UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Eldin Velagić

Razvoj informacijskega sistema za organizirano skupnost

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: 00001/2010
Datum: 01.10.2010

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

Kandidat: **ELDIN VELAGIĆ**

Naslov: **RAZVOJ INFORMACIJSKEGA SISTEMA ZA ORGANIZIRANO SKUPNOST**

INFORMATION SYSTEM FOR ORGANISED COMMUNITY

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte spletno aplikacijo, ki informacijsko podpira celotno delovanje organizirane skupnosti v tri-nivojski arhitekturi. Vaša naloga je razviti zgornji nivo (spletni vmesnik) in srednji nivo (poslovna logika).

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a ELDIN VELAGIĆ,

z vpisno številko 63060382,

sem avtor/-ica diplomskega dela z naslovom:

RAZVOJ INFORMACIJSKEGA SISTEMA ZA ORGANIZIRANO SKUPNOST

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 06.01.2011 Podpis avtorja/-ice: Velagić Eldin

Zahvala

Zahvaljujem se profesorju dr. Roku Rupniku za vodenje in vse nasvete pri izdelavi diplomske naloge. Zahvaljujem se tudi Islamski skupnosti v Republiki Sloveniji in muftiju dr. Nedžadu Grabusu, za projekt in odlično sodelovanje pri izdelavi informacijskega sistema. Še posebej pa se zahvaljujem svoji družini, ki mi je stala ob strani v času študija in dekletu za motivacijo in pomoč pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic in pojmov

ISVRS	Islamska skupnost v Republiki Sloveniji
MEŠIHAT	Najvišji verski ter administrativni organ Islamske skupnosti v Sloveniji
IT	Informacijska tehnologija
MVC	Model View Controler
ASP.NET	Active Server Pages ogrodja .NET
XML	Extensible Markup Language
AJAX	Asynchronous JavaScript and XML
JQUERY	Javascript knjižnica
JQUERY UI	Javascript knjižnica za izdelavo uporabniških vmesnikov
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
IPC	Interprocess Communication
MAC OS-X	Operacijski sistem narejen za računalnike Macintosh
PDF	Portable Document Format
API	Application Program Interface
W3C	World Wide Web Consortium
ERAN-UCC	European Article Numbering - Uniform Code Council
GIF	Graphic Interchange Format
PNG	Portable Network Graphics
JPEG	Joint Photographic Experts Group
ASCX	Active Server Custom Control

Povzetek

Diplomsko delo opisuje postopek razvoja informacijskega sistema za Islamsko skupnost v Republiki Sloveniji. Podrobneje zajema proces razvoja obličja spletne aplikacije skozi celoten proces razvoja informacijskega sistema, od faze zajema zahtev do razvoja in testiranja.

Za razvoj novega sistema so se v skupnosti odločili predvsem zato, da bi bolj povezali delovanje in pospešili ter izboljšali delovne procese znotraj skupnosti. Poglavitni cilj novega informacijskega sistema je bil priprava centralnega registra, ki bo predstavljal podatkovno bazo za vse odbore znotraj skupnosti. Podatki se bodo nahajali na enem mestu, zaposleni pa bodo, do podatkov dostopali iz svojih delovnih mest.

Spletna aplikacija je razvita z orodjem Microsoft Visual Studio 2010 z uporabo tehnologije ASP.NET (omogoča kreiranje dinamičnih spletnih strani), na osnovi arhitekture MVC. Diplomsko delo opisuje tudi glavna orodja in tehnologije, ki smo jih uporabljali pri izdelavi sistema. Razvoj samega obličja spletne aplikacije je opisan v štirih glavnih fazah. V prvi fazi smo zbrali zahteve naročnika, ki smo jih uporabili za zasnovo zaslonskih mask. V drugi fazi smo pripravili in oblikovali zaslonske maske uporabniškega vmesnika, v tretji fazi pa pripravili dialoge za vnos in urejanje podatkov. Zadnja, četrta faza, pa govori o postopkih testiranja uporabniškega vmesnika spletne aplikacije.

Ključne besede

spletna aplikacija, obličje sistema, uporabniški vmesnik, razvoj informacijskega sistema

Abstract

The thesis describes the process of developing an information system for the Islamic community in Slovenia. It more specifically covers the process of developing a front end of the web application, through the whole IT development process, covering the stages from requirements definition to development and testing.

The reasons that the community decided to develop a new system, ware primarily to better link the performance, speed up and improve work processes within the community. The main objective of the new information system was to prepare the central registry, which will represent the database for all boards within the community. Data will be located in one place; employees will be able to access these data from their office chairs.

The web application was developed in Microsoft Visual Studio 2010 using ASP.NET technology (enables us to create dynamic web pages) based on MVC architecture. The thesis also describes the main tools and technologies that were used in the development process. The development of the front end of the web application is described in four main phases. In the first phase, we collected the customer's requirements, which we then used for planning the user interface. In the second phase we designed and developed the user interface screen masks, in the third phase, we constructed the dialogues for entering and editing data. The last, fourth phase, talks about the testing procedures of the web applications user interface.

Key words

web application, front end of the system, user interface, information system development

1. Uvod

Danes živimo v dobi, kjer so informacije ključnega značaja, informatiko pa srečujemo na vseh področjih, tako zasebnih, kot poslovnih. Pomaga nam združevati podatke, procese, ljudi in okolja v nek celovit sistem, s čimer pa lažje in nedvomno hitreje dosežemo želene cilje.

Tako so se tudi vodilni v Islamski skupnosti v Republiki Sloveniji odločili, da svoje delovne procese, oziroma organizacijo, informatizirajo. Rodila se je ideja o razvoju informacijskega sistema, ki bo karseda dobro povezal organe islamske skupnosti v Sloveniji, njihove člane in zaposlenim omogočil vpogled v evidenco.

Islamska skupnost je razdeljena na posamezne odbore, kjer vsak odbor vodi svojo evidenco. Vodenje je do sedaj potekalo tako, da so se podatki najprej beležili ročno na papir in kasneje prepisovali v Excelovo tabelo. Excelove tabele iz posameznih odborov pa je nekdo moral ročno sinhronizirati v eno skupno tabelo. Takšen način vodenja evidence je seveda zelo zamuden in dolgotrajen proces, da ne govorimo o naporu, ki ga mora zaposleni vložiti, da, na primer, poišče neko poročilo iz leta 2005. Tako so za glavne cilje novega informacijskega sistema postavili: odpravo zamudnih in nepotrebnih procesov, implementacijo arhiva podatkov, vpogled in iskanje po arhivu, standardizacijo postopkov in izboljšanje povezanosti ter komunikacije med odbori in zaposlenimi.

Razvoj informacijskega sistema je potekal v dveh iteracijah, projektna skupina pa je bila razdeljena na dva dela. Prva podskupina je bila zadolžena za obličje sistema, medtem, ko je druga podskupina razvijala zaledje sistema.

V diplomski nalogi bom podrobneje predstavil problem, opisal razvoj obličja informacijskega sistema, ter predstavil orodja, tehnologije in metode, ki sem jih pri razvoju uporabljal.

