UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Rok Grahut

Gradnja bogatih spletnih aplikacij z ogrodjem ADF in zrni EJB

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Zoran Bosnić

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: 00107/2011
Datum: 05.04.2011

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

Kandidat: **ROK GRAHUT**

Naslov: **GRADNJA BOGATIH SPLETNIH APLIKACIJ Z OGRODJEM ADF IN ZRNI EJB**

RICH WEB APPLICATION DEVELOPMENT WITH ADF AND EJB FRAMEWORKS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvoj bogatih spletnih aplikacij (aplikacij, ki jih odlikujejo mnoge lastnosti lokalnih, namiznih aplikacij) prehaja v vse večjo rabo na svetovnem spletu.
Kandidat naj v okviru diplomske naloge primerja različna orodja za razvoj tovrstnih aplikacij. Osredotoči naj se na ogrodje ADF, ki naj ga podrobneje opiše. V nadaljevanju naj prikaže razvoj lastne vzorčne aplikacije na osnovi tega ogrodja, ki uporablja poglavitne prednostne značilnosti tega ogrodja.

Mentor:

doc. dr. Zoran Bosnić

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Rok Grahut, z vpisno številko 63040042, sem avtor diplomskega dela z naslovom:

Gradnja bogatih spletnih aplikacij z ogrodjem ADF in zrni EJB

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Zorana Bosnića
- 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 15. 10. 2011 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Zoranu Bosniću za pomoč in vodenje pri izdelavi diplomske naloge. Zahvaljujem se tudi sodelavki Ivoni Marić Šantić in ostalim sodelavcem, ki so me seznanili z ogrodjem ADF.

Kazalo

Seznam uporabljenih kratic

API – Application programming interface
ADF – Application development framework
ADF BC – ADF business components
ADF Faces RC – ADF Faces rich client
AJAX – Asynchronous JavaScript and XML
BPEL – Business process execution language
CRUD – Create, read, update and delete
CSS – Cascading style sheets
CSV – Comma-separated values
EJB – Enterprise JavaBeans
EL – Expression language
HTML – Hyper text markup language
Java EE – Java platform, enterprise edition
Java SE – Java platform, standard edition
JMS – Java message service
JPA – Java persistence API
JPQL – Java persistence query language
JSF – JavaServer Faces
JSP – JavaServer Pages
JSR – Java specification request
MVC – Model-view-controller
PPR – Partial page rendering
RIA – Rich internet applications
SQL – Structured query language
SOA – Service-oriented architecture
SVG – Scalable vector graphics
URL – Uniform resource locators
XML – Extensible markup language

Povzetek

V diplomski nalogi smo predstavili ogrodje Aplication development framework (ADF) podjetja Oracle, njegovo tehnološko ozadje, arhitekturo ter splošne lastnosti. Ogrodje ADF za implementacijo uporabniškega vmesnika ponuja komponente ogrodja ADF Faces rich client (ADF faces RC), ki omogočajo razvoj bogatih spletnih aplikacij pod skupnim imenom Rich internet applications (RIA). Le-te združujejo najboljše lastnosti tradicionalnih namiznih aplikacij z dostopnostjo preko spleta. Opisali smo tudi integrirano razvojno okolje Oracle JDeveloper, ki pokriva celotno ogrodje ADF in je tako najprimernejše integrirano razvojno okolje za razvoj aplikacij v tej tehnologiji. V praktičnem delu naloge je prikazana uporaba ogrodja na primeru spletne aplikacije za vodenje podatkov o prodaji nepremičnin. Za razvoj smo uporabili orodje JDeveloper, tehnologije v okviru ogrodja ADF in zrna Enterprise JavaBeans (EJB). V zaključku smo primerjali ogrodje ADF z ogrodjem .NET.

Ključne besede:

bogate spletne aplikacije, ogrodje ADF, zrna EJB, arhitektura MVC, integrirano razvojno okolje.

Abstract

The thesis presents Oracle ADF framework, its technological background, architecture and general features. For user interface implementation, ADF framework provides ADF Faces RC Framework, a key component for developing RIA applications that combine the best features of traditional desktop applications with access via the Internet. We also describe an integrated development environment Oracle JDeveloper, which covers the entire ADF framework and is the preferred integrated development environment for developing applications in this technology. Practical part of thesis illustrates the use of the ADF framework in development of a sample web application for managing data of real estate for sale. For development we used the JDeveloper tool, technologies of the ADF Framework and EJB. In conclusion, we compared Oracle ADF framework with Microsoft .NET framework.

Key words:

Rich internet applications, ADF framework, EJB beans, MVC architecture, integrated development environment.

Poglavje 1

Uvod

S hitrim razvojem interneta se močno širi uporaba spletnih aplikacij. Potreba po razvoju poslovnih spletnih aplikacij zato stalno narašča, s tem pa se tudi povečuje število ponudnikov oziroma razvijalcev le-te. Aplikacije lahko postanejo precej obsežne in kompleksne ter s tem tudi drage. Postajajo trend tehnologije, ki omogočajo hiter, enostaven, a vseeno zanesljiv način gradnje poslovnih rešitev. Take tehnologije tako zagotavljajo cenejši razvoj in večjo produktivnost razvijalcev, ugodni in kakovostni izdelki pa povečujejo zadovoljstvo naročnikov. Tudi tehnične zahteve naročnikov so čedalje večje. Zaželene so aplikacije z lastnostmi namiznih aplikacij, a vendar dostopne preko spleta in podprte za večino brskalnikov. Take spletne aplikacije poznamo pod skupnim imenom Rich internet applications (RIA).

Naša naloga je bila, da smo izbrali tehnologijo, ki ima želene lastnosti, vam jo predstavili ter prikazali njene prednosti in slabosti na primeru enostavne spletne aplikacije. Izbrali smo ogrodje Aplication development framework (ADF), ki temelji na programskem jeziku Java in tehnologiji JavaServer Faces (JSF) in z uporabo komponent ogrodja ADF Faces RC omogoča razvoj bogatih spletnih aplikacij RIA. Seveda pa razvoj bogatih spletnih aplikacij ni možen brez ustreznega razvojnega okolja. Z integriranimi vsemi potrebnimi tehnologijami za razvoj aplikacij v ogrodju ADF je najprimernejše integrirano razvojno okolje JDeveloper, katerega splošne lastnosti in uporabo opisujemo v poglavju 3.

Uporabo predstavljenih tehnologij smo v poglavju 4 skušali v čimvečji meri prikazati na primeru razvoja enostavne spletne aplikacije za vodenje podatkov o prodaji nepremičnin. Za implementacijo poslovno-storitvenega nivoja smo izbrali tehnologijo izven ogrodja ADF, zrna EJB, ki so del arhitekture Java platform, enterprise edition (Java EE).

Za konec smo dodali še kratko primerjavo z razvojnim ogrodjem .NET, ki prikazuje nekaj prednosti ogrodja ADF in potencialno povečanje produktivnosti pri prehodu na ogrodje ADF.

Poglavje 2

Ogrodje ADF in razvojno okolje JDeveloper

ADF je inovativno razvojno ogrodje Java EE, ki poenostavlja razvoj aplikacij Java in Java EE za boljše sodelovanje podjetij in razvijalcev pri gradnji sodobnih programskih rešitev. Temelji torej na odprtokodnih tehnologijah in s tem omogoča enostavnejšo in hitrejšo implementacijo storitveno usmerjenih aplikacij. Predstavlja celovito rešitev za gradnjo aplikacij, saj pokriva vse segmente arhitekture Model-view-controller (MVC). Majhna povezanost nivojev te arhitekture omogoča izbor poljubne tehnologije za implementacijo posameznega nivoja [7, 2].

Tehnologija ADF tudi zmanjšuje potrebo po pisanju programske kode in s tem omogoča, da se razvijalec osredotoči na funkcionalnosti aplikacije. Poleg svobode pri izbiri tehnologij posameznega nivoja pa lahko razvijalci izbirajo tudi način razvoja (deklarativen, vizualen ali pisanje programske kode) ter razvojno okolje (JDeveloper, Eclipse, NetBeans ipd.) [2].

Oracle ADF 11g je del arhitekture Oracle Fusion, ki temelji na arhitekturi Service-oriented architecture (SOA). Ogrodje ADF 11g in orodje JDeveloper 11g skupaj tvorita celoten nabor programske opreme, imenovan Oracle Fusion middleware. Ta nabor ponuja infrastrukturo za celovite poslovne programske rešitve večjega formata in programerjem iz različnih ozadij omogoča produktivno delo preko integriranega razvojnega okolja, ki ponuja enako

funkcionalnost in zrelost, kot jo je ponujala tehnologija Oracle Forms v preteklosti [7]. Več o Oracle Fusion v sledečem poglavju.

2.1 Tehnološko ozadje

2.1.1 Java EE

Java EE je poslovna izdaja ogrodja, namenjenega razvoju, gradnji in namestitvi poslovnih spletnih aplikacij. Temelji na programskem jeziku Java, zato deluje neodvisno od operacijskega sistema. Od standardne izdaje ogrodja (Java platform, standard edition – Java SE) se razlikuje po tem, da dodaja knjižnice, ki omogočajo postavitev večnivojske, porazdeljene programske opreme Java, ki temelji na modularnih komponentah in se izvaja na aplikacijskem strežniku [25].

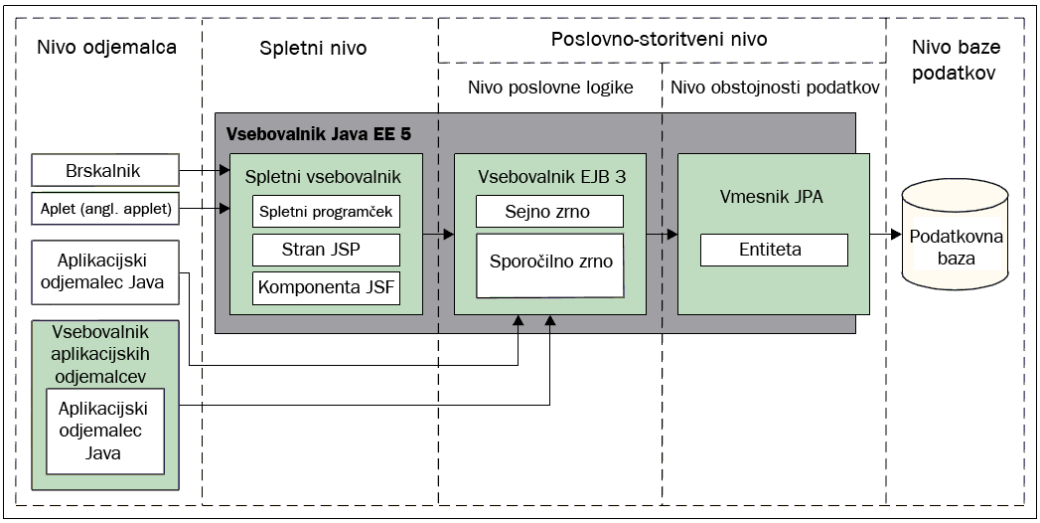
Arhitekturni model Java EE

Java EE definira komponente (npr. tehnologije, kot so JavaServer Pages – JSP in EJB), ki jih združujemo z drugimi komponentami, da bi ustvarili aplikacijo. Arhitekturni model Java EE deli te komponente v nivoje. Aplikacijska logika je razdeljena v komponente glede na funkcijo, komponente, ki sestavljajo aplikacijo Java EE, pa so nameščene na različne naprave, in to glede na nivo, ki mu pripadajo [1, 9].

Slika 1 prikazuje štiri nivoje arhitekturnega modela Java EE:

- nivo odjemalca (angl. client layer), ki se izvaja na računalniku odjemalca,
- spletni nivo (angl. web layer), ki se izvaja na strežniku Java EE,
- poslovno-storitveni nivo (angl. business layer), ki se izvaja na strežniku Java EE,
- nivo baze podatkov (angl. database layer), ki se izvaja na baznem strežniku.

Prikazane so tudi vrste objektov, ki se uporabljajo v okviru posameznega objekta.



Slika 1: Arhitekturni model ogrodja Java EE, verzija 5 [10]

2.1.2 EJB

EJB je tehnologija, ki jo ponuja ogrodje Java EE za realizacijo poslovno-storitvenega nivoja. Ta nivo je v ogrodju Java EE razdeljen na dva podnivoja, in sicer nivo poslovne logike in nivo »obstojnosti podatkov« (slika 1). Prvi skrbi za sejna zrna (angl. session bean) in sporočilna zrna (angl. message driven bean), ki so postavljena in zagnana v vsebovalniku EJB, drugi pa za entitete, za katere obstojnost in shranjevanje v podatkovno bazo skrbi vmesnik Java persistence API (JPA). Kot je razvidno iz slike 1, poznamo več vrst objektov EJB [11]:

- sejna zrna (angl. session beans) – poslovni objekti dveh vrst, s stanjem (angl. stateful) ali brez stanja (angl. stateless),
- sporočilna zrna (angl. message-driven beans) – objekti, namenjeni upravljanju s prihajajočimi sporočili asinhronih sistemov, kot je Java message service (JMS),
- entitete – objekti za dostop do podatkov (so del JPA specifikacije).

