UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Primož Cigoj

Aplikacija za podporo delovanja svetovalcev

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Rok Rupnik

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: 00023/2010
Datum: 01.10.2010

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

Kandidat: **PRIMOŽ CIGOJ**

Naslov: **APLIKACIJA ZA PODPORO DELOVANJA SVETOVALCEV**
APPLICATION SYSTEM TO SUPPORT WORK OF CONSULTANTS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte apletno aplikacijo, ki v svetovalnem podjetju podipra organiziranje delovanja svetovalcev in delovanje svetovalcev samih. Uporabite programska jezika PHP in Pyton ter podatkovno bazo MySQL.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a PRIMOŽ CIGOJ,

z vpisno številko 63040199,

sem avtor/-ica diplomskega dela z naslovom:

APLIKACIJA ZA PODPORO DELOVANJA SVETOVALCEV

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc.dr. Rok Rupnik

  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 11.3.2011 Podpis avtorja/-ice: 

ZAHVALA

Rad bi se iskreno zahvalil mentorju doc. dr. Roku Rupniku na Fakulteti za računalništvo in informatiko v Ljubljani. Njegov širok spekter znanja, spodbujanja in osebna svetovanja so zagotovili dobro podlago za moje diplomsko delo.

Posebna zahvala za potrpežljivost in vztrajnost gre mojim staršem, ki so mi omogočili študij in me skozi vsa leta študija podpirali. Zahvalil bi se rad tudi svoji punci za spodbudo in pomoč pri urejanju diplomske naloge.

Nazadnje se zahvaljujem sodelavcem in še vsem ostalim, ki so kakorkoli prispevali k nastanku te diplomske naloge.

Kazalo

Seznam uporabljenih kratic

AJAX – Asynchronous JavaScript and XML (tehnologija za ustvarjanje interaktivnih spletnih aplikacij)

B2B – Business to Business (interakcija podjetje - podjetje)

B2C – Business to Consumer (interakcija podjetje - potrošnik)

B2E – Business to Employee (interakcija podjetje - zaposleni)

BSD – Berkeley Software Distribution (Unix operacijski sistem)

CSS – Cascading Style Sheets (slogovna predloga, ki določa izgled spletnih strani napisanih v označevalnem jeziku)

CGI – Common Gateway Interface (standard za pisanje kode)

DOM – Document Object Model (drevesni programski vmesnik)

GUI – Graphical User Interface (Grafični uporabniški vmesnik)

HTML – HyperText Markup Language (označevalni jezik za izdelavo spletnih strani)

IDE – Integrated Development Environment (integrirano programsko okolje)

IIS – Internet Information Services (spletni strežnik)

JVM – Java Virtual Machine (tolmač za javansko vmesno kodo)

MS – Microsoft

OS – Operacijski Sistem

OUP – Objektno Usmerjeno Programiranje

PHP – Hypertext Preprocessor (odprtokodni skriptni programski jezik za izdelavo spletnih strani)

RIP – Računalniška Izmenjava Podatkov

RUP – Rational Unified Process (iterativen proces razvoja programske opreme)

SGML – Standard Generalized Markup Language (sistem za organizacijo in označevanje elementov dokumenta)

SQL – Structured Query Language (strukturirani povpraševalni jezik za delo s podatkovnimi bazami)

SVG – Scalable Vector Graphics (standard, ki je bil razvit pod okriljem W3C)

UML – Unified Modeling Language (standardenjezik, namenjen modeliranju računalniških sistemov)

UTF – Unicode Transformation Format (način zapisa kodnih točk v standardu unicode)

W3C – World Wide Web Consortium (mednarodni konzorcij podjetij, ki sodelujejo z internetom in spletom)

XHTML – Extensible HyperText Markup Language (html zapisan po pravilih xml)

XML – Extensible Markup Language (razširljiv jezik za definiranje strukture podatkov in dokumentov)

XUL – XML User Interface Language (jezik za za opisovanje vsebine oken, ki temelji na XML-ju)

1. POVZETEK

Diplomsko delo predstavlja novo rešitev za eno večjih strokovnih organizacij v Republiki Sloveniji, katere temeljne naloge so organiziranje in izvajanje svetovalnega dela na področju računovodstva in poslovnih financ. V diplomskem delu je podrobneje predstavljen primer izdelave poslovne spletne aplikacije za pomoč naročnikom, ki izboljšuje in olajša delovanje zaposlenih. V sodelovanju s podjetjem RSteam, d. o. o., smo s pomočjo orodij razvili novo aplikacijo.

Poslovne aplikacije so pomembne v današnjem svetu, tako poslovnem kot privatnem, saj si z njimi lajšamo delo oz. so nam v pomoč v vsakdanjem življenju. Aplikacije se med seboj razlikujejo po načinu uporabe, namembnosti in predvsem v sami zasnovi. Zgrajene so v različnih programskih jezikih, razlikujejo pa se tudi po vrsti okolja, za katera so zgrajena.

V nalogi predstavimo osnovne pojme elektronskega poslovanja, poslovnih aplikacij in z njimi povezanih konceptov. Seznanimo se z orodji in tehnologijami, ki sem jih uporabili pri razvoju aplikacije. Za razvoj aplikacije sem uporabili veliko osnovnih tehnologij, s katerimi sem se seznanil že v preteklosti. Poslovna aplikacija temelji na modernem programskem jeziku PHP in Python v povezavi s podatkovno bazo MySQL, kjer so shranjeni vsi podatki.

Pred glavnim delom diplomske naloge sem predstavil analizo in načrtovanje aplikacije, katera je bila načrtovana po metodologiji RUP, ki zajema štiri faze življenjskega cikla. Načrtovanje zajema zahteve z opisom in prikazom vseh primerov uporabe ter predstavitev podatkovnega in procesnega dela.

Po načrtovanju sem poslovno aplikacijo podrobno opisal. Opisal sem njeno funkcionalnost, način uporabe, kot jih vidi uporabnik. Bolj podrobno so opisani razviti moduli poštna knjiga, prenos naročnikov in iskanje po zgodovini. Moduli so bili izdelani predvsem zaradi lažjega dela, prenosa dokumentov in baze znanja v elektronsko obliko poslovanja.

V zaključku diplome sem izpostavil prednosti in slabosti nove poslovne aplikacije ter ideje in rešitve za nadaljnjo razširitev.

Ključne besede

Elektronsko poslovanje, B2E, spletna aplikacija, RUP, UML.

2. ABSTRACT

Work presents a new solution for one of the major professional organizations in the Republic of Slovenia, whose core functions are organizing and carrying out consultancy work in the field of accounting and corporate finance. The thesis presented details the making of business web applications to help clients to improve and facilitate the activities of their employees. In collaboration with RSteam, d. o. o., we have used tools to develop a new application.

Business applications are important in today's world, both business and private, since we work with them to alleviate problems. Applications help us in everyday life; however they differ in their use, purpose, and especially in their own design. Many have been built in different programming languages, but they differ also by type of environment for which they are built.

In this work we present the basic concepts of electronic commerce, enterprise applications and related concepts. Get acquainted with the tools and techniques that I use to build applications. For the development of applications I use many of the key technologies, which I have noted in the past. Business application based on a modern programming language PHP and Python in conjunction with the MySQL database where we store all data.

Before the main part of the thesis I presented the analysis and design applications, which was scheduled after the RUP methodology, which comprises four phases in life cycles. Planning involves a description and presentation of all cases of use and presentation of data and how the process works.

After designing the business application I described it in detail. I have described its functionality, the application method, as seen by the user. More details are described to develop modules; postal book, transfer clients and search history. The modules were designed primarily to facilitate the work of transfer of documents and knowledge base in electronic form of business.

In conclusion I drew degrees pros and cons of new business applications, and ideas and solutions for further extension.

Key words

Electronic commerce, B2E, web application, RUP, UML.

3. UVOD

Opredelitev problema

Živimo v času nenehnega razvoja elektronskega poslovanja. Število uporabnikov interneta narašča iz meseca v mesec. Prav tako narašča število uporabnikov in ponudnikov elektronskega poslovanja, ki uporablja za svoj način poslovanja elektronsko poslovanje. Tako kot večina poslovnih uporabnikov, je naš naročnik uporabljal aplikacijo že vrsto let, ki je bila potrebna temeljite prenove. Temeljita prenova aplikacije bi pripomogla k enostavnejšemu in hkrati učinkovitejšemu poslovanju. Aplikacija je bila razvita v zastarelem programskem jeziku in sama posodobitev te aplikacije bi bila skoraj nemogoča, zato smo se odločili za razvoj nove aplikacije, ki temelji na osnovi predhodne z dodatnimi funkcionalnostmi in moduli glede na zahtevo in želje naročnika.

Namen diplomske naloge

Spletne aplikacije so se v zadnjih letih zelo razvile. Pred tem so imele v primerjavi z namiznimi precej manj funkcionalnosti, ki bi uporabniku delo olajšale. Danes ni več tako, saj nam sodobna tehnologija omogoča, da na spletno stran dodamo praktično vse, kar nam srce poželi. Namen diplomske naloge je uporabniku olajšati delo z aplikacijo, saj spletna aplikacija za uporabo ne potrebuje nobenega dodatnega programa, temveč le poljuben spletni brskalnik. Deluje na vseh platformah, neodvisno od operacijskega sistema, in je dostopna kjerkoli in kadarkoli, ne glede na čas in lokacijo. Večina ljudi dandanes uporablja internet, kar pomeni, da imajo že skoraj vsi izkušnje z vsaj enim spletnim brskalnikom, kar olajša privajanje uporabnika na aplikacijo. Namen diplomske naloge ni le olepšati in olajšati delo z aplikacijo, temveč uporabniku omogočiti še vrsto drugih funkcionalnosti, kot so beleženje svetovanj, podroben pregled zgodovine, enostaven prenos podatkov med zaposlenimi in sinhronizacija baze naročnikov iz računovodskega programa Navision. Za dosego tega smo se odločili, da sta najprimernejša jezika PHP in Python.

Struktura diplomske naloge

V diplomski nalogi želim na splošno predstaviti poslovne aplikacije, jih opisati ter primerjati tehnologije in rešitve za njihov razvoj.

Teoretični del diplomske naloge bo zajemal opis vseh uporabljenih orodij, programske jezike in njihovo uporabo ter glavne lastnosti. V praktičnem delu bom na primeru razvoja poslovne aplikacije opisal celoten potek razvoja, od analize do končnega produkta. Predstavil bom posamezne postopke razvoja. Podrobneje bom predstavil razvite dodatne module in funkcionalnosti s strani uporabnika.

Cilji naloge so:

1. spoznati tehnologije in orodja za razvoj spletnih aplikacij,
2. konkreten prikaz razvoja spletne aplikacije po metodologiji RUP,

3. pripraviti delujoč sistem, ki temelji na modernih in preizkušenih tehnologijah in je primeren za takojšnjo uporabo,
4. razviti sistem, tako da bo omogočal dodajanje novih modulov in bo tako razširljiv za različne potrebe uporabnikov,
5. razviti sistem, ki bo komuniciral z drugimi obstoječimi programi v podjetju,
6. ugotoviti prednosti in slabosti predstavljenih rešitev pri razvoju nove spletne aplikacije.

4. ELEKTRONSKO POSLOVANJE

Hiter razvoj in razširjanje omrežnih ter komunikacijskih tehnologij, predvsem interneta, je bil eden od dejavnikov za nedavne pospešitve gospodarske rasti v večini držav. Številni potrošniki uporabljajo internet in druge omrežne računalniške sisteme za iskanje prodajalcev, vrednotenje proizvodov in nakupovanje. Eksplozija rasti na internetu je vidni pojav prenovljenega svetovnega trga, ki temeljiti na elektronskem poslovanju, v nadaljevanju e-poslovanje.

V angleščini se uporabljata dva izraza, ki označujeta e-poslovanje:

- e-commerce (se je uporabljal v preteklosti),
- e-business (uveljavljen v zadnjem času).

Greenstein in Feinman [1] opredeljujeta e-commerce kot »uporabo elektronskih prenosnih medijev (telekomunikacij) za menjavo izdelkov in storitev, ki terjajo fizični ali digitalni transport z ene lokacije na drugo, vključujoč kupovanje in prodajo« ter (2000, 1–2) ločita »e-commerce«, pri katerem gre za računalniško izmenjavo podatkov, vezano na nabavo in prodajo blaga oziroma storitev in ne obsega v celoti prave narave izmenjanih informacij s pomočjo telekomunikacijskih naprav. Pojem »e-business« vključuje tudi izmenjevanje podatkov, ki niso neposredno vezani na določen nakup in prodajo blaga/storitev, zagotavljajo pa distribucijo informacij in podporo strankam, npr. novice v elektronskih časopisih. Te aktivnosti niso »komercialne« aktivnosti, ampak so »poslovne« aktivnosti. Zato je izraz e-business veliko širši od pojma e-commerce in bolj ustreza slovenskemu izrazu e-poslovanje.

Elektronsko poslovanje temelji na računalniški izmenjavi podatkov (RIP) in vseh podobnih tehnologij. Zasnovano je za uporabo elektronskih in digitalnih komunikacij, ki se uporabljajo za podjetja, z namenom povečanja tržnega deleža in ohranitev dolgoročne sposobnosti preživetja v poslovnem okolju.

Izraz elektronsko poslovanje je eden izmed najpogosteje uporabljenih poslovnih izrazov in se uporablja v večini že na vseh področjih. E-poslovanje se nanaša na avtomatizacijo poslovnih procesov vseh vrst v elektronski obliki. E-mail, elektronske oglasne deske, faksimile, finančni prenosi sredstev so tipične oblike omenjenega poslovanja.

E-poslovanje lahko razdelimo na tri široka področja:

Ekstranet: interakcija podjetje – podjetje (B2B – Business to Business)

Internet: interakcija podjetje – potrošnik (B2C – Business to Consumer)

Intranet: interakcija podjetje – zaposleni (B2E – Business to Employee)

4.1. Delitev elektronskega poslovanja

B2B – poslovanje med podjetji

B2B je avtomatiziran proces med trgovinskimi partnerji in se izvaja v večji količini. Vključuje lahko tržne dejavnosti med podjetji in ne le končnih transakcij, ki izhajajo iz trženja. Uporablja se tudi za zagotavljanje prodajnih transakcij med podjetji oz. zagotavljanje poslovnih odločitev preko spletnih transakcij z drugimi podjetji. B2B e-poslovanje se bolj osredotoča na ustvarjanje visoko učinkovitih in preglednih trgov, da bi spremenilo strukturo verig vrednosti industrije.

B2C – poslovanje med podjetji in odjemalci

Opisuje dejavnosti v gospodarskih organizacijah, ki služijo končnemu potrošniku s proizvodi in/ali storitvami. Z drugimi besedami, gre za izmenjavo transakcij, informacij, izdelkov ali storitev med podjetjem in potrošnikom. B2C način e-poslovanja lahko pomaga omejiti stroške pri ustanavljanju trgovine, zavaruje prodajalca ter razvija bolj učinkovito dobavno verigo.

