboljšana navigacija, na novo zasnovani administrativni zagoni, obsežna uporaba interaktivnih poročil in vgrajene zmožnosti iskanja po aplikaciji.

- *Izboljšana interaktivna poročila*

  Interaktivna poročila zdaj podpirajo poglede ikon, podrobnosti in koledarja, notranje urejanje, sestavljene filtrirane izraze, obveščanje po elektronski pošti in novo skupino po funkcionalnosti. Poleg tega vsako poročilo vključuje izboljšane možnosti shranjevanja, lahko je preneseno na HTML z možnostjo iskanja in ponuja bolj zrnaste zmožnosti tiskanja.

- *Dinamične akcije*

  Dinamične akcije razvijalcem ponujajo način napovedi definiranja obnašanja stranke brez potrebe po JavaScriptu. Z uporabo preprostega čarovnika lahko razvijalci izberejo predmet strani, stanje, vnesejo vrednost in izberejo dejanje (npr. prikaži, skrij, omogoči, prikaži vrstico predmetov).

- *Priključki*

  Priključki razvijalcem omogočajo izboljšanje obstoječe vgrajene funkcionalnosti s pisanjem komponent PL/SQL za tipe predmetov, področij in procesov.

- *Oracle Application Express Poslušalec*

  Izdaja 4.0 vključuje nov spletni strežnih HTTP. Ta spletni strežnik temelji na Javi. Oracle Application Express Poslušalec vključuje predpomnjenje datotečnega sistema, podporo za pretvorbe v PDF, ponuja izboljšano nalaganje datotek in je certificiran proti Web Logic, Tomcat in OC4J z Oracle WebLogic strežnikom, OC4J in Oracle Glassfish strežnikom. Arhitekturo bomo podrobneje opisali v naslednjem poglavju.

- *Format časovnega žiga aplikacije in format časovnega žiga TZ*

  Izdaja 4.0 za razvijalce vključuje možnost nastavitve časovnega pasu za uporabnika med sejo Application Express in možnost nastavitve časovnega pasu samodejno iz brskalnika odjemalca.

- *Izboljšave »drevesa«*

Izdaja 4.0 vključuje podporo za nove »drevesne komponente« z uporabo jsTree, komponente brskalnika, ki temelji na JavaScriptu in podpira teme, opcijsko navigacijo s tipkovnico in možnost shranitve stanja.

- *Izboljšave tem*

Oracle Application Express vključuje 20 tem. Dvanajst tem je bilo posodobljenih z novim videzom. Te teme ponujajo moderen stil, ki uporablja strukturo strani DIV in teče v »standardnem načinu«, kot je definirano v DOCTYPE na vsaki predlogi strani.

2.6 Jezik PL/SQL

PL/SQL je kratica za proceduralni jezik, ki je podaljšek strukturiranega poizvedovalnega jezika – SQL [12]. Podjetje Oracle je predstavilo PL/SQL zato, da odpravi pomanjkljivosti in omejitve SQL in zagotovi popolnejše programske rešitve za tiste, ki nameravajo graditi projektno-kritične aplikacije na osnovi baznega sistema Oracle. PL/SQL ima določene karakteristike [12]:

- Je ugnezden programski jezik, kar pomeni, da ni mišljen kot samostojen jezik, ampak kot del jezika, ki ga pokličemo z gostiteljskega okolja. Gostiteljsko okolje lahko predstavlja baza prek vmesnika SQL*Plus ali Razvijalca na bazi Oracle (angl. Oracle Developer).
- Je visoko zmogljiv in visoko integriran bazni jezik – lahko uporabimo veliko programov, ki tečejo na bazi Oracle. Uporabimo lahko Javo, Visual Basic, Delphi, C++ za izvrševanje programskih rešitev na osnovi Oracle. Vendar je v primerjavi z drugimi jeziki mnogo laže pisati učinkovitejšo kodo s programskim jezikom PL/SQL.
- Je visoko strukturiran, berljiv in dostopen jezik – modeliran je bil po programskem jeziku Ada. Je odličen jezik za tiste, ki začenjajo programirati, in zelo hitro prilagodljiv za tiste, ki so programirali že v drugih jezikih. Dostopnost jezika pomeni tudi to, da na lahek način pišemo kodo in jo skozi čas enostavneje vzdržujemo.
- Je standarden in prenosljiv jezik za razvoj, ki dela na osnovi podatkovne baze Oracle. Če pišemo proceduro ali funkcijo na osebnem računalniku, jo z lahkoto prenesemo na omrežje podjetja, kjer delamo, ob predpostavki, da sta podatkovni bazi Oracle enakih verzij. »Napiši enkrat, poženi kjer koli« je bilo osnovno vodilo jezika PL/SQL, še preden se je pojavila Java.

Prav tako sistem Oracle nudi veliko zmogljivostnih lastnosti v okviru PL/SQL jezika (npr. zanka FORALL) in kmalu ugotovimo, da je pisanje v PL/SQL preprostejše, dosegamo višje dostopne čase baze in lažje vzdržujemo programsko kodo. Zgodovina razvoja programske kode PL/SQL sega v leto 1991, ko je podjetje Oracle izdalo bazni sistem Oracle verzije 6.0.

Istega leta je podjetje Oracle izdalo pričakovani programski paket SQL*Forms, ki se danes imenuje Oracle Forms ali Forms Developer. Prva izdaja programskega jezika PL/SQL je bila zelo omejena, saj nisi mogel shranjevati procedur ali funkcij za kasnejšo izvedbo, vendar je bil kljub mnogim pomanjkljivostim programski jezik PL/SQL lepo sprejet med razvijalskimi skupinami.

3 Oracle Application Express

3.1 Kaj je Oracle Application Express (APEX)

Oracle Application Express, na kratko APEX, ali, kot se je imenoval nekoč, HTML DB, je **deklarativen**, **spletni RAD** (Rapid Application development – orodje za hiter razvoj aplikacij). Uporablja se za razvoj **spletnih** aplikacij, ki so **osredotočene na podatke** [4]**.**
Razložili bomo nekaj izrazov, da bo lažje razumeti, kaj pomenijo v kontekstu APEX.

Spletni

Omemba »spletnega« se pojavi tako v opisu razvijalskega orodja kot tudi izdelka – aplikacije APEX. Tako za razvoj kot uporabo naše aplikacije namreč uporabljamo spletni brskalnik. Prednosti so očitne. Spretni brskalniki so skoraj privzeto nameščeni na večini delovnih postaj, ki jih uporabljamo, neodvisno od strojne opreme in operacijskega sistema. To pomeni, da na strani uporabnika ne potrebujemo ničesar več. Da lahko začnemo razvijati in uporabljati aplikacije APEX. Poleg tega pomeni tudi izredno lahek dostop do naših aplikacij, tako prek intraneta kot prek interneta. Če dodamo še sodobne pametne telefone in podobne izdelke s trga, ki imajo že nameščene spletne brskalnike, bomo prišli do zaključka, da so možnosti za dostop do aplikacij APEX aplikacij. Še ena pomembna prednost ob uporabi spletnega brskalnika je svoboda, saj nismo omejeni z lokalno strojno opremo in operacijskim sistemom. Ker so aplikacije APEX generirane v kodi HTML, nam (razvijalcem) ni treba skrbeti za strojno opremo in operacijski sistem naših končnih uporabnikov. Dokler imamo dostop do brskalnika, ki podpira HTML, ne bo težav.

Osredotočenost na podatke

APEX je bil ustvarjen z namenom izdelovanja aplikacij, ki bi hranile, iskale, spreminjale in prikazovale podatke iz Oraclovih podatkovnih baz. To pomeni, da če glavna logika naše aplikacije ni delo s podatki, potem APEX ni optimalno orodje za njen razvoj. Če pa sta delo in manipulacija s podatki v središču logičnih procesov naše aplikacije, bo APEX najvarnejša

stava. APEX je pravzaprav zbirka paketov PL/SQL, ki »živijo« znotraj Oraclovih podatkovnih baz. To pomeni, da APEX avtomatično pridobi vse lastnosti, pogosto omenjane z okoljem Oraclovih podatkovnih baz: vzdržljivost, zanesljivost, izboljšana varnost, dobro delovanje in še več. Poleg tega APEX lahko izkoristi bogato okolje SQL in PL/SQL in z uporabo vgrajenih paketov, ki jih ponujajo Oraclove podatkovne baze, manipulira s podatki na kar najboljši način, kar ga RDBMS (sistem za upravljanje relacijskim podatkovnim bazam) lahko ponudi.

Deklarativno razvijalsko orodje

APEX je deklarativno orodje. To pomeni, da se kot razvijalci aplikacij bolj zberemo na »kaj je treba narediti« in manj na »kako to narediti«. Pomislimo za trenutek na SQL. V izjavi SELECT, ko uporabimo ukaz ORDER BY, pravzaprav podatkovni bazi s tem povemo, kaj potrebujemo – razvrščene podatke, ne povemo pa ji, kako naj te podatke razvršča. Deklarativno delovanje v APEX pomeni, da ne generiramo tradicionalne programske kode (3GL). Nasprotno, delamo s serijo čarovnikov in listov z lastnostmi ter tako lahko določimo vsa pravila, ki jih potrebujemo za generacijo kode. V APEX je vključena serija čarovnikov, podprtih objektov HTML, podprtih objektov in podatkovnih tipov za podatkovne baze, postopkov in možnosti za prikaz strani, procesov po potrditvi in DML možnosti in še mnogo več. Z vsem tem lahko povežemo naše obrazce, poročila, grafikone, itd. Z njihovim videzom in uporabo/poslovno logiko. APEX jih prevede v kodo HTML na uporabnikovi strani in v kodo SQL ter PL/SQL na strani strežnika.

3.2 Arhitektura APEX

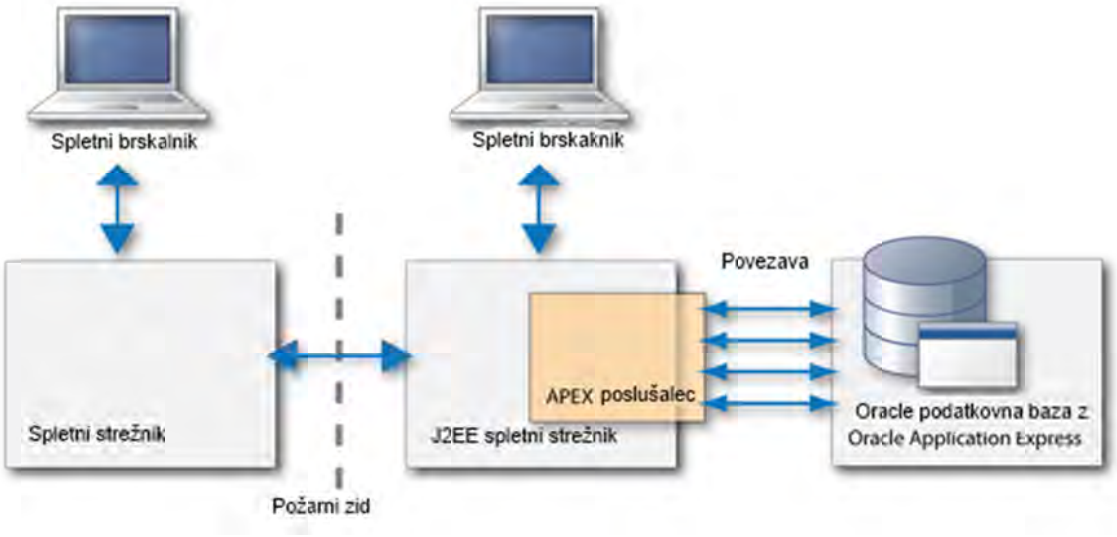
Mehanizem Application Expressa naredi aplikacije v realnem času iz podatkov, ki so shranjeni v tabelah v podatkovni bazi. Ko ustvarimo ali razširimo aplikacijo, Oracle Application Express ustvari ali spremeni metapodatke, shranjene v tabelah. Ko se zažene aplikacija, Application Express prebere metapodatke in prikaže aplikacijo. Application Express živi povsem v podatkovni bazi Oracle. Sestavljen je iz istega števila podatkov, ki so v tabelah, in velike količine kode PL/SQL. Jedro Oracle Application Express sestavlja približno 450 tabel in 230 paketov PL/SQL, ki vsebujejo več kot 425.000 vrstic kode. Za dostop do Oracle Application Express v podatkovni bazi Oracle moramo konfigurirati Oracle HTTP Server (Apache), ki vsebuje mod_plsql dodatek. Dodatek deluje kot posrednik za komunikacijo med spletnim strežnikom in Oracle Application Express objekti v podatkovni

bazi Oracle. Z Oracle podatkovno bazo 11.1 je v arhitekturi strežnik HTTP (Apache) zamenjan z vgrajenim prehodom PL/SQL. Vgrajeni prehod PL/SQL prehod teče na XML DB HTTP strežniku v podatkovni bazi Oracle in vsebuje osnovne funkcije mod_plsql, vendar ne zahteva strežnika Apache HTTP.

Obstajajo tri možnosti spletnega strežnika, ki se lahko uporablja z Oracle Application Express [13]:

1. Oracle Application Express Listener (APEX poslušalec)

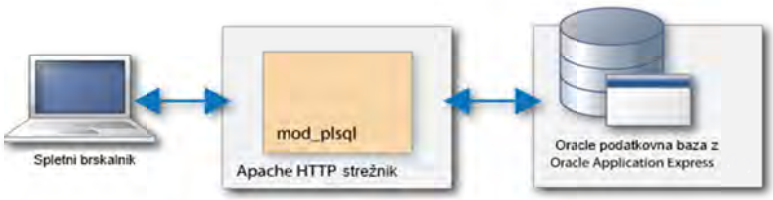
APEX poslušalec je zgrajen v Javi in nameščen v strežniku J2EE Web (slika 3).