Vmesnik JPA je del specifikacije EJB in skrbi za objektno-relacijsko preslikavanje v aplikacijah Java EE. Obstaja tudi več implementacij vmesnika JPA: TopLink, Hibernate, OpenJPA. Za poizvedbe podatkov iz entitet vmesnik JPA uporablja objektno orientiran poizvedbeni jezik Java persistence

query language (JPQL). Jezik je zelo podoben poizvedbenemu jeziku Structured query language (SQL), le da namesto s tabelami operira z entitetnimi objekti JPA.

2.1.3 AJAX

AJAX (Asynchronous JavaScript and XML) je skupina med seboj povezanih tehnologij za spletni razvoj, ki omogočajo kreiranje interaktivnih spletnih aplikacij. Z uporabo funkcionalnosti AJAX lahko aplikacija pošilja podatke na strežnik in sprejema podatke iz strežnika asinhrono (v ozadju), brez osveževanja celotne strani [13]. AJAX uporablja tehnologijo JavaScript in objekt XMLHttpRequest, ki omogočata asinhrono izmenjavo med brskalnikom in strežnikom v izogib osveževanju celotne strani.

2.1.4 Predznanja, potrebna za razvoj

Razvoj aplikacij Java EE temelji na mnogih tehnologijah in s tem omogoča kreiranje aplikacije z vsemi potrebnimi funkcionalnostmi. Orodja, kot je JDeveloper, sicer precej pomagajo, vendar mora spletni razvijalec vseeno poznati več različnih programskih jezikov [1].

Nekaj teh tehnologij je značilnih za spletne aplikacije (npr. Hyper text markup language – HTML, Cascading style sheets – CSS, JavScript), zato je njihovo poznavanje potrebno tudi pri drugih ogrodjih. Znanje, potrebno za razvoj, je odvisno tudi od izbire tehnologije za implementacijo posameznega nivoja ter od velikosti in tipa spletne aplikacije. V tabeli 1 so našteti programski jeziki, stopnja potrebnega znanja posameznega jezika ter opis primarne uporabe.

Programski jezik (uporaba)	**Stopnja znanja**	**Primarna uporaba**
Java (uporaba ogrodij)	osnovno	Poslovna logika, potrjevanje, pogojni tok strani.
Java (razširjevanje in spreminjanje značilnosti ogrodja)	napredno	Spreminjanje funkcionalnsti ogrodja.
HTML	osnovno/ni potrebno	Spreminjanje programske kode generirane z vlečenjem komponent na stran.
Extensible markup language (XML)	osnovno/vmesno	Urejanje generirane kode.
JavaScript	osnovno/ni potrebno	Kreiranje specifičnih interaktivnih funkcij.
CSS	vmesno/ni potrebno	Spreminjanje izgleda aplikacije. Običajno uporabljamo teme.
Expression language (EL)	osnovno/ni potrebno	Oskrba komponent s podatki iz aplikacijskih objektov.
Groovy	osnovno	Potrjevanje, sporočila o napakah. Samo pri uporabi ADF Business Components

Tabela 1: Znanja, potrebna za razvoj Java EE aplikacij [1]

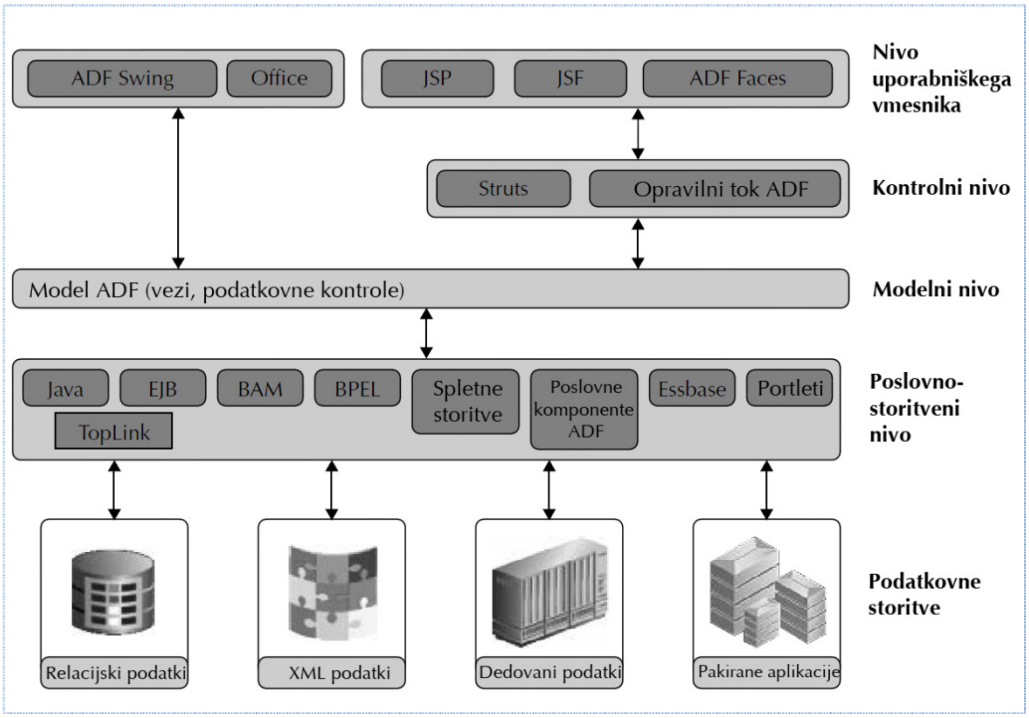
2.2 Arhitektura ogrodja ADF

Arhitektura ogrodja ADF temelji na arhitekturi MVC, ki aplikacijo razdeli na tri nivoje:

- modelni nivo, ki skrbi za dostop do podatkov in poslovno logiko,
- nivo uporabniškega vmesnika, ki skrbi za interakcijo z uporabnikom,
- kontrolni nivo, ki skrbi za aplikacijski tok in povezuje modelni nivo ter nivo uporabniškega vmesnika.

Ločevanje aplikacije na te tri nivoje poenostavlja vzdrževanje in ponovno uporabo komponent. Rezultat neodvisnosti posameznega nivoja je storitveno-orientirana arhitektura SOA.

Arhitektura ADF še izboljša arhitekturo MVC, tako da modelni nivo loči od poslovne logike. Modelni nivo lahko tako deluje z različnimi implementacijami poslovne logike na konsistenten način [2]. Slika 2 prikazuje arhitekturo z vsemi štirimi nivoji in podatkovnimi storitvami, ki so vir podatkov aplikacije. Prikazane so povezave med posameznimi nivoji in tehnologije, ki so na voljo za uporabo na posameznem nivoju.



Slika 2: Arhitektura ADF [1]

Kot smo omenili v uvodu, lahko razvijalci izbirajo različne tehnologije za implementacijo vsakega od nivojev. Ne glede na izbiro tehnologije posameznega nivoja ogrodje ADF v kombinaciji z orodjem JDeveloper zagotovlja enako produktivno izkušnjo pri razvoju aplikacij. Sledi podroben opis posameznih nivojev.

2.3 Poslovno-storitveni nivo

Poslovno-storitveni nivo (angl. business services layer) skrbi za interakcijo s podatkovnim nivojem. Zagotavlja dostop do podatkovne baze, skrbi za obstojnost podatkov, izvajanje poslovne logike in upravljanje s transakcijami. V tehnologiji ADF je ta nivo lahko realiziran na različne načine: kot zrna Java (JavaBeans), zrna EJB, poslovne komponente ADF (angl. ADF business components – ADF BC), spletne storitve (angl. web services), objekti JPA [6]. Nabor tehnologij tega nivoja je viden tudi na sliki 1.

Za implementacijo poslovno-storitvenega modela pri razvoju aplikacij ADF se najpogosteje uporabljajo poslovne komponente ADF BC, ki so del ogrodja ADF in s tem tudi enostavnejše za uporabo od ostalih tehnologij. Izbira ADF BC se priporoča bazno usmerjenim programerjem in razvijalcem s poznavanjem tehnologije Oracle Forms.

Pri alternativnih implementacijah je najpogostejša uporaba sejnih oz. strežniških zrn EJB. Zrna EJB so del ogrodja Java EE in predstavljajo standard za poslovno-storitveni nivo aplikacij Java EE. Ta izbira omogoča nekaj več fleksibilnosti in je pogostejša med Java razvijalci.

Za prototip aplikacije ADF, ki smo jo predstavili v šestem poglavju, smo izbrali zrna EJB, zato smo jih tudi podrobneje opisali v razdelku 2.1.2.

Poslovne komponente ADF BC

Poslovne komponente ADF BC so tehnologija arhitekture Oracle Fusion za kreiranje poslovnih storitev. Tehnologija je bazno naravnana (tabele, poizvedbe, transakcije, tipi objektov), zato je pisana na kožo razvijalcem s poznavanjem baznih tehnologij.

Vrste poslovnih komponent:

- definicije entitetnih objektov (angl. entity object definitions),
- povezave (angl. associations),
- definicije objektov pogleda (angl. view object definitions),
- definicije povezav pogleda (angl. view link definitions),
- definicije aplikacijskih modulov (angl. application module definitions).

Komponente ADF BC so lahko postavljene na več načinov, preklapljanje med njimi je deklarativno, brez spreminjanja izvorne kode:

- v vsebovalnik Java EE kot del aplikacije,
- v vsebovalnik Java EE kot del sejnega zrna EJB in so tako na voljo drugim aplikacijam Java EE,
- kot spletna storitev za uporabo v sistemih BPEL (Business process execution language),
- v knjižnico, ki jo lahko vključimo v drug JDeveloper projekt.

2.4 Modelni nivo

Modelni nivo (angl. model layer) povezuje uporabniški vmesnik preko različnih objektov aplikacije s poslovnimi storitvami. V ogrodju ADF se ta nivo imenuje model ADF (angl. ADF Model) in je ključna komponenta ogrodja, saj deluje kot lepilo med različnimi implementacijami poslovnih storitev, kot je EJB, in uporabniki teh storitev, kot je uporabniški vmesnik.

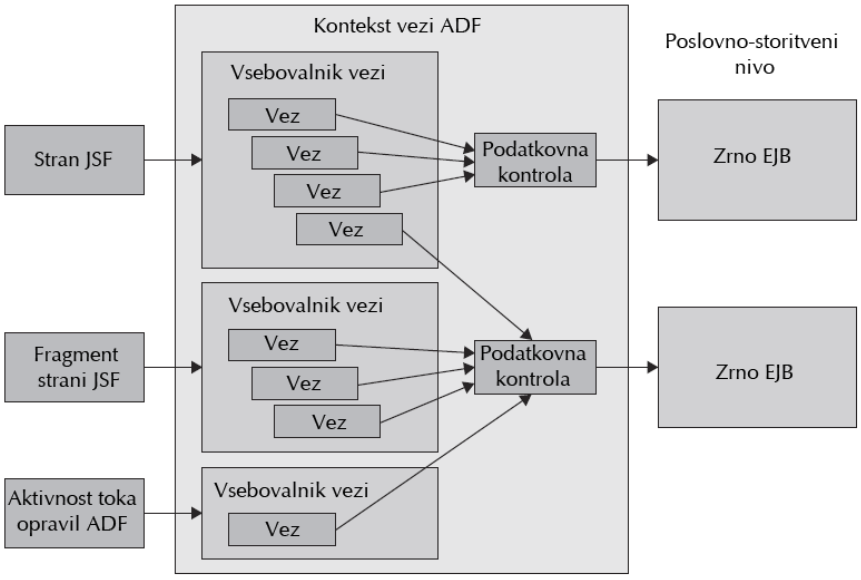
2.4.1 Model ADF

Ogrodje ADF implementira modelni nivo kot enoten vmesnik, ki ga lahko uporabimo za dostop do kateregakoli tipa poslovnih storitev. Nivo je sestavljen iz dveh tipov komponent, podatkovnih kontrol (angl. data controls) in vezi (angl. bindings). Podatkovna kontrola je implementirana iz specifikacije Java EE za dostop do podatkov, Java specification request (JSR) 227. Podatkovne kontrole uporabljajo standardne vmesnike za predstavitev poslovnih storitev. Z uporabo okolja JDeveloper so te kontrole vidne kot ikone, ki jih lahko povlečemo na stran. Takrat JDeveloper avtomatsko naredi povezave med stranjo in poslovnimi storitvami. Med izvajanjem aplikacije model ADF bere informacije, ki opisujejo podatkovne kontrole in vezi iz ustreznih datotek XML, in implementira dvosmerno povezavo med uporabniškim vmesnikom in poslovnimi storitvami [2, 7].

Oracle ADF ponuja implementacije podatkovnih kontrol za večino pogosto uporabljenih poslovno-storitvenih tehnologij. Poleg podpore za aplikacijske module ADF model ADF ponuja tudi podporo za servisne tehnologije, kot so [7]:

- sejna zrna EJB in entitete JPA,
- zrna Java (JavaBeans),
- spletne storitve,
- XML,
- datoteke CSV (angl. Comma-separated values).