2. Islamske skupnosti v Republiki Sloveniji

2.1. Predstavitev skupnosti

Islamska skupnost v Republiki Sloveniji je javna, enkratna in samostojna verska skupnost vseh prebivalcev Republike Slovenije, ki sprejemajo islam za svojo vero. Zavzema se za duhovnost in človekovo dostojanstvo v zasebnem in javnem življenju, prizadeva si za osmišljanje bivanja na področju verskega življenja in ima s svojim delovanjem hkrati tudi pomembno vlogo v javnem življenju. Z razvijanjem svojih kulturnih, vzgojnih, izobraževalnih, solidarnostnih, dobrodelnih in drugih dejavnosti s področja socialne države bogati nacionalno identiteto in s tem opravlja pomembno družbeno vlogo [1].

Islamska skupnosti v Republiki Sloveniji je splošno koristna, verska organizacija, ki svobodno in avtonomno uči svojo vero, opravlja verske obrede in posle, samostojno in svobodno oblikuje svojo organizacijsko strukturo, skrbi za versko in kulturno-izobraževalno dejavnost, rešuje finančna, administrativna, lastniška ter druga vprašanja, povezana z delovanjem Islamske skupnosti. Avtonomija in svoboda delovanja Islamske skupnosti sta zagotovljena s pravnim redom Republike Slovenije [1].

2.2. Organizacijska struktura

Islamsko skupnost sestavljajo odbori kot osnovne organizacijske enote. Odbor je v tradicionalni organizacijski formi Islamske skupnosti medžlis, a v organizaciji Islamske skupnosti v Sloveniji ustreza džematu. Odbor obsega najmanj 250 muslimanskih gospodinjstev na določenem področju, ki so med seboj povezani v izvrševanju skupnih islamskih dolžnosti. Odbor je pravna oseba zasebnega prava, vendar ne more delovati brez soglasja Mešihata Islamske skupnosti. Pravno osebnost pridobi z registracijo pri Uradu Vlade Republike Slovenije za verske skupnosti z zahtevo, ki jo vloži Mešihat [1].

Mešihat islamske skupnosti je izvršilni, kolegijski in najvišji verski ter administrativni organ Islamske skupnosti v Republiki Sloveniji [1].

V Sloveniji imamo 18 odborov, v večjih slovenskih mestih, kjer v praksi praviloma en odbor pokriva področje enega slovenskega mesta. Vsak odbor ima svoje člane in svoje zaposlene. Član odbora je lahko vsaka polnoletna oseba, ki sprejema islam kot svoje versko prepričanje. Praviloma se včlani v odbor, ki teritorialno pokriva območje njegovega prebivališča in s tem postane aktivni član. Aktivni član Islamske skupnosti je vsaka oseba, ki redno poravnava svoje materialne obveznosti oziroma redno plačuje članarino. S plačilom članarine pa postanejo člani Islamske skupnosti tudi vsi člani družine in skupaj se v evidenci vodijo kot gospodinjstvo.

3. Opredelitev problema

3.1. Predstavitev problema

Pred uvedbo novega informacijskega sistema je evidentiranje in beleženje potekalo ročno v posameznih odborih. Zaposlene osebe v posameznem odboru so evidenco vodile, s pisanjem v za to posebej namenjene zvezke, kasneje pa so se podatki prepisovali v Excelove tabele. V teh tabelah so videli rešitev za lažje iskanje in boljšo preglednost dela, saj so lahko z uporabo filtrov iskani zapis hitro našli.

Največji problem pri takšnem načinu vodenja je pomanjkanje centralnega nadzora. Predstavniki odborov so enkrat mesečno dostavili podatke v Mešihat, kjer je bilo potrebno, ponovno ročno, sinhronizirati podatke iz različnih odborov v centralni register. Ta centralni register pa je zopet predstavljala neka tabela, kamor je nekdo ročno vnašal podatke. Predstavnikom mešihata veliko pomeni, da lahko spremljajo, kaj se dogaja po odborih in kakšno je stanje nalog, ki jih zaposleni izvajajo. Tako lahko odbore usmerjajo k boljšemu delovanju, povečajo učinkovitost delovanja in okrepijo interakcijo s svojimi člani.

Kot drugi veliki problem se izpostavlja pomanjkanje enotnosti dela. Zaposleni v odboru Maribor, tako kot zaposleni v odboru Koper, vodijo evidenco svojih članov in imajo svoj popis gospodinjstev, pri čemer pa vsak evidenco vodi na svojstven način. V Mariboru, na primer, za potrditev vplačila ali račun uporabljajo paragonske bloke, medtem, ko so zaposleni v Kopru računalniško bolj vešči in za izdajo potrdil ali računov uporabljajo tiskalnik. To je le eden od številnih problemov pomanjkanja standardizacije postopkov, ki se pojavljajo znotraj skupnosti.

3.2. Cilji novega informacijskega sistema

Poglavitni cilj novega informacijskega sistema je priprava centralnega registra, ki bo predstavljal podatkovno bazo za vse odbore znotraj skupnosti. Podatki se bodo nahajali na enem mestu, zaposleni pa bodo do podatkov dostopali iz svojih delovnih mest. Pripraviti je potrebno arhiv podatkov in ga pravilno strukturirati ter indeksirati. Pomembno je, da so podatki vedno dostopni in da je iskanje po registru hitro in učinkovito. Ko bo predsednik mešihata želel videti dnevni pregled dogajanja v odboru Ljubljana, odboru Kranj ali pa npr. odboru Nova Gorica, mu za to ne bo potrebno klicati zaposlene, da mu dostavijo dnevno poročilo, ampak bo to storil preprosto z nekaj kliki.

Z novim informacijskim sistemom želijo doseči standardizacijo procesov znotraj skupnosti. Novi sistem mora vsebovati enotne elektronske obrazce za vnos podatkov za vse odbore. To pa še posebej velja za redna poročila in naloge, ki jih oddajajo ter izvajajo zaposleni. Člani skupnosti pa dobijo nove članske izkaznice, personalizirane z unikatno črtno kodo. Te izkaznice služijo za lažje in hitrejše identificiranje članov.

Cilj sistema je tudi skrajšanje oziroma odprava dolgotrajnih in nepotrebnih procesov. V kolikor želijo zaposleni v odboru Ljubljana svojim članom poslati neko obvestilo, morajo poiskati in prepisati naslove, pripraviti in kuvertirati pošiljke, ter jih odnesti na pošto v pošiljanje. Z novim sistemom želijo uvesti elektronsko obveščanje članov preko elektronske pošte ali sms sporočil. Mislim, da ni potrebno govoriti o času in denarju, ki se bo na ta način prihranil.

Izboljšati je potrebno tudi komunikacijo in obveščanje med zaposlenimi. Sistem mora omogočati neposredno izmenjavo sporočil oziroma dokumentov med zaposlenimi, še posebej pa na relaciji nadrejeni – podrejeni.