Slika 3: Oracle Application Express uporablja APEX poslušalec

2. Oracle HTTP strežnik (Apache) in mod_plsql

Mod_plsql je Apachejeva razširitev za Oracle HTTP strežnik in prek njega komuniciramo z podatkovno bazo. Oracle HTTP strežnik lahko namestimo na istem strežniku kot podatkovno bazo ali na več strežnikov (slika 4).



Slika 4: Oracle APEX uporablja Oracle HTTP strežnik (Apache) z mod_plsql

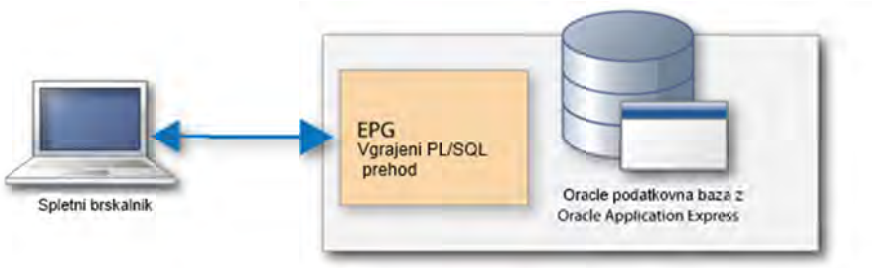
Mod_plsql je bil prej imenovan Oracle PL/SQL kartuša in OWA (Oracle Web Agent – Oracle spletni zastopnik). Privzeto se mod_plsql zaganja, ko se zažene Oracle HTTP Strežnik (Apache). Ni posebnih ukazov, da se ga ustavi in zažene. Standardni Oracle PL/SQL programi se lahko razširijo na mod_plsql programe. Programiranje se izvede v PL/SQL z uporabo naslednjih pakiranih postopkov:

- HTP – Hipertekst procedure,
- HTF – Hipertekst funkcije,
- OWA_UTIL – Oracle spletni zastopnik pripomočkov,
- OWA_COOKIE – Pošlji in pridobi piškote.

Jezik PL/SQL je podrobneje opisan v poglavju 2.6.

3. Vgrajeni PL/SQL prehod (EPG)

Vgrajeni PL/SQL prehod je značilnost Oracle Database 11g. EPG sestavlja Oracle podatkovna baza s spletnim strežnikom preostalih potrebnih infrastruktur za ustvarjanje dinamičnih aplikacij. EPG deluje v XML Database HTTP strežniku, ki je del Oracle Database in vsebuje osnovne funkcije mod_plsql, vendar ne zahteva ločenega spletnega strežnika. Oracle Database 10g Express Edition (XE) prav tako uporablja EPG, kot je prikazano na sliki 5.



Slika 5: Oracle Application Express uporablja vgrajeni PL/SQL prehod

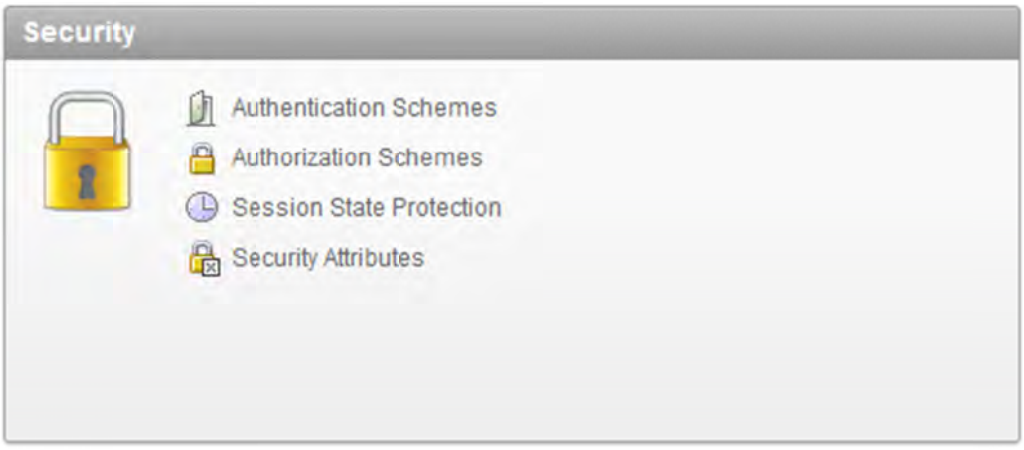
3.3 Varnost

Pri ustvarjanju aplikacij je varnost izredno pomembna, saj preprečuje nepooblaščen dostop in delovanje v naših aplikacijah. Varnostni ukrepi niso potrebni za vse aplikacije, javna spletna stran jih na primer ne potrebuje. A vseeno obstaja ogromno aplikacij, kjer je treba postaviti pravila glede tega, kdo lahko dostopa do njih in jih zažene. Potem ko uporabnik zažene aplikacijo, je treba določiti pravila, do katerih funkcij lahko dostopa. V APEX so te varnostne nastavitve vzpostavljene prek avtentifikacijskih in avtorizacijskih shem. S temi shemami lahko hitro in enostavno deklarativno določimo varnost naših aplikacij.

3.3.1 Pregled varnostnih možnosti APEX

Ko v naše aplikacije vpeljujemo varnostne ukrepe, je potrebnih precej – avtentifikacija, avtorizacija itd. V APEX jih lahko najdemo v odseku »Komponente v skupni rabi (Shared Components)« [4]. Ta del aplikacije je središče našega vzpostavljanja varnosti aplikacije. Na sliki 6 lahko vidimo povezave do varnostnih funkcij, ki jih ponuja APEX, vključno z:

- avtentifikacijskimi shemami (Authentication Schemes): ugotavljajo identiteto uporabnikov in nadzorujejo, kdo lahko dostopa do naših aplikacij,
- avtorizacijskimi shemami (Authorization Schemes): nadzorujejo, katere funkcije naših aplikacij uporabnik lahko uporablja ob prijavi,
- zaščito stanja seje (Session State Protection): preprečuje spreminjanje naslovov in vrednosti, shranjenih v APEX,
- varnostne lastnosti (Security Attributes): tu lahko določimo varnostne lastnosti za vso aplikacijo.



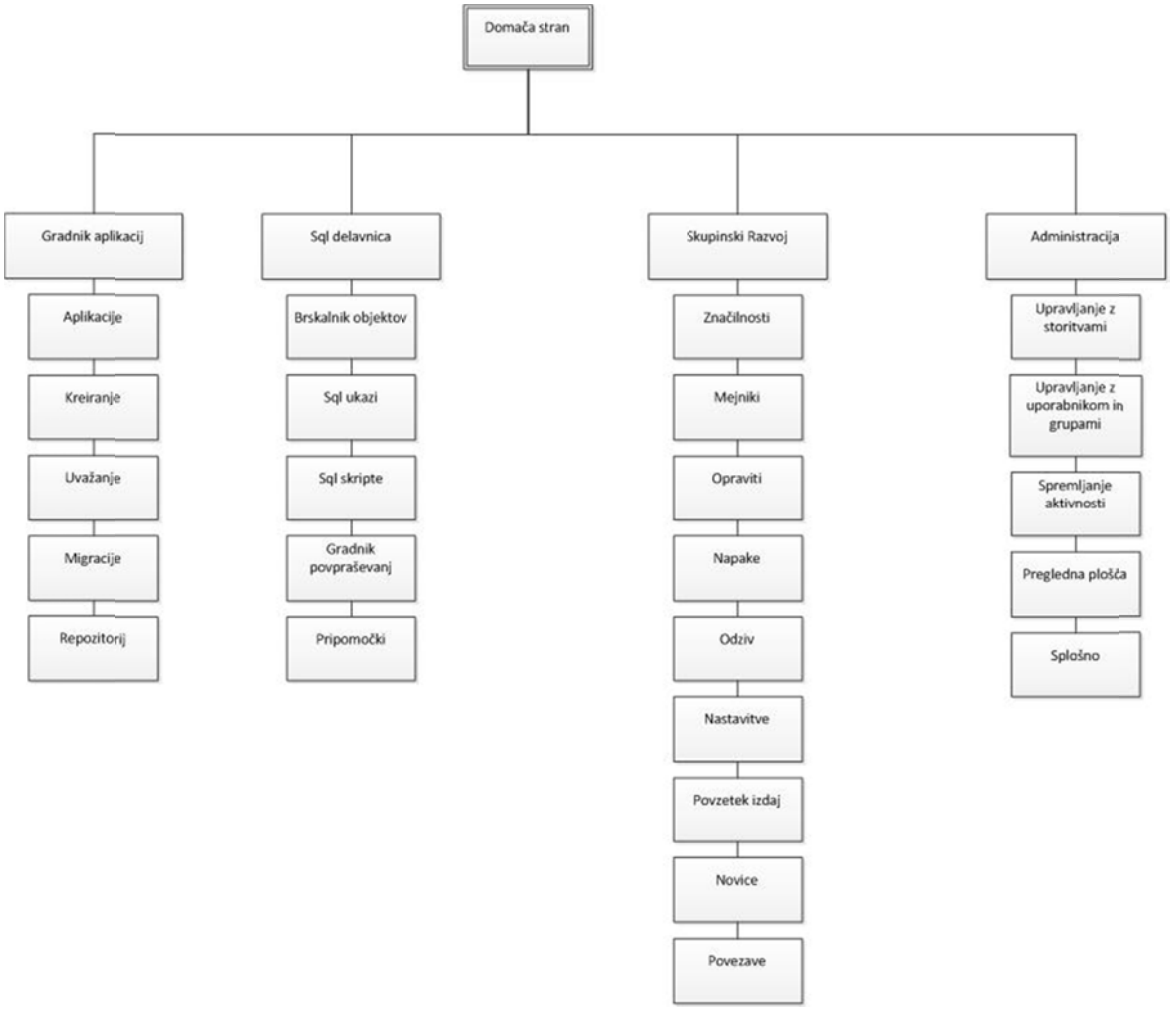
Slika 6: Varnostne funkcije APEX

3.4 Komponente APEX

APEX ima veliko število komponent, ki jih lahko razporedimo v 4 glavne skupine:

1. gradnik aplikacij (Application Builder),
2. sql delavnica (Sql workshop),
3. skupinski razvoj (Team development),
4. administracija (Administration).

Na sliki 7 so hierahnično prikazane glavne komponente APEX, kratek opis glavnih komponent je podan v nadaljevanju [3].



Slika 7: Hierarhična struktura

• **Gradnik aplikacij –** osrednji del APEX in v njem najdemo vse, kar potrebujemo za izdelavo aplikacije. Poln je čarovnikov za izdelavo grafikonov, poročil, regij, dresnih struktur itd.

• **SQL delavnica** – lahko se ukvarjamo neposredno s podatkovno bazo in podatki. Ta del lahko vidimo kot spletno verzijo SQL*PLUS z nekaj GUI dobrotami za lažjo uporabo. V osnovi je to orodje, ki nam omogoča interakcijo in dostop do podatkovne baze prek spletnega brskalnika. Na sliki 8 lahko vidimo, da je okolje SQL delavnica razdeljeno na štiri odseke:

brskalnik objektov – omogoča nam brskanje po že obstoječih objektih v bazi podatkov in ustvarjanje novih,

SQL ukazi – s temi ukazi lahko zaženemo SQL in PL/SQL kodo znotraj okolja APEX,

SQL skripte – tukaj lahko ustvarimo in vzdržujemo shrambo SQL in PL/SQL skript,

gradnik povpraševanj – s tem orodjem lahko ustvarjamo SQL poizvedbe z uporabo grafičnih pripomočkov.

• **Skupinski razvoj** – tu lahko spremljamo podatke v zvezi z razvojem APEX aplikacij, kot so recimo nove funkcionalnosti, seznam opravkov, hrošči in razni dosežki. Uporabniki lahko v realnem času podajajo povratne informacije, ki jih potem razvrstimo po njihovem tipu.

• **Administracija –** tu lahko upravljamo s podrobnostmi delovnega okolja, privzetimi nastavitvami, uporabniki, skupinami ... Vredno je omeniti, da bo imel administrator delovnega okolja na voljo več možnosti kot navaden razvijalec.

3.5 Prednosti in slabosti APEX

Čeprav smo že v prejšnjih podpoglavjih omenili osnovne prednosti APEX, bomo zdaj podrobneje opredelili tako prednosti kot slabosti orodja APEX.

Prednosti:

- hiter razvoj
- stoodstotno spletno orientiran,
- pripravljen za uporabo komponent,
- profesionalen videz,
- enostaven za ustvarjanje modelov,
- enostaven za uporabo (končni uporabnik mora le odpreti URL za dostop in uporabo aplikacije APEX),
- enostaven za razumevanje,

- hiter,
- vsa procesiranja in potrjevanja so narejena na strežniški strani,
- močna in podporna skupnost uporabnikov (posebno Oracle APEX forum),
- osnovna podpora za skupinski razvoj,
- brezplačno gostovanje za demo aplikacije.

Slabosti:

- stran v aplikaciji APEX lahko izpiše največ 100 elementov,
- aplikacije APEX so ustvarjene s pomočjo Oraclovih lastnih orodij in tako je aplikacija zaklenjena le na Oracle podatkovno bazo. APEX aplikacija ne bo delovala, če imamo za bazo MySql ali kakšno podobno podatkovno bazo,
- kot aplikacijsko ogrodje je včasih težko prilagoditi zahtevek zunaj sklopa pričakovanj o tem, kako naj bi ta zahtevek v APEX deloval,
- obsežna namestitev,
- zelo malo spletnih gostiteljev ponuja storitev gostovanja za APEX (Oracle podatkovno bazo), večina od njih ponuja PHP + MySQL ali ASP + SQL Server. Kot rezultat so aplikacije APEX omejene pri izbiri spletnih gostiteljev.