B2E – poslovanje znotraj podjetja

Oblika B2E elektronskega poslovanja nakazuje, da je elektronsko poslovanje v prvi vrsti način delovanja in organiziranosti v podjetju, ki ni nujno povezano z eksternostjo. V obravnavano obliko poslovanja sta vpletena podjetje in zaposleni. Gre torej za interno obliko poslovanja, ki je praviloma podprta s portalom.

4.2. Opredelitev razvite poslovne aplikacije

Aplikacijo uvrščamo med B2E aplikacije, saj je dosegljiva samo na lokaciji podjetja. Za izdelavo aplikacije te vrste je potrebno najprej analizirati potrebe kupca in na podlagi njegovih želja izdelati dober načrt. Gre za spletno aplikacijo, postavljeno na serverju, ki se nahaja na lokaciji podjetja. Prednost take aplikacije je, da se vsi podatki nahajajo na enem mestu in so na voljo uporabnikom za iskanje po njih. Za prijavo v sistem je potrebno vnesti uporabniško ime in geslo, katero se dodeli uporabnikom aplikacije. Poleg tega ima uporabnik točno določene funkcionalnosti pri uporabi spletne aplikacije. Namen aplikacije je uporabnikom olajšati delo in prihraniti pomemben čas, katerega izgubijo s trenutno aplikacijo. Glavna funkcionalnost aplikacije je arhiviranje telefonskih, pisnih in osebnih svetovanj, kar omogoča uporabnikom vpogled v vsa opravljena svetovanja in njegove podatke. Ključna prednost aplikacije te vrste je, da je za spremembo dela aplikacije potrebno spremeniti aplikacijo samo na serverju in ne na vseh računalnikih, ki jih uporabljajo uporabniki.

5. UPORABLJENA TEHNOLOGIJA IN ORODJA

5.1. Predstavitev uporabljenih tehnologij

Pravilna izbira tehnologij je pri razvoju spletne aplikacije zelo pomembna. Izbira primerne tehnologije za vse aplikacije se vrši v sodelovanju z naročnikom, ki razvijalcem posreduje podrobne podatke o funkcionalnosti in izgledu aplikacije.

Pri tem je potrebno biti pozoren, saj je potrebno pomisliti tudi na izvedbo dodatnih modulov, ki bi lahko kasneje razširili aplikacijo, z izbrano tehnologijo. Po pridobljenih podatkih smo se odločili za uporabo tehnologij, katere sem na kratko tudi predstavil v nadaljevanju.

5.1.1. Označevalni jezik HTML

Hyper Text Markup Language (slovensko: jezik za označevanje nadbesedila, kratica HTML) je označevalni jezik za izdelavo spletnih strani [2]. Nadbesedilo (hípertékst) je način označevanja besedila ali grafičnih elementov (slika ali del slike), ki omogočajo povezavo (skok) na drugi del besedila ali večpredstavni element. HTML je standardiziran; za ta standard skrbi W3C ter sodi v družino SGML (Standard Generalized Markup Language). Značilnost jezika HTML je enostavnost, saj je bil zasnovan tekstovno z minimalnim številom oznak, da se ga lahko vsakdo hitro nauči in s poljubnim urejevalnikom napiše lasten dokument. HTML dokument je tekstovna datoteka, ki vsebuje oznake (tags), s pomočjo katerih različnim brskalnikom (Firefox, Internet Explorer, Google Chrome ipd.) povemo, kako naj prikažejo vsebino naše spletne strani.

HTML oznake

Vsak HTML dokument vsebuje oznake. Vse oznake so definirane med znakoma za manjše (<) in pa večje (>) npr. <td>. Nekatere oznake so enodelne, spet druge dvodelne. Dvodelne oznake začnemo pisati s prvim delom, npr. (<td>), in zaključnim delom (</td>). Zaključni del vedno vsebuje poševno črto in se tako tudi razlikuje od začetnega dela. Oznake lahko vsebujejo atribute, kateri nudijo dodatne informacije brskalnikom (npr. align=”center”).

5.1.2. Stilski jezik CSS

Cascading Style Sheets (kratica CSS) je slogovna predloga, ki določa izgled spletnih strani (videz in format), napisanih v označevalnem jeziku. Izdelovalcem spletnih strani omogoča ločevanje vsebine spletnih strani od sloga strani.

Preboj CSS-ja se je začel, ko se je pojavila potreba po lepših, boljših in predvsem zahtevnejših spletnih straneh. Najpogosteje se uporablja za slog spletne strani, napisane v HTML in XHTML. Jezik se lahko uporablja tudi za kakršen koli dokument, kot npr. XML, vključno s SVG in XUL. Slogi v HTML določajo, kako so elementi oblikovani. Slogovne

predloge so ponavadi shranjene v posebnem ločenemu dokumentu zaradi lažje preglednosti in urejanja. Eno slogovno predlogo lahko uporabimo na več drugih straneh. CSS nam omogoča določitev pisave, velikost črk in vizualno podobo spletne strani.

5.1.3. Odprtokodni programski jezik PHP

PHP (PHP Hypertext Preprocessor) [3] je razširjen odprtokodni skriptni programski jezik, ki se uporablja za strežniške uporabe oziroma za razvoj dinamičnih spletnih vsebin. Sam princip programiranja v PHP je zelo podoben strukturiranim programskim jezikom C in Perl. Glavni cilj piscev PHP je bil, da omogočijo izdelovalcem spletnih strani hitro izdelavo.

PHP je zelo prilagodljiv in ga je preprosto kombinirati s številnimi jeziki. Lahko se uporablja na vseh popularnejših operacijskih sistemih (Linux, *BSD, Mac Os, MS Windows ipd.). PHP podpirajo skoraj vsi spletni strežniki (Apache, IIS, Lighttpd, Nginx ipd.). Večina teh ima PHP modul, lahko pa dela tudi preko CGI. S PHP-jem imamo na izbiro proceduralno ali objektno programiranje oz. kombinacijo obojega. Posebno močan je v sodelovanju z raznimi zbirkami podatkov. Podpira vse popularnejše baze (MySQL, PostgreSQL, MSSQL, Oracle, SQLite).

5.1.4. Objektni skriptni jezik JavaScript

JavaScript je objektni skriptni programski jezik, ki ga je razvil Netscape z namenom, da pomaga spletnim programerjem pri ustvarjanju interaktivnih spletnih strani [4]. Omogoča OUP (objektno usmerjeno programiranje), saj podpira dedovanja s pomočjo prototipov, kot tudi lastnosti in metod. Pisanje objektne orientirane kode nam daje moč, da lahko pišemo kodo, katero lahko ponovno uporabimo in je zaprtega tipa.

Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko uporabljamo s HTML-kodo in s tem poživimo stran z dinamičnim izvajanjem. Podprt je s strani velikih programskih podjetij in kot odprt jezik ga lahko uporablja vsakdo, ne da bi pri tem potreboval licenco. Podpirajo ga vsi novejši spletni brskalniki.

5.1.5. Programski jezik Python

Python je tolmačeni programski jezik. Ime je dobil po priljubljeni televizijski nanizanki Leteči cirkus Montyja Pythona (Monthy Python's Flying Circus). Je izredno močan dinamičen programski jezik, ki se uporablja na različnih področjih. Zaradi dinamičnih podatkovnih tipov je podoben programskim jezikom Ruby, Perl, Scheme, SmallTalk in Tcl. Ker je zelo dinamičen jezik, dovoljuje preproste rešitve vseh problemov, ki se lahko pojavijo pri programiranju.

Python je interpretativni, objektno orientiran jezik in je označen kot visoko nivojski jezik [5]. Namenjen je predvsem objektno orientiranem programiranju, v manjši meri pa tudi funkcijskem programiranju.

Za spreminjanje kode v programu ne potrebujemo nič drugega kot poznavanje skripte in pa Python Interpreter, katerega lahko dobimo na Pythonovi uradni strani.

5.2. Predstavitev orodij

Razvojno orodje je integrirano razvojno okolje, tako imenovano IDE (Integrated Development Environment). Je programska aplikacija, ki programerju zagotavlja celovito podporo pri razvoju. IDE običajno vsebuje:

- urejevalnik izvorne kode,
- prevajalnik oz. tolmača
- orodje za avtomatizacijo izgradnje programa,
- razhroščevalnik.

V razvojno orodje je včasih vgrajen tudi sistem za nadzor verzij programov in orodje za tvorbo grafičnih uporabniških vmesnikov (GUI, Graphical User Interface).

Mnoga sodobna razvojna okolja, ki predvidevajo objektno usmerjeno programiranje, vsebujejo tudi brkljalnik razredov (class browser), inšpektor objektov in diagram hierarhije razredov.

Nekatera izmed orodij so osredotočena na pomoč pri načrtovanju in kreiranju podatkovne baze s pomočjo povpraševalnega jezika SQL.

V tem poglavju bom opisal:

- Netbeans IDE,
- Komodo,
- phpMyAdmina,
- Power Designerja.

5.2.1. Razvojno okolje NetBeans

NetBeans IDE je odprto-kodno integrirano razvojno okolje za razvoj Javanskih aplikacij [6]. Napisan je v celoti v Javi in lahko deluje na več platformah (vključno z Windows, Linux, Mac OS X in Solaris), kjer je nameščen javanski virtualni računalnik JVM (ang. Java Virtual Machine).

NetBeans omogoča tudi izdelavo grafičnih vmesnikov (GUI) po sistemu povleci in spusti, kjer se v ozadju sproti generira tudi koda. Iz menija vzamemo različne gumbe ali polja in jih namestimo na osnovni panel. NetBeans nam naredi ogrodje, v katerega potem samo še napišemo našo kodo. Možen je preprost preklop med kodnim izpisom aplikacije in grafičnim prikazom ter razvoj v kombinaciji obeh načinov.

NetBeans vsebuje integrirano razvojno okolje (IDE) za razvoj aplikacij v različnih programskih jezikih, kot so npr. JavaScript, PHP, Python, Ruby, C, C++ itd. Kot razvojno okolje za razvoj aplikacije v diplomski nalogi sem uporabil NetBeans PHP, različico 6.9.1.

PHP podpora

Spletno aplikacijo sem se odločil izdelati v jeziku PHP v razvojnem okolju NetBeans, saj ponuja IDE različico nalašč za razvijanje PHP kode, katera obsega različne skripte in označevalne jezike. PHP urednik je dinamično povezano s HTML, CSS in JavaScript funkcijami za urejanje. PHP podpora je omogočena vse od različice 6.5.

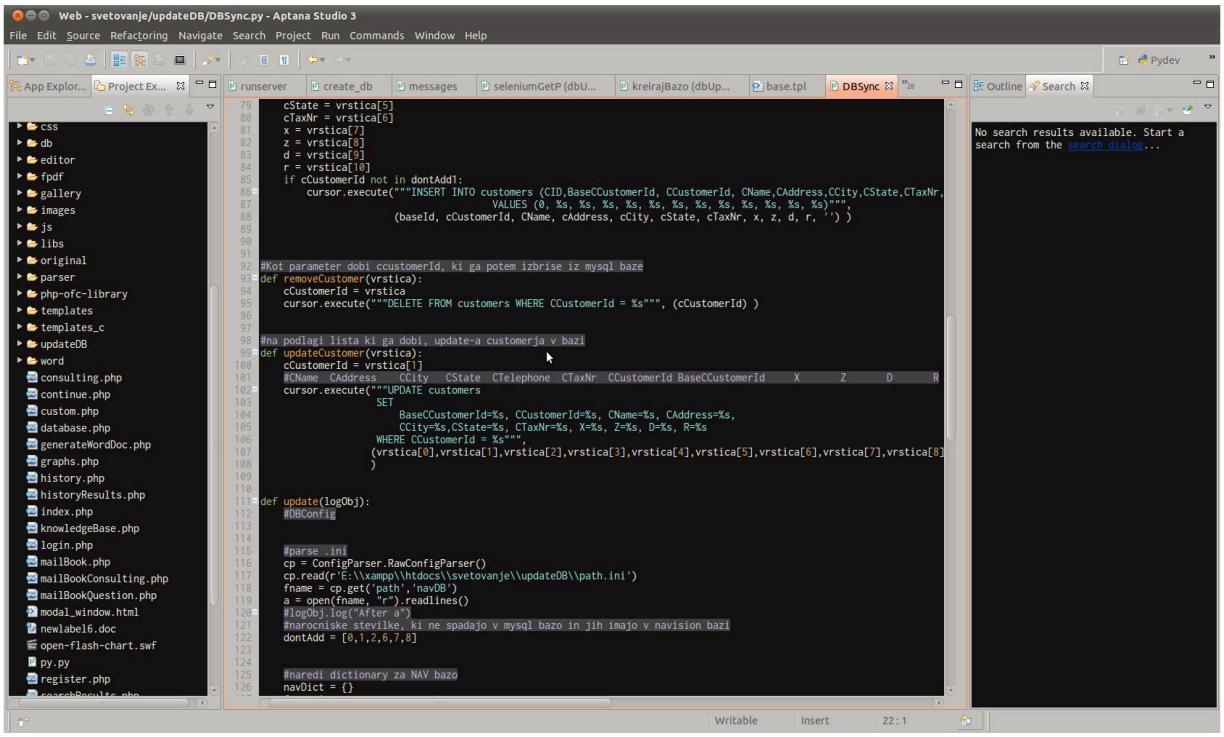
Netbeans PHP IDE vsebuje:

- osnovno urejanje,
- poudarjanje sintakse,
- zaključevanje besed,
- refaktoriranje kode,
- razhroščevalnik PHP kode z xdebug,
- testiranje z PHPUnit in selen,
- vključevanje različnih podatkovnih baz,
- pokritost kode,
- Zend Framework (od različice 6.9),
- Symfony Framework (od različice 6.9),
- PHP 5,3 namespace (od različice 6.8).

5.2.2. Aptana Studio

Aptana Studio (Slika 1) je robustno razvojno okolje za spletne strani in aplikacije, ki podpira tehnologije Python, JavaScript, Ruby on Rails, PHP, Ajax, DOM, HTML in CSS, na voljo sta tudi vstavka za Adobov AIR in razvoj programske opreme za Applov iPhone [7]. Za naprednejše razvijalce je na voljo tudi Aptana Studio Pro s plačljivo tehnično podporo, obe različici pa vsebujeta strežnik Ajax-Jaxer. Aptana temelji na Eclipsu.

Gre za brezplačno, odprtokodno in od operacijskega sistema neodvisno razvojno okolje, primerno predvsem za razvoj aplikacij za težke odjemalce.



Slika 1: Razvojno okolje Aptana Studio

Python podpora

Določene dele aplikacije, ki niso bili neposredno povezani s spletno stranjo, sem programiral v programskem jeziku Python. Mednje spadajo skripta za sinhronizacijo baz, ki so namenjena za vsakodnevno shranjevanje varnostnih kopij baze ipd. Vse to smo počeli v Aptana Studio, saj zagotavlja podporo za Python v obliki PyDev vtičnika. Aptana je uradni vzdrževalec Pydev (http://aptana.com/python) projekta za Eclipse. Aptana Pydev je razviti Python modul, ki omogoča boljše preverjanje sintakse in podporo pri razvoju Python in Jython kode. Ta omogoča poudarjanje barvne sintakse, zaključevanje kode in opombe za odpravljanje napak ter vgrajeno podporo pri pisanju kode. Vgrajeno ima tudi podporo za Python, Jython in IronPython.