3.3. Glavni funkcionalni sklopi

Kot glavne funkcionalne sklope, ki jih bo sistem vseboval, so postavili sledeče zahteve:

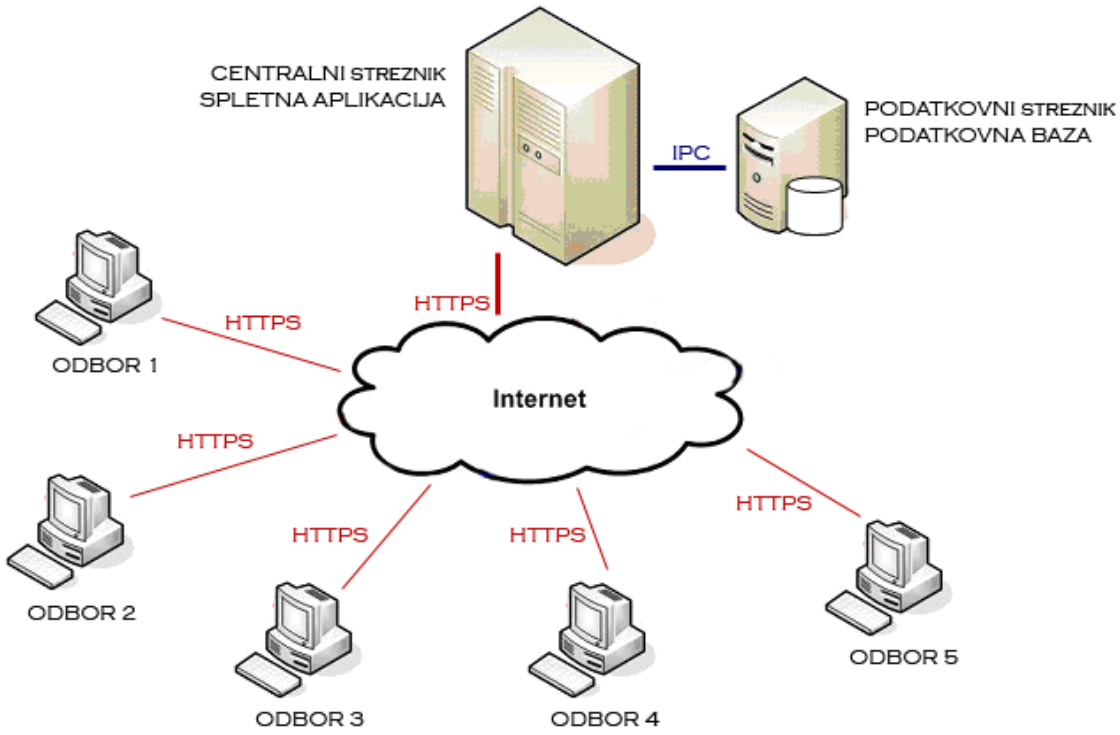
- Evidenca članstva, gospodinjstev in članarin
- Evidenca zaposlenih
- Evidenca nalog zaposlenih
- Spremljanje poteka nalog zaposlenih
- Obrazci za tedenska, mesečna in letna poročila
- Evidenca aktivnosti članov
- Evidenca porok in pogrebov
- Evidenca izvajanja verouka
- Evidenca dogodkov, prireditev in versko-izobraževalnih aktivnosti
- Evidenca donacij in dobrodelnih akcij
- Sistem komuniciranja in obveščanja zaposlenih in članov
- Spremljanje dogodkov v posameznih odborih s strani centralnega organa
- Finančna in druga poročila
- Statistika

3.4. Ponujena rešitev

Kot končni produkt smo definirali spletno aplikacijo, ki bo vsebovala vse zgoraj naštete funkcionalne sklope. Postaviti je bilo potrebno centralni strežnik, podatkovni strežnik in pripraviti spletno aplikacijo informacijskega sistema. Ideja je bila namreč takšna, da odjemalcem, v našem primeru zaposlenim v odborih, ne bo potrebno nameščati dodatne programske opreme, ampak bodo do sistema enostavno dostopali preko svojega priljubljenega brskalnika.

Razvoj aplikacije je bil razdeljen na dva dela. Razvoj obličja spletne aplikacije, oziroma dela, ki je v interakciji z uporabniki in razvoj zaledja spletne aplikacije, ki je uporabniku skrit. Sam

sem sodeloval v razvoju obličja sistema, ter vseh modulov in funkcionalnih sklopov povezanih z njim.



Slika 1: Struktura informacijskega sistema.