3.6 Priporoč il

V tem podpoglavju bomo opisali priporočila in kdaj ter kako uporabljati orodje APEX.

- Na začetku moramo omejiti razvoj aplikacije le na uporabnike s tehničnim znanjem. Čeprav APEX lahko uporabljajo tako končni uporabniki kot podjetja, je priporočljivo omejiti razvoj le na tiste, ki imajo tehnično znanje, tako programerje kot DBA-je (administratorje podatkovne baze). Razumevanje podatkovnega modela in poznavanje podatkov za dostop sta ključna pogoja za razvijanje aplikacij s pomočjo APEX.
- Za cilj si zastavimo združevanje delovnih listov in namiznih podatkovnih baz. Ena izmed ključnih možnosti APEX je namreč skupna raba podatkov in podjetja to lahko izkoristijo tako, da združijo svoje delovne liste s podatkovnimi bazami.
- Začnimo z majhnim in nadgrajujmo. Za začetek uporabimo APEX na eni ali dveh Oracle podatkovnih bazah, predvsem zato, da pridemo do razumevanja, kaj lahko s tem pridobimo in kakšne so lahko težave.

- Najdimo težave pri varnosti podatkov. Ko uporabljamo APEX, je treba uvesti dodatne varnostne ukrepe, še posebno pri podatkovnih bazah z zasebnimi podatki. Vloge in računi vseh uporabnikov, ki uporabljajo aplikacije APEX, naj bodo jasno določeni.
- Omejimo razvoj APEX na produkcijske podatkovne baze. Tako kot pri politiki razvoja za druga programerska okolja tudi APEX ni primeren za uporabo neposredno v produkcijskem okolju.
- Usposabljajmo razvijalce in DBA-je. Čeprav je APEX dokaj enostaven za učenje in uporabo, razmislimo o primernem usposabljanju, da iz njega lahko iztisnemo kar največ.

4 Modeliranje in razvoj aplikacije

V tem poglavju bo predstavljen razvoj dela aplikacije. Spoznali bomo trenutno stanje in probleme, povezane s strankami in knjiženjem artiklov, ter predlagano in delno razvito rešitev v elektronski obliki. To je namen naše aplikacije. Predstavljena bodo orodja in tehnologije, uporabljeni za izdelavo spletne aplikacije, ter načrtovanje podatkovne baze, ki jo uporablja aplikacija.

4.1 Orodja in tehnologije, uporabljeni pri izdelavi spletne aplikacije

V tem podpoglavju bomo opisali tehnologije in orodja za razvoj aplikacije, ki smo jih uporabljali za načrtovanje in razvoj spletne aplikacije. Podpoglavje bomo razdelili na razvojna orodja in opis tehnologij, uporabljenih za izdelavo aplikacije.

4.1.1 **Razvojna orodja**

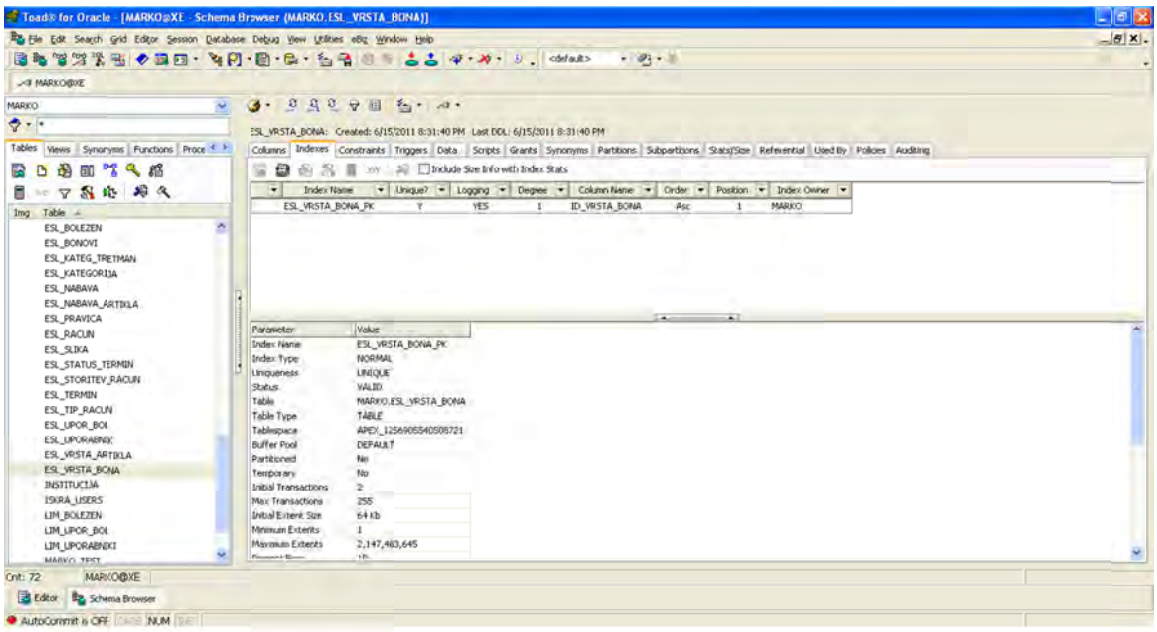
Razvojna orodja so namenjena razvoju uporabniške programske opreme, kamor sodijo prevajalniki (angl. compiler), programska okolja in sistemi za upravljanje podatkovnih baz. Spletna aplikacije je v celoti narejena v orodju APEX, ki smo ga že opisali v 3 poglavju. Poleg APEX sta uporabljena :

- Toad for Oracle,
- Sybase Power Designer.

Na kratko ju bomo opisali v nadaljevanju.

4.1.1.1 Toad for Oracle

Toad for Oracle (v nadeljevanju Toad) predstavlja programsko orodje za delo s podatkovno bazo Oracle. Obstajajo tudi drugi komercialni programi za upravljanje z Oraclovo bazo, ampak Toad je eden izmed najzmogljivejših, najuporabnejših in najučinkovitejših orodij, narejenih za upravljanje, razvoj in administracijo. Razvijalci Toada trdijo, da orodje podpira največ SQL ukazov izmed vseh zunanjih razvijalcev aplikacij za podporo Oraclovim bazam. Koristen je tako za razvijalce aplikacij, razvijalce PL/SQL in sistemske analitike kot tudi za administratorje podatkovnih baz (slika 8).



Slika 8: Programsko orodje Toad

V diplomski nalogi je bilo uporabljeno orodje Toad for Oracle 9.6.

4.1.1.2 Sybase Power Designer

Power Designer je grafično orodje za modeliranje, ki ga je razvilo podjetje Sybase. Omogoča nam lažjo vizualizacijo, analizo in manipulacijo metapodatkov (slika 9). Z njim lahko oblikujemo konceptualne modele, fizične modele, objektno usmerjene modele, poslovne

procese, xml modele, proste modele itn. V diplomski nalogi bomo uporabljali objektno usmerjeni model, ki podpira tudi razne z jezikom UML implementirane diagramske tehnike:

· diagrame primerov uporabe,

· diagrame aktivnosti,

· diagrame stanj,

· objektnw diagrami,

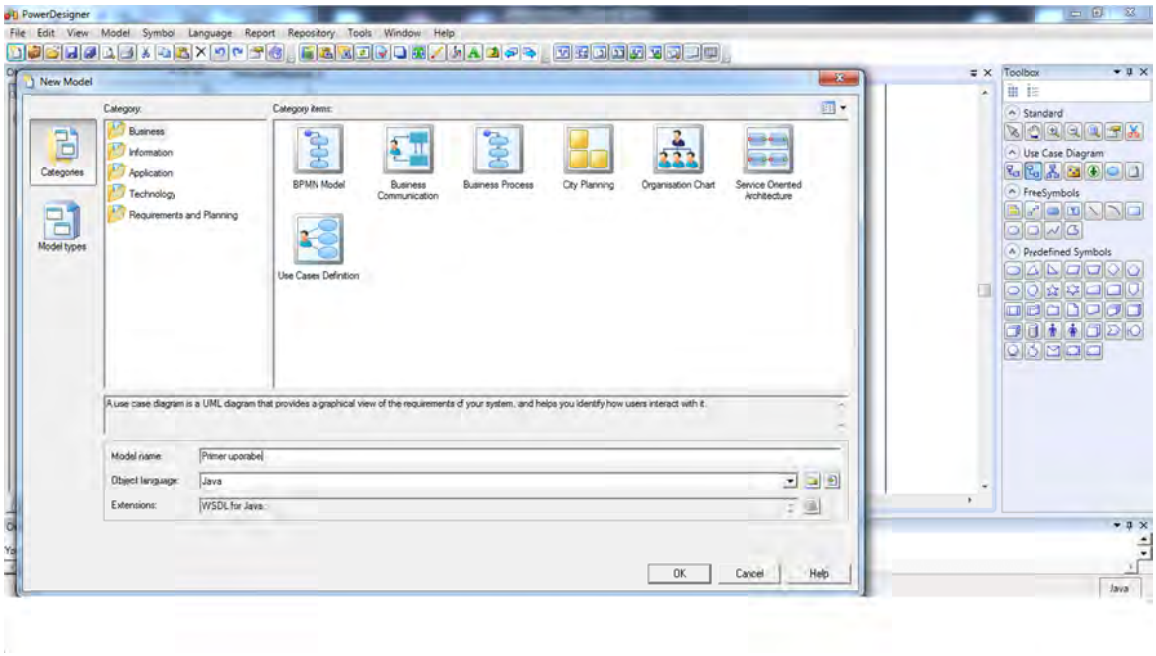
· diagrame komunikacije,

· diagrame zaporedij,

· razredne diagrame,

· komponentne diagrame,

· diagrame postavitev.



Slika 9: Programsko orodje Power Designer

V diplomski nalogi je bilo uporabljeno orodje Sybase Power Designer 16.0.

4.1.2 Uporabljene tehnologije

V tem podpoglavju bomo opisali tehnologije, ki smo jih uporabili za izdelavo spletne aplikacije.

4.1.2.1 JavaScript

JavaScript je programski jezik in nepogrešljiv del spletnih strani in aplikacij. Omogoča večjo dinamičnost in interaktivnost spletnih strani, npr. prikazovanje menijev, zavihkov, pop-up oken, preverjanje obrazcev in podobno. JavaScript se izvaja na uporabnikovi strani, zato ga je treba za pravilno delovanje naše spletne aplikacije omogočiti v brskalniku.
Glavni namen JavaScripta je omogočanje interaktivnosti HTML. Navadno je vgrajen neposredno v HTML. Pri razvoju aplikacije je bil JavaScript uporabljen predvsem za validacijo obrazcev in implementacijo virtualne tipkovnice.

4.2 Opis obstoječega stanja

Obstoječi informacijski sistem je v celoti narejen samo na papirju. Vsaka nova stranka in njeni osebni podatki se beležijo v glavno knjigo »Stranke«. Za vsako novo stranko je treba izpolniti obrazec oziroma anketo, kjer so še dodatna vprašanja glede osebnega zdravja in kratka medicinska zgodovina določene stranke. (Primer slika 10)
Velika pomanjkljivost obstoječega stanja je Knjiga naročanj. Knjiga naročanj vsebuje podatke razpisanih terminov, tretmajev in odgovorne osebe, ki bo naredila potrebno storitev. Dosedanji informacijski sistem Knjiga naročanj je dejansko »koledar«, kjer se beležijo zgoraj navedeni podatki. Nakup novih proizvodov in izdelkov je zapisan v MS excel datoteki in to je za zdaj edini del informacijskega sistema, ki je računalniško podprt. Tako mora pri prodaji določenega izdelka odgovorna oseba ročno zmanjšati število tega izdelka v že omenjeni excel datoteki. Računi za prodane storitve in izdelke so izdelani ročno, kar pomeni, da je treba za vsak nov račun pogledati naslednjo zaporedno številko računa in potem ročno napisati dva računa (enega za stranko, drugega za firmo).

Slika 10: Obstoječe stanje

4.3 Opis problema

Že v opisu obstoječega stanja je več kot očitno, da je problemov veliko. Vsaka nova prihajajoča oseba mora, kot je prikazano na sliki 10, ročno izpolniti anketo.

Zaposleni v estetskem studiu Limes mora ob prihodu stranke najprej preveriti, ali je prihajajoča stranka nova ali že obstoječa stranka. Ker pa je knjiga strank v papirnati obliki, se lahko hitro zgodi, da ne najdemo podatkov o stranki in zna priti do neugodnih vprašanj, kot so: »Ali ste že bili pri nas?« in podobno, kar ni prijetna situacija tako za stranko kot za zaposleno osebo. Problem nastane tudi pri anketah, saj se njihova vsebina ne prenese v računalniško podprto obliko. V tej obliki bi nam te omogočale tudi samo svetovanje za oblike zdravljenja za določeno stranko. Pri ročnem vnosu v koledar se hitro pojavi zmeda, saj lahko za isto uro naročimo več strank, čeprav zmogljivosti estetskega studia tega ne omogočajo. Posledici tega sta prekomerno črtanje zabeleženih terminov in nepreglednost. Samo ugotavljanje obiskanosti studia, priljubljenost tretmajev, najpogostejših strank in najbolj zasedenih terminov je težko analizirati na podlagi ročne obdelave koledarja. Dnevno, tedensko in mesečno poročilo o izvedenih storitvah za določenega zaposlenega je pri ročni analizi težko, dolgotrajno in utrujajoče delo.