5.2.3. Orodje za delo z bazami phpMyAdmin

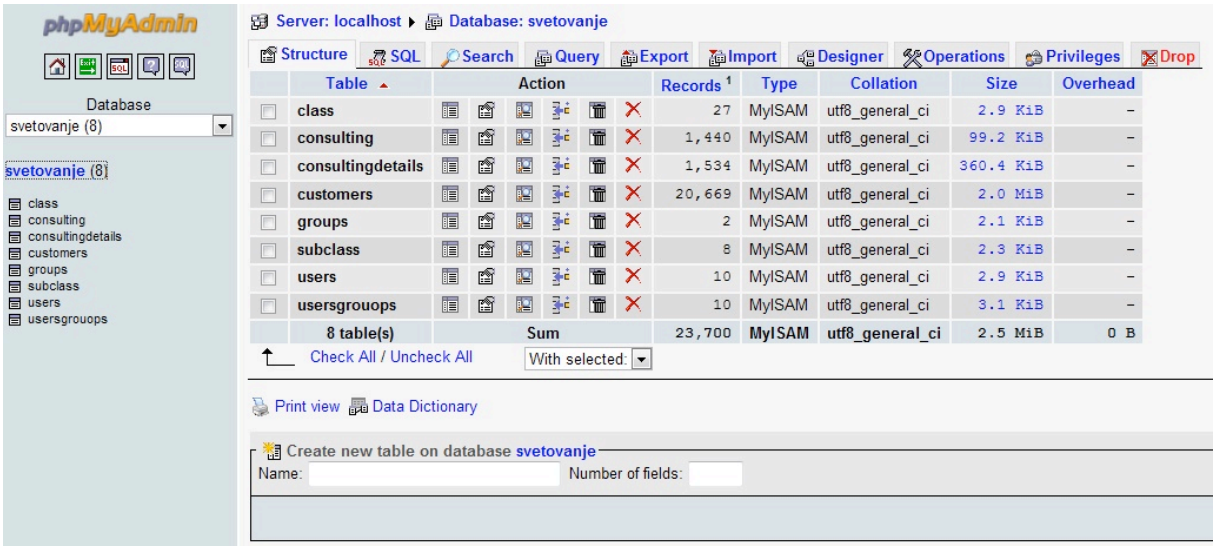
PhpMyAdmin je odprtokodna aplikacija, napisana v PHP, ki nam omogoča upravljanje MySQL baz kar preko spletnega brskalnika (Slika 2). MySQL [8] je sistem, namenjen upravljanju s podatkovnimi bazami, ki za delo s podatki uporablja SQL stavke. PhpMyAdmin je na voljo v 57 jezikih, med drugimi tudi v slovenščini.

S tem orodjem lahko opravljamo različne naloge:

- kreiranje in brisanje celotne podatkovne baze,

- izvoz in uvoz celotne podatkovne baze,
- kreiranje, brisanje in spreminjanje table,
- kreiranje, brisanje in spreminjanje polj v tabelah,
- upravljanje s ključi na poljih,
- upravljanje uporabnikov in določanje pravic,
- globalno iskanje po celotni bazi,
- izvajanje, urejanje in shranjevanje SQL stavkov.

Zelo podobno orodje phpMyAdminu je phpPgadmin, le da ta omogoča delo s PostGreSQL bazami. Za MS SQL podatkovne baze pa obstaja orodje phpMSAdmin. Obstaja tudi oskubljena verzija phpMyAdmina, ki sestoji iz ene same PHP datoteke in podpira vse glavne funkcije phpMyAdmin – phpMinAdmin.



Slika 2: Upravljanje MySQL baze preko brskalnika s phpMyAdmin

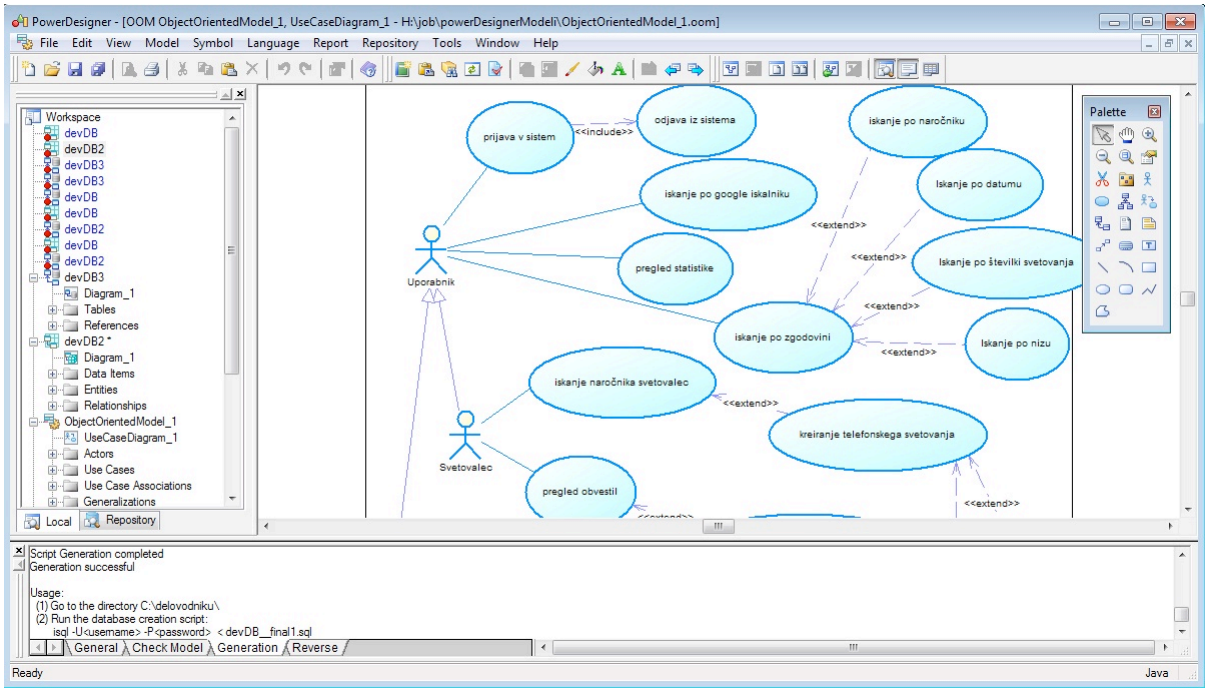
5.2.4. Orodje PowerDesigner

PowerDesigner [9] je orodje za modeliranje, ki ga je izdelalo podjetje Sybase. Deluje na operacijskih sistemih Windows. Uporabniku omogoča povečanje produktivnosti, razna orodja za analiziranje in predvsem enostavno izdelavo različnih diagramov [10]. Izdelava diagramov je zelo pomembna za podjetje, saj lahko praktično celotno strukturo baze, izmenjavo podatkov ter vsako funkcionalnostjo predstavimo z diagramom. S pomočjo orodja PowerDesigner lahko izdelamo različne diagrame, ki so potrebni pri razvoju programske opreme, hitro, učinkovito in dosledno.

PowerDesigner (Slika 3) je enostaven za namestitev in uporabo in je popolnoma neodvisen od vse večje programske opreme ali zbirke podatkov. Vključuje poslovno modeliranje, modeliranje podatkov in objektno modeliranje, tako da lahko z njim razvijamo konceptualne, logične in fizične modele. PowerDesigner je tudi preprost za uporabo, tako da se lahko bolj osredotočimo na sam model in manj na orodje.

Ima podporo za vse glavne platforme: več kot 60 podatkovnih baz, vodilne platforme za razvoj aplikacij, kot so Java J2EE, Microsoft.NET, spletne storitve in PowerBuilder.

Pri aplikaciji sem ga uporabil za izdelavo konceptualnega podatkovnega modela, diagrama primerov uporabe in diagramov zaporedja.



Slika 3: PowerDesigner, orodje za modeliranje

Med drugim omogoča tudi diagrame, kot so:

- entitetni diagram

Entitetni diagram grafično predstavi relacije med entitetami in atributi v podatkovni bazi.

Entiteta je objekt, ki ima shranjene podatke. Relacija definira, kako sta dve entiteti povezani.

Poznamo tri tipe relacij med entitetami, in sicer one-to-one, one-to-many in many-to-many.

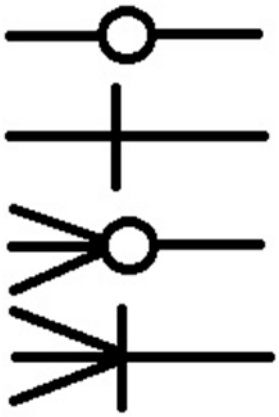
- diagram podatkovnih tokov

Diagram podatkovnih tokov nam prikaže, kako informacije potekajo od objekta do objekta. Gradniki, ki sestavljajo ta diagram, so tok podatkov, proces, podatkovno skladišče in zunanji izvori ter ponori podatkov.

- relacijski diagram

Relacijski diagram lahko PowerDesigner sam generira na podlagi entitetnega diagrama. Razlika je, da je entiteta predstavljena kot tabela, atribut kot stolpec, identifikacijski atribut kot primarni ključ in povezava kot tuj ključ.

Povezave med entitetami (Slika 4) so lahko naslednje:



- Interval 0,1
- Interval 1,1
- Interval 0..n
- Interval 1..n

Slika 4: Prikaz povezav med entitetami

5.3. Metodologije razvoja RUP

Rational Unified Process (RUP) je iterativen proces razvoja programske opreme, je ogrodje procesa za razvoj programske opreme, ki natančno določa, kaj in kdaj je potrebno storiti, kaj potrebujemo in kaj bo pri tem nastalo. Določa tudi, kdo bo posamezno aktivnost izvedel. Kljub obsežnosti in primernemu nivoju opisa delovnih napotkov so določena pomembna področja le bežno opisana z izdatnim seznamom referenc na dodatno literaturo o področju.

RUP ni strogo določen, ampak predstavlja zgolj okvir. Ta okvir zagotavlja smernice, predloge in uspešne primere za različne vidike in faze razvoja, pri čemer pa razvijalsko podjetje samo izbere elemente procesa, ki ustrezajo njihovim potrebam.

Razvoj programske opreme po metodi RUP temelji na iteracijah. Ker je pri razvijanju kompleksnejših programskih rešitev skoraj nemogoče zajeti vse cilje in izvesti cel razvoj v enem samem koraku, nam metoda RUP delo razbije na manjše dele, ki jim pravimo iteracije.

Za uspešen projekt je potrebno, da so iteracije:

- nadzorovane,
- načrtovane,
- v pravilnem vrstnem redu.

Vrstni red iteracij je tak, da se najprej izvede bolj tvegane iteracije, šele kasneje pa tiste bolj rutinske.

Proces razvoja je razbit v razvojne cikle, vsak cikel pa je razbit v štiri zaporedne faze:

- zametek – začetna faza (angl. “inception”),
- dodelava – zbiranje informacij (angl. “elaboration”),

- izgradnja – konstrukcija (angl. “construction”),
- prehod – prevzem (angl. “transition”).

Po vsaki fazi pregledamo, ali smo dosegli načrtovane rezultate faze in ali je možen zaključek trenutne faze in prehod v naslednjo. Vsaka izmed faz je razdeljena na več iteracij, vsaka iteracija pa gre skozi šest različnih delovnih tokov.

RUP je razdeljen na tehnični (opis proizvodov, aktivnosti in odgovornosti) in dinamični pogled (opis faz, ciklov, mejnikov, osredotočenosti na čas, denarja, virov, kakovosti in tveganja).

Opisuje način učinkovite uporabe šestih najboljših izkušenj s področja razvoja programske opreme, in sicer:

- literativni razvoj,
- obvladovanje zahtev,
- komponentna arhitektura,
- vizualno modeliranje,
- preverjanje kakovosti,
- nadzor sprememb.

5.3.1. Jezik za modeliranje UML

Unified Modeling Language (UML) je standarden jezik [11], namenjen modeliranju računalniških sistemov. Jezik je neodvisen od uporabljenih programskih jezikov ter razvojnega procesa. Omogoča določitev, predstavitev in dokumentacijo sistemov. Zajema več vrst diagramov, s katerimi lahko sistem predstavimo iz različnih vidikov [12].

Cilj jezika UML je zagotoviti uporabnikom standardiziran ter razširljiv opisni mehanizem za prikaz različnih vidikov sistema.

UML je jezik za specifikacijo, vizualizacijo, konstrukcijo in dokumentacijo izdelkov programsko-intenzivnega sistema.

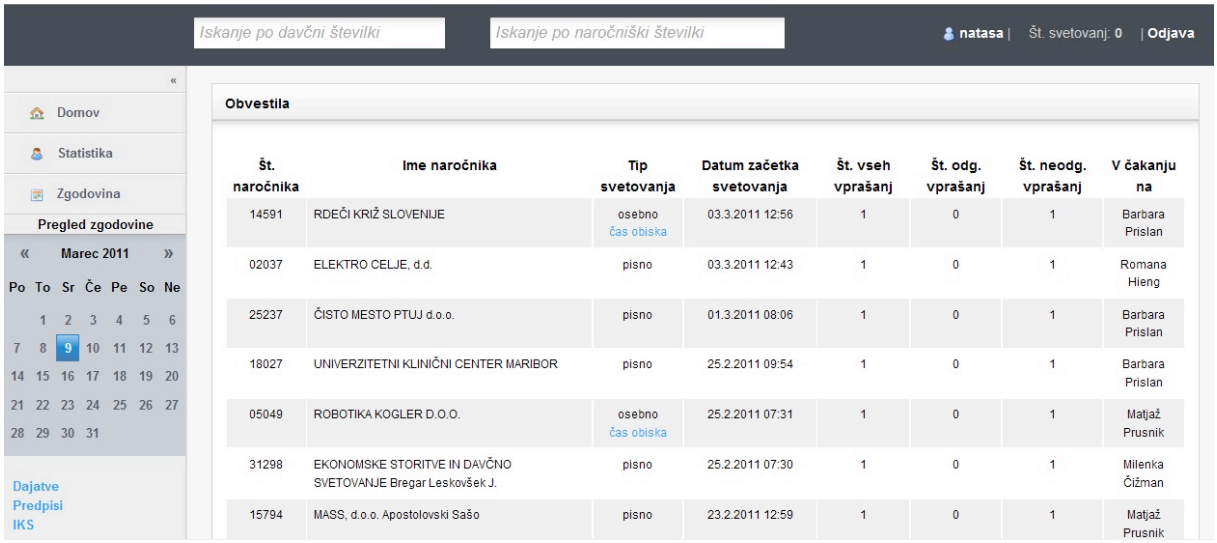
Poznamo 5 pogledov na problem s stališča UML:

- uporabniške zahteve (diagram primerov uporabe),
- statična slika sistema (razredni diagram),
- obnašanje programskega sistema (diagram prehajanja stanj, diagram aktivnosti),
- interakcija objektov (diagram zaporedja, diagram sodelovanja),
- implementacijski konstrukti (komponentni diagram, paketni diagram, diagram razvoja in dobave).