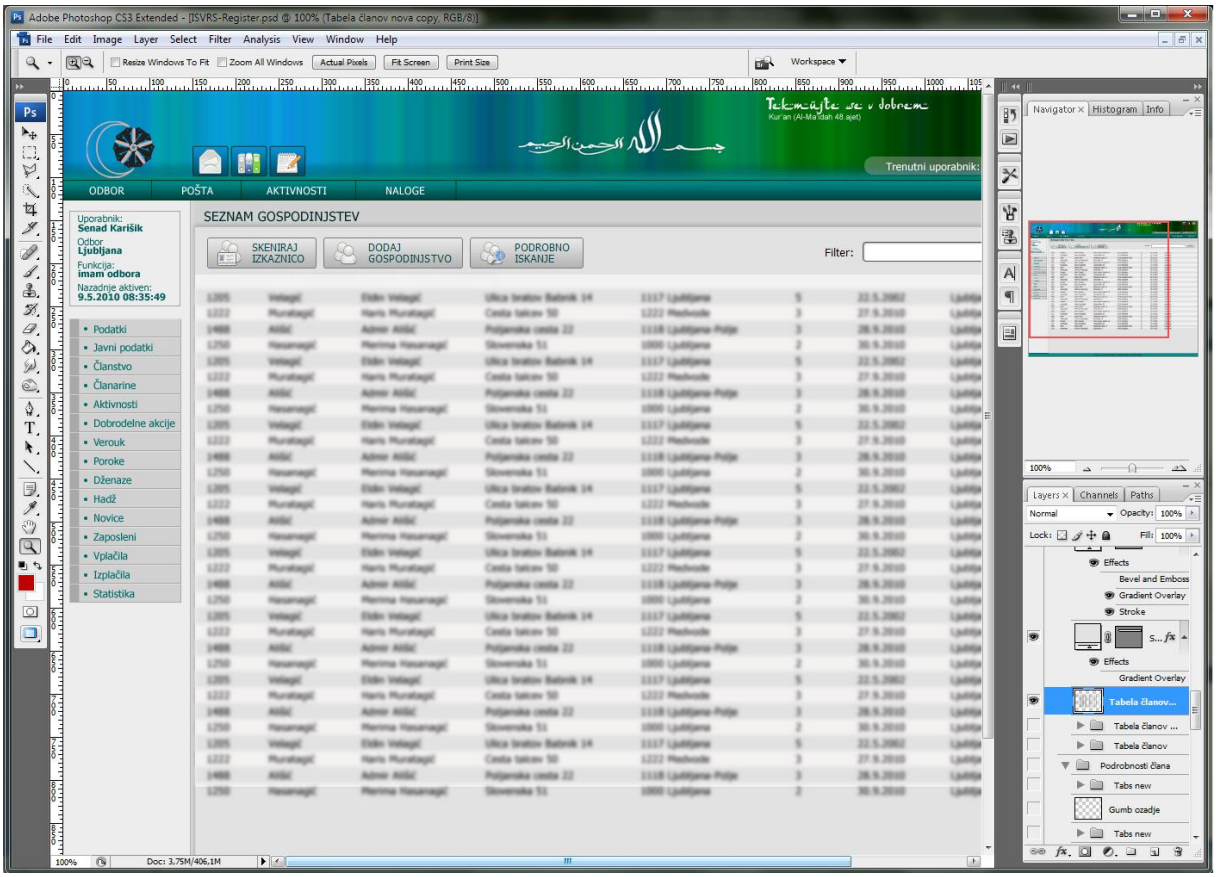
4. Uporabljena orodja in tehnologije

Za razvoj obličja informacijskega sistema smo uporabili več različnih orodij in tehnologij. Sama implementacija spletne aplikacije je potekala v programskem orodju Microsoft Visual Studio 2010 z uporabo Microsoftove spletne tehnologije za izdelavo dinamičnih spletnih strani in spletnih aplikacij asp.net MVC. Razvoj je potekal v programskem jeziku C#. Samo obličje aplikacije pa je bilo zasnovano in oblikovano s programom Adobe Photoshop CS3. Uporabili smo tudi sledeče tehnologije: HTML, CSS, JavaScript, AJAX, JQuery, JQuery UI, JQGrid…

4.1. Adobe Photoshop

Adobe Photoshop je profesionalni računalniški program za obdelavo fotografij in drugih grafik. Razvili so ga pri podjetju Adobe Systems, ki je trenutno eden izmed vodilnih izdelovalcev grafičnih programov na svetu.

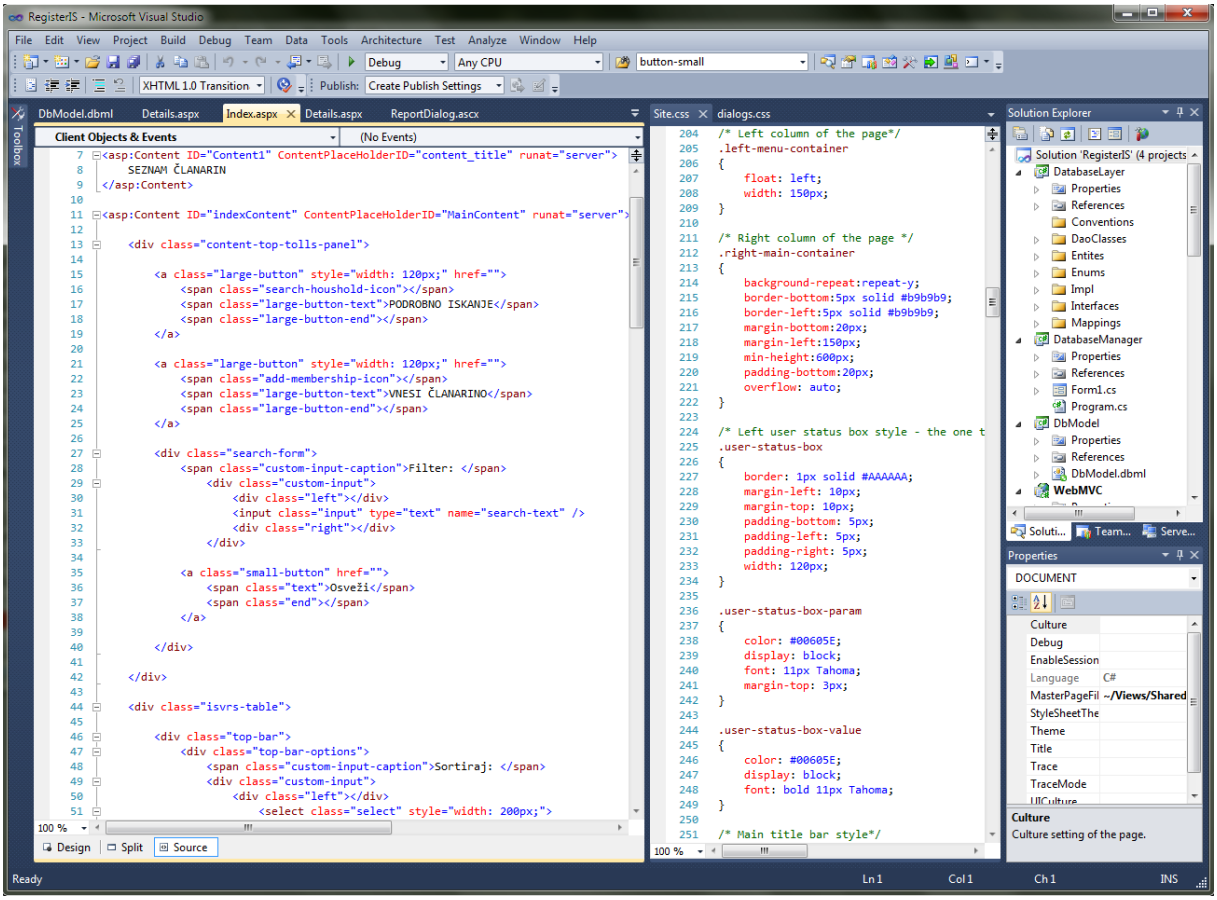
Leta 1987 je Thomas Knoll napisal prvi grafični program na Mac Plus-u. V istem letu, sta skupaj z bratom Johnom, ustvarila še program imenovan Display, ki sta ga, leto kasneje, preimenovala v ImagePro. V letu 1989 se je podjetje Adobe začelo zanimati za ImagePro, tako se je program ponovno preimenoval, tokrat v Adobe Photoshop, in začel se je razvoj. Tako je leta 1990 izšla prva različica, danes zelo priljubljenega, programa Adobe Photoshop [3].



Slika 2: Program Adobe Photoshop.

4.2. Visual Studio 2010

Microsoft Visual Studio 2010 je integrirano okolje, ki poenostavi celoten življenjski cikel razvijanja aplikacije, od načrtovanja do uvajanja. Visual Studio 2010 ponuja orodja za izdelovanje prototipov, modeliranje in vizualno načrtovanje, s katerimi programer lažje oživi svoje vizije. Nudi integrirano okolje, v katerem lahko razvijalci s svojim znanjem modelirajo, kodirajo, odpravljajo napake, preskušajo in uvajajo vse več vrst programov. Visual Studio 2010 poenostavi pogosta opravila in omogoča razvijalcem raziskovanje v globino platforme. Ponuja zmogljiva orodja za upravljanje projekta, ohranjanje izvorne kode in iskanje napak. Preizkuševalci in razvijalci lahko uporabljajo ročno in samodejno preizkušanje ter napredna orodja za odpravljanje napak in tako zagotovijo, da sestavljajo pravi program in na pravi način [2].