4.4 Analiza in zajem zahtev

Zdaj smo ugotovili obstoječe stanje pri vnosu nove stranke, blaga in storitev v estetskem studiu Limes in se seznanili s postopkom naročanja na ustrezni tretma. Po zajemu zahtev za aplikacijo se bomo lotili razvoja aplikacije.

4.4.1 Nefunkcionalne zahteve

Poleg zahtev, ki funkcionalno določajo sistem, je potrebno poskrbeti tudi za nefunkcionalne zahteve sistema. Nefunkcionalne zahteve neposredno ne določajo sistema, ampak so zelo pomembne in je od njih odvisno nemoteno delovanje aplikacije.
V nadaljevanju bomo podali nekaj najpomembnejših nefunkcionalnih zahtev sistema.

1. Strojna in programska oprema

Zaradi relativno nezahtevne aplikacije so zahteve s tehnične strani bolj skromno gledane. Strojna oprema zahteva osebni računalnik, ki ima omogočen dostop do interneta ali lokalnega omrežja, kjer je nameščen strežnik. V okviru programske opreme potrebujemo strežnik Oracle, orodje APEX in spletni brskalnik, ki mora imeti javascript podporo. Moramo imeti uspešno nameščeni in delujoči tiskalnik.

2. Performanse

Performanse so pogojene s performansami strežnika in odjemalca. To pomeni, da so v največji meri odvisne od hitrosti in zmogljivosti povezave oziroma prenosa podatkov.

3. Uporabnost

Aplikacija mora biti uporabniku prijazna in enostavna za uporabo. Omogočati mora, da lahko v vsakem trenutku enostavno v dveh, treh korakih pridemo do vseh funkcij in modulov aplikacije, ki so nam na voljo, odvisno od pravic, s katerimi smo prijavljeni.

4. Dostopnost

Zaradi zagotavljanja čim večje dostopnosti je smiselno uporabiti spletno tehnologijo, ki to v največji meri omogoča. Dostopnost je pogojena z dostopnostjo strežnika in odjemalca.

5. Odzivnost

Uporabniki zahtevajo, da so moduli aplikacije, ki jih redno uporabljajo, dovolj hitri oziroma odzivni. To je pomembno zgolj za iskanje uporabnikov in za modul izdelave in tiskanja računov.

6. Varnost

Podatkovna baza je ključnega pomena za delovanje spletne aplikacije, zato je zelo pomembno, da je ta zaščitena pred dostopom nepooblaščenih oseb. Sistem mora zagotavljati varno dostopanje do posameznih podatkov na bazi, odvisno od pravic posameznega uporabnika.

4.4.2 **Funkcionalne zahteve**

Funkcionalne zahteve bomo predstavili s pomočjo diagrama primerov uporabe. Pred opisom primerov uporabe moramo določiti akterje sistema.

4.4.2.1 Akterji sistema

V aplikaciji bomo definirali tri vrste uporabnikov. Od statusa uporabnika je odvisen dostop do posameznih segmentov in funkcij aplikacije.
Statuse pravic bomo opredelili po padajočem vrstnem redu; status 0 pomeni, da ima uporabnik vse pravice (celotna funkcionalnost).
Akterji sistema so:

1. administrator – uporabnik, ki ima status 0 in po uspešni prijavi v aplikacijo dostop do vseh funkcij in modulov ter pregled nad celotnim sistemom,

2. pooblaščena oseba – uporabnik (zaposleni), ki ima status 1, lahko po uspešni prijavi v aplikacijo dodaja in spreminja podatke o strankah oziroma uporabnikih. Lahko sprejme in potrdi zahtevek za storitev (brez odobritev drugih oseb),

3. uporabnik – uporabnik (stranka), ki ima status 2, lahko po uspešni prijavi v aplikacijo kreira zahtevek za določeno storitev, ki potem čaka na pregled pooblaščene osebe. Na voljo ima pregled svojega osebnega profila in zgodovino tretmanov.

4. tiskalnik – naprava, primarno namenjena tiskanju računov in preostalih tedenskih in mesečnih statistik za posameznega uporabnika ali pooblaščeno osebo.

4.4.2.2 Primeri uporabe

Zaradi lažje predstavitve funkcionalnosti sistema bomo v nadaljevanju podali tekstovni opis primera uporabe.

Primeri uporabe so:

1. Prijava v aplikacijo

Po nalaganju aplikacije se pojavi vstopna stran s poljema za vnos uporabniškega imena in gesla. Vnos je možen tudi z uporabo navidezne tipkovnice. Uporabniško ime in geslo je predhodno določil administrator sistema. Po uspešni prijavi se prikaže osnovna stran aplikacije.

2. Odjava iz aplikacije

Vsak uporabnik ima možnost odjave iz aplikacije. S tem onemogoči uporabo aplikacije drugi osebi v njegovem imenu.

3. Urejanje uporabnikovega profila

Uporabniku aplikacije omogoča spreminjanje lastnih podatkov in dodajanje slik.

4. Pregled uporabnikov

Osebam s statusom pooblaščene osebe ali administratorja omogoča iskanje in pregled uporabnikov aplikacije.

5. Prijava na tretma

Vsi uporabniki imajo možnost prijave na tretma.

6. Potrditev prijave uporabnika na želen termin

Potrditev prijave imata samo pooblaščena oseba in administrator sistema.

7. Pregled prijav

Uporabnik na spletni strani vidi le lastne kreirane prijave na termine, pooblaščena oseba vidi prijave samo za salon, v katerem je zaposlena, administrator ima vpogled v vse zahtevke.

8. Dodajanje uporabnika

Pooblaščenim osebam omogoča dodajanje uporabnikov z statusom 2 (navadne stranke), administratorju sistema dodajanje vseh uporabnikov v aplikacijo.

9. Urejanje podatkov o uporabnikih

Omogoča administratorju urejanje vseh podatkov o uporabnikih.

10. Izdelava računov

Omogoča pooblaščenim osebam in administratorju izdelavo računa.

11. Tiskanje računov

Omogoča pooblaščeni osebi in administratorju tiskanje računov

12. Vpogled v statistike

Samo administrator ima pravico do vpogleda in izračuna tedenskih, mesečnih in letnih statistik.

4.4.2.2.1 Opis toka dogodkov za primer Prijava v aplikacijo

Primer uporabe omogoča uporabniku prijavo v aplikacijo (administrator, pooblaščena oseba in stranka).

Osnovni tok:

0. Uporabnik začne postopek prijave, ko odpre naslov spletne aplikacije.
1. Sistem prikaže formo z vnosnimi polji za uporabniško ime in geslo.
2. Uporabnik vnese uporabniško ime in geslo.
3. Sistem preveri uporabniško ime in geslo.
4. Sistem prikaže formo za izbiro možnosti dela, ki so uporabniku omogočena glede na njegove pravice.

Alternativni tok:

0. Sistem prikaže zaslon z vnosnimi polji za uporabniško ime in geslo.
1. Uporabnik vnese uporabniško ime in geslo.
2. Sistem preveri uporabniško ime in geslo.
3. Sistem zavrne dostop do aplikacije zaradi napačnega uporabniškega imena ali gesla.

4.4.2.2.2 Opis toka dogodkov za primer Dodajanje uporabnika

Primer uporabe omogoča uporabniku (administratorju in pooblaščeni osebi) dodajanje uporabnika.

Osnovni tok:

0. Administrator ali pooblaščena oseba klikne na povezavo za vnos novega uporabnika.
1. Sistem prikaže formo za vnos novega uporabnika.
2. Administrator ali pooblaščena oseba vnese vse obvezne podatke o uporabniku.
3. Administrator ali pooblaščena oseba shrani/potrdi spremembe.
4. Sistem preveri obvezna polja in zasedenost uporabniškega imena.
5. Sistem zapiše spremembe v podatkovno bazo.
6. Sistem prikaže sporočilo o uspešni shranitvi podatkov in prikaže formo seznam uporabnikov.

Alternativni tok:

0. Administrator ali pooblaščena oseba klikne na povezavo za vnos novega uporabnika.
1. Sistem prikaže formo za vnos novega uporabnika.
2. Administrator ali pooblaščena oseba vnese vse obvezne podatke o uporabniku.
3. Administrator ali pooblaščena oseba shrani/potrdi spremembe.
4. Sistem preveri obvezna polja in zasedenost uporabniškega imena.
5. Sistem sporoči napako o zasedenosti uporabniškega imena.

4.4.2.2.3 Opis toka dogodkov za primer Urejanje podatkov o uporabnikih

Primer uporabe omogoča uporabniku (administratorju) urejanje podatkov o uporabnikih.

Osnovni tok:

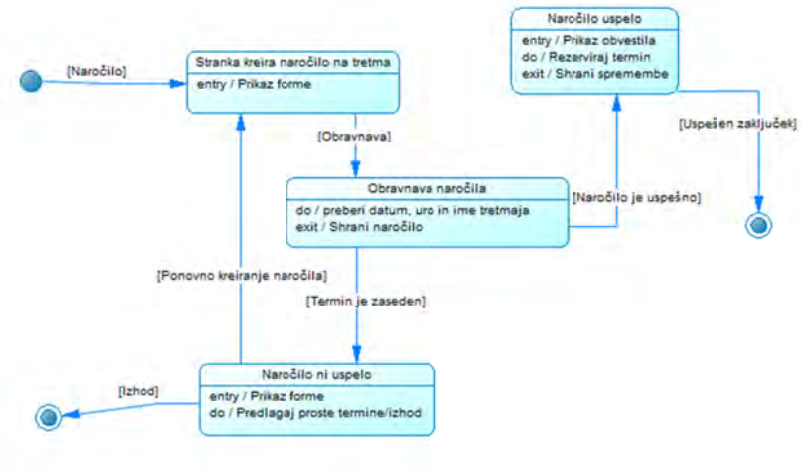
0. Administrator klikne gumb uredi ob izbranem uporabniku na formi seznam uporabnikov.
1. Sistem prikaže formo za popravek podatkov o uporabnikih.
2. Administrator popravi podatke uporabnika.
3. Administrator shrani/potrdi spremembe.
4. Sistem zapiše spremembe v podatkovno bazo
5. Sistem prikaže sporočilo o uspešni shranitvi podatkov in prikaže formo seznam uporabnikov.

Alternativni tok:

Ni predviden.

4.4.2.3 Diagram stanj za primer Prijava na tretma

Primer uporabe omogoča uporabniku prijavo na tretma na želeni datum in uro (slika 11), scenarij pa je opisan v nadeljevanju.



Slika 11: Diagram stanj za prijavo na tretma

Scenarij diagrama stanj:

0. Stranka kreira naročilo na tretma.
1. Sledi postopek obravnave naročila.
2. Naročilo je uspelo.
3. Naročilo je zaključeno.

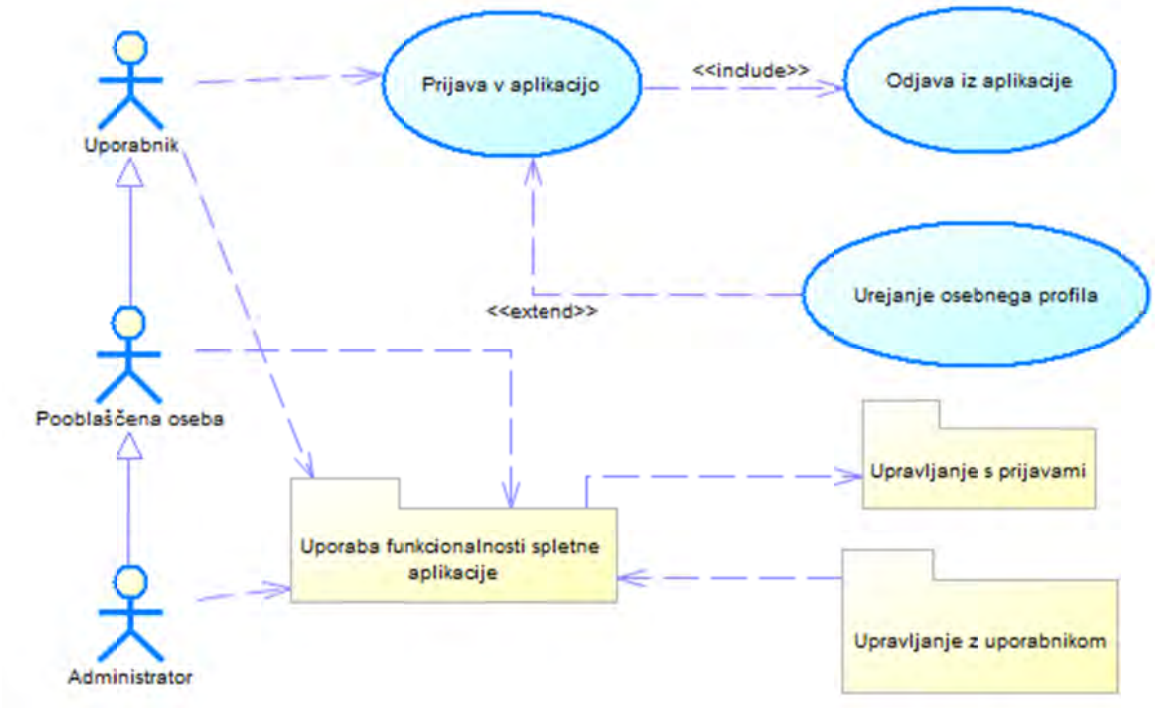
Alternativni scenarij:

0. Stranka kreira naročilo na tretma.
1. Sledi postopek obravnave naročila.
2. Naročilo ni uspelo (termin je zaseden).
3. Sistem ponovno preusmeri stranko na formo za kreiranje naročila.