6. RAZVOJ MODULOV APLIKACIJE

Moduli so dandanes zelo pomemben del vsake aplikacije. Uporabniku omogočajo različne funkcionalnosti, katere jim lahko poenostavijo pridobitev želenih informacij, pregled nad raznimi podatki, hranjenje novih informacij ipd. V aplikaciji (Slika 5) sem izdelal naslednje module za uporabnike, in sicer iskanje po zgodovini, pregled statistike, vgrajeni Google iskalnik, prenos naročnikov iz Navision baze v bazo spletne aplikacije in poštna knjiga.

Delovanje, opis in način izvedbe najpomembnejših sem v nadaljevanju tudi podrobneje predstavil.



Slika 5: Aplikacija za podporo delovanja svetovalcev

6.1. Opis obstoječe aplikacije

Obstoječa aplikacija, namenjena beleženju svetovanj, je stara okoli 20 let. Bila je precej okrnjena in nepregledna. Ni omogočala vnosa vprašanj in odgovorov, torej ni hranila zgodovine, kar je zelo moteče, saj svetovalci velikokrat odgovarjajo na ista vprašanja. Ob izdaji popravkov je bilo potrebno posodobiti aplikacijo na vsakem računalniku posebej, kar je pomenilo čakanje svetovalcev na razvijalce, saj sami tega niso znali storiti. Poleg tega aplikacija ni vsebovala poštne knjige, kar pomeni, da so svetovalci morali podatke o pisnih in osebnih svetovanjih osebno oddati pristojni osebi za to področje. Ta pa je morala podatke urediti in vnesti v nov Word dokument. Aplikacija je bila programirana v programskem jeziku Visual Basic.

6.2. Analiza zahtev in načrtovanje

Zahteve je potrebno analizirati, da lahko omogočimo implementacijo rešitve, ki bo dejansko zadoščala potrebam naročnikov. Potrebno je določiti prioritete zahtevam, jih organizirati in modelirati ter opredeliti omejitve, ki vplivajo na izbor rešitve. Zatem se presodi ustreznost zbranih zahtev in preveri njihovo veljavnost.

Za uspešno izvedbo razvoja aplikacija sta analiza in načrtovanje najpomembnejša dejavnika, saj nam omogočata doseganje boljšega razumevanja sistema, ga poenostavimo ter zagotovimo ponovno uporabo, vizualizacijo in nadzor nad arhitekturo sistema.

Naše zahteve so bile, da mora aplikacija:

- delovati v spletnem brskalniku Internet Explorer na operacijskem sistemu Windows,
- biti vidna samo znotraj podjetja,
- biti enostavna za uporabo,
- biti skalabilna za lažje dodajanje novih modulov.

6.3. Načrtovanje sistema

Na uspešno, učinkovito in varno delovanje informacijskega sistema najbolj vpliva kvaliteta izvedenega načrtovanja, v katerem morajo biti podrobno analizirane naročnikove želje, potrebe in cilji.

Načrtovanje sistema je zelo pomemben del razvoja aplikacije, saj nam vsaka napaka pri samem načrtovanju lahko oteži implementacijo ali pa celo povzroči spremembo dela strukture sistema, kar podaljša čas, potreben za razvoj aplikacije.

6.3.1. Primer uporabe

Diagram primera uporabe (use-case)

Diagram primera uporabe predstavlja komunikacijo med uporabniki in sistemom [13]. Gradniki diagrama primera uporabe so:

- akterji,
- primeri uporabe,
- povezave med akterji ter primeri uporabe,
- povezave med primeri uporabe in medsebojne povezave med akterji.

Ta diagram uporabe nam pomaga pri definiranju sistemskega in kontrolnega načrtovanja kot sredstvo za zajemanje in sledenje zahtevam, komunikacijo s končnim uporabnikom in strankami, razgradnji sistema in ocenjevanju velikosti projekta in potrebnih virov.

Sistem modeliramo kot množico procesov, ki predstavljajo funkcionalnosti katere lahko določen uporabnik uporablja.

Zaradi lažje predstave funkcionalnosti spletne aplikacije sem izdelal diagram primerov uporabe, ki prikazuje funkcionalnost celotne aplikacije (Slika 6).

Sistem vsebuje akterje: uporabnik, svetovalec in uporabnik poštne knjige.

Primere uporabe, ki so na voljo tako svetovalcem kot uporabniku poštne knjige, sem povezal z akterjem "Uporabnik", da je stvar preglednejša.

Uporabnik ima naslednje dodatne funkcionalnosti:

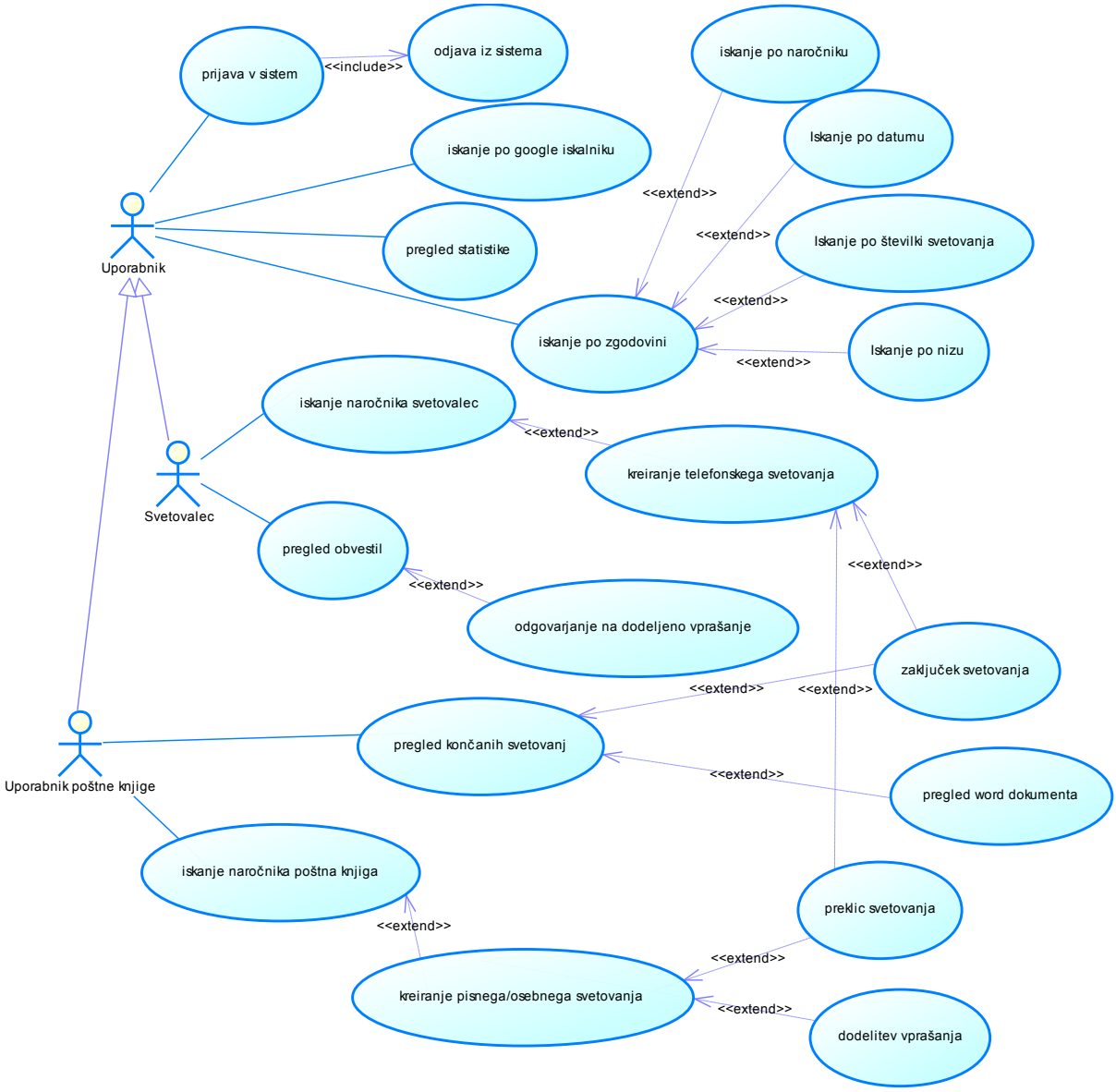
- prijava v sistem (po prijavi se mora uporabnik, ko konča z delom, odjaviti),
- iskanje po Google iskalniku,
- pregled statistike,
- iskanje po zgodovini (lahko iščemo po naročniku, datumu, številki svetovanja ter izbranem nizu).

Svetovalec ima poleg vseh možnosti prijavljenega uporabnika še naslednji možnosti:

- pregled obvestil, kjer lahko odgovori na dodeljeno vprašanje,
- iskanje naročnikov. Po predhodnem iskanju naročnika ima svetovalec možnost kreiranja telefonskega svetovanja, katerega lahko zaključi oziroma prekliče.

Uporabnik poštne knjige tudi generalizira uporabnika. Njegove dodatne funkcionalnosti so:

- iskanje naročnika (sledi kreiranje osebnega oziroma pisnega svetovanja ter preklica oziroma dodelitve vprašanja),
- pregled končanih svetovanj (poljubnega lahko zaključimo in generiramo Word dokument)



Slika 6: Primer uporabe za aplikacijo

6.3.2. Procesni model

Je hierarhični prikaz funkcij, procesov ter njihovih odvisnosti. Procesni model lahko predstavimo z diagramom funkcionalne razgradnje ali funkcionalne dekompozicije in diagramom podatkovnih tokov.

Lastnik procesa mora zelo dobro poznati potek procesa in akterje, ki v njem sodelujejo. Dobro mora tudi preučiti faze, v katerih lahko pričakuje zastoje, da lahko pravočasno ukrepa. Zato je zelo pomembno, da si lastnik izdela procesni model, kateri mu omogoča lažje vodenje procesa.

6.3.2.1. Diagram zaporedja

Diagram zaporedja se nanaša na točno določen primer uporabe. Prikazuje sodelovanje in izmenjavo medsebojnih sporočil objektov na osnovi časovnega zaporedja, ne prikazuje pa povezav med objekti.

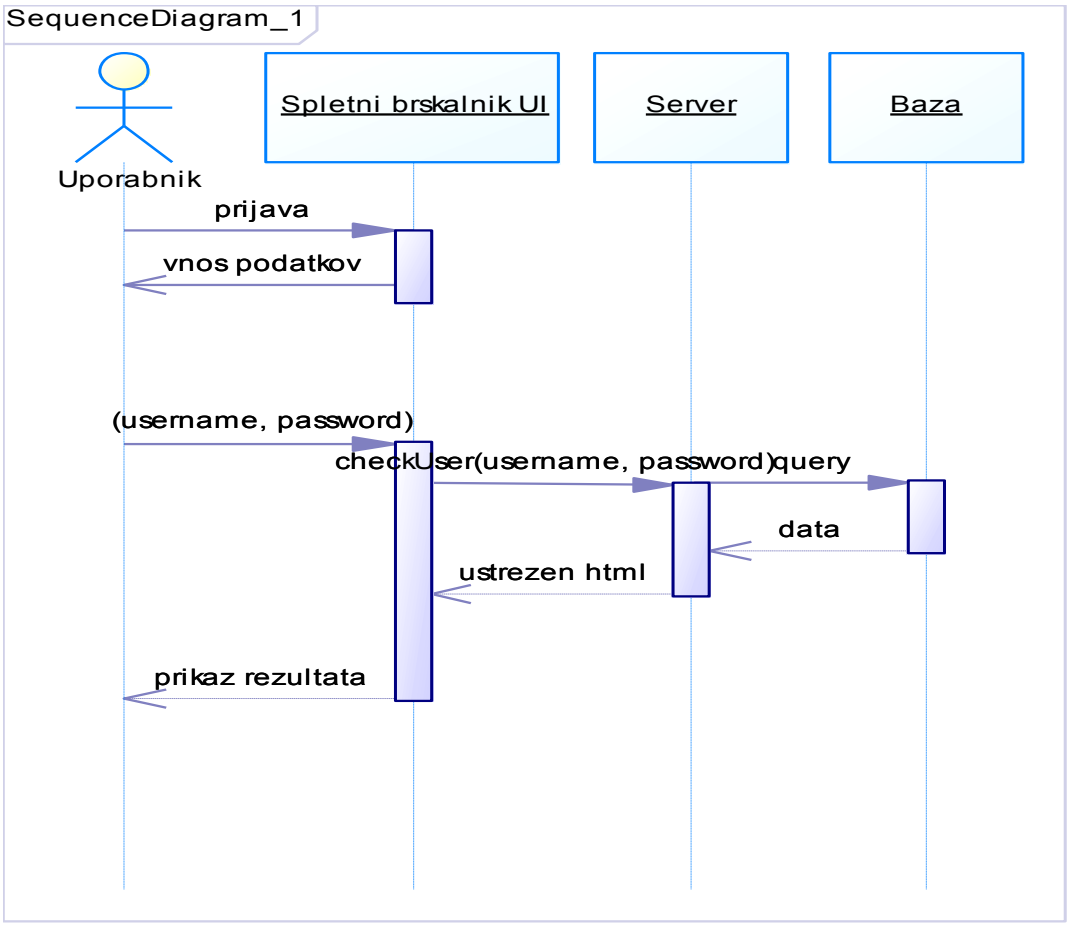
Diagram zaporedja ima **splošno obliko**, ki opisuje vsa možna zaporedja in **obliko primerkov**, ki opisuje konkretno zaporedje. Ima tudi dve dimenziji. **Vertikalna dimenzija** prikazuje čas, **horizontalna** pa različne objekte, ki sodelujejo v procesu.

Poznamo štiri osnovne gradnike diagrama zaporedja:

- življenjska črta objekta,
- aktivacija,
- sporočilo,
- časi prehoda.

Za spletno aplikacijo sem izdelal dva diagrama zaporedja. Prvi predstavlja postopek prijave v sistem, drugi pa postopek iskanja po zgodovini.

Slika 7 predstavlja diagram zaporedja za prijavo v sistem, ki najprej zahteva od uporabnika vnos uporabniškega imena ter gesla. Ko uporabnik to vnese, se kliče metoda checkUser, ki sprejme parametra uporabniško ime in geslo. Nato server sestavi poizvedbo in dobi povratno informacijo od baze. Preveri, če ta uporabnik obstaja, in vrne HTML kodo za prikaz v brskalniku.