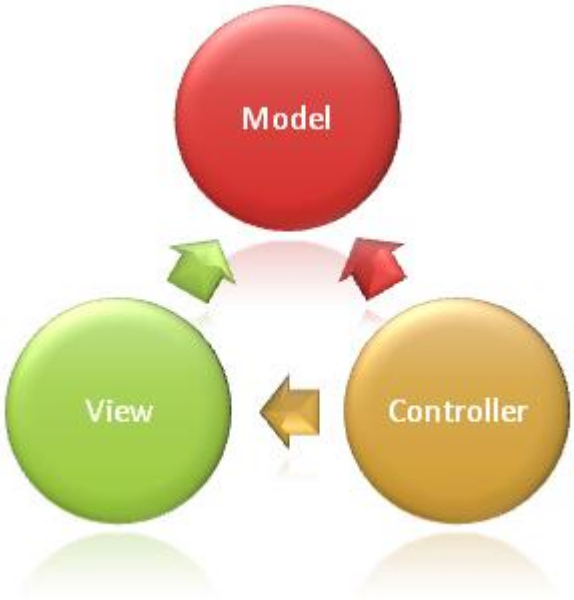


Slika 3: Razvojno orodje Microsoft Visual Studio 2010.

4.3. ASP .NET MVC

MVC, kratica za Model-View-Controller, je arhitekturni vzorec v računalniškem inženiringu, ki spletno aplikacijo razbije na 3 glavne komponente: model (model), pogled (view) in nadzornik (controller). Ogrodje ASP.NET MVC trenutno predstavlja alternativo svojemu predhodniku ASP.NET Web Forms pri razvoju spletnih aplikacij [4].

V zapletenih spletnih sistemih, ki uporabnikom ponujajo velike količine podatkov želi razvijalec pogosto ločiti podatkovni del (model) od uporabniškega vmesnika (view). Na takšen način zagotovi, da spremembe narejene na uporabniškem vmesniku ne bodo vplivale na ravnanje s podatki in, da prestrukturiranje podatkov ne bo prineslo dodatnih sprememb na uporabniškem vmesniku. MVC reši ta problem s striktno ločitvijo dostopa do podatkov in poslovne logike od predstavitve podatkov in uporabniškega vmesnika, kar pa doseže z uvedbo vmesne komponente nadzornika (controller) [5].



Slika 4: Grafični prikaz MVC komponent.

4.3.1. Model (model)

Model je komponenta, oziroma to so objekti, ki predstavljajo podatkovni del aplikacije ter implementirajo podatkovno domeno sistema. Navadno model objekti naložijo in shranijo stanje v podatkovno bazo. Na primer: objekt Artikel lahko pridobi informacije iz podatkovne baze, jih obdela in posodobljene informacije ponovno zapiše v podatkovno bazo [4].

4.3.2. View (pogled)

View je komponenta MVC, ki skrbi za prikaz podatkov uporabnikom sistema. Predstavlja uporabniški vmesnik. Tipično se ta uporabniški vmesnik ustvari s pomočjo objektov iz model komponente. View je lahko zaslonska maska, ki, na primer, uporabniku prikaže podrobnosti nekega Artikla, ter mu ponudi možnosti za urejanje le tega [4].

4.3.3. Controller (nadzornik)

Controller je komponenta, ki skrbi za upravljanje podatkov in reagira na interakcije uporabnika. Controller sproža ter obdeluje spremembe na objektih model-a in skrbi za izbiro uporabniškega vmesnika, ki se bo prikazal uporabniku.

4.3.4. Zanimiva predstavitev MVC arhitekture

Najprej si predstavljajmo banko. Sef je baza podatkov, kjer so shranjene najpomembnejše dobrote in so tako dobro zaščitene pred zunanjim svetom.

Potem imamo tu bankirje ali v programerskem izrazu model-e. Bankirji so edini, ki imajo dostop do sefa (baza podatkov). Po navadi so debelejši, starejši in leni, kar se lepo ujema z enim od MVC pravil: »debeli modeli, suhi nadzorniki«.

Nato imamo še povprečne delavce na banki, hitre in odzivne tekače - nadzornike. Nadzorniki ali tekači postorijo vso tekanje naokoli, zato morajo biti zdravi in suhi. Od bankirjev (model-ov) pridobijo informacije in jih prenesejo do bančnih strank – pogled-ov.

Bankirji (model-i) so na tem delovnem mestu že nekaj časa, zato tudi odločajo o pomembnih odločitvah, ki nas pripeljejo do naslednjega pravila: »čim več poslovne logike naj bo v model-u«. Nadzornik ali tekač, naš povprečen delavec, naj ne bi odločal o takih odločitvah ampak za detajle povprašal bankirja. Vendar pa tudi nadzornik ne sme biti povsem enostaven ali celo neumen (v redu je, da nadzornik vsebuje nekaj poslovne logike). Vendar pa kadarkoli tekač preveč razmišlja se bankir ujezi in vaša banka (ali aplikacija) izgublja posel. Zato ponovno poudarimo; vedno naj se naloži čim več možne poslovne logike (in izvajanja odločitev) na model.

Bankirji vsekakor ne bodo direktno govorili s strankami (pogled-i), ker so v njihovih udobnih naslonjalih vsekakor prepomembni za to početje. S tem sledi še eno pravilo: »Modeli naj ne govorijo s pogledi«. Ta komunikacija med bankirjem in stranko (med model-om in pogled-om) je vedno podprta s tekačem (nadzornik). (Da, obstajajo tudi nekatere izjeme tega pravila za VIP stranke, ampak ostanimo za zdaj pri osnovah).

Dogaja se tudi, da mora posamezen delavec (nadzornik) pridobiti informacije od več bankirjev, in to je popolnoma sprejemljivo. Če so bankirji sorodni (kako bi sicer imeli tako lepe poklice?), potem bodo bankirji (model-i) komunicirali najprej sami s sabo in zatem podali zbrane informacije tekaču, ki jih bo veselo dostavil stranki (pogled-u). Zato tu nastopi še eno pravilo: »modeli priskrbijo informacije nadzorniku s pomočjo njihovih vez«.

V naši banki bi to bilo nekako tako:

Delavec Andrej -> vpraša bankirja Primoža -> ki vpraša drugega bankirja Janeza -> kdo dobi zaslužek s sefa (podatkovne baze).

Kaj pa naša stranka (pogled)? No, tudi banke delajo napake in stranka naj bo dovolj pametna, da pretehta informacijo o svojem računu in naredi svojo odločitev. V MVC vzorcu tako dobimo še eno preprosto pravilo: »prav je, da pogled vsebuje nekaj logike, ki se ukvarja s pogledom oziroma predstavitvijo«.

Če sledimo naši analogiji to pomeni, da stranka ne bo pozabila obleči hlač, ne bo pa govorila bankirju kako naj le ta izvaja transakcije [6].

4.4. JavaScript

Leta 1995 so pri podjetju Sun razvili nov programski jezik, imenovan Java. Namenjen je pisanju programov, ki jih lahko nespremenjene izvajamo na vsakem računalniku, ne glede na vrsto procesorja ali operacijski sistem. Zaradi te lastnosti so ti programi idealni za vključitev na spletne strani, saj tam ne vemo vnaprej, na kakšnih vrstah računalnikov bodo uporabniki prebirali naše strani.