Slika 3 prikazuje arhitekturo nivoja Model ADF z uporabo strani JSF, toka opravil ADF in zrn EJB.

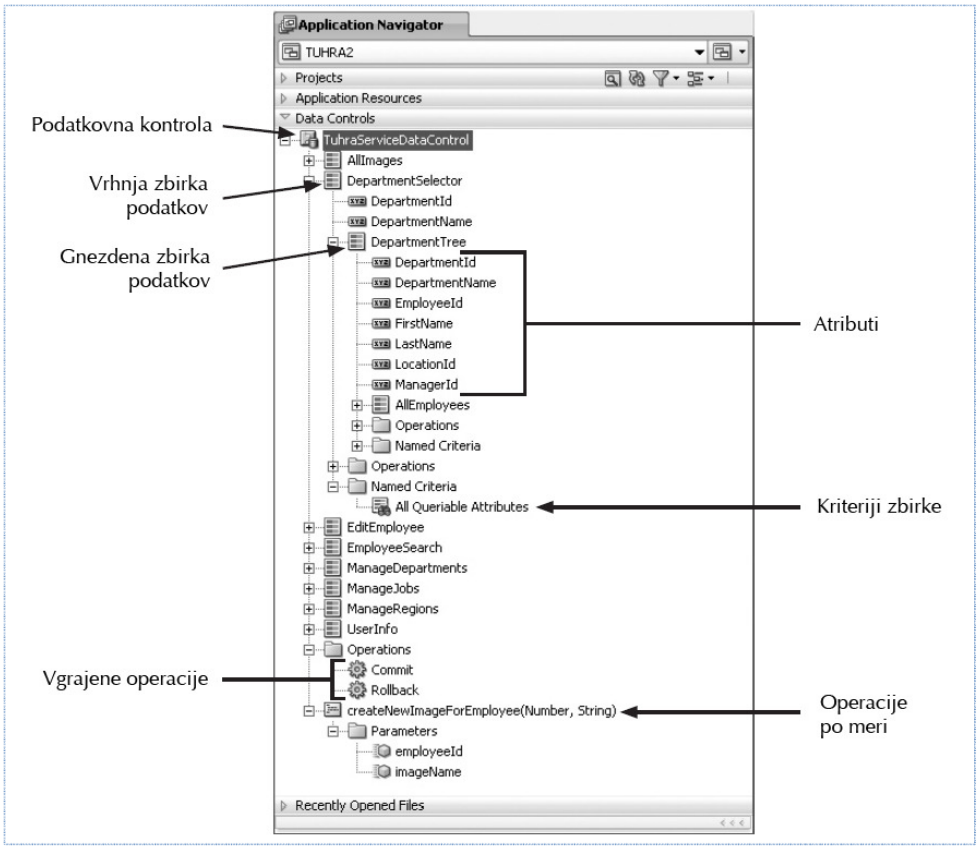


Slika 3: Arhitektura modelnega nivoja »model ADF« [1]

2.4.2 Podatkovne kontrole

Podatkovna kontrola (angl. data control) je Java objekt na modelnem nivoju, ki deluje kot dodaten nivo kode za posredno komunikacijo s poslovnimi storitvami. Brez uporabe podatkovnih kontrol bi morali za komunikacijo z vsakim tipom poslovnih storitev uporabljati svojo vrsto ukazov. Podatkovne kontrole ponujajo poenoten vmesnik in tako omogočajo konsistenten pregled

ter razvoj z uporabo različnih implementacij poslovno-storitvenega nivoja [1]. Na sliki 4 je okno za pregled podatkovnih kontrol v razvojnem okolju JDeveloper. Iz tega okna med razvojem s pomočjo metode »povleci in spusti« (angl. drag-and-drop) dodajamo povezave iz določene strani JSF na želeno podatkovno kontrolo.



Slika 4: Podatkovne kontrole [1]

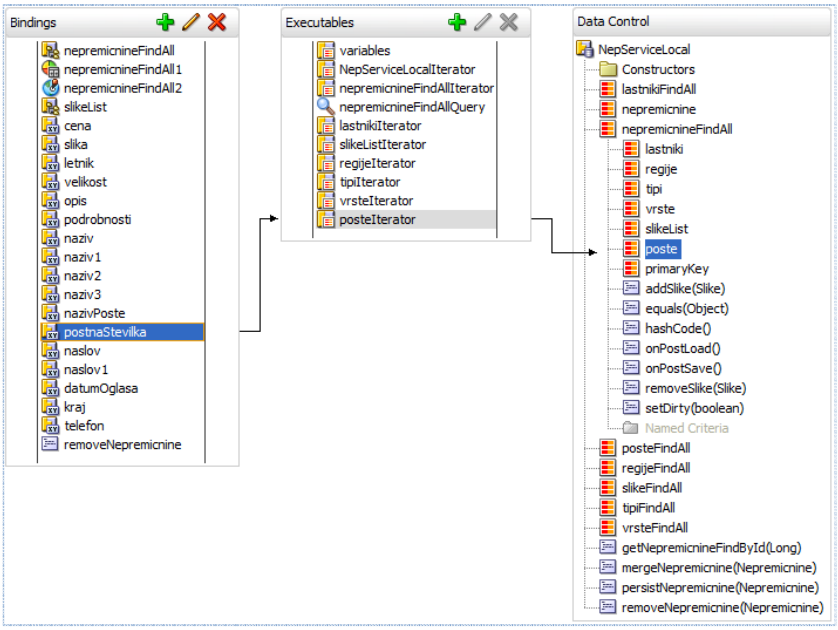
2.4.3 Vezi

Vezi omogočajo interakcijo med komponento uporabniškega vmesnika ali aktivnostjo toka opravil in podatkovno kontrolo. Vezi so shranjene v vsebovalniku vezi (angl. binding container). Vsebovalnik vezi je objekt jezika Java na modelnem nivoju, ki strani, fragmentu strani ali toku opravil omogoča dostop do podatkov. Stran ali aktivnost z definiranim ustreznim vsebovalnikom vezi lahko dostopa do vsebovalnika preko ustreznega izraza

Expression language (EL) #{bindings} [1]. Definicija vsebovalnika vezi za določeno stran je shranjena v datoteki definicij strani (angl. page definition file). To je datoteka XML, ki vsebuje podatke o vezeh za določeno stran. Datoteka se kreira avtomatsko z vlečenjem podatkovnih kontrol na stran. Povezava med stranjo JSF in datoteko definicij je opisana v datoteki DataBindings.cpx.

Vezi običajno dodajamo na stran JSF s pomočjo vlečenja zbirk ali atributov iz podatkovnih kontrol na mesto, kjer jih želimo prikazati ali urejati. Pri tem moramo izbrati, v okviru katere komponente ADF Faces bodo podatki prikazani. Nato se poleg novih elementov na strani JSF kreirajo tudi vse potrebne vezi ADF, ki jih lahko pregledujemo in urejamo preko pregleda vezi (slika 5). Vezi lahko dodajamo in urejamo tudi z urejanjem programske kode v datoteki definicij strani.

Slika 5 prikazuje tudi vmesnik med vezmi in podatkovnimi kontrolami, izvršljive povezave (angl. executables). Izvršljiva povezava je običajno iterator, ki deluje kot kazalec na trenutni zapis zbirke podatkov.



Slika 5: Pregled vsebovalnika vezi

2.5 Kontrolni nivo

Kontrolni nivo (angl. controller layer) skrbi za aplikacijski tok in posredovanje uporabnikovih akcij modelu aplikacije. Za spletne aplikacije imamo v orodju JDeveloper na voljo tri vrste kontrolerjev: kontroler Oracle ADF, kontroler JSF in Apache Struts. Zaradi največje funkcionalnosti je logična izbira kontroler ADF, katerega podrobnejši opis sledi v nadaljevanju. Ne glede na izbiro kontrolerja se aplikacijski tok tipično oblikuje s postavitvijo posameznih strani in navigacijskih pravil na diagram [6].

2.5.1 Kontroler ADF

Kontroler ADF je ena najpomembnejših značilnosti ogrodja ADF, predstavljenih v verziji 11g, in predstavlja ključno vlogo v vseh aplikacijah, ki temeljijo na tehnologiji Oracle Fusion. Za razliko od ostalih večjih delov ogrodja, kot je ADF BC, je kontroler ADF popolnoma nova funkcionalnost, čeprav ima nekaj podobnih konceptov kot Apache Struts in kontrolerji JSF v preteklosti [1].

Kontroler ADF omogoča razbitje aplikacijkega toka na manjše tokove opravil (angl. task flow) z možnostjo ponovne uporabe, vključevanje nevizualnih komponent in kreiranje tokov fragmentov strani (angl. page fragment), ki tečejo znotraj regije, vsebovane v posamezni strani [6].

Prednosti pred kontrolerjem JSF

Kontroler JSF omogoča osnoven tok strani, ki je definiran v datoteki faces-config.xml. Le-ta je dovolj za osnovne aplikacije, pri aplikacijah z bogatim uporabniškim vmesnikom pa nastane kar nekaj težav, ki jih odpravi kontroler ADF [1]:

- spremembe samo enega dela spletne strani so možne samo z uporabo manjših enot, fragmentov strani, ki jih navigacija JSF ne podpira,
- navigacija JSF pozna samo dva tipa elementov, strani in povezave med njimi, kontroler ADF pa omogoča tudi pogojno usmerjanje, parametre ipd.,
- kontroler ADF omogoča ponovno uporabo strani in tokov.

2.5.2 Tok opravil

Neomejeni tokovi opravil

Neomejen tok opravil nima eksplicitno definiranega začetka in konca. Navigacija znotraj takega toka opravil se lahko začne kjerkoli v toku tako, da uporabnik usmeri brskalnik na določeno aktivnost. Primer neomejenega toka opravil je datoteka adfc-concig.xml, ki je avtomatsko ustvarjena pri vsakem novem ADF projektu. Aplikacija ima lahko definiranih več neomejenih tokov opravil, a so pri zagonu aplikacije vsi združeni v enega globalnega [8].

Omejeni tokovi opravil

Omejen tok opravil predstavlja eno bistvenih komponent ogrodja ADF. Pospešuje produktivnost, vzpodbuja ponovno uporabo in omogoča enostavnejši razvoj kompleksnejših spletnih aplikacij [1].

Od neomejenega toka se razlikuje v več pomembnih lastnostih [8]:

- ima eno vstopno in nič ali več izstopnih točk,
- lahko je parametriziran in obdrži privatno stanje,
- lahko je klican preko naslova URL (Uniform resource locators), lahko je ugnezden v drugem toku opravil z uporabo aktivnosti za klic tokov opravil, ali pa je ugnezden na strani JSF ali fragmentu strani JSF z uporabo regijske komponente ADF,
- lahko uporablja celotne strani JSF ali samo fragmente strani JSF,
- če je klican iz drugega toka opravil, lahko prejme vhodne parametre in vrača izhodne.

Aktivnosti in kontrole toka opravil

Pri kreiranju tokov opravil imamo na voljo več različnih aktivnosti ter kontrol toka opravil. V nadaljevanju smo jih našteli in opisali njihove bistvene funkcije.

Aktivnosti:

- pogled (angl. view) – vsebovalnik za strani in fragmente strani,
- klic drugega toka opravil (angl. task flow call) – usmeritev na nov tok opravil,
- usmerjevalnik (angl. router) – namenjen pogojnemu usmerjanju toka opravil,
- vračanje na predhodni tok (angl. task flow return) – vračanje na klicoči tok opravil,
- klic metode (angl. method call) – aktivnost, namenjena klicu določene metode,
- pogled URL (angl. Uniform resource locators (URL) view) – povezava na stran izven aplikacije.

Kontrole toka opravil:

- povezava kontrole toka opravil (angl. control flow case) – povezuje aktivnosti med seboj. Vsaka povezava ima definirano izvorno aktivnost, ciljno aktivnost ter akcijo, ki povzroči navigacijo toka opravil od izvorne do ciljne aktivnosti. Te akcije so dostopne iz komponent strani JSF preko lastnosti »action«,
- splošno pravilo kontrole toka opravil (angl. wildcard control flow rule) se uporablja za klice aktivnosti iz različnih delov aplikacije, npr. za menjavanje globalnih zavihkov ali kot vstopna točka aplikacije.

Upravljana zrna

Upravljana zrna (angl. managed beans) so razredi jezika Java, namenjeni shranjevanju in delu s podatki med delovanjem aplikacije. Lahko hranijo podatke poslovne logike ali podatke, ki jih vnesemo preko obrazca. Tipično so zrna sestavljena iz atributov privatnega tipa (angl. private), do katerih vrednosti dostopamo preko metod »get« in jim nastavljamo vrednost preko

metod »set«. Zrna so tudi mesto za programske implementacije akcij komponent uporabniškega vmesnika (npr. akcije, poslušalci sprememb vrednosti, poslušalci akcij, metode za shranjevanje ipd.).