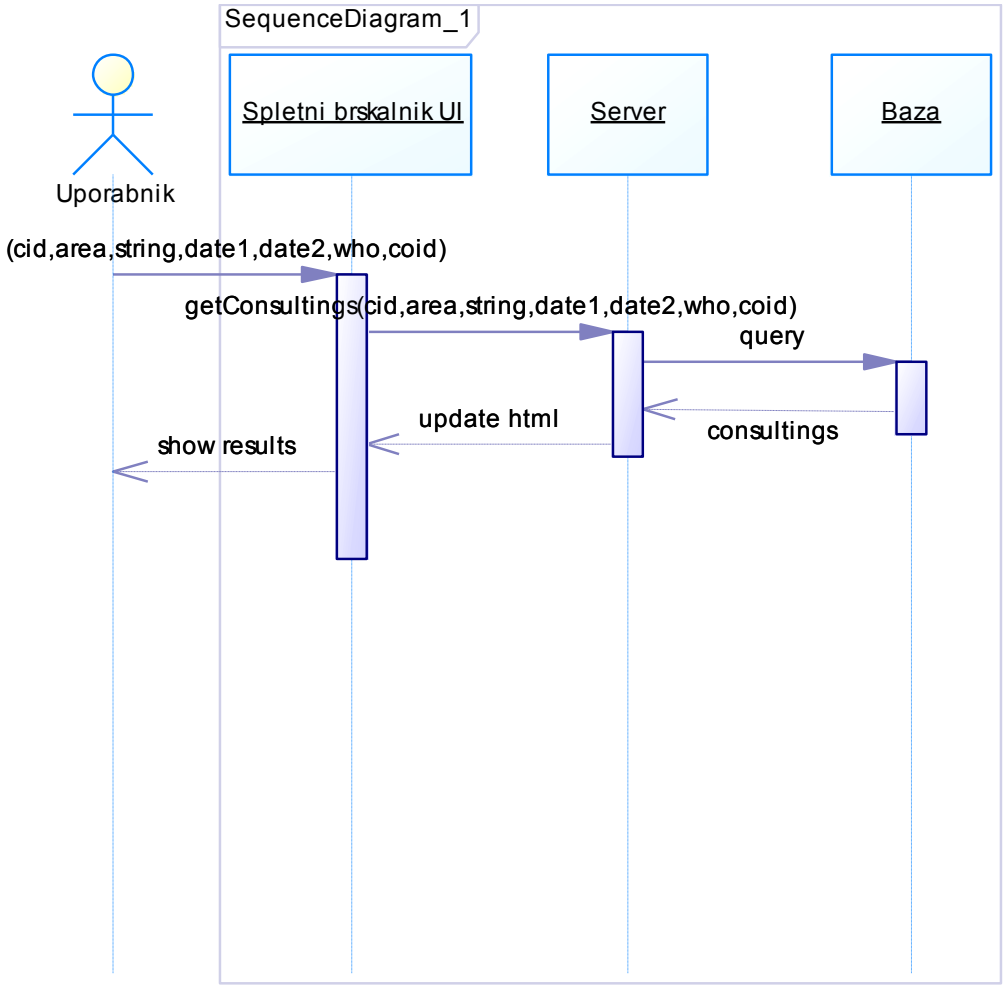


Slika 7: Diagram zaporedja za prijavo v sistem

Diagram zaporedja za iskanje po zgodovini je prikazan v Sliki 8. Pri iskanju po zgodovini uporabnik najprej poda podatke o samem iskanju, ki so:

- cid (naročniška številka stranke, katere svetovanja se išče),
- date1 ter date2 (časovni interval, na katerem išče svetovanja),
- string (če želi iskati določen besedni niz),
- area (kje želi iskati vpisani besedni niz),
- who (ali išče samo svetovanja prijavljenega uporabnika ali od vseh),
- coid (poda, v kolikor pozna številko svetovanja, katero bi rad videl).

Nato server sestavi poizvedbo ter dobi povratno informacijo od baze. Rezultate vključi v html kodo, katero posreduje brskalniku. Zatem brskalnik rezultate prikaže uporabniku.



Slika 8: Diagram zaporedja iskanja po zgodovini

6.3.3. Podatkovni model

Podatkovni model je zbirka konceptov, ki je namenjena opisu podatkov in manipulaciji z njimi. Opisan je z entitetami, njihovimi atributi ter ključi in relacijami med entitetami. Je eden izmed najpomembnejših delov analize, saj predstavlja vse podatke o področju, kjer se podatki spremljajo, obdelujejo in hranijo.

Najbolj znani podatkovni modeli so hierarhični, mrežni, relacijski, entitetno-relacijski, objektno-relacijski in objektno orientirani.

6.3.3.1. Konceptualni podatkovni modeli

Podatkovni model je zbirka konceptov, uporabljena za opis podatkov in operacij nad njimi.

Podatkovni model, ki opisuje množico konceptov iz dane realnosti, imenujemo konceptualni podatkovni model.

Sestavni deli konceptualnega modela so:

- entitetni tipi (posamezne instance tipov objektov iz poslovne domene),
- atributi (totalni, parcialni, enovrednostni, večvrednostni),
- razmerje (entitete so med seboj povezane z razmerji, ki imajo določen pomen),
- enolični identifikator entitete.

Lastnosti konceptualnega modela so izraznost, preprostost, minimalnost, formalnost, grafična popolnost in berljivost.

Konceptualni podatkovni model se v današnji dobi uporablja pri praktično vsakem resnem projektu, saj nam pametna izvedba samega modela omogoča lažje in predvsem pravočasno odpravljanje napak. Posledično so projekti končani hitreje, učinkoviteje ter je sama aplikacija bolj skalabilna. Med samo izdelavo konceptualnega modela ali med poskusom pretvorbe v fizičnega nas večina orodij (npr. Power Designer) opozori na napake, kar prepreči napačno predstavitev baze in posledično kasnejše popravljanje same strukture baze, kar ponavadi precej zavleče projekt.

Konceptualni model aplikacije, ki je prikazan na Sliki 9, vsebuje osem med seboj povezanih entitet, in sicer:

class – vsebuje enolični identifikator CLID ter atributa CLName (predstavlja ime teme) in CLCategory (predstavlja kategorijo teme);

subclass – poleg enoličnega identifikatorja SCLID vsebuje tudi tuji ključ CLID, s katerim je določena podtema, ki označuje, kateri temi pripada. Entiteta vsebuje tudi atribut SCLName, ta pa nam pove ime določene podteme;

consulting – glavna entiteta aplikacije, ki vsebuje informacije o svetovanjih, ima primarni ključ COID ter atribute:

- COType (številčna predstavitev tipa svetovanja),
- CODate (datum kreiranja svetovanja),
- COTimeSum (skupen čas svetovanja),
- NrOfQuestions (pri telefonskih svetovanjih nam pove, koliko je vseh vprašanj, pri pisnih in osebnih pa, na koliko vprašanj morajo svetovalci še odgovoriti),
- IsClosed (podatek o pravilnem končanju svetovanja),
- QuestionSender (podatek o pošiljatelju),
- mailArrived (datum dospelosti pisma/maila),
- note (opomba svetovalcu),

- visitDate (datum osebnega obiska pri svetovalcu).

Poleg tega ima tudi dva tuja ključa, in sicer UID, ki predstavlja svetovalca, ter CID, kateri nam pove naročnika, kateremu se mu svetuje.

consultingdetails – entiteta hrani podatke o vprašanjih določenega svetovanja. Vsebuje primarni ključ CDID, ki unikatno določa vprašanje. Tuji ključ COID določa svetovanje, h kateremu spada določeno vprašanje, UID določa svetovalca tega vprašanja, CLID pa temo svetovanja. Poleg tega ima entiteta še naslednje atribute:

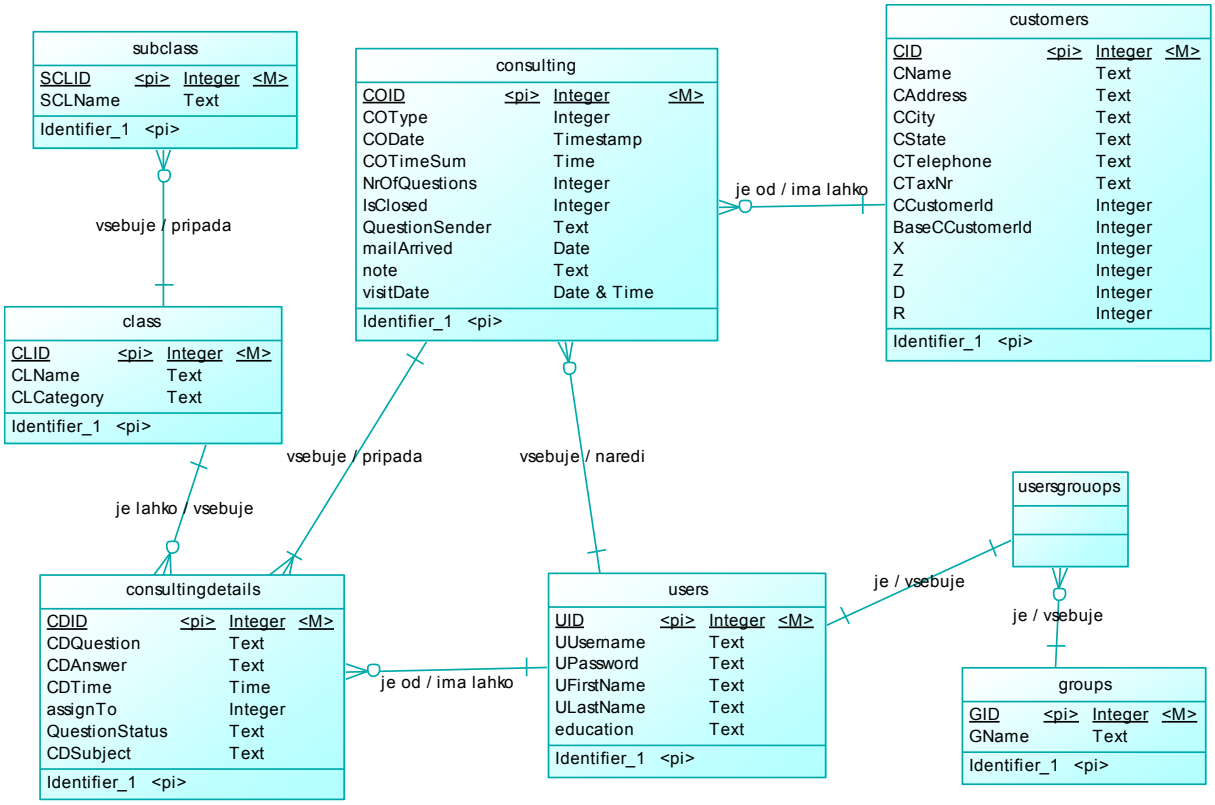
- CDQuestion (hrani besedilo vprašanja),
- CDAnswer (hrani besedilo odgovora),
- CDTime (podatek o času vprašanja),
- assignTo (privzeta vrednost je 0, pri pisnih in osebnih pa ima vrednost UID svetovalca, kateremu je vprašanje dodeljeno),
- QuestionStatus (pove ali je vprašanje nedotaknjeno, shranjeno oziroma odgovorjeno),
- CDSubject (zadeva pri pisnem/osebnem svetovanju);

customers – hrani podatke o naročnikih. Vsebuje primarni ključ CID, ki predstavlja zaporedno številko svetovanja. Atributi, ki predstavljajo posameznega naročnika, so: CName, CAddress, CCity, CState, CTelephone, CTaxNr, CCustomerId, BaseCCustomerId, X, Z, D, R;

users – vsebuje podatke o uporabnikih. UID enolično določa uporabnika, poleg tega entiteta vsebuje še atribute UUsername (uporabniško ime), UPassword (kriptiran password), UFirstName (ime), ULastName (priimek), education (naziv);

groups – hrani različne tipe uporabnika. GID enolično določa tip uporabnika, poleg tega ima tudi atribut GName, ki vsebuje ime tipa;

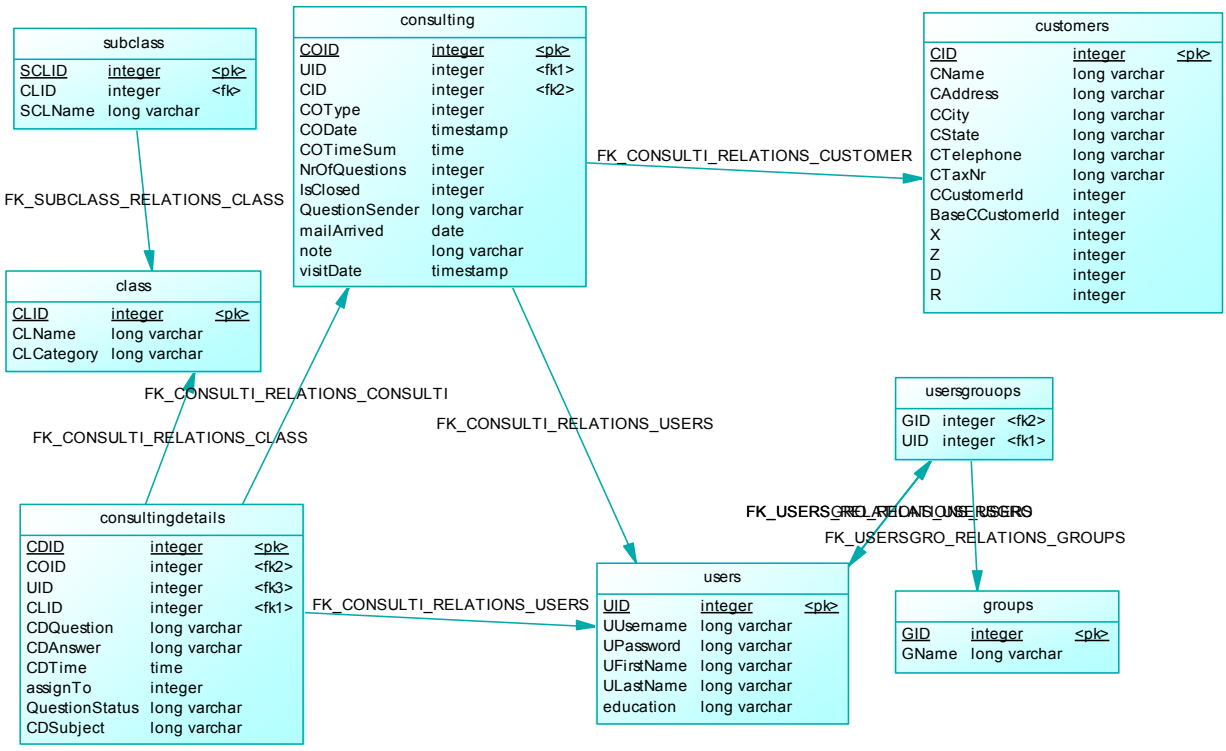
usersgroups – vsebuje samo dva tuja ključa, in sicer GID ter UID. Skupaj nam podata informacijo za vsakega uporabnika, katerega tipa je.



Slika 9: Konceptualni podatkovni model aplikacije

6.3.3.2. Fizični podatkovni model

Fizični podatkovni model je prikaz dejanske baze podatkov, ki je lahko izvedena z različnimi programskimi orodji. Model se od logičnega modela razlikuje po tem, da z njim prikazanim atributom določimo ustrezen tip vsakega atributa. Tipe atributov, ki kažejo, kakšnega tipa podatkov bodo polja v bazi, moramo pravilno določiti, sicer informacijski sistem ne bo pravilno deloval. Fizični model aplikacije (Slika 10) sem s pomočjo PowerDesignerja generiral iz konceptualnega. Poleg vsega, kar vsebuje konceptualni podatkovni model, vsebuje fizični, kjer so prikazani tudi tuji ključi, kar poenostavi predstavitev strukture baze aplikacije.



Slika 10: Fizični podatkovni model

6.4. Razvoj in implementacija modulov

Pri izdelavi modulov je potrebno biti pozoren na sam vrstni red izdelave modulov, saj so nekateri lahko odvisni od drugih in posledično jih ni mogoče testirati, zato morajo biti izdelani kasneje. Tako sem moral pred začetkom razvoja modulov določiti prioritete modulom. Prišel sem do ugotovitve, da je potrebno najprej za pravilno delovanje aplikacije izdelati modul za prenos naročnikov (iz Navision baze v MySQL bazo), nato modul poštna knjiga in zatem še preostale.