Da bi izboljšali podporo programčkom, napisanim v jeziku Java, so pri podjetju Netscape še istega leta razvili programski jezik, ki so ga sprva poimenovali LiveScript, iz tržnih razlogov pa so ga kmalu preimenovali v JavaScript (zaradi popularnosti Jave). To ni bila ravno dobra poteza, saj je preveč uporabnikov mislilo, da je Java in JavaScript isto. Kljub začetnim pomanjkljivostim (slaba varnost, nestandardiziranost, pomanjkanje razvojnih orodij) je JavaScript sčasoma postal, eno od najbolj priljubljenih orodij za izdelavo dinamičnih spletnih strani [8].

Sintaksa jezika JavaScript ohlapno sledi programskemu jeziku C. Prav tako kot C, JavaScript nima vgrajenih vhodno izhodnih funkcij, zato je izvedba teh funkcij odvisna od gostitelja. JavaScript se veliko uporablja za ustvarjanje dinamičnih spletnih strani. Program se vgradi ali pa vključi v HTML z namenom, da opravlja naloge, ki niso mogoče s samo statično stranjo. Na primer, odpiranje novih oken, preverjanje pravilnosti vnesenih podatkov, enostavni izračuni, itd. Na žalost različni spletni brskalniki izpostavijo različne objekte za uporabo. Za podporo vseh brskalnikov je zato potrebno napisati več različic funkcij.

Zunaj spleta se JavaScript uporablja v različnih orodijh. Adobe Acrobat in Adobe Reader podpirata jezik v PDF-datotekah. Podpirata ga tudi operacijska sistema Microsoft Windows in Mac OS X. Programi imajo svoje objektne modele, ki zagotavljajo dostop do gostiteljevega okolja, samo jedro jezika JavaScript pa je v vseh programih večinoma enako [9].

4.5. AJAX

AJAX (asinhroni JavaScript in XML) je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij. Z AJAX-om si lahko spletne aplikacije izmenjujejo podatke s strežnikom asinhrono v ozadju, brez potrebe po ponovnem nalaganju strani. S tem je mogoče tekoče in hitrejše spremljanje ter spreminjanje vsebine na spletni strani. Podatki se prenašajo s pomočjo objektov XMLHttpRequest ali s pomočjo Remote Scriptinga (v starejših brskalnikih, ki ne podpirajo tehnologije AJAX).

Uporaba tehnologije AJAX je značilna za Web 2.0. Navkljub imenu, uporaba tehnologij Javascript in XML ni pogoj za izvajanje AJAX-a.

AJAX aplikacije dajejo vtis, kot da v celoti tečejo na računalniku uporabnika. Običajna spletna aplikacija namreč za vsako spremembo na strani pošlje zahtevo HTTP, strežnik pa kot odgovor pošlje celotno stran. Brskalnik mora zato osvežiti celotno stran, s tem pa pride do motečega obnavljanja strani. Spletna aplikacija je zato počasna in čisto nič podobna namiznim. Medtem, ko so aplikacije AJAX prirejene za generiranje poizvedb za strežnik, tako da pošljejo samo tiste podatke, ki jih dejansko potrebujejo. Klic se opravi kot asinhrona komunikacija, torej medtem ko aplikacija čaka podatke iz strežnika, lahko uporabnik nemoteno uporablja spletno stran. Ko so podatki pripravljeni, določena funkcija v JavaScriptu prikaže podatke na strani, brez potrebe po ponovnem nalaganju. Posledica tega je uporabniški vmesnik, ki se veliko hitreje odziva na vnose uporabnika.

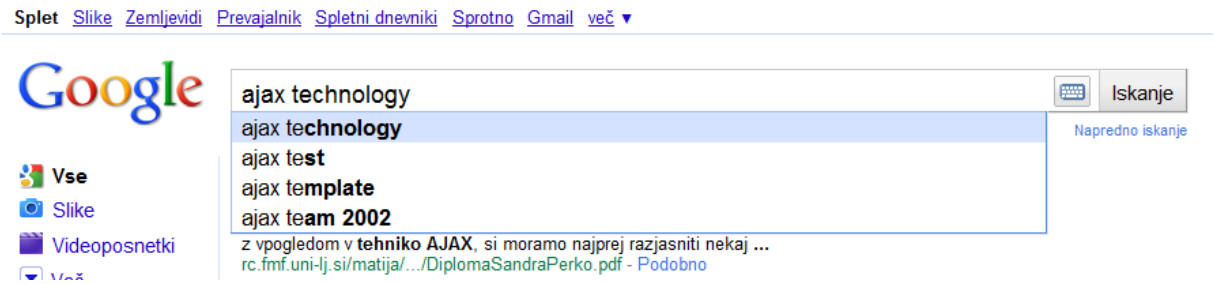
Razlog za vpeljavo te tehnologije je tudi dejstvo, da se med odjemalcem (brskalnikom) in strežnikom prenese veliko manj podatkov ter, da poteka nalaganje podatkov asinhrono. Poleg tega se zmanjša obremenitev spletnega strežnika, ker se veliko obdelave lahko naredi na strani odjemalca [7].

Prednosti AJAX-a:

- Možnost izdelave hitrejših, boljših in uporabniku bolj prijaznih spletnih aplikacij
- Prenašanje samo določenih podatkov, ne pa celotne strani, kar drastično zmanjša količino prometa med strežnikom in odjemalcem.

Slabosti AJAX-a:

- Podatki se osvežujejo samo znotraj strani, zato je navigacija po strani otežena
- Brskalniki, ki ne uporabljajo skriptnih jezikov JavaScript ali podobnih, imajo težave s prikazom strani
- Neenotno delovanje na različnih brskalnikih, kar otežuje programiranje aplikacij



Slika 5: Uporaba tehnologije AJAX v Google iskalniku.

4.6. JQuery

JQuery je knjižnica za skriptni jezik JavaScript, ki prinaša zbirko funkcij za hitrejši razvoj spletnih aplikacij. Ideja je, da z manj kode naredimo več, kar je zapisano tudi v sloganu "Write Less, Do More". Poleg tega rešuje težave v zvezi s spletnimi brskalniki, kjer se isto stvar v različnih brskalnikih implementira drugače. Če jQuery določen brskalnik podpira, potem iste jQuery funkcije delujejo v vseh brskalnikih enako. V nadaljevanju je nekaj enostavnih primerov uporabe knjižnice.