4.4.2.4 Diagrami primerov uporabe

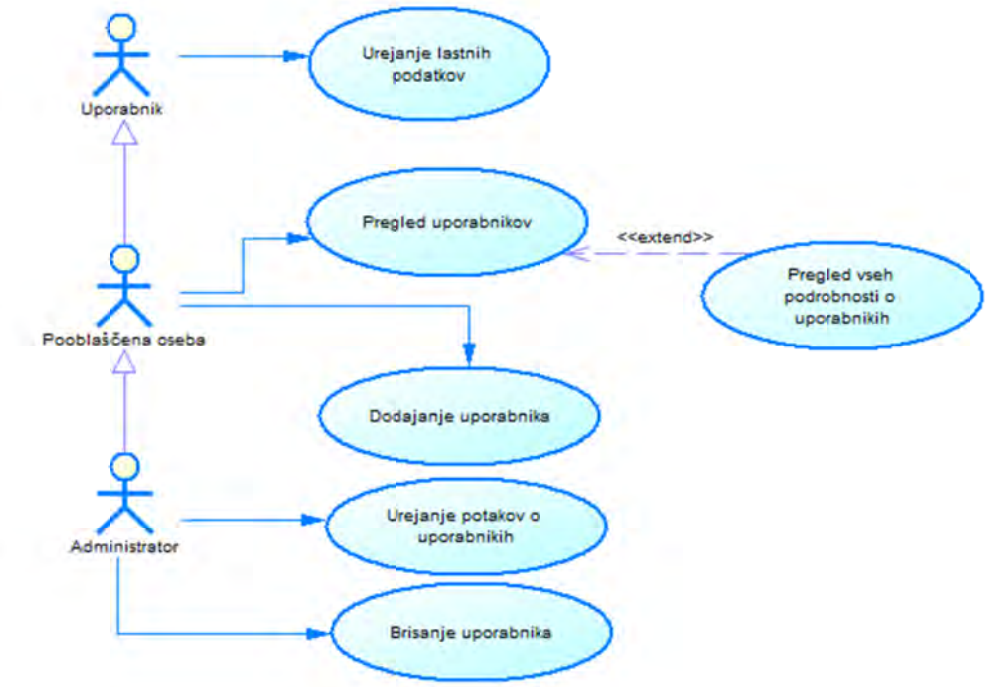
Diagrami primerov uporabe predstavljajo komunikacijo med uporabnikom in informacijskim sistemom ter opisujejo funkcionalno obnašanje sistema, kot ga vidi uporabnik.

Modeli primerov uporabe prikazujejo možne primere uporabe aplikacije. Na spodnji sliki (slika 12) bomo prikazali primere uporabe za vso aplikacijo. Primer uporabe vsebuje prijavo v aplikacijo, urejanje profila uporabnika in uporabo paketa za upravljanje s prijavami. Power Designer je programsko orodje, v katerem bomo narisali primere uporabe.

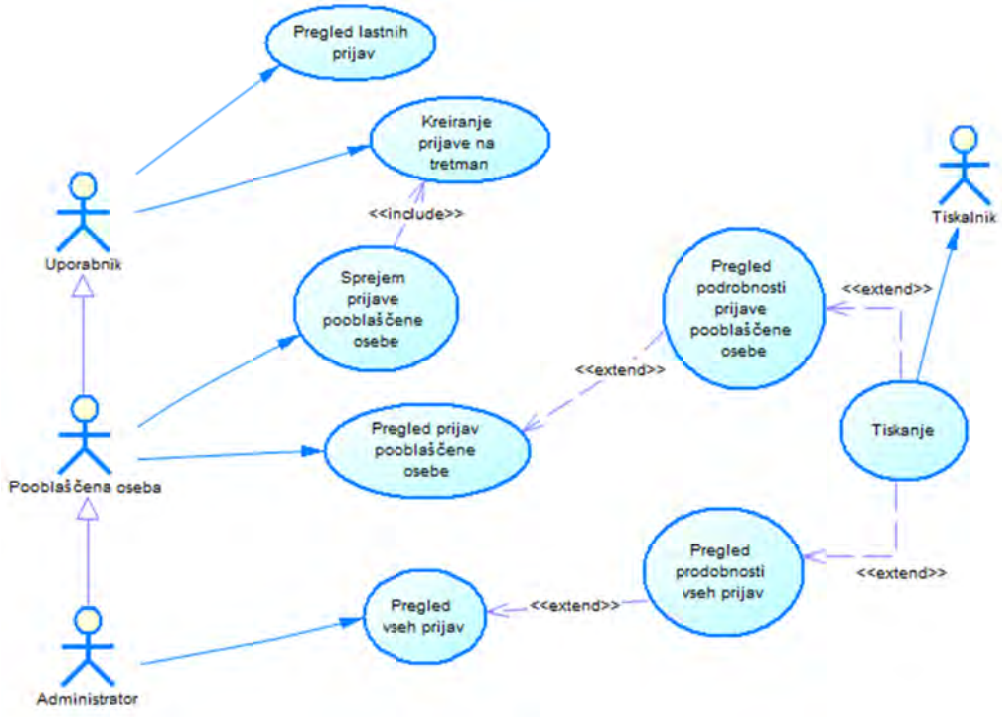


Slika 12: Model primerov uporabe za vso aplikacijo

V nadaljevanju sta prikazana modela primerov uporabe za paket upravljanja z zahtevkom za termin in paket upravljanja z uporabniki (slika 13, 14).



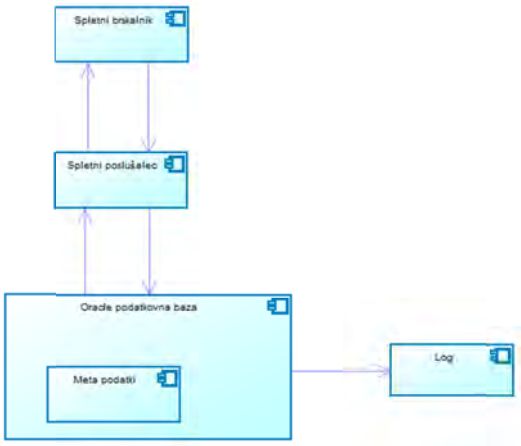
Slika 13: Model primerov uporabe za upravljanje z uporabniki



Slika 14: Model primerov uporabe za upravljanje s termini

4.4.3 Komponentni diagram

Komponentni diagram je eden od devetih diagramov UML, ki ponujajo grafični prikaz odvisnosti in posplošitev med komponentami programske opreme. Vozlišča grafa so komponente, povezave pa relacije med njimi. Na sliki 15 bomo pokazali komponentni diagram naše aplikacije, opis posameznih komponenet sledi v nadeljevanju. Komponentni diagram je narisan v orodju Power Designer.



Sika 15: Komponentni diagram

Spletni brskalnik ter splet in spletne aplikacije kot je podrobnejše opisan v poglavju 3.1 in 2, priporočljiva pa je uporaba brskalnika Internet Explorer. Spletni poslušalec mod_plsql je že podrobnejše opisan v poglavju 3.2. Oracle je sistem za upravljanje z relacijskimi bazami podatkov (RDBMS). Prva verzija programa Oracle je nastala že 1977 leta. Osnovne prednosti Oracle podatkovne baze so neodvistnost od platforme (lahko jo namestimo na vse znane OS: WIN, Unix, Linux, AIX, Sun Solaris ...), hitrost pri večji količini podatkov, prilagodljivost, stabilnost, programski jezik PL/SQL z več možnostmi, možnost shranjevanja podatkov pri delujoči bazi ...(več o oraclovi podatkovni bazi je dostopno na www.oracle.com)

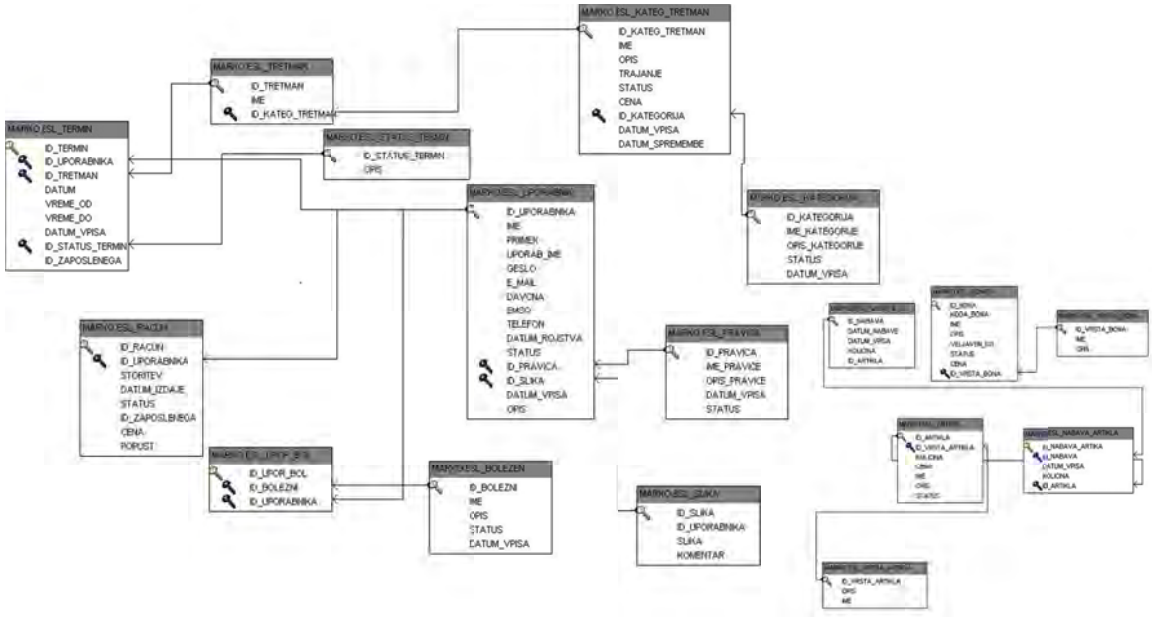
5 Podatkovni model

Podatkovni model je zbirka konceptov, s katerimi skušamo izraziti statične in dinamične lastnosti podatkov v okviru informacijskega sistema, obenem pa mora realno predstavljati izsek realnosti. Predstavlja množico pravil, ki določajo, kako smejo biti podatki v bazi podatkov organizirani oziroma strukturirani in dovoljene operacije nad podatki. Dober podatkovni model je osnova za razvoj (dobre) aplikacije. Čeprav dober podatkovni model še

ne more zagotoviti dobre aplikacije, lahko trdimo, da slab podatkovni model nujno pripelje do slabe aplikacije in nekakovostnih podatkov.

5.1.1 Diagram entiteta-razmerje

Za izdelavo diagrama smo si pomagali s programom Toad, ki omogoča samodejno izrisovanje diagramov entiteta-razmerje neposredno iz baze, s katero je trenutno povezan. Diagram (slika 16) zajema 17 podatkovnih entitetnih tipov. Vsak entitetni tip ima svoje ime, pod katerim so našteti atributi, in eno ali več relacij, s katerimi je povezan z drugimi entitetnimi tipi. Rumeni ključi pred atributom označujejo enolični identifikator oziroma primarni ključ. Opis posameznih entitetnih tipov in njihovih relacij sledi v nadaljevanju.



Slika 16: Diagram entiteta-razmerje

Podatkovni model vsebuje 17 entitet, ki predstavljajo:

- ESL_TIP_RACUN vsebuje kratko ime, polno ime in status tipa računa (npr. gotovina, kartica, itn.),
- ESL_KATEGORIJA šifrant za kategorije, določa kategorijo tretmana (masaža, depilacija, itn.); gledano s tehničnega vidika je tabela precej enaka kot šifrant esl_tip_racun,
- ESL_STATUS_TERMIN opisuje status termina, poleg primarnega kljuca ima polje opis, kamor vpišemo opis za status (odkazan, preložen, itn.),
- ESL_SLIKA vsebuje tuji ključ uporabnika (id_uporabnika) in polji slika (tipa blob) in komentar; uporablja se za nalaganje slik uporabnika,
- ESL_PRAVICA se uporablja za vnos pravic (uporabnik, pooblaščena oseba, administrator),

- ESL_BOLEZEN je šifrant za vrste bolezni; poleg enoličnega identifikatorja ima še polja ime, opis in status (alergija, diabetes, itn.),
- ESL_NABAVA opisuje nabavo, vsebuje polja datum nabave, tuji ključ id_artikla in količino (gel pevonia – količina 5, itn.),
- ESL_VRSTA_ARTIKLA opisuje vrste artiklov (lonsion, krema, gel, itn.),
- ESL_ARTIKEL opisuje artikle, vsebuje tuji ključ na šifrant esl_vrsta_artikla in polja za ime, opis, količino in ceno ter ali je artikel na voljo (hidratantna krema 50ml),
- ESL_STORITEV_RACUN vsebuje tuje ključe id_tip_racun in id_racun ter storitev,
- ESL_KATEG_TRETMAN vsebuje tuji ključ id_kategorija in polja za ime, ceno, trajanje in status tretmaja (hot stone masaža, 35 €, 75 min),
- ESL_UPORABNIK opisuje uporabnika, vsebuje tuja ključa id_pravica in id_slika ter polja ime, priimek, uporab_ime, geslo, e-mail, davčna, emšo, telefonska, datum rojstva, dodatno polje za opis (npr. vedno zamuja 30 min) ter kontrolno polje prva prijava, ki se izpolni pri prvi prijavi uporabnika,
- ESL_UPOR_BOL je vmesna tabela med tabelo esl_uporabnik in esl_bolezen, vsebuje samo tuja ključa od teh tabel,
- ESL_BONOVI opisuje bone, vsebuje polja koda bona, ime, opis, veljaven od in status (poklon bon, tajska masaža, velja do 29. 12. 2011),
- ESL_NABAVA_ARTIKLA je vmesna tabela med nabavo in artiklom, poleg dveh tujih ključev (id_artikla in id_nabava) vsebuje še polje za vpis količine,
- ESL_RACUN opisuje račune, vsebuje dva tuja ključa, ki sta vezana na isti entitetni tip esl_uporabnik, in polje za izbiro storitve ali artikla, ceno in popust,
- ESL_TERMIN vsebuje 3 tuje ključe, in sicer id_uporabnika, id_tretman ter id_status_termin. V šifrantu so še datum in vreme od in vreme do in opombe za določeni termin

5.1.2 **Fizični model**

Fizični podatkovni model je prikaz dejanske baze podatkov, ki je lahko izvedena z različnimi programskimi orodji. V našem primeru ga ni treba ustvariti ročno, saj orodje Toad vsebuje funkcijo, kjer lahko izvozimo celotno shemo in generiramo skripte za fizični podatkovni model**.** Poleg tabel so v bazi podatkov potrebni še drugi objekti, kot so sekvenca, indeks in sprožilec. Sekvenca je objekt v podatkovni bazi, ki omogoča ustvarjanje unikatnih števil, in jo

najpogosteje koristimo za avtomatično ustvarjanje primarnih ključev. Sprožilce smo v naši aplikaciji uporabili za dodeljevanje novih vrednosti ob vsakem vnosu v tabelo.