Pri definiranju upravljalnega zrna na toku podatkov moramo podati ime in lokacijo razreda, lastnosti ter želen spominski obseg (angl. memory scope). Različni objekti, ki jih potrebujemo, so shranjeni v spominu za dostop med delovanjem aplikacije. Ti objekti so vezi, upravljana zrna, atributi ipd. Spominski obseg definira, kdaj in koliko časa je dostopen nek objekt.

Standardni JSF in Servlet spominski obsegi:

- obseg aplikacije (angl. application scope) – dostopnost v celotni aplikaciji v času delovanja aplikacije,
- obseg seje (angl. session scope) – dostopnost v celotni aplikaciji v času seje,
- obseg zahtevka (angl. request scope) – dostopnost na strani, ki ji posredujemo zahtevek.

Dodatni ADF spominski obsegi:

- obseg pogleda (angl. view scope) – dostopnost na strani, s katere dostopamo,
- obseg toka strani (angl. pageflow scope) – dostopnost, dokler je aktiven primerek toka podatkov, preko katerega dostopamo do objekta,
- obseg podpornega zrna (angl. backingbean scope) – dostopnost v okviru fragmenta strani oz. deklarativne komponente.

2.6 Nivo uporabniškega vmesnika

V osnovni izvedbi nivo uporabniškega vmesnika (angl. view layer) spletne aplikacije predstavlja stran z dodanimi komponentami (npr. vnosno polje, tabela, gumb) za prikaz podatkov ter interakcijo. Ogrodje ADF pa v okviru ogrodja ADF Faces RC ponuja dodatne komponente in funkcionalnosti, ki omogočajo kreiranje bogatih uporabniških vmesnikov z možnostjo ponovne uporabe [12].

Komponente ADF Faces so del najvišjega nivoja arhitekture MVC in tako najbolj direktno predstavljajo uporabniško izkušnjo, funkcionalnosti ter uporabnost aplikacije, zato smo v naslednjem razdelku opisali nekaj njihovih najpomembnejših lastnosti.

2.6.1 Ogrodje ADF Faces RC

Ogrodje ADF Faces RC je implementacija standarda JavaServer Faces (JSF). JSF je ogrodje, ki ga je razvilo podjetje Sun Microsystems, namenjeno hitremu in enostavnemu razvijanju spletnih aplikacij. ADF Faces vključuje tudi funkcionalnosti AJAX (Asinhroni JavaScript in XML) [3].

ADF Faces komponente ločimo v več kategorij, glavni dve kategoriji sta [1]:

- **atomarne komponente** (angl. atomic compnents), namenjene delu z vsebinskimi objekti (npr. vnosna polja, spustni meniji, gumbi),
- **razporeditvene komponente** (angl. layout components), ki obkrožajo atomarne komponente in druge razporeditvene vsebovalnike in ponujajo različne možnosti avtomatske razporeditve.

ADF Faces RC ponuja več kot 100 različnih komponent RIA, vključno s hierarhičnimi podatkovnimi tabelami, drevesnimi meniji, vgrajenimi dialogi, zložljivimi vsebovalniki, razdelilniki in dinamičnimi tabelami. Vključuje pa tudi komponente za vizualizacijo podatkov, ki vključujejo skalabilno vektorsko grafiko (angl. scalable vector graphics – SVG) in animacije Flash, kar omogoča izris dinamičnih grafikonov, merilnikov, grafov in drugih grafičnih komponent, ki omogočajo realno-časovni pogled na podatke. Vsaka komponenta ponuja tudi prilagodljivost delovanja in izgleda, podporo internacionalizaciji in dostopnst [3].

2.6.2 Življenjski cikel komponent

Ogrodje ADF Faces RC temelji na ogrodju JSF, zato za osnovo uporablja standarden življenjski cikel JSF z nekaj dodatnimi funkcionalnostmi (dodatna komponenta »podobrazec« (angl. subform), dodatni časovni obsegi ipd.).

Ko je stran JSF posredovana in je zahtevana nova stran, je sprožen življenjski cikel JSF. Ta nato poskrbi za posredovanje vrednosti, potrjevanje komponent, navigacijo, prikaz komponent na ciljni strani ter shranjevanje vrednosti in obnovitev stanja.

2.6.3 Potrjevanje in pretvorba vnosa

Komponente ADF Faces omogočajo avtomatsko pretvorbo podatkov (angl. conversion). Podatki so v aplikaciji na modelnem nivoju shranjeni kot spremenljivke različnih tipov (cela števila, decimalna števila, datumi ipd.). Uporabniški vmesnik mora te podatke predstaviti v taki obliki, da jih lahko uporabnik vidi in ureja, pri shranjevanju sprememb pa spet pretvoriti nazaj v tip, ki ga zahteva aplikacija. Za to nalogo skrbijo pretvorniki, ki jih lahko dodajamo ročno, nekatere komponente pa jih imajo avtomatsko (npr. komponenta za prikaz in urejanje datumov af:inputDate) [3].

Komponente za vnos podatkov podpirajo tudi potrjevanje vnosa (angl. validation). Potrjevalna pravila lahko dodajamo preko atributov komponente, preko komponent ADF Faces za potrjevanje ali pa na modelnem nivoju definiramo pravila potrjevanja po meri.

Pravila pretvorbe in potrjevanja se lahko na komponentah prikazujejo preko vgrajenih namigov. Ob neuspešni pretvorbi oz. potrjevanju se poleg komponent izpišejo ustrezna sporočila o napakah. Sporočila so lahko sprožena sproti (ko zapustimo polje) ali po pritisku na gumb za shranjevanje obrazca.

2.6.4 Delno osveževanje strani

Delno osveževanje strani (angl. partial page rendering – PPR) je ena najpomembnejših funkcionalnosti ogrodja ADF. Omogoča osveževanje določenih komponent brez osveževanja celotne strani in s tem izboljša interaktivnost, hitrost in uporabnost aplikacij. Nekatere komponente imajo to lastnost že vgrajeno, drugim pa jo lahko dodamo deklarativno tako, da določimo, katera komponenta bo povzročila osveževanje druge.

2.7 Integrirano razvojno okolje Oracle JDeveloper

Razvojno okolje Oracle JDeveloper je celostno, standardno, produktivno in brezplačno [5].

Oracle JDeveloper je orodje, namenjeno razvoju Java aplikacij, ki ponuja zelo široko paleto funkcij in tehnologij. Ostala razvojna okolja, kot sta npr. Eclipse in NetBeans, so v splošnem sicer bolj popularna med razvijalci, je pa JDeveloper edini, ki pokriva celotno Oracle platformo, vključno s podporo ogrodju ADF. Predvsem zaradi tega je JDeveloper najširše uporabljen za razvoj aplikacij znotraj podjetja Oracle in v podjetjih, ki pogosto uporabljajo Oracle tehnologije. ADF je sicer možno uporabljati tudi v drugih okoljih, ampak je potrebno nameniti več časa postavitvi razvojnega okolja in precej pisanja kode za naloge, za katere lahko v orodju JDeveloper preprosto uporabiš npr. vlečenje (angl. drag-and-drop) [1].

JDeveloper združuje značilnosti razvoja aplikacij (Java, SOA Web 2.0), podatkovne baze in spletnih storitev v eno razvojno orodje. Razvoj teh različnih tehnologij združuje skupna projektna struktura in razvojna izkušnja, kar poenostavlja učni proces ter razvoj zahtevnejših aplikacij, ki združujejo različne tehnologije [4].

Funkcionalnosti

JDeveloper vsebuje veliko funkcionalnosti in tehnologij, zato je primeren za različne tipe razvijalcev in oblikovalcev programske opreme. Za boljši pregled smo nekatere lastnosti in tehnologije našteli v skupinah po tipih uporabnikov, katerim so namenjene [5]:

Za razvijalce poslovne logike:

- EJB 3.0/JPA, TopLink, ADF BC,
- vizualni urejevalnik modela,
- generiranje entitet iz baze,
- deklarativno urejanje,
- ustvarjanje sejne fasade (angl. Session facade),
- avtomatsko ustvarjanje testnega klienta.

Za analitike in oblikovalce:

- UML (modeliranje razredov, sekvenc, primerov uporabe, aktivnosti),
- vizualzacija programske kode (Java razredi, EJB, ADF BC, podatkovna baza, tok strani).

Za XML razvijalce:

- urejanje ogrodja,
- vizualni urejevalnik XML sheme,
- primerjava XML datotek.

Za RIA razvijalce:

- podpora JSF 1.2,
- AJAX osveževanje,
- komponente ADF Faces.

Za JavaScript razvijalce:

- JavaScript razhroščevalnik,
- nov urejevalnik z označevanjem sintakse, ujemanjem oklepajev ipd.

Za SOA razvijalce:

- razvoj BPEL,
- razvoj ESB.

Za razvijalce podatkovne baze:

- brskalnik baze,
- ustvarjanje baznih modelov in objektov, optimizacija SQL,
- urejevalnik in razhroščevalnik PL/SQL.

Poglavje 3

Primer razvoja spletne aplikacije

V tem poglavju sledi predstavitev razvoja enostavne spletne aplikacije. Namen aplikacije je prikaz uporabe funkcionalnosti ogrodja ADF in razvojnega okolja JDeveloper, ki smo ju predstavili v poglavju 2 in 3. V nadaljevanju sledi prikaz celotnega postopka razvoja, od vsebinske zasnove do prikaza delovanja v spletnem brskalniku. Tabela 2 prikazuje vso programsko opremo, ki smo jo uporabljali pri razvoju aplikacije. Za vsako orodje smo navedli naziv in verzijo ter uporabo in naziv podjetja.

Uporaba	**Naziv podjetja**	**Naziv orodja**	**Verzija**
Podatkovna baza	Oracle	Database 10g XE	10.2.0.1.0
Relacijski in podatkovni model, generiranje baznih skript	Oracle	Designer 6i	6.5.95.5.6
Poganjanje baznih skript	Oracle	SQL*Plus	10.1.0.4.2
Polnjenje podatkov, testiranje	Allround Automations	PL/SQL Developer	8.0.3.1510
Razvoj aplikacije	Oracle	JDeveloper 11g	11.1.1.3.0

Tabela 2: Seznam uporabljene programske opreme

3.1 Opredelitev zahtev

Za nazoren prikaz funkcionalnosti vseh nivojev ogrodja mora imeti aplikacija naslednje lastnosti:

- enostaven podatkovni model,
- možnost posodabljanja (angl. update) podatkov v bazi,
- možnost vstavljanja (angl. insert) podatkov v bazo,
- možnost brisanja (angl. delete) podatkov iz baze,
- uporaba različnih tipov vezi, iteratorjev in izvršljivih povezav,
- uporaba ločenih tokov podatkov (omejeni in neomejeni),
- uporaba različnih razporeditvenih komponent ADF Faces,
- uporaba različnih enostavnih in kompleksnih atomskih komponent ADF Faces,
- glavna stran naj bo razdeljena na več funkcionalnih delov z delnim osveževanjem in možnostjo interaktivnega spreminjanja postavitve,
- vsaj ena prikazni obrazec in ena vnosni obrazec s potrjevanjem po meri,
- uporaba tabele z možnostjo urejanja, sortiranja, filtriranja in preurejanja stolpcev,
- uporaba vgrajenega dialoga,
- vsi elementi, meniji in sporočila morajo biti v slovenskem jeziku.

3.2 Vsebinska zasnova aplikacije

Področje, ki se nam je zdelo najbolj primerno in tudi aktualno, je prodaja nepremičnin. Nepremičnina kot predmet prodaje ima kot entiteta naslednje lastnosti:

- več enostavnih podatkov različnih tipov,
- nekaj enostavnih šifrantov,
- zunanje entitete,
- slike nepremičnine,
- lokacijo nepremičnine.

S stališča bogatih spletnih aplikacij so nepremičnine zanimive zaradi večpredstavnostnih podatkov. Poleg interaktivnega prikaza slik komponente ADF Faces omogočajo tudi prikaz lokacije nepremičnine na interaktivnem zemljevidu, prikaz cen nepremičnin na animiranem grafu ipd.

Želeli smo naslednje sklope uporabniškega vmesnika:

- osnovno stran za pregled nepremičnin s prikazom vseh podrobnosti trenutno izbrane nepremičnine, ločenih na več sklopov,
- obrazec za urejanje podatkov izbrane nepremičnine,
- obrazec za kreiranje nove nepremičnine.

Osnovna stran naj vsebuje:

- tabelo nepremičnin s prikazom osnovnih podatkov in vsemi zahtevanimi lastnostmi,
- prikaz vseh podrobnosti izbrane nepremičnine, slik, lokacije in grafa cen v štirih sklopih, ločenih z zavihki: podrobnosti, slike, graf, zemljevid.

3.3 Definiranje podatkovnega modela ter generiranje podatkovne baze

Za potrebe načrtovanja relacijskega, podatkovnega in fizičnega modela smo uporabili orodja Entity Relationship Diagrammer in Design Editor, oba sta del orodja Oracle Designer.