V nadaljevanju bom predstavil tri module, in sicer modul za prenos naročnikov, modul poštna knjiga in modul za pregled zgodovine.

6.4.1. Modul za prenos naročnikov

Za posodobitev MySQL baze skrbi skripta checkNarocnik.py. Skripta opazuje datum spremembe datoteke, imenovane narocniki.txt, ki se nahaja na strežniku. V primeru, da so se zgodile spremembe na datoteki, pomeni, da je prišlo do sprememb v bazi in je potrebno našo MySQL bazo posodobiti z novimi podatki. Za to poskrbi fukncija startDBSync(), ki se nahaja v datoteki convert.py. Zaradi različnega kodiranja znakov datoteke in MySQL baze skripta najprej kreira novo datoteko, ki je identična datoteki narocniki.txt, le da so podatki zapisani v

encodingu utf 8, katerega uporablja tudi MySQL baza. Po ustvarjeni datoteki skripta kliče funkcijo update(), ki se nahaja v datoteki DBSync.py. Funkcija najprej prebere kreirano datoteko in shrani podatke v slovar, imenovan navDict. Ključi elementov so ID-ji strank, ostali podatki so pa v seznamu, ki je vrednost tega ključa. Nato na isti način pridobi še stranke iz MySQL baze in jih shrani v slovar, imenovan mysqlDict. Ko ima vse podatke na voljo, mora skripta ugotoviti razlike med slovarjema. Ta del je spodaj na kratko opisan.

Podatkovna struktura slovar bo nosila podatke o strankah. Za uporabo slovarja sem se odločil, ker ID-ji strank niso bili zaporedne številke in bi v primeru uporabe seznama imel na nekaterih indeksih prazne vrednosti.

Slovar:

Slovar v programskem jeziku python je zbirka neurejenih vrednosti, do katerih lahko dostopamo preko ključa. Ključi v zbirki morajo biti različni. Časovna kompleksnost dostopa do vrednosti preko ključa je v povprečju O(1).

Za uporabo slovarja sem se odločil, ker ID-ji niso bili zaporedne številke in bi v primeru uporabe seznama imel na nekaterih indeksih prazne vrednosti.

V kolikor skripta checkNarocnik.py ne more dostopati do strežnika, nam pošlje sporočilo po elektronski pošti.

Ugotavljanje razlik med slovarjema navDict in mysqlDict:

- **Dodajanje novih strank v bazo**

Spodaj dodani seznam »zaDodati« generiram s tako imenovanim »list comprehension«, kjer je vsak element seznama svoj seznam s podatki, ki bodo shranjeni v bazo. Kar naredi, je to, da se sprehodi čez vsak element slovarja navDict, in v kolikor slovar mysqlDict nima istega ključa, kot je ključ trenutnega elementa, in ključa ni v seznamu dontAdd, je le-ta dodan.

```
zaDodati = [v for k,v in navDict.items() if not mysqlDict.has_key(k) and k not in dontAdd]
```

- **Brisanje starih strank**

Na podoben način generiram seznam »zaZbrisati«, ki vsebuje elemente, katere je potrebno izbrisati. Za razliko od seznama zaDodati so tukaj elementi seznama cela števila, ki predstavljajo ID stranke, katero je potrebno izbrisati iz MySQL baze.

```
zaZbrisati = [v[1] for k,v in mysqlDict.items() if not navDict.has_key(k) ]
```

- **Posodabljanje obstoječih strank**

Za posodobitev podatkov obstoječih strank generiram še seznam »zaUpdate«, kjer je vsak element seznama svoj seznam z novimi podatki. Najprej se sprehodim čez elemente slovarja navDict. V primeru, da slovar mysqlDict ima isti ključ, kot je ključ trenutnega elementa, in da je vrednost ključa v slovarju mysqlDict različna od vrednosti istega ključa v slovarju navDict, je vrednost trenutnega ključa dodana v seznam.

```
zaUpdate = [v for k,v in navDict.items() if mysqlDict.has_key(k) and mysqlDict[k] != navDict[k] ]
```

Ko generiram vse potrebne sezname, se skripta sprehodi čez elemente vsakega seznama in izvede potreben SQL stavek (na primer vsak element iz seznama zaDodat doda v MySQL bazo). Ko skripta konča z delom, imata bazi enake vnose in skripta ponovno čaka na spremembe datoteke narocniki.txt.

Ker strežnik deluje na operacijskem sistemu Windows in skripta nima interakcije z uporabnikom, sem se odločil, da bo skripta Windows service. Definiral sem objekt, kateri je imel lastnosti Windows servica. Za delovanje je bilo potrebno definirati funkcijo start, katera se kliče ob zagonu servica in znotraj funkcije klicati že obstoječe procedure. Objekt Service je imel že sam po sebi možnost pisanja logov, zato sem to uporabil za lažji nadzor dogajanja.

Windows service:

Windows service je dolgotrajen program, ki opravlja posamezne nalogo in ne zahteva posredovanja uporabnika. Lahko se ga konfigurira tako, da se zažene ob zagonu operacijskega sistema Windows in teče v ozadju, dokler teče operacijski sistem. Windows service je mogoče tudi ročno ustaviti ali zagnati tako, da v Run ukazu vpišete “services.msc” in nato na želenem servicu kliknete Start oziroma Stop.

6.4.2. Modul poštna knjiga

Na osnovni strani lahko uporabnik poštne knjige vidi svoja obvestila ter graf, ki prikazuje statistiko uporabnika zadnjega meseca.

Modul je namenjen uporabniku poštne knjige, kateri nadzoruje vsa pisna in osebna svetovanja. Njegovi glavni funkciji sta dodeljevanje vprašanj pisnih ter osebnih svetovanj in zaključevanje teh svetovanj.

Obvestila

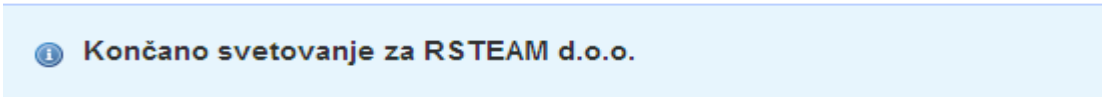
Pod obvestili ima uporabnik poštne knjige vpogled v končana svetovanja (to so svetovanja, ki jih mora še zaključiti), svetovanja, ki se niso pravilno zaključila ter status njegovih nedokončanih svetovanj. Poleg tega ima možnost pregleda statistike, ki je prikazana z različnimi grafi.

Za prikazovanje vseh vrst obvestil smo uporabili javascript knjižnico JQuery. Knjižnica vsebuje zbirko funkcij, katere omogočajo hiter razvoj aplikacije z zelo malo kode in truda. Ker JQuery temelji na Ajax tehnologiji, uporabniku ni potrebno osvežiti strani, da vidi nova obvestila.

Pod obvestili se uporabniku prikazujejo končana svetovanja in tabela nedokončanih svetovanj.

Končana svetovanja

Ko so vsa dodeljena vprašanja nekega svetovanja odgovorjena, se prikaže modro obvestilo (Slika 11), ki uporabnika opozori na končano svetovanje, njegovo stanje pa se odstrani iz tabele.

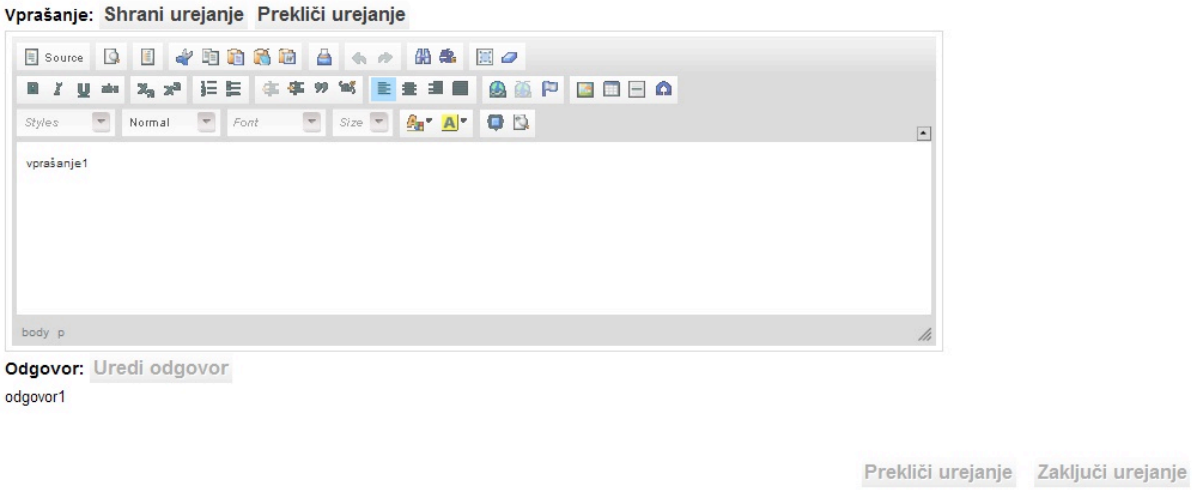


Slika 11: Obvestilo o končanem svetovanju

Ob kliku na želeno svetovanje se uporabniku odpre stran z informacijami o izbranem svetovanju, kjer lahko uporabnik ureja vprašanja in odgovore svetovanja, zahteva predogled svetovanja v Wordu in zaključi oziroma prekliče svetovanje.

- **Urejanje vprašanj in odgovorov**

Ob kliku na gumb "uredi" uporabnika poštne knjige preusmeri na stran za urejanje, kjer ima na voljo dva gumba, in sicer "Uredi vprašanje" in "Uredi odgovor". Ob kliku na enega izmed njiju se mu odpre urejevalnik (podoben Wordu), nad njim pa se pojavita gumba "Shrani urejanje" in "Prekliči urejanje" (Slika 12). Zaradi varnosti so takrat ostale povezave (izjema so iskalnik Google ter povezave na dajatve, predpise in IKS) onemogočene. Gumb "Shrani urejanje" je namenjen shranjevanju sprememb v besedilu. V kolikor pa sprememb, ki jih je uporabnik poštne knjige naredil v urejevalniku, ne želi shraniti, klikne na gumb »Prekliči urejanje«. Ko uredi vsa vprašanja in odgovore, ima spodaj še dva gumba, in sicer "Prekliči urejanje" in "Zaključi urejanje". Če sprememb ne želi shraniti dokončno v bazo oziroma želi zavreči spremembe, klikne na gumb "Prekliči urejanje", v nasprotnem primeru pa na gumb "Zaključi urejanje".



Slika 12: Urejanje izbranega vprašanja

- **Predogled v Wordu**

Na strani s podatki o zaključenem svetovanju ima svetovalec možnost tudi predogleda svetovanja v Word dokumentu. Ob izbiri te opcije se kliče funkcija generateDocument, ki se nahaja v datoteki generateWord.php. Funkcija dobi vse podatke o svetovanju v tabeli, ki jo dobi kot parameter. Nato programsko ugotovi število svetovalcev in temu primerno dodeli podpis dokumenta (npr. pripravil/pripravila/pripravili). Za tem se kliče funkcija generateDocBody, ki generira kodo vseh odgovorov, ločenih z novo vrstico. Na koncu doda še imena in nazive svetovalcev, ki so sodelovali pri svetovanju. V primeru, da je eden, je potrebno izpisati njegovo šifro (prva črka imena in priimka), za kar poskrbi funkcija getInitials. Na koncu se koda združi, dodajo se stili ter »headerji«, potrebni za generiranje Word dokumenta, in to se zapiše v datoteko s končnico ».doc«, ki se nahaja na strežniku. Aplikacija ponudi uporabniku možnost prenosa te datoteke. Generiranje Word dokumenta z vsemi podatki o svetovanju je zelo dobra funkcionalnost, saj uporabniku poštne knjige prihrani ogromno časa ter hkrati zagotavlja pravilnost vnesenih podatkov. Primer generiranega Word dokumenta prikazuje Slika 13.

RSTEAM d.o.o.

Linhartova 2A

1234 Mengeš

542/2011

4. 3. 2011

Na vaše vprašanje, postavljeno v dopisu z dne 31. 1. 2011, vam sporočamo svoje mnenje.

Zadeva: zadeva1

odgovor1

Zadeva: zadeva2

odgovor2

Lep pozdrav!

Pripravil:

samostojni strokovni svetovalec

Direktor strokovne službe Zveze:

Slika 13: Generiran Word dokument končanega svetovanja

- **Zaključek svetovanja in preklic**

S klikom na modro obvestilo o končanem svetovanju se uporabniku poštne knjige odpre stran z vsemi informacijami izbranega svetovanja. Tu ima možnost urejanja vprašanj in odgovorov, ki so jih svetovalci vnesli. Poleg tega ima uporabnik možnost generiranja Word dokumenta tega svetovanja. Ob kliku na "Zaključi svetovanje" se odstrani modro okence (obvestilo), izračunajo se točke, informacije se pošljejo v datoteko, katero se lahko uvozi v Navision ter označi v MySQL bazi kot končano. S klikom na "Prekliči svetovanje" se ta ne zaključi, ampak ostane še vedno dosegljiv za urejanje v modrem okencu (obvestilu) na osnovni strani.

Tabela nedokončanih svetovanj

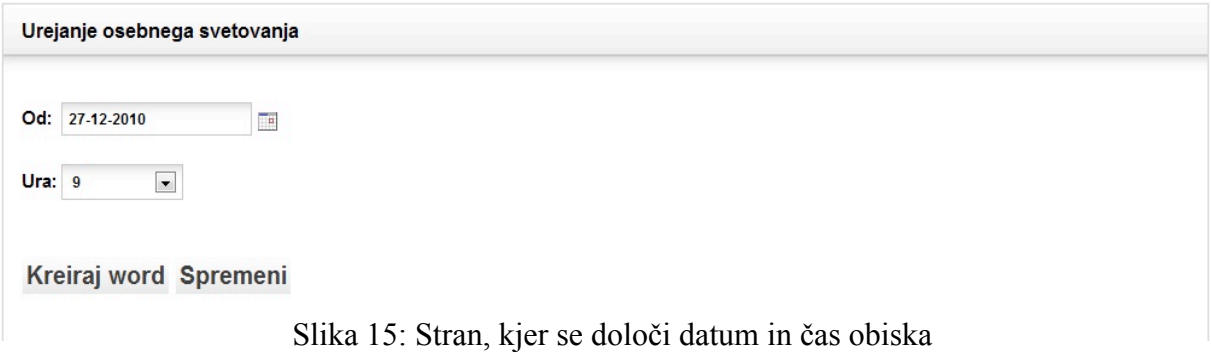
Tabela prikazuje stanje svetovanj, pri katerih še niso bila vsa vprašanja odgovorjena (Slika 14). Stanje vsakega takega svetovanja je opisano v svoji vrstici, kjer nam številke povedo, na koliko vprašanj mora svetovalec še odgovoriti.