Primer skripte fizičnega podatkovnega modela za tabelo `ESL_TERMIN:`

```
CREATE TABLE ESL_TERMIN
(
  ID_TERMIN         NUMBER                      NOT NULL,
  ID_UPORABNIKA     NUMBER,
  ID_TRETMAN        NUMBER,
  DATUM             DATE,
  VREME_OD          VARCHAR2(10 BYTE),
  VREME_DO          VARCHAR2(10 BYTE),
  DATUM_VPISA       DATE                        DEFAULT sysdate,
  ID_STATUS_TERMIN  VARCHAR2(10 BYTE),
  ID_ZAPOSLENEGA    NUMBER                      DEFAULT 1
)
CREATE UNIQUE INDEX ESL_TERMIN_PK ON ESL_TERMIN (ID_TERMIN);
ALTER TABLE ESL_TERMIN ADD (
  CONSTRAINT ESL_TERMIN_PK
 PRIMARY KEY
 (ID_TERMIN));
```

Primer sekvence in sprožilca za tabelo `ESL_TERMIN:`

```
CREATE SEQUENCE ESL_TERMIN_SEQ
  START WITH 1
  MAXVALUE 999999999999999999999999999
  MINVALUE 1
  NOCYCLE
  CACHE 20
  NOORDER;
CREATE OR REPLACE TRIGGER "BI_ESL_TERMIN"
  before insert on "ESL_TERMIN"
  for each row
begin
  if :NEW."ID_TERMIN" is null then
    select "ESL_TERMIN_SEQ".nextval into :NEW."ID_TERMIN" from dual;
  end if;
end;
```

Skripto za fizični podatkovni model vnesemo v Oracle bazo prek sqlplus @ funkcije,
`sqlplus user/pass @ skripta.sql.`

Skripta nam nato generira prazno tabelo `ESL_TERMIN` z naslednjimi stolpci:
`ID_TERMIN` tipa `NUMBER`, ki predstavlja enolični ključ,
`ID_UPORABNIKA` tipa `NUMBER`, ki predstavlja tuji ključ,
`ID_TRETMAN` tipa `NUMBER`, ki predstavlja tuji ključ,
`DATUM` tipa `date`,
`VREME_OD` tipa `VARCHAR2,`
`VREME_DO` tipa `VARCHAR2,`
`DATUM_VPISA` tipa `date,`
`ID_STATUS_TERMIN` tipa `VARCHAR2`, ki predstavlja tuji ključ in
`ID_ZAPOSLENEGA` tipa `NUMBER`, ki predstavlja tuji ključ.

5.2 Opis aplikacije

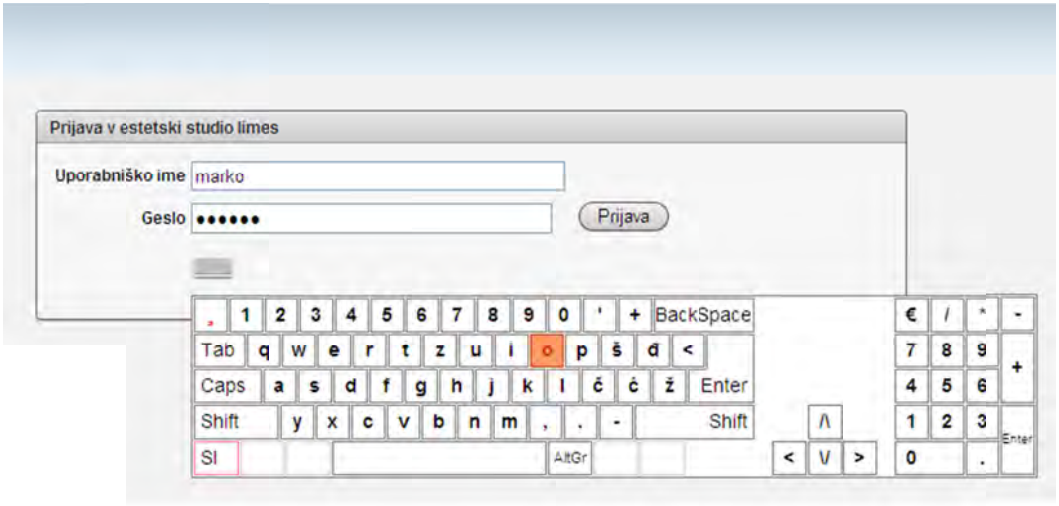
V naslednjem poglavju bomo predstavili razvito spletno aplikacijo. Spoznali bomo njen namen, uporabniški vmesnik, funkcionalnosti in možnosti za nadaljnjo nadgradnjo, razvoj in uporabo.

5.2.1 Splošno o aplikaciji

Aplikacija je namenjena uporabi zaposlenih v estetskem studiu namesto obstoječega sistema, ki poteka izključno v tiskani obliki. Razvita aplikacija omogoča učinkovito spremljanje in lažje vodenje storitvene dejavnosti. Aplikacija omogoča vnos uporabnikov, pregled zgodovine tretmajev in osebnega profila ter pošiljanje zahteve za določeni tretma na želeni termin. Uporabniki z ustreznimi pravicami lahko dodajajo, brišejo in spreminjajo izdelke, tretmaje in uporabnike. Omogoča iskanje, vodenje evidence in izdajo računov za redne stranke in pravne osebe. Poleg tega aplikacija omogoča pregled prometa po dnevih, mesecih, letih ali zaposlenih.

5.2.2 **Prijava v sistem**

Uporabniku se na prvi strani spletne aplikacije prikaže forma za prijavo v aplikacijo (slika 17). Forma vsebuje dve vnosni polji uporabniško ime in geslo (tipa `password`), ter gumb za prikaz navidezne tipkovnice ki nam omogoča prijavo brez tipkovnice. Orginalna navidezna tipkovnica je javascript knjižnica, ki je dostopna na http://www.codeproject.com/KB/scripting/jvk.aspx, ki je za potrebe spletne aplikacije ustrezno dopolnjena (več v prilogi). Za nove stranke administrator sistema predhodno določi geslo in uporabniško ime, s katerim se lahko potencialna stranka prijavi in dokonča postopek registracije.



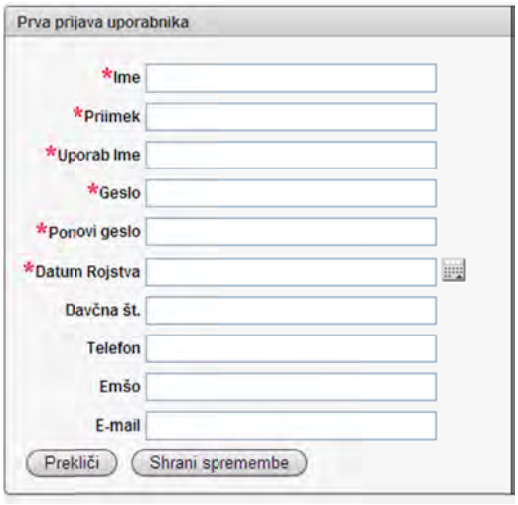
Slika 17: Forma za prijavo v aplikacijo

Prijava je uspešna, če je v podatkovni bazi natanko takšno uporabniško ime in geslo, kot ga uporabnik vtipka na prijavni formi. Za preverjanje vhodnih podatkov poskrbi funkcija PL/SQL `preveri_uporabime_geslo (p_uporab_ime IN VARCHAR2, p_geslo IN VARCHAR2)`

5.2.3 **Registracija uporabnika**

Ko se uporabnik prvič prijavi v aplikacijo, se prikaže osnovna forma, kjer mora uporabnik obvezno spremeniti geslo in uporabniško ime. Kot je razvidno iz slike 18, mora uporabnik vnesti novo geslo dvakrat in izpolniti preostala obvezna polja, ki so označena s zvezdico (ime,

priimek in datum rojstva). Tak pristop je smiseln, saj bi se lahko uporabnik pri vnosu novega gesla zatipkal, kar bi mu seveda onemogočilo nadaljnjo prijavo.



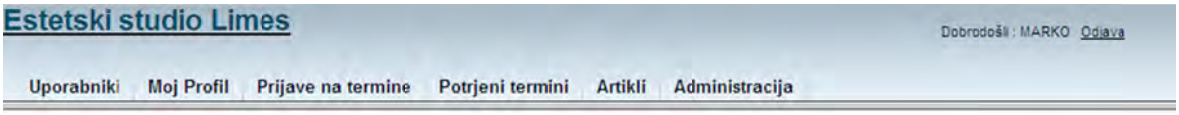
Slika 18: Prva prijava uporabnika

Pri postopku dokončanja registracije lahko pride do naslednjih napak in s tem povezanih sporočil:

- sporočilo o napaki neujemanja gesel (narejeno z APEX validacijo),
- sporočilo o napaki neustrezne dolžine uporabniškega imena ali gesla (funkcija `preveri_dolzino(input_string in varchar2)`),
- sporočilo o neizpolnjenih obveznih poljih,
- aporočilo o napaki zasedenega uporabniškega imena (funkcija `preveri_uporabIme(uporab_ime in varchar2)`).

5.2.4 **Uporabniški meni**

Po uspešni prijavi v aplikacijo se uporabniku prikaže uporabniški meni (slika 19). Na vrhu vsake strani so prikazani zavihki, v katerih so navedene povezave do različnih opravil, ki so na voljo posameznemu uporabniku, odvisno od pravice, ki jo ima uporabnik. Tako uporabnik s pravico 2 (stranka) vidi samo svoj profil, ki ga lahko ureja (dodaja slike itn), ima možnost prijave na tretma in pregled zgodovine. Vsak uporabnik ima v zgornjem kotu prikazano svoje uporabniško ime in povezavo na odjavo. Odjava je implementirana z APEX `LOGOUT_URL` funkcijo, ki uporabnika vrne na vhodno prijavno stran aplikacije.