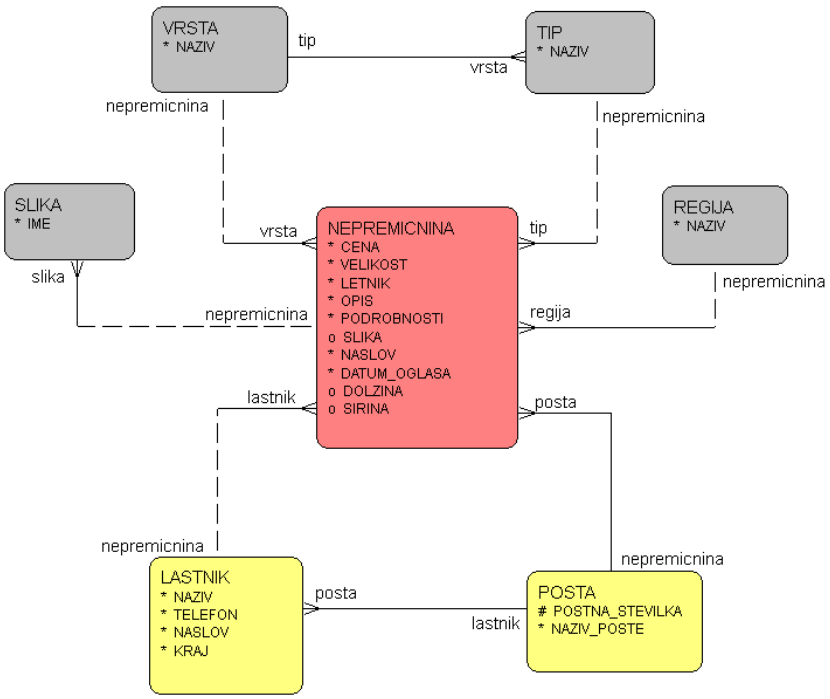
Relacijski model smo načrtovali v orodju Entity Relationship Diagrammer, prikazuje ga slika 6. Z vgrajeno funkcijo Database Design Transformer smo relacijski model pretvorili v podatkovni model. Iz podatkovnega modela smo nato v orodju Design Editor zgenerirali skripte za kreiranje baznih objektov (kreiranje tabel, primarnih ključev, tujih ključev, sekvenc).

Za podatkovno bazo smo uporabili Oracle Database 10g XE, na kateri smo ustvarili novega uprabnika »nepremicnine« z ustreznimi pravicami. Na novem

uporabniku smo pognali prej ustvarjene skripte s pomočjo programa SQL*Plus.

Opis tabel:

- tabela »Nepremicnine« je namenjena shranjevanju vseh podatkov o nepremičnini razen slik, ki jih ima ena nepremičnina lahko več, zato so hranjene v svoji tabeli. Vsebuje več vrst enostavnih podatkov in tuje ključe iz tabel »Lastniki«, »Vrste«, »Tipi«, »Regije« ter »Poste«,
- tabela »Lastniki« je namenjena hranjenju podatkov o lastnikih nepremičnin. Ti podatki so naziv, telefon, naslov, kraj in tuji ključ iz tabele »Poste«,
- tabela »Poste« je šifrant vseh slovenskih pošt. V aplikaciji se uporablja tako za naslov nepremičnine kot za naslov lastnika. Poleg primarnega ključa, ki je poštna številka, vsebuje še naziv pošte,
- tabela »Regije« je šifrant vseh slovenskih regij. Uporablja se za lažje grupiranje in sortiranje nepremičnin,
- tabela »Vrste« je šifrant vrst nepremičnin,
- tabela »Tipi« je šifrant tipov nepremičnin. Je sestavljen šifrant - vsaka vrsta nepremičnine ima svoj nabor tipov nepremičnine,
- tabela »Slike« vsebuje slike nepremičnin. Nepremičnina ima lahko nič ali več slik.



Slika 6: Relacijski model

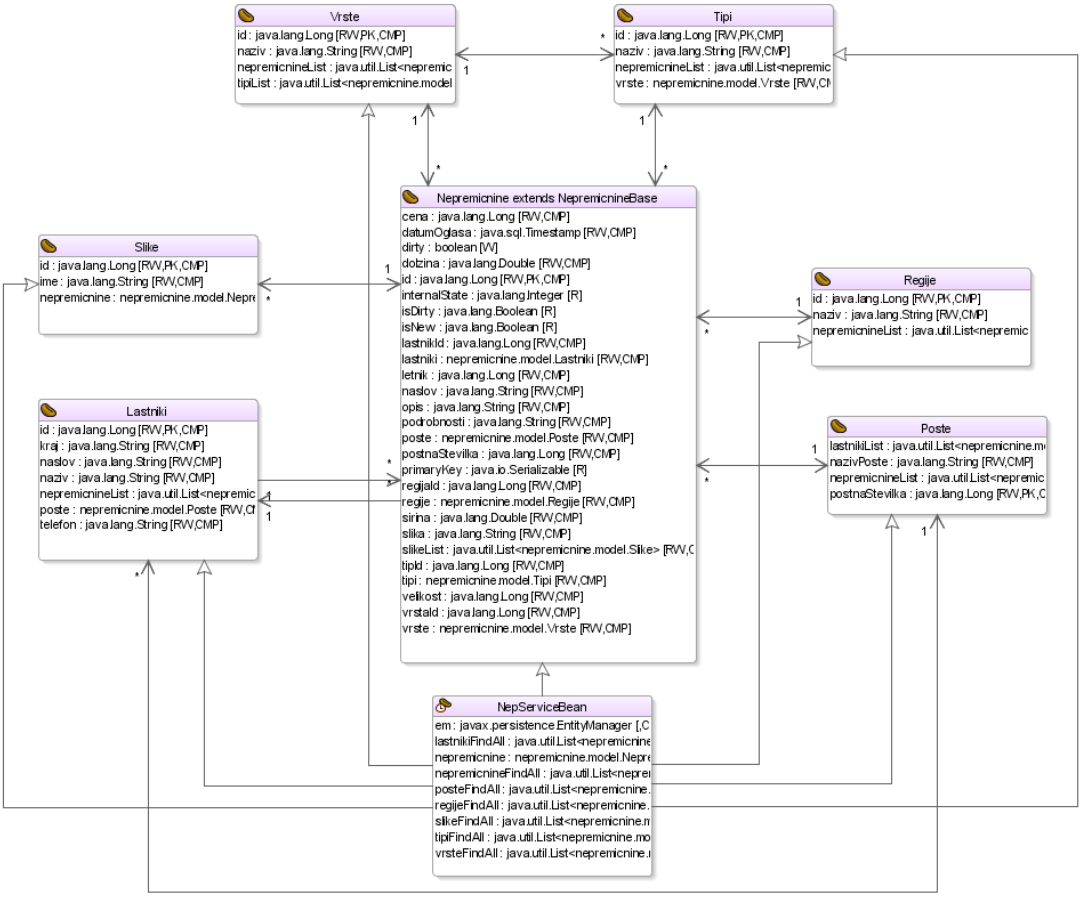
Po uspešno generirani podatkovni bazi smo s pomočjo orodja PL/SQL Developer v tabele napolnili testne podatke za potrebe razvoja. Sledi opis razvoja aplikacije.

3.4 Razvoj aplikacije

V orodju JDeveloper smo ustvarili novo aplikacijo tipa »Fusion web application (ADF)«. V okviru aplikacije se avtomatsko ustvarita dva projekta, Model in ViewControler. Prvi je namenjen poslovno-storitvenemu in modelnemu nivoju, drugi pa nivoju uporabniškega vmesnika ter kontrolnemu nivoju.

3.4.1 Poslovno-storitveni nivo

Definirali smo povezavo na podatkovno bazo ter v projektu Model zgenerirali EJB/JPA entitete preko vgrajene funkcije »entities from tables«. Nato smo ustvarili EJB diagram ter sejno zrno EJB z vsemi potrebnimi funkcijami za komunikacijo z bazo. Diagram z vsemi entitetami in sejnim zrnom je prikazan na sliki 7.



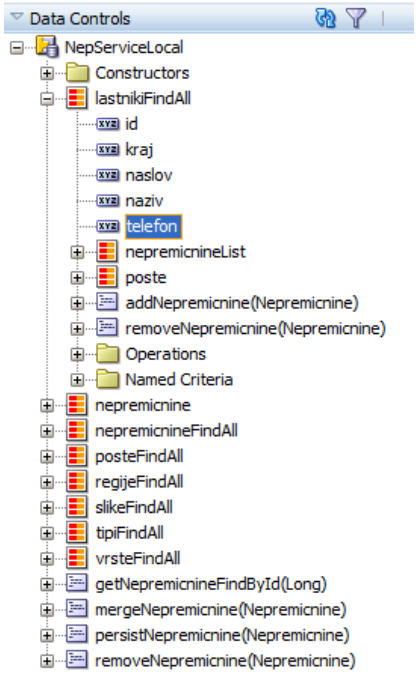
Slika 7: Diagram EJB

Osnovne metode za pridobivanje podatkov iz baze so generirane avtomatsko. Primer take metode je »getNepremicnineFindAll«, ki nam vrne vse nepremičnine. Za potrebe obrazca za urejanje nepremičnin pa smo potrebovali še metodo, ki vrne nepremičnino za določeno vrednost atributa »id«. V sejnem zrnu smo zato ustvarili novo metodo z imenom

»getNepremicnineFindById«. Za potrebe obrazca za kreiranje pa smo ustvarili metodo z imenom »getNepremicnine«, ki vrne novo, prazno nepremičnino.

3.4.2 Modelni nivo

Ko je bilo sejno zrno EJB pripravljeno, je bilo potrebno omogočiti dostop do entitet in metod višje ležečim nivojem. Z desnim miškinim gumbom smo kliknili na sejno zrno (v naši aplikaciji se imenuje »NepServiceBean«) in izbrali ustvarjanje podatkovnih kontrol. Podatkovne kontrole so postale dostopne v pregledovalniku podatkovnih kontrol, od kjer smo jih lahko povlekli na želene strani JSF (slika 8).



Slika 8: Podatkovne kontrole

Urejanje lastnosti atributov, dodajanje potrjevalnikov

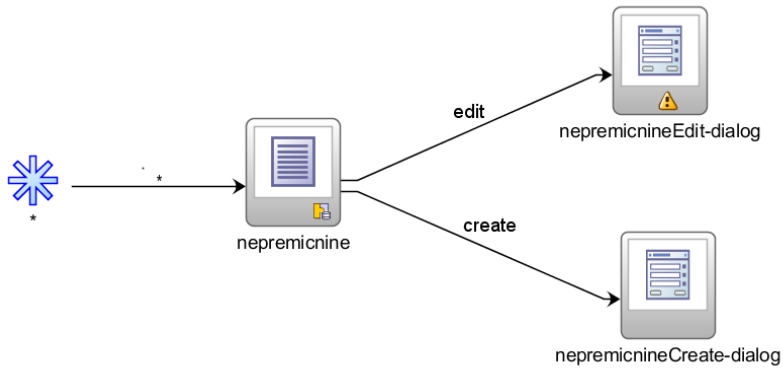
Pri kreiranju podatkovnih kontrol se za vsako entiteto kreira datoteka XML, v kateri se lahko nastavi najrazličnejše lastnosti atributov entitete. Del teh

lastnosti vpliva na prikaz atributov na uporabniškem vmesniku. Za atribute entitete nepremičnine, ki so prikazani na obrazcih, smo nastavili oznake, namige ter širino izpisa.

Na tem nivoju lahko na atribute dodamo tudi potrjevalnike. Le-te smo dodali na atribute: cena (vrednost mora biti pozitivna), letnik (vrednost mora biti večja od 1000), velikost (vrednost mora biti pozitivna), opis in podrobnosti (omejitev dolžine vnosa glede omejitev dolžine polja v bazi).

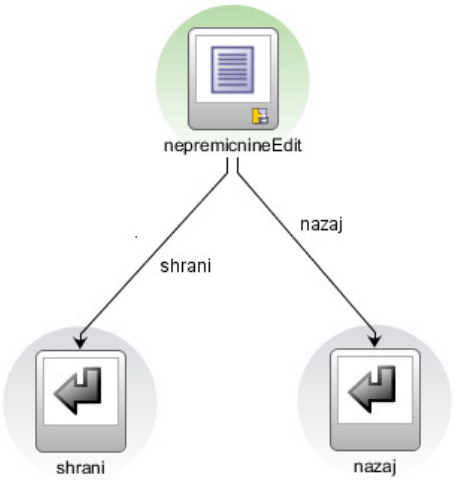
3.4.3 Kontrolni nivo

Na avtomatsko ustvarjen osnovni tok opravil smo dodali vstopno točko aplikacije, aktivnost »pogled« za osnovno stran aplikacije, klic toka opravil za urejanje nepremičnin ter klic toka pravil za ustvarjanje nove nepremičnine. Aktivnosti smo med seboj ustrezno povezali ter za oba klica toka opravil nastavili, da se nov tok pravil odpira v vgrajenem dialogu. Tok podatkov, ki smo ga ustvarili, je prikazan na sliki 9.