V primeru, da oklepaja ni, mora svetovalec odgovoriti na eno vprašanje. Pri osebnem svetovanju ima uporabnik poštne knjige povezavo do določanja časa obiska stranke za določeno svetovanje. S klikom na povezavo »čas obiska« se uporabniku odpre stran (Slika 15), kjer lahko za izbrano osebno svetovanje spremeni datum in uro osebnega svetovanja.

Št. naročnika	Ime naročnika	Tip svetovanja	Datum začetka svetovanja	Št. vseh vprašanj	Št. odg. vprašanj	Št. neodg. vprašanj	V čakanju na
53782	RSTEAM d.o.o.	osebno čas obiska	27.12.2010 13:24	1	0	1	Primož Kariž
53782	RSTEAM d.o.o.	pisno	27.12.2010 09:28	1	0	1	Primož Kariž
53782	RSTEAM d.o.o.	osebno čas obiska	21.12.2010 12:27	1	0	1	Primož Kariž
53782	RSTEAM d.o.o.	osebno čas obiska	15.12.2010 09:02	1	0	1	Primož Kariž
01016	TERME ČATEŽ,d.d.	osebno čas obiska	14.12.2010 08:24	1	0	1	Janez Novak

Slika 14: Primer tabele nedokončanih svetovanj

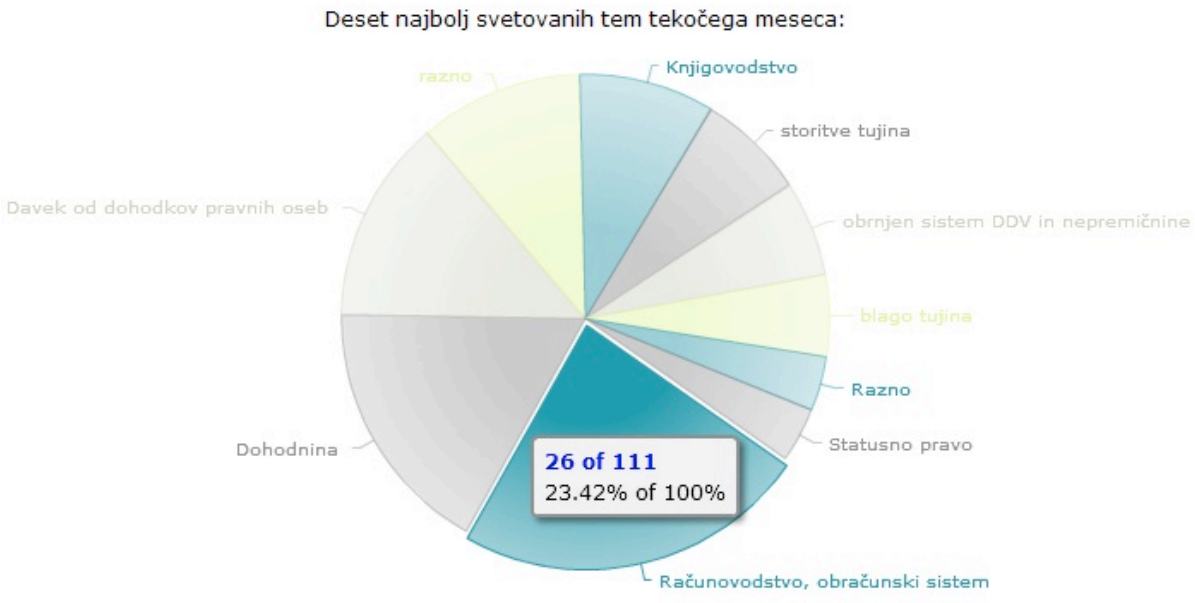
Ko je uporabnik poštne knjige na strani za določitev časa obiska določenega podjetja, izbere datum obiska ter uro obiska. Če pozabi te izbrati, ga na to opozori program. Ob kliku na gumb »Kreiraj Word« se ustvari Word dokument za osebno svetovanje, katerega lahko odpre ter natisne. V primeru, da želi samo shraniti spremembe, klikne gumb »Spremeni«.



Slika 15: Stran, kjer se določi datum in čas obiska

Grafi

Uporabnik poštne knjige ima dostop do štirih vrst grafov. Do **grafa svetovalcev**, kjer lahko uporabnik pogleda grafe svetovanj vseh svetovalcev za tekoči mesec. Vsak graf je predstavljen s krivuljo, ki predstavlja število svetovanj po dnevih. **Graf tipov svetovanj** predstavlja statistiko telefonskih, pisnih ter osebnih svetovanj za tekoči mesec s krivuljo za posamezne dni. Do **grafa področij svetovanj**, ki ima obliko histograma, katerega horizontalna os predstavlja teme, vertikalna pa število vprašanj. Pri **grafu desetih najpogostejših področij svetovanj** pa ima uporabnik možnost ogleda tortnega grafa, kjer se lahko vidi statistiko desetih najpogostejših področij svetovanj (Slika 16). Za kreiranje grafov smo uporabili PHP knjižnico open flash chart 2. Knjižnica je lahka za uporabo in omogoča uporabniku prikazovanje podatkov v obliki različnih grafov.



Slika 16: Graf desetih najpogostejših področij svetovanj

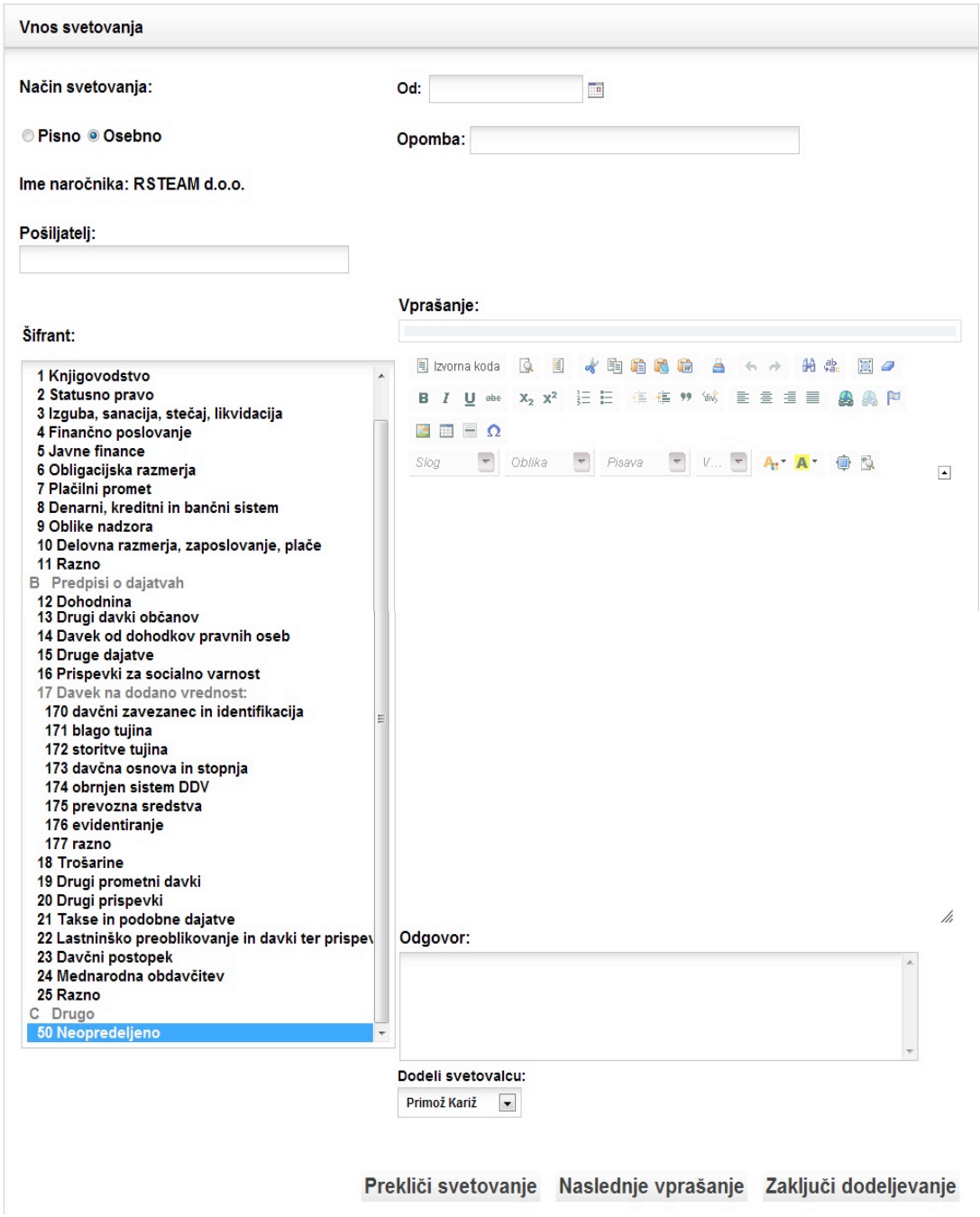
Kreiranje svetovanja in dodeljevanje vprašanj svetovalcem

Uporabnik poštne knjige ima možnost kreiranja svetovanja za stranko, pri čemer njeno naročniško stanje ni pomembno. Uporabnik mora najprej stranko poiskati. V primeru več zadetkov lahko uporabnik poštne knjige s klikom na želeno stranko ustvari svetovanje. Nato uporabnika poštne knjige preusmeri na stran (Slika 17), kjer lahko dodeljuje vprašanja svetovalcem. Za dodelitev vprašanja je potrebno:

- izbrati tip svetovanja, ki je lahko pisno ali osebno (se izbere samo pri prvem vprašanju svetovanja),
- izbrati datum prispelosti e-pošte/pošte (se izbere samo pri prvem vprašanju),
- vpisati pošiljatelja (samo pri prvem vprašanju),
- vpisati opombo (samo pri prvem vprašanju, polje ni obvezno),
- vpisati vprašanje (vnos v urejevalnik CKEditor),
- izbrati svetovalca, kateremu se vprašanje dodeli.

Če želi, lahko uporabnik poštne knjige izbere tudi šifrant. V primeru, da ga ne izbere, je šifrant neopredeljen in ga mora izbrati svetovalec. S klikom na gumb »Prekliči svetovanje« svetovalec prekliče svetovanje, s klikom na gumb »Zaključi svetovanje« ga zaključi, s klikom na gumb »Naslednje vprašanje« odpre stran za novo vprašanje tega svetovanja (na novi strani se uporabniku poštne knjige izpiše podatek, komu je bilo prejšnje vprašanje dodeljeno).

Urejevalnik besedila CKEditor je eden izmed najboljših urejevalnikov, katere se uporablja znotraj spletne strani. Gre za WYSIWYG (what you see is what you get) urejevalnik, kar pomeni, da je izgled vnosa enak izgledu dokumenta. Zaradi velike podobnosti namiznemu urejevalniku Microsoft Word uporabniki načeloma nimajo težav z učenjem uporabe urejevalnika.



Slika 17: Primer novega svetovanja za naročnika RSTEAM, d. o. o.

6.4.3. Modul za pregled zgodovine

Splošno iskanje

Za pregled celotne zgodovine lahko uporabnik poštne knjige v levem meniju klikne na gumb z napisom »Zgodovina«. Ko to stori, ga preusmeri na stran (Slika 18), kjer lahko vpiše šifro naročnika oziroma ime naročnika. V primeru več zadetkov izbere želenega, določiti mora obdobje, v katerem so iskana svetovanja kreirana, in označiti, ali naj išče samo njegova svetovanja oz. od vseh.

Obdobje določi s klikom na ikono  ter izbere datum. Uporabnik ima tudi možnost iskanja določene besede ali besedne zveze. To stori tako, da jo vpiše v polje »Iskani niz« ter izbere področje iskanja (vprašanja, odgovori, teme, vse). Na koncu je potrebno klikniti gumb »Išči«, ki se nahaja desno od datumov. V primeru, da uporabnik pozabi izpolniti eno izmed obveznih polj ali pa vnese napačen vnos datumov, ga aplikacija na to opozori.

Obvezna polja sta le:

- začetni datum in
- končni datum.

Pseudo koda za splošno iskanje

```
on form_submit:
 //server-side validacija
 if not validFields():
 print "napacen vnos"
 else:
        //funkcija getHistoryResultsSql vrne niz, ki predstavlja sql stavek za podane parametre
 sql = getHistoryResultsSql(cCName, whichUsers, date1, date2, searchedString, area)
 if mysql_nrOfRows(sql) == 0:
 print "ni zadetkov"
```

javascript pseudo:

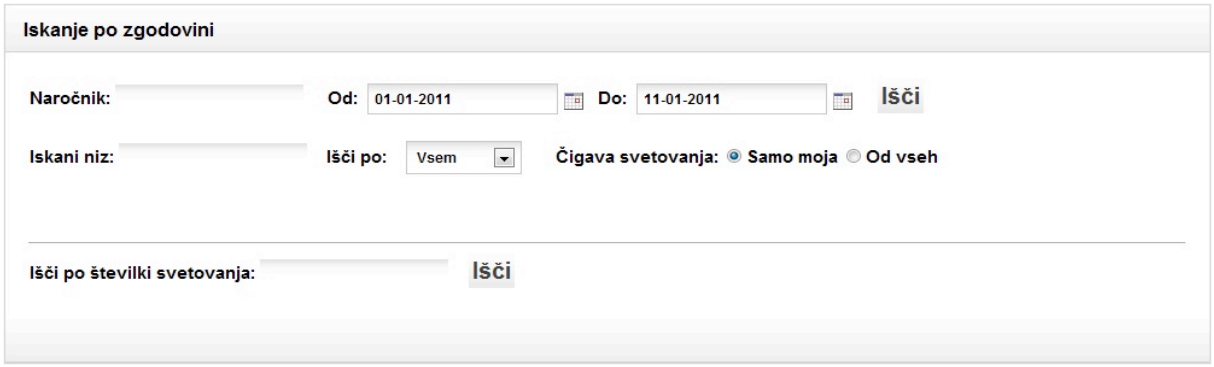
```
if on historyResultsPage:
 //kliče php funkcijo ki vrne celoten html za prikaz zadetkov
 html = callAjaxToGetResults("historyResults",searchString,page_num,order,sorting)
 //prikaže rezultate v določen div, skupaj s pagination-om
 displayResults(html)
```

Funkcija validFields preveri, ali so bili vsi potrebni podatki vneseni, in če so bili, preveri tudi pravilnost podatkov. Če funkcija vrne vrednost true, se kliče funkcija getHistoryResultsSql, ki na podlagi parametrov, katere prejme, generira SQL stavek za pridobitev zadetkov, kateri ustrezajo vnesenim podatkom.

Ko se stran naloži in javascript izvede klic funkcije –»document.ready«, se v primeru, da se uporabnik nahaja na strani historyResults.php, kliče funkcija callAjaxToGetResults. Funkcija izvede klic php skripte s potrebnimi podatki, ta pa najprej generira sql stavek za pridobitev želenih zadetkov ter ga nato še izvede. Na podlagi pridobljenih rezultatov php skripta generira html in ga vrne v obliki niza. Za prikaz rezultatov poskrbi funkcija displayResults, katera s pomočjo knjižnice JQuery vstavi html na uporabnikovo stran.