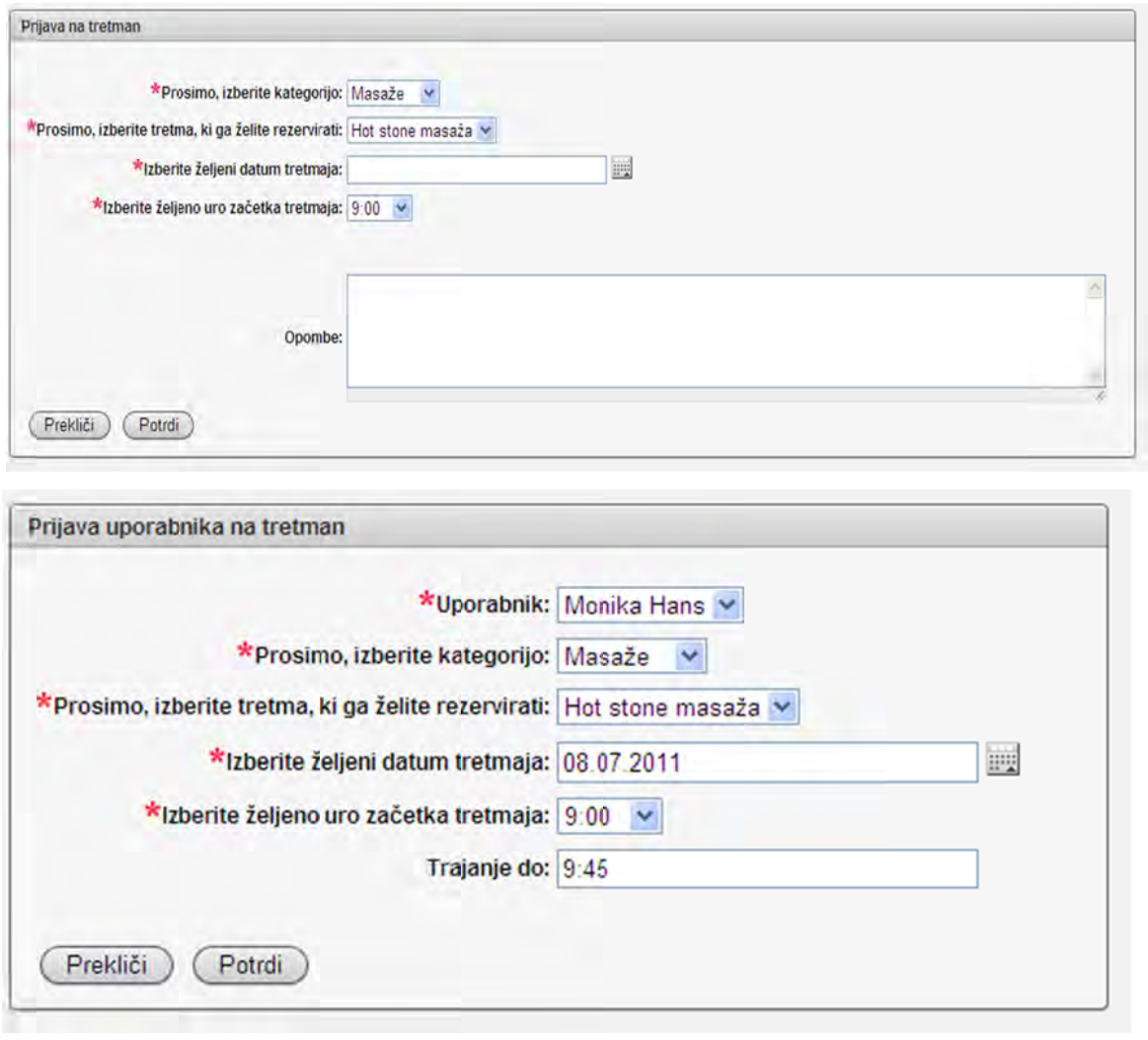


Slika 19: Uporabniški meni

5.2.5 **Ustvarjanje in pregled zahtevkov na želeni tretma**

Ustvarjanje in pregled zahtevkov ločimo na dva dela: na tistega za administratorje in pooblaščene osebe ter tistega za uporabnike. Uporabniki lahko kreirajo zahtevek za določeno storitev in imajo na voljo vpogled samo v svoje potrjene in nepotrjene termine.

Administrator in pooblaščena oseba imata na voljo vpogled v vse zahtevke in možnost ustvarjanja zahtevka v imenu drugega uporabnika. Na sliki 20 sta prikazana modela za kreiranje zahtevka za uporabnika in za pooblaščeno osebo oziroma administratorja.



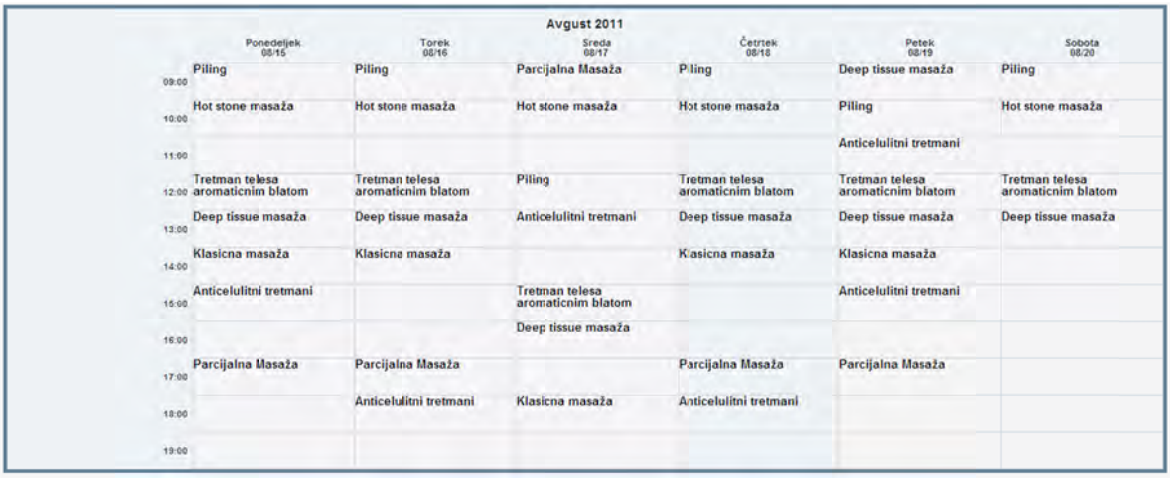
Slika 20: Prijava uporabnika na tretma

Za ustvarjanje zahtevka uporabnik najprej izbere kategorijo in nato tretma ter datum in uro želenega termina, pooblaščena oseba oziroma administrator ima na voljo še izbiro uporabnika, oziroma zaposleno osebo ki bo izvajala tretma. Vsa polja na formi razen opomb so obvezna,

validacija pa je narejena z uporabo APEX vgrajene komponente za validacije (naziv validacije: Value of item is not null). Poleg polja za vnos datuma imamo na voljo gumb APEX koledar, kjer lahko lažje izberemo želeni datum.

5.2.6 Koledar

Najpomembnejši modul vsake aplikacije, ki se ukvarja s časom, je koledar. V naši aplikaciji imamo na voljo tri vrste koledarjev: mesečni, tedenski in dnevni prikaz. Koledarji so narejeni z uporabo APEX forme za koledarje in so vidni le pooblaščenim osebam in administratorju. Za izdelavo koledarja v APEX je treba napisati sql vprašanje, ki nam vrne vsaj eno polje, ki vsebuje datum oziroma je tipa date. Od vseh treh vrst koledarjev je najpomebnejši dnevni, oziroma tedenski koledar, kjer se točno vidi ura in ime naročenega tretmana (slika 21). Če kliknemo na zapis v koledarju, nas aplikacija preusmeri na formo za vpogled zahtevka na tretma, kjer lahko vidimo dodatne opombe, ki jih je napisala stranka. Administrator sistema lahko določi in spremeni zaposlenega za izvajanje tretmaja.



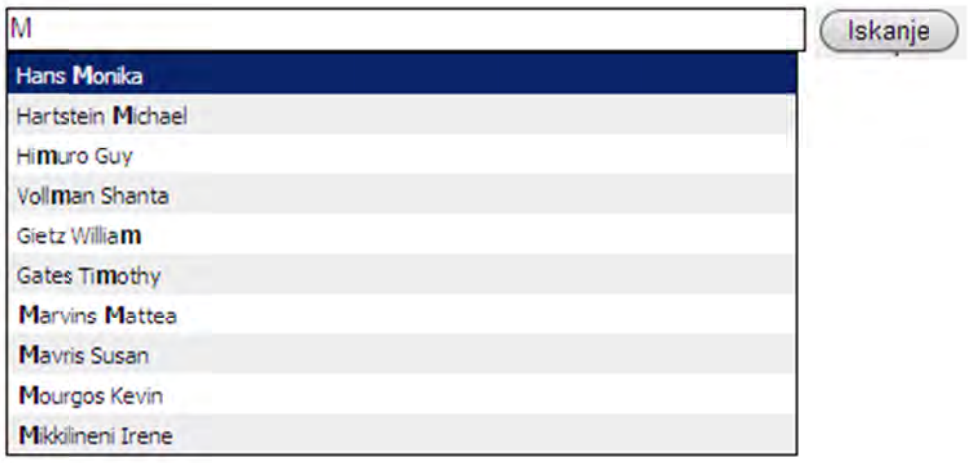
Slika 21: Tedenski koledar

5.2.7 Dodajanje uporabnikov in pooblaščenih oseb

Administrator sistema in pooblaščena oseba imata pravico dodajanja uporabnikov. Funkcionalnost je za oba akterja sistema skoraj enaka, edina razlika je v tem, da pooblaščena oseba lahko dodaja samo uporabnike s statusom 2 (navadna stranka), administrator pa lahko ustvarja tako uporabnike kot pooblaščene osebe in nove administratorje. Če administrator dodaja novega zaposlenega ali novega administratorja so obvezna vsa polja na formi kot je že prikazano na sliki 18.

5.2.8 Iskanje uporabnikov

Iskalnik uporabnikov je namenjen hitremu iskanju uporabnikov. Narejen je iz enega iskalnega polja, ki ima vgrajeno APEX funkcijo samodokončanja (ang. autocmplete), ki ima za izvor podatkov sql vprašanje, v našem primeru ime in priimek osebe. Primer: select ime, priimek from esl_uporabnik; poleg iskalnega polja imamo na voljo gumb za začetek iskanja (v primeru, da ne koristimo funkcije samodokončanja), kot je prikazano na sliki 22. Iskanje je narejeno z PL/SQL funkcijo ki nam omogoča iskanje po 4 poljih: ime, priimek, id uporabnika ali zadnje 4 cifre telefonske številke.



Slika 22: Hitro iskanje uporabnikov

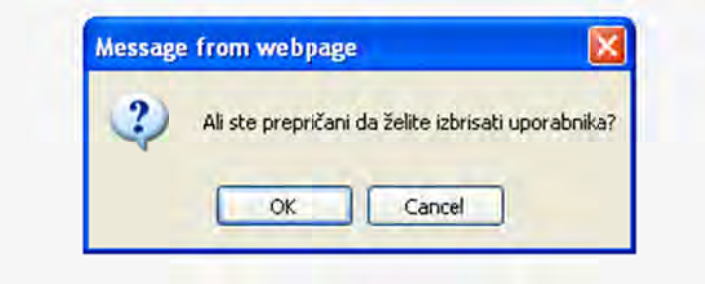
Če iskalnik ne najde nobenega uporabnika, ki ustreza iskalnim pogojem, nas o tem primerno obvesti.

5.2.9 Brisanje

Pravico za brisanje uporabnikov, splošnih šifrantov, naročil itn. ima samo administrator sistema. Brisanje je implementirano na dva načina:

- pri brisanju uporabnikov se dejansko ne briše iz podatkovne baze, ampak se spremeni status uporabnika,
- pri brisanju šifrantov kategorij, statusov in artiklov se dejansko briše zapis iz podatkovne baze.

Preden se zapis dejansko zbriše, nas sistem vpraša za potrditev, kot je prikazano na sliki 23.



Slika 23: Potrditev brisanja zapisa

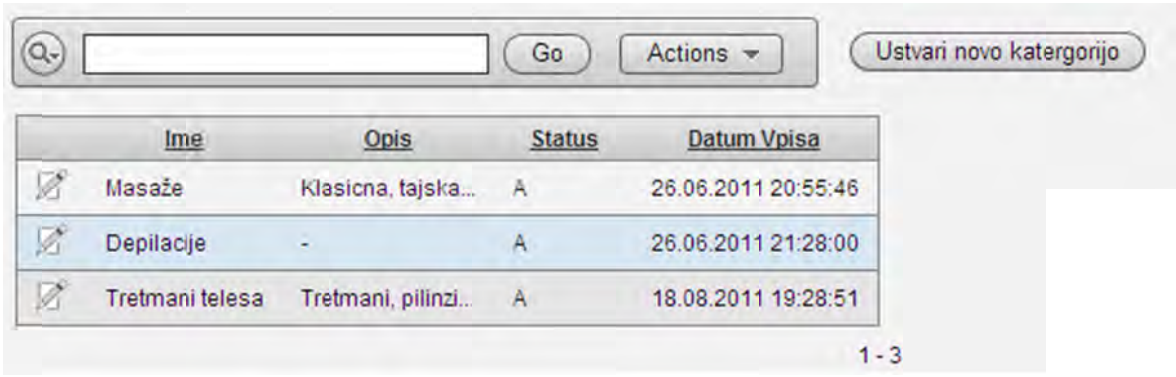
Po potrditvi se v tem primeru zapis izbriše in njegov priklic ni več možen. Podobno opozorilo se prikaže tudi ob brisanju šifrantov, artiklov, tretmajev itn. (velja isti princip brisanja).

5.2.10 Pregled, dodajanje in brisanje šifrantov

Šifranti so načeloma namenjeni enkratnemu vnosu podatkov, ki jih potrebujemo pri poslovanju. Dostop do šifrantov je v naši aplikaciji na zavihku administracija, ki je seveda viden samo administratorju sistema. Na sliki 24 je prikazan seznam vseh šifrantov v aplikaciji.

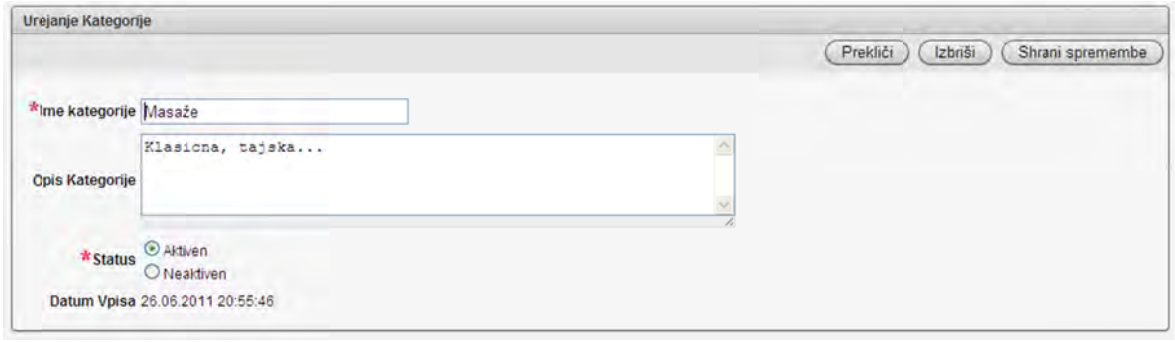
Slika 24: Seznam šifrantov

Po kliku na ime šifranta nas aplikacija avtomatično preusmeri na formo, kjer je prikazan seznam podatkov v tem šifrantu, kot je prikazano na sliki 25. Če želimo ustvariti novo kategorijo kliknemo na gumb Ustvari novo kategorijo in nas aplikacija preusmerja na formo za ustvarjanje nove kategorije.