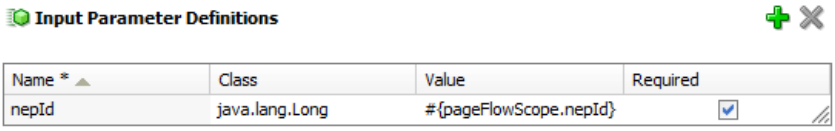
Slika 9: Osnovni, neomejeni tok opravil (adfc-config.xml)

Z dvoklikom na klic toka opravil za urejanje nepremičnin smo ustvarili nov, omejeni tok opravil, ki smo mu dodali privzeto aktivnost (aktivnost »pogled« za povezavo na stran za urejanje nepremičnin) ter dve aktivnosti za vračanje na predhodni tok opravil (slika 10). Enak postopek smo ponovili tudi na klicu toka opravil za kreiranje nove nepremičnine.



Slika 10: Omejeni tok opravil za urejanje nepremičnin

Ker je klic toka opravil za urejanje nepremičnin ločen od osnovnega toka opravil, ne vemo, katera je trenutno izbrana nepremičnina. Ta problem smo rešili tako, da smo toku opravil za urejanje definirali vhodni atribut, kot to prikazuje slika 11. Vrednost izraza #{pageFlowScope.nepId}, ki predstavlja id trenutno izbrane površine, služi kot vhodni parameter pri klicu metode »getNepremicnineFindById«.



Slika 11: Definicija vhodnega parametra

Vrednost izraza #{pageFlowScope.nepId} se nastavi ob kliku gumba za urejanje nepremičnine in pošlje iz osnovnega toka opravil. Več o tem pri kreiranju gumba v razdelku 4.4.4.

3.4.4 Nivo uporabniškega vmesnika

Uporabniški vmesnik za pregled nepremičnin

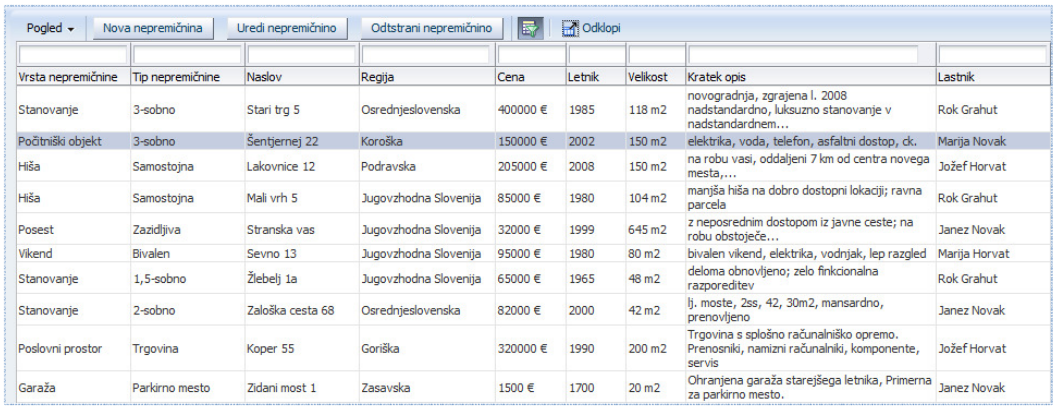
Z dvoklikom na aktivnost nepremičnine v osnovnem toku opravil smo ustvarili novo stran (nepremičnine.jspx) za pregled nepremičnin. Stran smo najprej razdelili na dva dela z uporabo komponente PanelSplitter, ki tudi omogoča, da se del strani skrije ali spet prikaže.

Na levi strani komponente PanelSplitter smo s pomočjo vlečenja ustrezne podatkovne kontrole ustvarili komponento PanelCollection, ki vsebuje tabelo z nekaj dodatnimi možnostmi (izklapljanje/vklapljanje prikaza posamičnih stolpcev, vklop/izklop filtra stolpcev, možnost odklopa tabele). Tabela omogoča sortiranje, filtriranje in preureditev vrstnega reda stolpcev.

V orodno vrstico komponente PanelCollection smo dodali še gumbe za urejanje, dodajanje ter brisanje nepremičnin. Gumbu za brisanje smo za akcijo nastavili klic metode removeNepremicnine, ki izbriše trenutno izbrano nepremičnino. Gumbu za urejanje smo nastavili akcijo »edit«, gumbu za dodajanje pa akcijo »create«. Obema gumboma smo nastavili še lastnost, da uporabljata okno ter širino in višino okna. Na ta način se bosta obrazca za urejanje in dodajanje nepremičnin odpirala v vgrajenem dialogu želenih dimenzij.

Na gumb za urejanje smo dodali še komponento SetPropertyListener, s pomočjo katere smo postavili vrednost izraza #{pageFlowScope.nepId} na identifikacijsko številko trenutno izbrane nepremičnine. Pri kliku gumba se vrednost izraza pošlje kot parameter v tok opravil za urejanje nepremičnin.

Slika 12 prikazuje končni izgled ustvarjene komponente PanelCollection s tabelo nepremičnin in orodno vrstico z gumbi za urejanje podatkov.



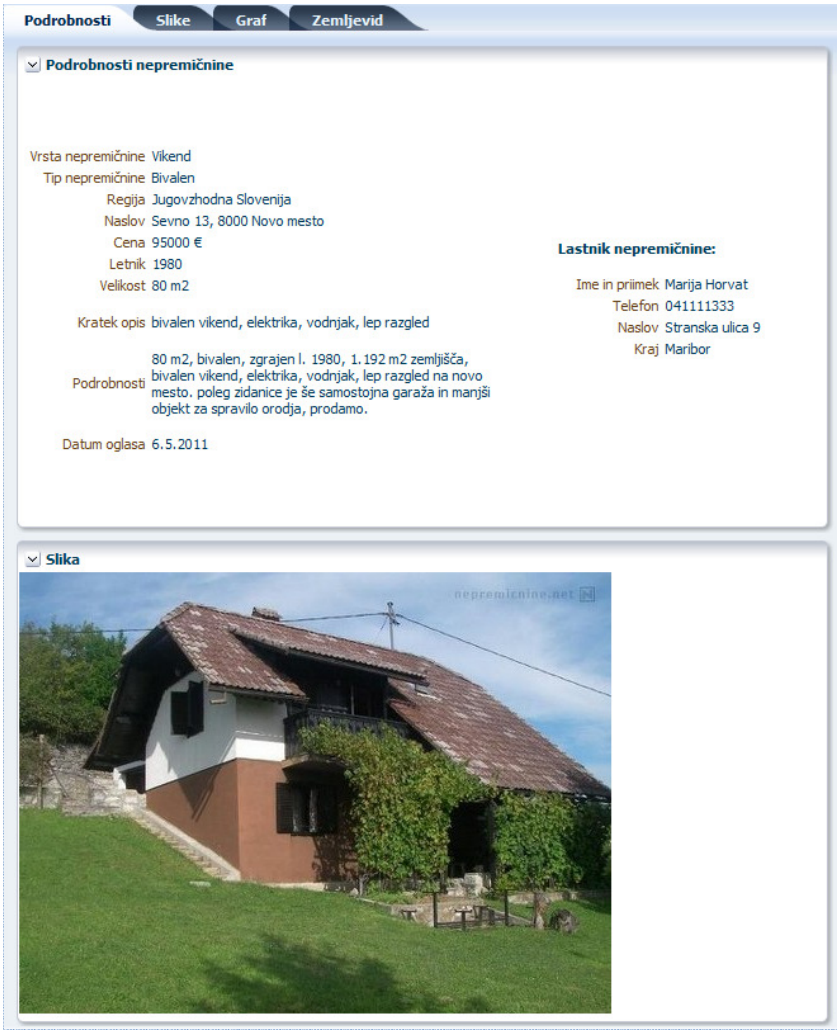
Vrsta nepremičnine	Tip nepremičnine	Naslov	Regija	Cena	Letnik	Velikost	Kratek opis	Lastnik
Stanovanje	3-sobno	Stari trg 5	Osrednjeslovenska	400000 €	1985	118 m2	novogradnja, zgrajena l. 2008 nadstandardno, luksuzno stanovanje v nadstandardnem...	Rok Grahut
Počitniški objekt	3-sobno	Šentjernej 22	Koroška	150000 €	2002	150 m2	elektrika, voda, telefon, asfaltni dostop, ck.	Marija Novak
Hiša	Samostojna	Lakovnice 12	Podravska	205000 €	2008	150 m2	na robu vasi, oddaljeni 7 km od centra novega mesta,...	Jožef Horvat
Hiša	Samostojna	Mali vrh 5	Jugovzhodna Slovenija	85000 €	1980	104 m2	manjša hiša na dobro dostopni lokaciji; ravna parcela	Rok Grahut
Posest	Zazidljiva	Stranska vas	Jugovzhodna Slovenija	32000 €	1999	645 m2	z neposrednim dostopom iz javne ceste; na robu obstoječe...	Janez Novak
Vikend	Bivalen	Sevno 13	Jugovzhodna Slovenija	95000 €	1980	80 m2	bivalen vikend, elektrika, vodnjak, lep razgled	Marija Horvat
Stanovanje	1,5-sobno	Žlebelj 1a	Jugovzhodna Slovenija	65000 €	1965	48 m2	deloma obnovljeno; zelo finkcionalna razporeditev	Rok Grahut
Stanovanje	2-sobno	Zaloška cesta 68	Osrednjeslovenska	82000 €	2000	42 m2	lj. moste, 2ss, 42, 30m2, mansardno, prenovljeno	Janez Novak
Poslovni prostor	Trgovina	Koper 55	Goriška	320000 €	1990	200 m2	Trgovina s splošno računalniško opremo. Prenosniki, namizni računalniki, komponente, servis	Jožef Horvat
Garaža	Parkirno mesto	Zidani most 1	Zasavska	1500 €	1700	20 m2	Ohranjena garaža starejšega letnika, Primerna za parkirno mesto.	Janez Novak

Slika 12: Tabela nepremičnin

Na desni strani komponente PanelSplitter smo ustvarili komponento PanelTabbed, ki omogoča uporabo zavihkov. Ustvarili smo štiri zavihke: Podrobnosti, Slike, Graf, Zemljevid. Vsi zavihki morajo vsebovati podrobnosti trenutno izbrane nepremičnine, zato smo na komponenti PanelTabbed deklarativno določili, da se osvežuje glede na tabelo nepremičnin in s tem vključili funkcionalnost delnega osveževanja PPR.

Zavihek »Podrobnosti«

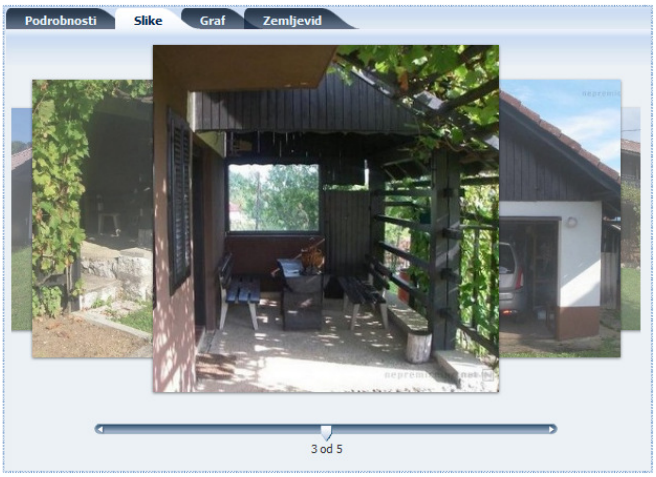
V zavihku smo ustvarili komponento PanelDashboard, ki omogoča razporeditev vsebujočih elementov v stolpce in vrstice. Nastavili smo na en stolpec in dve vrstici ter ustvarili dve komponenti PanelBox. Vsaka komponenta PanelBox ima poleg naslova puščico, s katero lahko skrijemo vsebino. V prvi smo naredili izpis vseh podrobnosti nepremičnine, ki se jih da predstaviti s tekstom, v drugi pa prikaz glavne, predstavitvene slike nepremičnine (slika 13).



Slika 13: Zavihek »Podrobnosti«

Zavihek »Slike«

V zavihku smo ustvarili komponento Carousel, ki omogoča prikaz elementov v »vrtiljaku«, skozi katerega se lahko premikamo s klikanjem na slike, uporabo drsnika ali s tipkovnico. Za seznam elementov izberemo listo slik trenutno izbrane nepremičnine. Na sliki 14 je prikazan vrtiljak petih slik nepremičnine.