Iskanje po številki svetovanja

Uporabnik poštne knjige ima tudi možnost iskanja svetovanja po številki svetovanja. To iskanje je ločeno od splošnega iskanja s horizontalno črto. Za iskanje po številki svetovanja pride v poštev le polje »Išči po številki svetovanja«, ki je obvezno. Dovoljen je le vnos številke. Za iskanje je potrebno klikniti gumb, ki se nahaja desno od omenjenega polja. V primeru, da je vnos napačen, uporabnika na to opozori aplikacija z opozorilnim oknom, saj ima aplikacija v ozadju narejeno javascript validacijo in validacijo strežnika.



Slika 18: Stran, ki se odpre na klik gumba »Zgodovina«. Vodoravna črta ločuje 2 vrsti iskanja.

Validacija

Za zagotovitev pravilnega vnosa podatkov aplikacija vsebuje tako »client-side« validacijo kot tudi server-side.

- **Client-side validacija**

Validacija je narejena v jeziku javascript in je namenjena boljši in hitrejši interakciji med uporabnikom in spletno stranjo. V primeru, da uporabnik pozabi vnesti eno izmed zahtevanih polj, ga javascript opozori s pojavnim oknom. Prednost javascript validacije je, da preveri podatke, preden se pošlje zahteva na strežnik, saj javascript deluje na odjemalčevem računalniku. Prav to je tudi razlog, da take vrste validacija ni dovolj dobra, saj lahko odjemalec z lahkoto onemogoči javascript in tako ogrozi varnost podatkov.

- **Server-side validacija**

Najpomembnejša in obvezna validacija se zgodi na serverju v PHP skriptah. V trenutku, ko odjemalec pošlje vnesene podatke, skripta pregleda vsak vnos posebej in preveri pravilnost vnosa. V primeru, da vnos ni pravilen, se zaradi varnosti podatkov določene operacije ne smejo zgoditi in tako dobi odjemalec opozorilo, da je bil vnos napačen.

7. SKLEPNE UGOTOVITVE

Diplomsko delo se nanaša na načrtovanje in razvoj dela spletne aplikacije. Izdelava aplikacije je posledica želje podjetja, katero se je odločilo, da je njihova obstoječa aplikacija zastarela. Na podlagi podrobne analize aplikacije, ki jo je podjetje uporabljalo, smo v podjetju bili mnenja, da nam razvoj nove aplikacije, ki bi jo odlikovala večja funkcionalnost ter lahka uporaba, ne bi smel predstavljati velikega problema. Tekom razvoja smo naleteli na dve manjši težavi, in sicer s kreiranjem dveh instanc urejevalnika CKEditor na isti strani in generiranjem Microsoft Word dokumenta. Za rešitev težav sem si moral pomagati z literaturo, ki sem jo dobil na medmrežju.

Nekaj težav smo imeli z zahtevo naročnika aplikacije, da bi svetovalci pričeli novo verzijo aplikacije uporabljati predčasno, vendar nam je kljub stiski s časom uspelo uspešno zaključiti do roka. Tako svetovalci že uporabljajo nov program in so po prvih odzivih zelo zadovoljni z njim.

Nova aplikacija močno olajša uporabnikom delo ter prihrani veliko časa, kar je bil tudi glavni cilj aplikacije. Pogoj za uspešno izveden projekt je dobra komunikacija med uporabniki aplikacije in razvijalci ter pametno načrtovanje. Med izdelavo aplikacije sem pridobil nova znanja in izkušnje predvsem s programskim jezikom Python in razvojnim okoljem Aptana.

Prišli smo do tudi do nekaterih idej, katere bi lahko spletno aplikacijo še izboljšale, in sicer:

- izdelava grafa, ki prikazuje število svetovanj po mesecih,
- graf s krivuljo, ki prikazuje število naročnikov konec vsakega meseca,
- pri iskanju po zgodovini bi lahko dodali možnost izbire tipa svetovanja katerega naj išče,
- možnost spremembe svetovalca dodeljenemu vprašanju.

Z uporabniki aplikacije smo se dogovorili, da bomo po poteku določenega časa organizirali sestanek, kjer bi uporabniki izrazili želje po spremembah, ki so jih pridobili tekom koriščenja aplikacije. Tako bomo imeli ponovno možnost za nadgradnje, nove izzive in razvoj, kar pa je na računalniškem področju v velikem razcvetu.

8. PRILOGE

DODATEK A

Programska koda

checkNarocniki.py - skripta za opazovanje sprememb željene datoteke

```
from winservice import Service, instart
import time, os.path, ConfigParser, convert
import MySQLdb

class checkFolder(Service):
    def start(self):
        self.runflag=True
        self.log("I'm alive ...")
        cp = ConfigParser.RawConfigParser()
        cp.read(r'E:\\updateDB\\parse.ini')
        fileName = cp.get('path','customers');

        print "last modified: %s" % time.ctime(os.path.getmtime(fileName))

        lastModification = time.ctime(os.path.getmtime(fileName))

        while 1:
            if (time.ctime(os.path.getmtime(fileName)) !=
lastModification):
                self.log("I'm in 1st while ...")
                lastModification = time.ctime(os.path.getmtime(fileName))
                time.sleep(3)
                goOn = True
                while goOn:
                    try:
                        fTmp = open(fileName,'r')
                        goOn = False
                    except:
                        pass
                fTmp.close()
                lastModification = time.ctime(os.path.getmtime(fileName))
```

```
                self.log("I'm after 2nd while ...")
                conn = MySQLdb.connect (host = "localhost",user =
"root",passwd = "root",db = "svetovanje",use_unicode = False)

                conn.set_character_set('utf8')
                cursor = conn.cursor(MySQLdb.cursors.DictCursor)
                cursor.execute('SET NAMES utf8;')
                cursor.execute('SET CHARACTER SET utf8;')
                cursor.execute('SET
COLLATION_CONNECTION=utf8_slovenian_ci;')

                convert.startDBSync(self,cursor)
                cursor.close()
                conn.close()
                self.log("I'm done.")
    def stop(self):
        self.runflag=False
        self.log("I'm done")

instart(checkFolder, 'checkFolder', 'Check Navision Folder')
```

DBSync.py – skripta za sinhronizacijo podatkovnih baz

```
# -*- coding: utf-8 -*-
#Skripta, ki dobi .txt datoteko kot parameter, to je Navision baza s
sumniki, se pravi convertana in naredi potrebne update-e (update customer,
add customer, delete customer)
#Dictionary-ji so predstavljeni kot key => ccustomerId, value pa [int, int,
str, str, str, str, str, int, int, int, int], ki so podatki za bazo

import ConfigParser
#Functions
def checkIfInt(x):
    try:
        return int(x)
    except:
        return 0



def pretvoriNavVVrstico(navVrstica):
    #izgled navVrstice: 01009;01009;KOMUNALNO STANOVANJSKO;PODJETJE BREICE
D.D.,;Cesta prvih borcev
9;;Breice;;;;8250;;88063615;1;0;0;0;0;0;0;0;0;0;Ne;0;0;0;0;0;0;0;0;0;0
    navArr = navVrstica.split(";")
    #0,1,2+ " "+3, 4, 6, 10, 12, 13,14,15,16
    vrstica = [navArr[0] ] + [int(navArr[1])] + [navArr[i] for i in
range(0,len(navArr)) if i in [2,3,4,6,10,12,13,14,15,16] ]
```

```
    #Opomba: 2 in 3 -> zdruzi 2 in 3 s presledkom
    #zdruzim dva + preverim ce slucajno je x,z,r,d prazen ker majo cudno
bazo... ce je prazna vrednost dam 0, drugace castam v int in dam tisto
vrednost
    vrstica = [int(x) for x in vrstica[:2] ] + ['
'.join([vrstica[2],vrstica[3]]) ] + vrstica[4:8] + [checkIfInt(x) for x in
vrstica[-4:] ]
    return vrstica

#prebere customerje iz mysql baze in naredi dictionary (ccustomerId:
[elementi customerja vkljucno z ccustomerId]), ki ga tudi vrne
def narediMysqlDict(cursor):
    #pobere iz baze
    cursor.execute("""SELECT BaseCCustomerId, CCustomerId, CName, CAddress,
CCity, CState, CTaxNr, X, Z, D, R   FROM customers""")
    rows = cursor.fetchall()
    #naredi prazen dictionary kjer bo shranjeval podatke
    mysqlDict = {}
    for row in rows:
        tmp1 = list(row)
        #['Z', 'D', 'CState', 'CCity', 'R', 'CName', 'CTaxNr', 'CAddress',
'X', 'BaseCCustomerId', 'CCustomerId']
        mysqlDict[int(row["CCustomerId"])] = [ int(row['BaseCCustomerId']),
int(row["CCustomerId"]),
row["CName"],row["CAddress"],row["CCity"],row["CState"],row["CTaxNr"],int(r
ow["X"]),int(row["Z"]),int(row["D"]),int(row["R"]) ]
        #mysqlDict[int(row["CCustomerId"])] = [int(x) for x in tmp1[:2] ] +
tmp1[2:7] + [int(x) for x in tmp1[7:] ]
    return mysqlDict

#na podlagi list-a ki ga dobi, doda customerja v bazo
def addCustomer(vrstica, dontAdd1,cursor):
    #[1009, 1009, 'KOMUNALNO STANOVANJSKO PODJETJE BRE\xc2\xa6ICE D.D.,',
'Cesta prvih borcev 9', 'Bre\xc2\xa7ice', '8250', '88063615', 1, 0, 0, 0]
    baseId = vrstica[0]
    cCustomerId = vrstica[1]
    CName = vrstica[2]
    cAddress = vrstica[3]
    cCity = vrstica[4]
    cState = vrstica[5]
    cTaxNr = vrstica[6]
    x = vrstica[7]
    z = vrstica[8]
    d = vrstica[9]
    r = vrstica[10]
    if cCustomerId not in dontAdd1:
        cursor.execute("""INSERT INTO customers (CID,BaseCCustomerId,
CCustomerId, CName,CAddress,CCity,CState,CTaxNr,X,Z,D,R, CTelephone)
                                        VALUES (0, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s)""",
                        (baseId, cCustomerId, CName, cAddress, cCity,
cState, cTaxNr, x, z, d, r, '') )

#Kot parameter dobi ccustomerId, ki ga potem izbrise iz mysql baze
def removeCustomer(vrstica,cursor):
    cCustomerId = vrstica
    cursor.execute("""DELETE FROM customers WHERE CCustomerId = %s""",
(cCustomerId) )

#na podlagi lista ki ga dobi, update-a customerja v bazi
def updateCustomer(vrstica,cursor):
```

```
    cCustomerId = vrstica[1]
    #CName  CAddress    CCity   CState  CTelephone  CTaxNr  CCustomerId
BaseCCustomerId     X       Z       D       R
    cursor.execute("""UPDATE customers
                      SET BaseCCustomerId=%s, CCustomerId=%s, CName=%s,
CAddress=%s,  CCity=%s,CState=%s, CTaxNr=%s, X=%s, Z=%s, D=%s, R=%s
                      WHERE CCustomerId = %s""",

(vrstica[0],vrstica[1],vrstica[2],vrstica[3],vrstica[4],vrstica[5],vrstica[
6],vrstica[7],vrstica[8],vrstica[9],vrstica[10],cCustomerId))

def update(logObj,cursor):
    #DBConfig
    cp = ConfigParser.RawConfigParser()
    cp.read(r'E:\\xampp\\htdocs\\svetovanje\\updateDB\\path.ini')
    fname = cp.get('path','navDB')
    a = open(fname, "r").readlines()
    #narocniske stevilke, ki ne spadajo v mysql bazo in jih imajo v
navision bazi
    dontAdd = [0,1,2,6,7,8]

    #naredi dictionary za NAV bazo
    navDict = {}
    for x in a:
        if x[:5] != "AVANS" and x[:6] != "NEZNAN":
            tmp = pretvoriNavVVrstico(x)
            navDict[int(tmp[1])] = tmp

    #naredi dictionary za MYSQL bazo
    mysqlDict = narediMysqlDict(cursor)
    #ZA IZPIS VRSTIC OD OBEH
    """
    print "NAV:"
    for k,v in navDict.items():
        print "key: %s, value: %s, len = %d" % (k,v, len(v))

    print "MYSQL:"
    for k,v in mysqlDict.items():
        print "key: %s, value: %s, len = %d" % (k,v,len(v))

    """

    #dobi seznam katere moram dodat v mysql bazo
    zaDodat = [v for k,v in navDict.items() if not mysqlDict.has_key(k) and
k not in dontAdd]
    #dobi seznam ccustomerId-jev katere je treba izbrisat iz mysql baze
    zaZbrisat = [v[1] for k,v in mysqlDict.items() if not
navDict.has_key(k) ]
    #dobi seznam katere mora update-at v mysql bazi
    zaUpdate = [v for k,v in navDict.items()  if mysqlDict.has_key(k) and
mysqlDict[k] != navDict[k] ]
    #logObj.log("za monadami")

    print "-----------------ADD----------------------"
    print "Stevilo narocnikov za dodat: %d" % len(zaDodat)

    print "-----------------DELETE-------------------"
    print "Stevilo narocnikov za zbrisat: %d" % len(zaZbrisat)

    print "-----------------UPDATE-------------------"
    print "Stevilo narocnikov za update-at: %d" % len(zaUpdate)
```

```
print "-----------------PROGRESS---------------------"
#dodaj potrebne
for x in zaDodat:
    addCustomer(x, dontAdd,cursor)
print "Dodal narocnike v bazo..."
#zbrisi potrebne
for x in zaZbrisat:
    removeCustomer(x,cursor)

print "Izbrisal odvecne narocnike iz baze..."

#update-aj potrebne
for x in zaUpdate:
    updateCustomer(x,cursor)

print "Update-al narocnike."
print "finished DBSync"
```

9. KAZALO SLIK

10. LITERATURA IN VIRI

[1] Greenstein, Marilyn, Todd M. Feinman, Electronic Commerce: Security, Risk Management and Control, ISBN 978-0072292893

[2] HTML. Dostopno na:

http://en.wikipedia.org/wiki/HTML

[3] PHP. Dostopno na:

http://www.php.net

[4] JavaScript. Dostopno na:

http://sl.wikipedia.org/wiki/JavaScript

[5] Python. Dostopno na:

http://www.python.org

[6] Netbeans. Dostopno na:

http://www.netbeans.org

[7] Aptana studio. Dostopno na:

http://www.aptana.com

[8] MySQL. Dostopno na:

http://www.mysql.com

[9] PowerDesigner. Dostopno na:

http://www.sybase.com/products/modelingdevelopment/powerdesigner

[10] Notacija v programskem orodju PowerDesigner. Dostopno na:

http://www2.gov.si/mju/emris.nsf/0/37FC42B69C63A413C1256EF6003ACAC1?OpenDocument

[11] Objektno načrtovanje podatkovnih baz. Dostopno na:

http://www.drustvo-informatika.si/fileadmin/dsi2001/sekcija_a/lahajnar_rozanec.doc

[12] UML. Dostopno na:

http://en.wikipedia.org/wiki/Unified_Modeling_Language

[13] Modeliranje primerov uporabe. Dostopno na:

http://mitjab.homelinux.net:8080/~mitjab/Sola/VAJE/OIS/Modeliranje%20primerov%20uporabe%20(PU)/modeliranje_USE_CASE.pdf