Preden lahko začnemo z uporabo, moramo knjižnico vključiti v svojo spletno stran. Na uradni spletni strani http://jquery.com/ lahko dobimo dve različici, minimizirano ali v izvorni obliki. Običajno se uporablja minimizirana oblika, ki je na poseben način predelana (stisnjena) JavaScript izvorna koda z namenom, da knjižnica (datoteka) zasede čim manj prostora. Prepoznamo jo po besedici "min" v imenu datoteke, ki je jquery.min.js in po nerazumljivi izvorni kodi. Na voljo je tudi izvorna oblika, ki jo prenesemo le, če želimo videti, kako je jQuery dejansko izdelan in se pri tem še kaj naučimo. Izbrano različico si prenesemo na svoj računalnik oz. strežnik in vključimo v spletno stran kot vsako drugo JavaScript kodo. Obstaja tudi možnost uporabe knjižnice preko Google Libraries API, kjer se jQuery naloži z Google strežnikov. Prednost je ta, da se sam jQuery ne prenaša ponovno, če je uporabnik že obiskal katerokoli spletno stran, ki že uporablja jQuery z Google strežnikov, kar pospeši nalaganje.

Predpostavimo, da imamo naslednjo HTML kodo:

```
<div id="Polja">
   <input id="poljeA" type="text" value="ena" />
   <input id="poljeB" type="text" value="dva" />
   <input id="poljeC" type="text" value="tri" />
</div>
```

V kolikor želimo izpisati vrednost prvega vnosnega polja (z ID-jem poljeA). Običajno to storimo takole:

```
alert(document.getElementById("poljeA").value);
```

S pomočjo jQuery-ja lahko to zapišemo krajše:

```
alert($("#poljeA").val());
```

Razlaga je naslednja: znak $ je ime privzete jQuery funkcije. Ta funkcija lahko prejme poljuben CSS selektor, ki je v našem primeru ID poljeA. Če je selektor naveden, potem dobimo seznam elementov, v našem primeru pa za vsak objekt pokličemo jQuery funkcijo val(), ki vrne vrednost iz HTML atributa value. V omenjenem primeru je samo en tak element, zato dobimo tudi samo eno vrednost.

V kolikor želimo zbrisati vse vrednosti vseh treh vnosnih polj, to z jQuery-jem izvedemo z:

```
$("input").val('');
```

S pomočjo CSS selektorja »input« izberemo vsa vnosna polja, nato pa nad vsakim spremenimo vrednost atributa »value« na prazen niz.

Primer uporabe jQuery-ja, kjer selektor ni podan je npr. izvedba AJAX klica po metodi get:

```
$.get("test.php", function(data){
   alert("Vrnjen rezultat: " + data);
 });
```

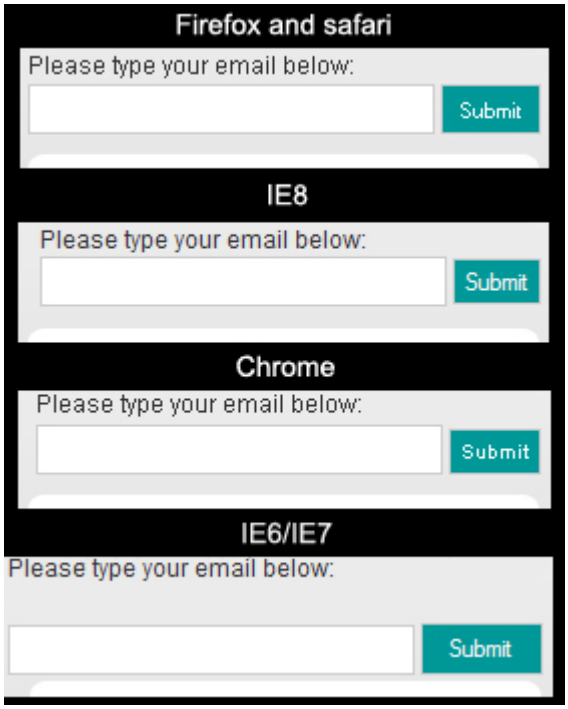
Za znakom $ je takoj pika, torej ni selektorja, nato sledi jQuery funkcija get, ki izvede AJAX klic po metodi get. Na vhodu prejme naslov spletne strani (v našem primeru test.php) in anonimno funkcijo (funkcija brez imena), ki se izvede, ko stran vrne rezultat. V našem primeru vrnjene podatke izpišemo preko funkcije »alert«. Bodimo pozorni na to, da omenjena koda deluje na vseh podprtih brskalnikih enako, čeprav se npr. za Internet Explorer AJAX klic implementira drugače kot v drugih brskalnikih [10].

4.7. CSS

CSS (Cascading Style Sheets) omogoča avtorjem spletnih strani, da predpišejo obliko posameznih elementov. Večina elementov v HTML je namreč namenjena logičnemu oblikovanju, kjer samo določimo, kakšne vrste je posamezen element (slika, tabela, vrstica v tabeli, celica v vrstici, seznam, točka seznama, odstavek, indeks, eksponent, naslov, aktivna povezava ...), brskalnik pa te elemente oblikuje po svoje.

Z uporabo stilov CSS lahko elementom določimo celo vrsto oblikovnih lastnosti, med katere spadajo ozadje, robovi, razmiki, odmiki, pisava, poravnava, barva, itd... CSS nam omogoča, da oblikovne lastnosti določimo ločeno od vsebine, kar poveča preglednost napisane kode.

Organizacija W3C, ki se ukvarja s spletnimi standardi, med drugim določa tudi priporočila, kako naj bi bili opisani stili CSS. Podobno kot drugi spletni standardi, se tudi CSS še vedno razvija. Začelo se je leta 1996 s CSS stopnje 1, ki je omogočal nastavitev pisave, barve, ozadja, robov, odmikov, itd... Dve leti kasneje so objavili priporočila za CSS stopnje 2, ki omogoča tudi absolutno postavitev elementov, različne vrste številčenja, prelome strani .... V pripravi so tudi že priporočila za CSS stopnje 3. Žal si avtorji brskalnikov priporočila razlagajo vsak po svoje in občasno se zgodi, da neka spletna stran v enem brskalniku zgleda precej drugače, kot v drugem (Slika 6) [11].



Slika 6: Razlike med brskalniki v prikazu elementov z enakim CSS-om.

5. Razvoj obličja aplikacije

Razvoj obličja aplikacije je potekal v štirih glavnih fazah. V prvi fazi je bilo potrebno zbrati zahteve naročnika. Ker se je sistem razvijal v dveh delih, obličje in zaledje sistema, smo pridobljene informacije uporabili na obeh straneh. Pri obličju sistema za oblikovanje in pripravo zaslonskih mask, pri zaledju sistema pa za pripravo podatkovne baze in model objektov. Izbrati je bilo potrebno tudi članske izkaznice in tehnologijo za delo z njimi.

V drugi fazi je bilo potrebno pripraviti in oblikovati zaslonske maske uporabniškega vmesnika, razrezati zasnovo sistema in pripraviti CSS in HTML datoteke.

V tretji fazi smo pripravili dialoge za vnos in urejanje podatkov in povezali zaslonske maske uporabniškega vmesnika z zaledjem informacijskega sistema.

V zadnji, četrti, fazi pa je bilo potrebno dobro testirati sistem. V kolikor se je pri testiranju pokazala kakšna pomanjkljivost ali napaka, je bilo potrebno napako odpraviti in ponovno testirati dotični modul.