Slika 14: Zavihek »Slike«

Zavihek »Graf«

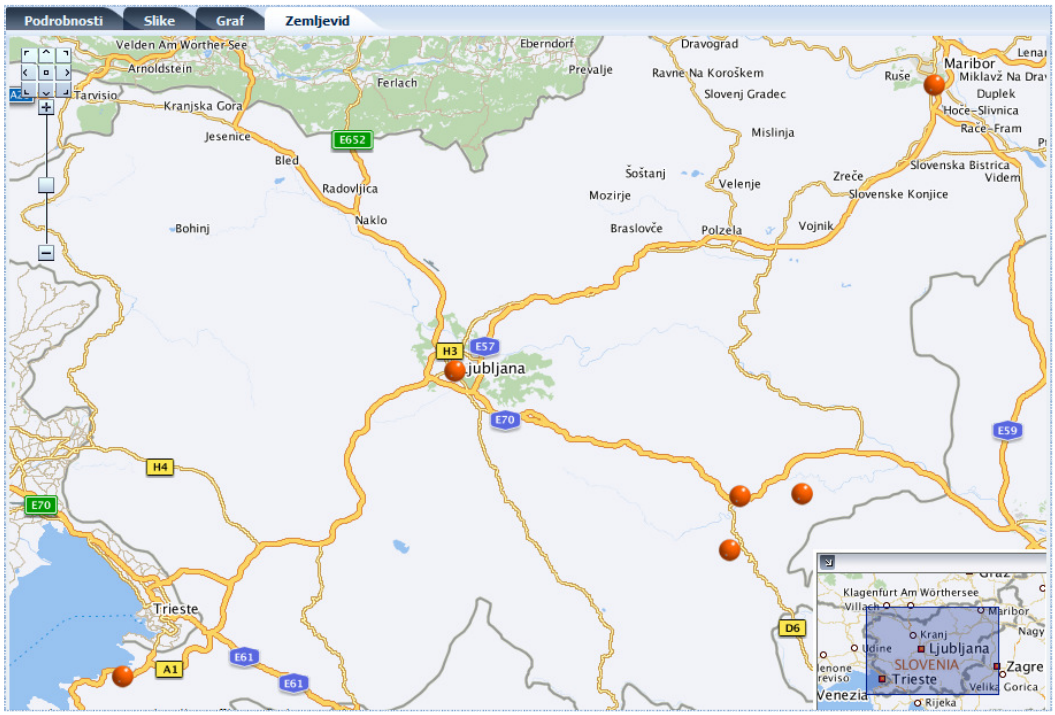
Ustvarili smo komponento BarGraph, ki izriše stolpčni graf. Za x-os smo izbrali naslov nepremičnine, za y-os pa ceno nepremičnine. Vključili smo možnost tridimenzionalnega izrisa ter animacijo grafa. Rezultat je interaktiven, tridimenzionalen, animiran graf s prikazom cen vseh nepremičnin (slika 15).

Slika 15: Zavihek »Graf«

Zavihek »Zemljevid«

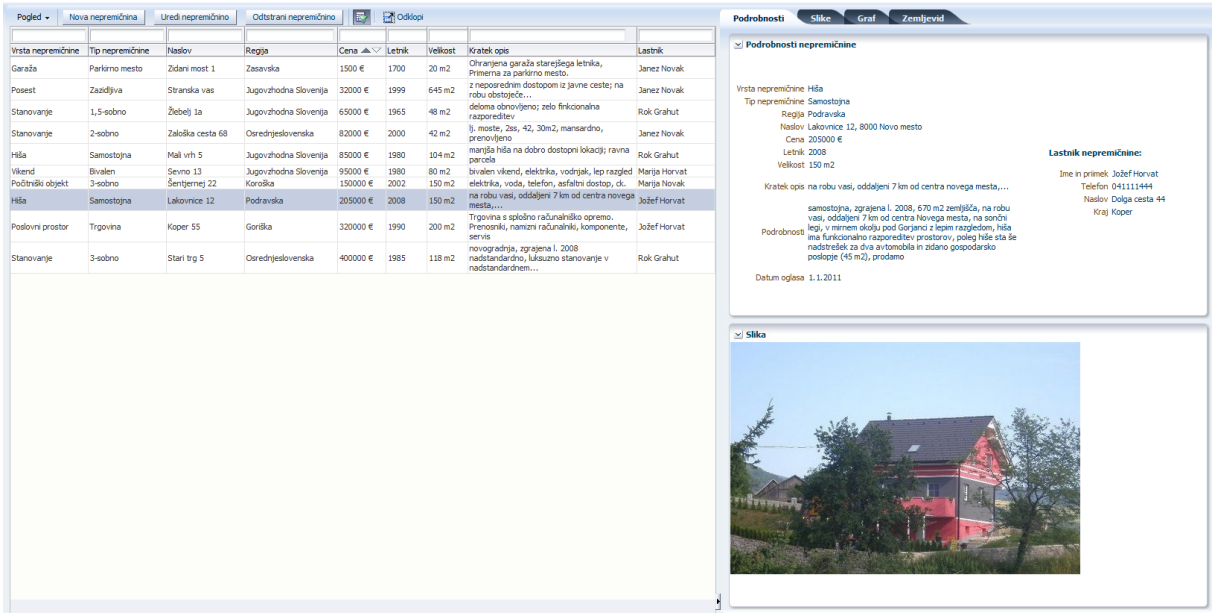
Ustvarili smo komponento Map, ki omogoča izris interaktivnega zemljevida. Komponenti smo določili, kateri zemljevid naj uporablja, začetne koordinate

ter stopnjo začetnega približanja. Določili smo tudi, naj bodo s točkami označene vse nepremičnine, ki imajo podatek o lokaciji. Pregled lokacij testnih nepremičnin, ki smo jih vključili v aplikacijo, je prikazan na sliki 16.



Slika 16: Zavihek »Zemljevid«

Končni izgled osnovne strani je prikazan na sliki 19. Vsi podatki na osnovni strani izvirajo iz podatkovne kontrole »nepremicnineFindAll«.

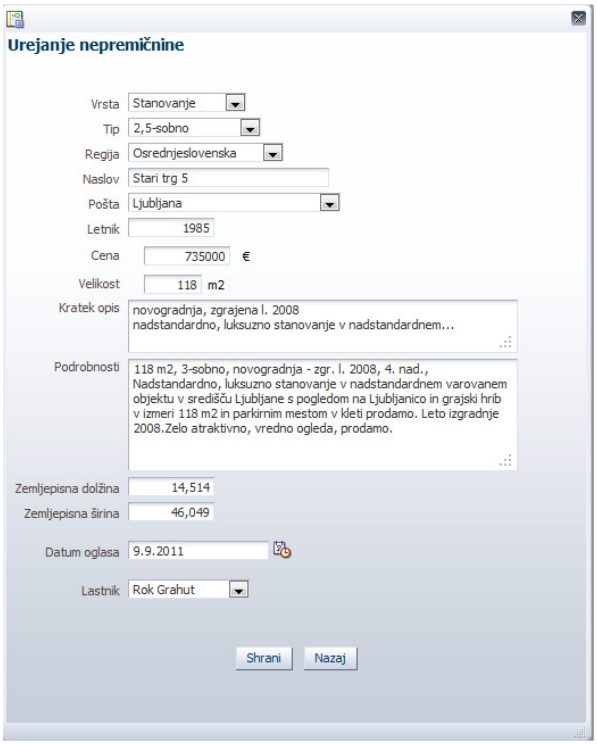


Slika 17: Osnovna stran z aktivnim zavihkom »Podrobnosti«

Uporabniški vmesnik za urejanje nepremičnin

V okviru toka opravil za urejanje nepremičnin smo ustvarili novo stran. Ustvarili smo obrazec za urejanje podatkov in smiselno uredili polja. Vnosna polja, ki temeljijo na šifrantih, preuredimo v spustne menije z naborom možnih izbir. Polji za opis in podrobnosti ustrezno razširimo. Na dno obrazca dodamo dva gumba. Gumb »Shrani« pokliče metodo »mergeNepremičnine«, ki shrani vnesene spremembe in nas vrne na osnovno stran. Gumb »Nazaj« nas vrne na osnovno stran brez shranjevanja sprememb. Končna oblika obrazca je prikazana na sliki 19.

Podatki vseh polj izvirajo iz podatkovne kontrole »getNepremicnine-FindById«, ki vrne nepremičnino glede na id, ki ga prejme tok opravil kot vhodni parameter.

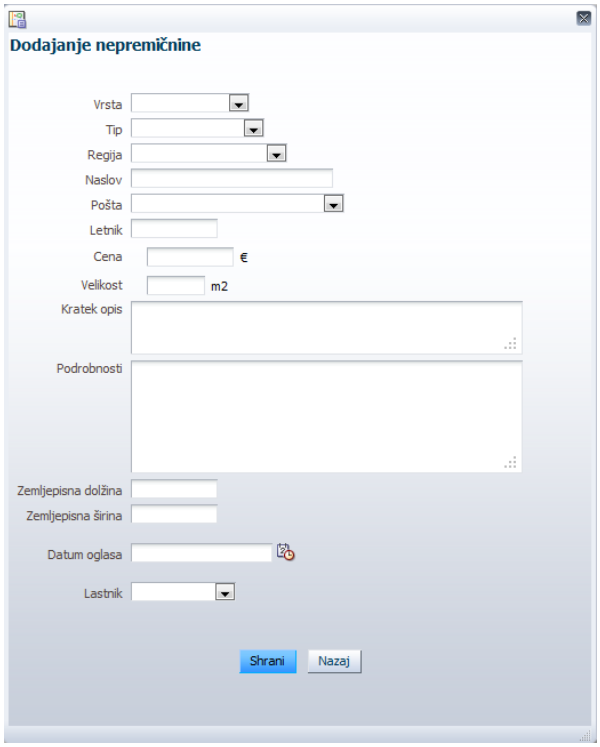


Slika 18: Urejanje nepremičnin

Uporabniški vmesnik za dodajanje nepremičnin

V okviru toka opravil za dodajanje nepremičnin smo ustvarili novo stran. Ustvarili smo obrazec za urejanje podatkov in smiselno uredili polja. Vnosna polja, ki temeljijo na šifrantih, smo preuredili v spustne menije z naborom možnih izbir. Polji za opis in podrobnosti ustrezno razširimo. Na dno obrazca dodamo dva gumba. Gumb »Shrani« pokliče metodo »persistNepremičnine«, ki ustvari novo nepremičnino z vnesenimi podatki in nas vrne na osnovno stran. Gumb »Nazaj« nas vrne na osnovno stran brez kreiranja nove nepremičnine. Končna oblika obrazca je prikazana na sliki 20.

Podatki vseh polj izvirajo iz podatkovne kontrole »getNepremicnine«, ki vrne novo, prazno nepremičnino.



Slika 19: Dodajanje nove nepremičnine

3.5 Lokalizacija

Pri aplikacijah, namenjenih širšemu krogu uporabnikov, je pomembno tudi, da so v celoti prevedene, zato smo v zahtevah dodali tudi ta pogoj. Aplikacije, zgrajene v ogrodju ADF, so privzeto angleške, ampak se jih da z uporabo t. i. svežnjev brez večjih težav prevesti. Prevesti je možno oznake, namige, menije ter sporočila posameznih komponent in tudi privzeta sporočila ter namige vgrajenih potrjevalnikov in pretvornikov.

Za potrebe naše aplikacije smo v svežnje vključili samo prevode tistih komponent, potrjevalnikov in pretvornikov, ki jih uporablja aplikacija.

Vključitev svežnja preoblek (angl. skin bundle):

Ustvarili smo nov sveženj SkinBundle.java, vanj skopirali privzete vrednosti uporabljenih komponent ter jih prevedli v slovenščino. Sveženj je bilo potrebno vključiti še v konfiguracijske datoteke preoblek, trinidad-config.xml in trinidad-skins.xml.

Vključitev svežnja sporočil (angl. message bundle):

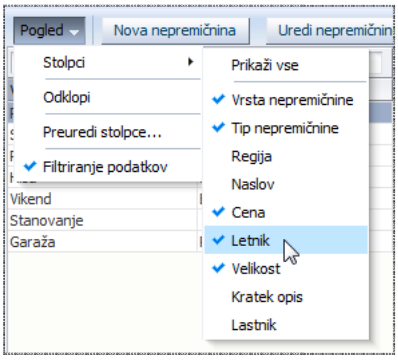
Ustvarili smo nov sveženj MessageBundle_sl.java, vanj skopirali privzete vrednosti sporočil in namigov uporabljenih potrjevalnikov in pretvornikov ter jih prevedli v slovenščino. Ta sveženj smo definirali za slovenski jezik, zato je bilo potrebno vključiti uporabo slovenskega jezika v nastavitvah aplikacije, v datoteki faces-config.xml.

3.6 Testiranje aplikacije

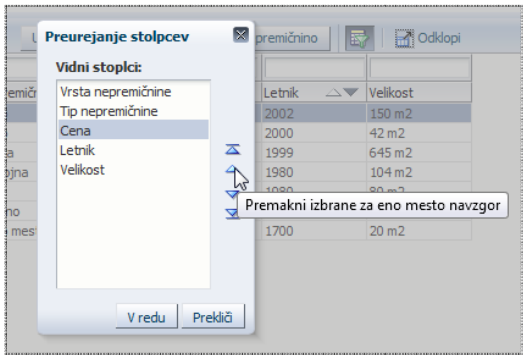
Za potrebe testiranja in razhroščevanja aplikacije ima orodje JDeveloper na voljo integriran strežnik WebLogic. Zaženemo lahko posamezno stran JSF ali posamezen projekt, ki mu določimo privzet tok pravil. Za testiranje aplikacije nepremičnin smo za privzet tok nastavili osnovni neomejen tok opravil adfc-config.xml.