Slika 25: Seznam kategorij

Ko kliknemo na ime šifranta ali na ikono zraven, nas aplikacija preusmeri na formo za urejanje tega podatka v šifrantu. Na voljo imamo še gumb prekliči, ki nas vrne na seznam kategorij (slika 25), gumb izbriši, ki izbriše zapis iz šifranta, in gumb shrani spremembe, kot je prikazano na sliki 26.



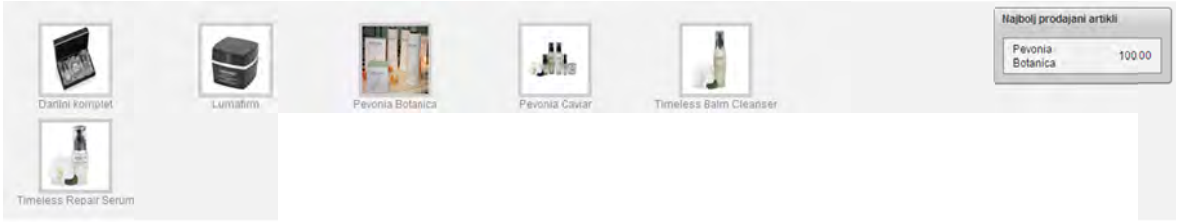
Slika 26: Urejanje podatkov v šifrantu

Dodajanje, spreminjanje in brisanje vseh preostalih šifrantov naše aplikacije poteka na podoben način. Preden začnemo uporabljati aplikacijo, je priporočljivo, da administrator napolni šifrante s podatki. V ta namen smo naredili SQL skripto, ki napolni osnovne podatke, potrebne za pravilno delovanje aplikacije.

5.2.11 Artikli

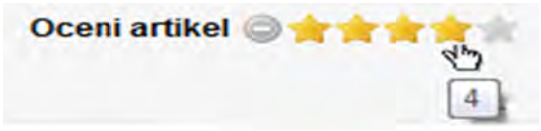
Poleg tretmajev imamo v naši aplikaciji še modul prodaja artiklov. Vsi uporabniki aplikacije imajo dostop do artiklov, ki so v ponudbi, tako lahko vidijo tudi opis, ceno in status, ki pove, ali je artikel trenutno na voljo. Na sliki 27 je prikazan seznam artiklov, ki je

za vse uporabnike aplikacije enak.



Slika 27: Artikli

Ko kliknemo na sliko artikla, se prikaže forma, kjer so podrobnosti artikla. Forma se po funkcionalnosti razlikuje v tem, da uporabnik vidi samo ceno, opis in ali je artikel na voljo, administrator sistema pa lahko zamenja sliko artikla, spremeni opis in ceno ter določi, ali je artikel na voljo. Spodaj imajo uporabniki na voljo še možnost, da podajo svoje mnenje oziroma oceno določenega produkta (slika 28). Ocenjevanje je narejeno z dodatkom jquerry, povprečno oceno za določeni izdelek dobimo s plsql funkcijo za izračun povprečja AVG.



Slika 28: Ocenjevanje artiklov

6 SKLEP

Glavni cilj diplomske naloge je bil sistematično razviti spletno aplikacijo za interno naročanje in vodenje strank v orodju Oracle APEX. Diplomska naloga prikazuje smernice, orodja, tehnike in tehnologije, ki smo jih uporabili pri izdelavi in razvoju spletne aplikacije.

Pri samem tehničnem razvoju aplikacije za vodenje strank nismo naleteli na bistvene težave, predvsem so bile težave z zahtevami naročitelja, ki za marsikakšno želeno funkcionalnost tudi sam ni vedel, kako naj bi se obnašala.

Spletna aplikacija za sprejem in vodenje strank je v splošnem namenjena zaposlenim, da imajo pregled uporabnikov in koledar naročanj v elektronski obliki, ki omogoča lažje in nemoteno poslovanje. Hkrati aplikacija podpira modul za uporabnika, ki se lahko sam prijavi na določen tretma, ureja svoj profil in pogleda novice, akcije, ponudbo, opis in cene tretmajev ter artiklov, ki so v prodaji. Poleg tega imamo na voljo modul za izpis računov. V bližnji prihodnosti pričakujemo, da bo aplikacija popolnoma nadomestila trenutni časovno in ekonomično neučinkoviti sistem.

V nadaljevanju podajamo določene ideje za izboljšavo spletne aplikacije, do katerih smo prišli med samim razvojem, izvedbo in testiranjem. To so:

- potrditev registracije uporabnikov po elektronski pošti,
- pošiljanje akcij, obvestil in tekočih novic uporabnikom po elektronski pošti,
- dodajanje košarice za spletno prodajo artiklov,
- dodajanje modula za elektronsko plačevanje tretmajev in konto uporabnika,
- dodajanje videov in kratkih flesh animacij na stran.

Literatura

[1] Beynon-Davies Paul, Williams Michael D., "The diffusion of information systems development methods ", Orlando (FL): Journal of Strategic Information Systems, str. 29-46, 2003.

[2] P. Fraternali, "Tools and approaches for developing dataintensive web applications: a survey ", ACM Comput. Surv, 257–263, 1999.

[3] Doug Gault, Karen Cannell, Patrick Cimolini, Martin Giffy D'Souza,and Timothy St. Hilaire, "Beginning Oracle Application Express 4", Apres, str. 13,270, 2011.

[4] Arie Geller Matthew Lyon, "Oracle Application Express 3.2 The Essentials and More", Birmingham –Mumbai, str. 7-11, 2010

[5] Keyes, J. , "Software Engineering Handbook (1st ed.) ", New York, Auerbach Publications.

[6] Richardson Lee, "Rapid Application Development", [URL: http://www.blueink.biz/RapidApplicationDevelopment.aspx], BlueInk, 10.9. 2011.

[7] Maner Walter, "Rapid Application Development", [URL: http://csweb.cs.bgsu.edu/maner/domains/RAD.htm], 1997.

[8] Martin James, "Rapid application development", New York, MacMillan publishing company, str. 788, 1991.

[9] McConnell, S., "Rapid Development: Taming Wild Software Schedules", Washington, Microsoft Press, 1996.

[10] Valacich J.S., George J.F. in Hoffer J.A., "Essentials of System Analysis and Design", New Jersey, Prentice Hall, str. 480 2005

[11] Whitten, J. L., Bentley, L. D., Dittman, K. C. , "Systems Analysis and Design Methods (6th ed.) ", London, McGraw-Hill, 2004.

[12] G. Žerak, "Izdelava in polnjenje področnega bančnega podatkovnega skladišča", Diplomsko delo, FAKULTETA ZA ORGANIZACIJSKE VEDE, Kranj, str. 18, 2006.

[13] (2011) Oracle Apex. Dostopno na: http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html

Priloge

Priloga A. Pl/Sql funkcije

```
/* preveri_uporabIme (uporab_ime IN VARCHAR2) */

CREATE OR REPLACE FUNCTION preveri_uporabIme (uporab_ime IN VARCHAR2)
RETURN BOOLEAN IS
BEGIN
    FOR c1 IN (SELECT uporab_ime FROM ESL_UPORABNIK)
    LOOP
        IF (upper(uporab_ime)=UPPER(c1.uporab_ime)) THEN
            RETURN FALSE;
        END IF;
    END LOOP;

    RETURN TRUE;

    EXCEPTION
    WHEN OTHERS THEN
      RETURN FALSE;
END;


/* preveri_uporabime_geslo  (p_uporab_ime IN VARCHAR2, p_geslo IN VARCHAR2)
*/


CREATE OR REPLACE FUNCTION preveri_uporabime_geslo  (p_uporab_ime IN
VARCHAR2, p_geslo IN VARCHAR2)
RETURN BOOLEAN IS
BEGIN
    FOR c1 IN (SELECT uporab_ime, geslo
                 FROM ESL_UPORABNIK
                WHERE PRVA_PRIJAVA IS NULL)
    LOOP
        IF((upper(p_uporab_ime)=upper(c1.uporab_ime)) AND
(p_geslo=c1.geslo)) THEN
            RETURN TRUE;
        END IF;
    END LOOP;
-- ce se uporabnik prvic prijavlja v sistem se v polje prva prijava vpiše -
-- začasno geslo
-- po uspešni prijavi se polje prva_prijava nastavi na null
    FOR c1 IN (SELECT prva_prijava
                 FROM ESL_UPORABNIK
                WHERE PRVA_PRIJAVA IS NOT NULL)
    LOOP
        IF(upper(p_uporab_ime)=upper(c1.prva_prijava) AND
upper(p_geslo)=upper(c1.prva_prijava)) THEN
            RETURN TRUE;
        END IF;
    END LOOP;

    RETURN FALSE;
    EXCEPTION
```

```
    WHEN OTHERS THEN
            RETURN FALSE;

      END;


/* preveri_dolzino (input_string varchar2) */

CREATE OR REPLACE FUNCTION preveri_dolzino (input_string varchar2) return
boolean is
    p_dolzina number;
    BEGIN

        SELECT LENGTH(to_char(input_string)) INTO p_dolzina FROM DUAL;
        If p_dolzina < 4 Or p_dolzina > 20  THEN
            RETURN FALSE;
        End If;
        RETURN TRUE;

        EXCEPTION
        WHEN OTHERS THEN
          RETURN FALSE;
    END;
```

Priloga B. javascript navidezna tipkovnica

```
/*  dodatek za slovensko-hrvaško tipkovnico (vkboard.js)*/

 avail_langs: [ ["Sl", "Slovenščina"],["Us", "English (US)"], ["Ca", "Canadian"], ["Hr", "Hrvatski"],

        ["De", "Deutsch"],["It", "Italiano"]]



/* del html kode vstopne forme v aplikacijo (slika 17) */

<html lang="sl" xmlns="http://www.w3.org/1999/xhtml" xmlns:htmldb="http://htmldb.oracle.com"
xmlns:apex="http://apex.oracle.com">
<head>
 <title>Login</title>
 <link rel="icon" href="/i/favicon.ico" type="image/x-icon">
 <link rel="shortcut icon" href="/i/favicon.ico" type="image/x-icon">
 <link rel="stylesheet" href="/i/css/apex_4_0.css" type="text/css" />
<!--[if IE]><link rel="stylesheet" href="/i/css/apex_ie_4_0.css" type="text/css" /><![endif]-->
<link rel="stylesheet" href="/i/libraries/jquery-ui/1.8/themes/base/jquery-ui-1.8.custom.min.css"
type="text/css" /><script type="text/javascript">var apex_img_dir = "/i/", htmldb_Img_Dir =
apex_img_dir;</script><script src="/i/libraries/jquery/1.4.2/jquery-1.4.2.min.js"
type="text/javascript"></script>
<script src="/i/javascript/apex_4_0.js" type="text/javascript"></script>
<script src="/i/javascript/apex_legacy_4_0.js" type="text/javascript"></script>
<script src="/i/javascript/apex_widget_4_0.js" type="text/javascript"></script>
<script src="/i/javascript/apex_dynamic_actions_4_0.js" type="text/javascript"></script>
<script src="/i/libraries/jquery-ui/1.8/ui/minified/jquery-ui-1.8.custom.min.js"
type="text/javascript"></script>
```

```
<script
src="wwv_flow_file_mgr.get_file?p_security_group_id=1256802015508696&p_flow_id=105&p_fname=vkbo
ard.js" type="text/javascript"></script>
<script type="text/javascript">
  var source = null, vkb = null, opened = true, insertionS = 0, insertionE = 0;

  var userstr = navigator.userAgent.toLowerCase();
  var safari = (userstr.indexOf('applewebkit') != -1);
  var gecko  = (userstr.indexOf('gecko') != -1) && !safari;
  var standr = gecko || window.opera || safari;
  function get_event_source(e)
  {
   var event = e || window.event;
   return event.srcElement || event.target;
  }
function init()
  {

     vkb = new VKeyboard("keyboard",    // container's id
                                                keyb_callback, // reference to the callback function
                  true,        // create the arrow keys or not? (this and the following params are optional)
                  true,        // create up and down arrow keys?
                  false,       // reserved
                  true,        // create the numpad or not?
                  "",          // font name ("" == system default)
                                                "14px",       // font size in px
                  "#000",      // font color
                  "#F00",      // font color for the dead keys
                  "#FFF",      // keyboard base background color
                  "#FFF",      // keys' background color
                  "#DDD",       // background color of switched/selected item
                  "#777",      // border color
                  "#CCC",      // border/font color of "inactive" key (key with no value/disabled)
                  "#FFF",      // background color of "inactive" key (key with no value/disabled)
                  "#F77",      // border color of the language selector's cell
                  true,        // show key flash on click? (false by default)
                  "#CC3300",    // font color during flash
                  "#FF9966",    // key background color during flash
                  "#CC3300",    // key border color during flash
                  false,       // embed VKeyboard into the page?
                  true,        // use 1-pixel gap between the keys?
                  0);          // index(0-based) of the initial layout

   // fokus polja
   source = document.getElementById("P101_USERNAME");
// REGISTRACIJA POLJA
   register_field("P101_USERNAME"); register_field("P101_PASSWORD");
   source.focus();
  }

  function keyb_change()
```

```
  {
    opened = !opened;
    vkb.Show(opened);
  }
 }

</script>
```