Pognali smo aplikacijo ter najprej preizkusili delovanje aplikacije brez urejanja podatkov: prikaz podrobnosti glede na izbrano vrstico v tabeli, preklapljanje med zavihki, pregled slik, prikaz zemljevida.

Preizkusili smo tudi funkcionalnosti komponente PanelCollection, ki omogočajo prilagoditev prikaza vsebujoče tabele. Preko vgrajenega menija smo odkljukali stolpce, ki smo jih želeli skriti (slika 20) in preko dialoga za preurejanje stolpcev nastavili želeni vrstni red stolpcev (slika 21). Informacija o številu skritih stolpcev se izpiše pod tabelo.

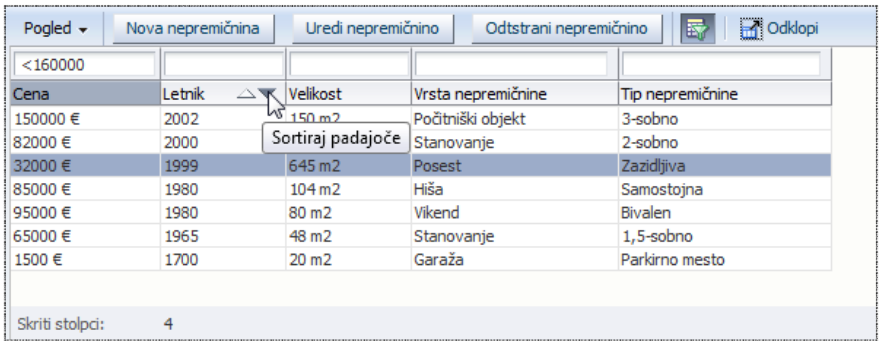


Slika 20: Izbira stolpcev



Slika 21: Preurejanje stolpcev

Tabelo smo še filtrirali glede na ceno (določili smo, da se morajo prikazati samo nepremičnine s ceno manjšo od 160.000 evrov) in jo uredili padajoče po letniku nepremičnine. Prilagojen prikaz tabele prikazuje slika 22, prikaz tabele pred prilagajanjem pa prikazuje slika 12 v razdelku 4.4.4.

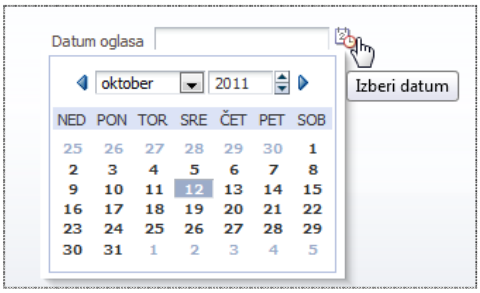


Slika 22: Prilagojen prikaz tabele

Nato smo preizkusili še dialoga za urejanje in dodajanje nepremičnin. Pri vnosu podatkov v obrazec za urejanje nepremičnin smo testirali prikaz namigov ter napak pri potrjevanju in pretvorbi. Sledi nekaj primerov delovanja vnosnih komponent s kratkimi opisi in zaslonskimi slikami. Slika 23 prikazuje izpis vgrajenega namiga pri vnosu v polje »Datum oglasa«. Namesto vgrajenega namiga lahko vsakemu atributu na poslovno-storitvenem nivoju določimo poljuben namig.

Slika 23: Vgrajen namig

Pri vnosu v datumska polja nam je v pomoč tudi vgrajen koledar za izbiro želenega datuma (slika 24).

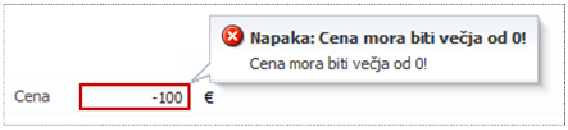


Slika 24: Koledar za izbiro datuma

Ob nepravilnem formatu vnosa v datumsko polje se ob zapustitvi polja sproži napaka vgrajenega pretvornika vnosa. Izpis napake je prikazan na sliki 25.

Slika 25: Pretvorba vnosa

Slika 26 prikazuje primer izpisa napake pri potrjevanju vnosa v polje »Cena«. Potrjevalnik, ki je sprožil to napako, smo na atributu »Cena« definirali na nivoju poslovno-storitvenega nivoja.



Slika 26: Potrjevanje vnosa

Poglavje 4

Primerjava tehnologij

Primerjava ogrodja ADF z ogrodjem .NET

Sledi prikaz nekaterih prednosti ogrodja ADF in potencialno povečanje produktivnosti pri prehodu iz ogrodja Microsoft .NET na ogrodje Oracle ADF.

Funkcionalnosti, ki jih ogrodje ADF omogoča »iz škatle«, ogrodje .NET brez razširitev in dodatkov pa ne [14]:

- implementacija modelnega nivoja oz. poslovne logike in poslovno-storitvenega nivoja z možnostjo enostavne ponovne uporabe za razvoj spletnih, namiznih ali mobilnih aplikacij,
- implementacija zbirke poslovnih pravil in pravil potrjevanja v enem nivoju in konsistentna uporaba le-teh na različnih nivojih aplikacije brez dodatnega pisanja programske kode,
- implementacija AJAX podatkovnih kontrol relacije »glavna entiteta/podrobnosti« z ohranjanjem konsistence podatkov.

Narejena je bila tudi primerjava implementacije tretje točke v obeh ogrodjih na primeru enostavne strani z izbiro strank in prikazom njihovih podatkov. Povzetek primerjave [14]:

- za razvoj v ogrodju ADF je bilo potrebno napisati 6 vrstic kode, v ogrodju .NET pa več kot 50,
- ko se je razvoj v ogrodju .NET dobro začel, je bil v ogrodju ADF že skoraj končan (po zaslugi vgrajenega potrjevanja in obstojnosti podatkov),
- za razvoj v ogrodju ADF ni bilo potrebno znanje nobene druge tehnologije, za razvoj v ogrodju .NET pa so bila potrebna znanja tehnologij JavaScript, DOM, Jquery in Microsoft AJAX library.

Razvoj v ogrodju ADF tako omogoča manj pisanja kode, razvoj je hitrejši in zahteva manj poznavanja drugih tehnologij, torej omogoča večjo produktivnost. Te lastnosti naredijo aplikacije ADF posledično tudi cenejše za vzdrževanje [14].

Produktivnost razvoja v tehnologiji Oracle ADF so preverjali tudi na delavnici ADF v Cipru, oktobra 2010. Udeleženci so bili razvijalci .NET aplikacij. Vodja delavnice je specificiral grobo specifikacijo strani za poslovno aplikacijo [14]:

- stran mora implementirati vse operacije CRUD (Create, read, update and delete),
- polja za iskanje podatkov,
- relacije »glavna entiteta/podrobnosti«, na primer zaposlenca in zgodovino njegovih zaposlitev,
- možnost izvajanja poizvedb po meri,
- potrjevanje na strani strežnika in klienta, vključno s potrjevanjem elektronskega naslova z uporabo regularnih izrazov,
- model in poslovna logika morata imeti možnost ponovne uporabe za različne tipe aplikacij,
- aplikacija mora biti spletna in omogočati AJAX.

Predpogoj je bila že postavljena podatkovna baza z ustreznimi podatki. Od programerjev so zahtevali približno oceno časa za razvoj aplikacije glede na zgornje specifikacije. Ocene so se gibale med dvema urama in enim dnevom. Vodja delavnice je nato prikazal razvoj strani v ogrodju ADF v samo osmih minutah. Produktivnost je tako glede na nekatere ocene lahko povečana tudi do petnajstkrat [14].

Poglavje 5

Sklepne ugotovitve

Glavni namen diplomske naloge je bila izbira in opis najbolj primernega ogrodja oziroma tehnologij za učinkovit razvoj poslovnih spletnih aplikacij z bogatim uporabniškim vmesnikom. Pri izbiri smo upoštevali lastnosti, kot so odprtokodnost, produktivnost, enostavnost uporabe, zanesljivost. Hiter razvoj uveljavljenih in pojavljanje novih tehnologij za razvoj spletnih aplikacij za razvijalce predstavlja precejšnje breme. Zato so v precejšnjo pomoč ogrodja, ki z različnimi metodami zmanjšujejo potrebo po poznavanju velikega števila tehnologij. Razvijalec se tako lažje osredotoči na poslovna pravila, uporabniški vmesnik ipd. Ravno tako pomembna lastnost je možnost razvoja aplikacije od zasnove do postavitve. Večino opisanih lastnosti pa ne omogoča ogrodje samo po sebi, ampak v kombinaciji s primernim razvojnim okoljem, zato je bila tudi izbira le tega pomemben dejavnik pri končnem rezultatu.

Kombinacija, ki smo jo izbrali (ogrodje ADF in razvojno okolje JDeveloper), se je v praktičnem delu izkazala kot dobra izbira glede na iskane lastnosti okolja in aplikacije. Razvoj je bil precej hiter, enostaven ter intuitiven, kot smo ga opisali v razdelku 4.4, izpuščenih je bilo samo nekaj manjših podrobnosti. V pomoč nam je bila tudi velika izbira literature, ki je dostopna na strani podjetja Oracle.

Rezultat naloge je bogata spletna aplikacija, ki omogoča pregled in urejanje podatkov nepremičnin preko naprednega interaktivnega uporabniškega vmesnika z delnim osveževanjem, animacijami, pregledovalnikom slik, zemljevidom ipd.

Aplikacija bi lahko bila uporabna za nepremičninske agencije ali za spletne oglasnike, ob predpostavki, da bi jo bilo potrebno ustrezno nadgraditi oziroma prilagoditi. Nadgradnja bi lahko vključevala vstopno stran, dodajanje uporabniških vlog in prilagajanje pravic, ločitev aplikacije na administratorski in uporabniški del ipd. V primeru resne uporabe bi bilo aplikacijo potrebno tudi postaviti v produkcijsko okolje, kar pa seveda zahteva nakup določenih licenc kljub temu, da je razvoj brezplačen.

Ogrodje ADF je torej dobra rešitev za razvoj bogatih poslovnih spletnih aplikacij. Potrebno pa je upoštevati, da se z velikostjo aplikacije običajno poveča tudi kompleksnost, kar pa zahteva nekaj več izkušenj in boljše poznavanje ogrodja.

Slike

Tabele

Literatura

[1] D. Mills, P. Koletzke, A. Roy-Faderman, Oracle JDeveloper 11g Handbook: A Guide to Oracle Fusion Web Development, New York: McGraw-Hill, 2010

[2] (2009) Oracle Application DevelopmentFramework Overview. Dostopno na: http://www.oracle.com/technetwork/developer-tools/adf/adf-11-overview-1-129504.pdf

[3] (2009) Oracle Fusion Middleware: Web User Interface Developer's Guide for Oracle Application Development Framework. Dostopno na: http://download.oracle.com/docs/cd/E17904_01/web.1111/b31973.pdf

[4] (2009) Oracle JDeveloper 11g Data Sheet. Dostopno na: http://www.oracle.com/technetwork/developer-tools/jdev/jdeveloper11g-datasheet-1-133040.pdf

[5] (2011) IDE in Action: Oracle JDeveloper 11. Dostopno na: http://www.docstoc.com/docs/71212585/Creating-Visual-Web-Applications-Jsf-Ajax-and-Data-Binding

[6] (2011) Oracle Application Development Framework Overview. Dostopno na: http://www.oracle.com/technetwork/developer-tools/adf/adf-11-overview-1-129504.pdf

[7] (2009) Oracle Fusion Middleware: Fusion Developer's Guide For Oracle Application Development Framework. Dostopno na: http://download.oracle.com/docs/cd/E12839_01/web.1111/b31974.pdf

[8] F. Nimphius, L. Munsinger, Oracle Fusion Developer Guide, New York: McGraw-Hill, 2010

[9] E. Jendrock, I. Evans, D. Gollapudi, K. Haase, C. Srivathsa, The Java EE 6 Tutorial: Basic Concepts, Fourth Edition, Boston: Addison-Wesley, 2010

[10] M. Sikora, EJB 3 Developer Guide: A Practical Guide for developers and architects to the Enterprise Java Beans Standard, Birmingham-Mumbai: Packt Publishing, 2008

[11] (2007) JSF, JPA, and EJB, Dostopno na: http://www.capcourse.com/Library/JSF+EJB/JSF+EJB_50.pdf

[12] (2007) Oracle ADF 11g Primer. Dostopno na: http://www.oracle.com/technetwork/developer-tools/adf/learnmore/oracle-adf-11g-primer-154277.pdf

[13] (2011) Ajax (programming). Dostopno na: http://en.wikipedia.org/wiki/Ajax_%28programming%29

[14] (2010) ADF vs .Net. Dostopno na: http://jernejkase.blogspot.com/search/label/ADF%20vs%20.Net