5.1. Zbiranje zahtev naročnika

Preden smo lahko začeli z pripravo in oblikovanjem zaslonskih mask je bilo od naročnika potrebno izvedeti, katere entitete oziroma področja sploh želi zajeti v sistem. Skupaj smo preučili področja, ki bi jih lahko novi sistem zajemal. Področja je bilo potrebno definirati v skladu z obstoječimi praksami a vseeno tako, da bo nov sistem olajšal in pospešil delo v skupnosti.

Za vsako entiteto je bilo potrebno zajeti vse funkcionalne in nefunkcionalne sklope, kot tudi definirati informacije in podatke, ki jih entiteta obkroža. Določili smo prioritete razvoja, katere dele razviti najprej, katere pozneje in katere odložiti za naslednje različice sistema. Kot rezultat zajema zahtev, je nastal dokument z opisom vseh entitet sistema, njihovih funkcionalnosti in podatkov ter informacij vezanih za dotično entiteto.

Seznam entitet oziroma funkcionalnih sklopov za prvo različico aplikacije je sledeči:

- Odbori
- Zaposleni
- Gospodinjstva
- Člani
- Članarine
- Aktivnosti
- Naloge
- Poroke
- Pogrebi
- Izobraževalne dejavnosti
- Dobrodelne akcije
- Donacije
- Statistika

Kot primer bom navedel glavne tri entitete iz nastalega dokumenta:

ODBORI	
Funkcionalna enota, ki pokriva določeno teritorialno področje članov ISVRS. V sistemu bo več odborov, ki bodo spadali pod okrilje centralne enote mešihata. Funkcionalnosti celotnega sistema so grupirane po odborih. Vse aktivnosti, finance, članstva in ostale podenote se vodijo pod okriljem teh odborov.	
Podatki in funkcionalnosti:	
Ime oz. naziv odbora	Imena in kratice, ki jih posamezni odbor uporablja za svojo identifikacijo.
Podatki o odboru	Naslov, poštna številka, kraj, opis odbora, funkcije, kontaktni podatki, …

Javni podatki	Podatki o odboru, ki bodo javno vidni na spletni strani. To so lahko opis odbora, opis funkcij odbora, zaposleni, člani, urniki, dokumenti, slikovno gradivo, …
Zaposleni	Pod zaposlene osebe spadajo imami, člani odborov, blagajniki, predsedniki... vsi tisti, ki bodo imeli dostop do sistema. Vsak zaposleni dobi podatke za prijavo v sistem, ter pravice, ki jih ima v sistemu. *Več pod točko zaposleni...*
Članstvo	Seznam gospodinjstev in oseb, ki so člani odbora.
Članarine	Seznam članarin za tekoče leto in nazaj. Vpogled v zgodovino članarin. Razni seštevki in kalkulacije.
Umrli in pogrebi	Seznam umrlih, ki so bili člani odbora, oziroma tistih ki, so bili pokopani pod okriljem odbora.
Poroke	Seznam poročnih obredov v odboru.
Aktivnosti	Seznam aktivnosti na nivoju odbora. Aktivnosti so lahko verouki, izobraževalne aktivnosti, predavanja in ostale prireditve....
Dobrodelne akcije	Dobrodelne akcije na nivoju odbora. Dobrodelno akcijo lahko razpiše mešihat za vse odbore, kar pomeni, da posamezni odbori le izpolnijo spletni obrazec o dobrodelni akciji, ki ga izda mešihat, ali pa jo razpiše sam odbor. Dobrodelne akcije lahko trajajo različno (lahko se enkratne ali pa npr. trajajo mesec, leto).
Vplačila in izplačila	Seznam prihodkov in odhodkov.
Finančna poročila	Različni seštevki, kalkulacije, izpiski....
Statistika	Prikaz raznih statistik na podlagi zbranih podatkov.
Analiza uspešnosti	Analiza uspešnosti med posameznimi odbori, glede na aktivnosti, članstvo, itd…
Novice	Vnašanje in urejanje novic, s strani posameznega odbora, ki bodo javno objavljene na spletni strani.

Tabela 1: Funkcionalni sklopi entitete odbor.

ZAPOSLENI

Vsak odbor ima pristojne in odgovorne ljudi. Vsak odbor ima tudi imama, predsednika, blagajnika, in člane zborov, ter druge zaposlene, ki bodo lahko dostopali do sistema. Pod zaposlene se štejejo predstavniki mešihata, ter vsi zaposleni, ki jih vnesejo za to pristojne osebe.

Zaposlenim se dodelijo pravice, ki jih imajo v sistemu.

Podatki in funkcionalnosti:

Podatki o zaposlenem	Ime, priimek, naslov, izobrazba,…
Podatki za sistem	Uporabniško ime, geslo, nastavitve, profil, pravice…
Urejanje odbora	Urejanje podatkov na nivoju odbora. Potrebne pravice.
Urejanje članstva.	Vnašanje novih članov, razporejanje članov in urejanje članstva.
Urejanje članarin	Vnos vplačil članarin.
Urejanje donacij in dobrodelnih akcij	Vnašanje in urejanje dobrodelnih akcij. Vnašanje in urejanje donacij za posamezno gospodinjstvo oziroma člana gospodinjstva.
Urejanje aktivnosti	Vnos aktivnosti, zaključevanje aktivnosti, urejanje in brisanje aktivnosti, itd…
Urejanje in izvajanje poročnih obredov	Vnos in urejanje podatkov o poročnih obredih, ki jih izvedejo zaposleni imami.
Urejanje in izvajanje pogrebnih obredov	Vnos in urejanje podatkov o pogrebnih obredih, ki jih izvajajo zaposleni imami.
Urejanje vplačil in izplačil	Vnos in urejanje podatkov o finančnih prihodkih in odhodkih v odboru.
Izmenjava sporočil	Zaposleni si lahko med seboj izmenjujejo sporočila, ki so tekstovne oblike ali pa kot priponka.
Tedenska, mesečna, letna poročil	Oddaja in izmenjava raznih poročil nadrejenim...
Urejanje javnih podatkov in novic	Urejanje, dodajanje, brisanje podatkov in novic, objavljenih na spletni strani.
Analiza dela	Izpis poročila o opravljenem delu za določeno časovno obdobje. Možnost primerjanja količine dela in uspešnosti med zaposlenimi ter temu primerno

	nagrajevanje…

Tabela 2: Funkcionalni sklopi entitete zaposleni.

GOSPODINJSTVA OZ. ČLANSTVO V ODBORIH	
Članstvo v odboru je vezano na gospodinjstvo, ki ga lahko sestavlja ena ali več oseb. Posamezno gospodinjstvo je lahko včlanjeno v enega ali več odborov. Gospodinjstvo ob včlanitvi dobi unikatno identifikacijsko kartico, s pomočjo katere se vodi članstvo.	
Podatki in funkcionalnosti:	
Podatki o gospodinjstvu	Priimek družine, naslov gospodinjstva, pošta, kraj.
Kontaktni podatki	Telefon, mail, skype,...
Seznam odborov	Prikaz odborov katerih član je gospodinjstvo.
Seznam oseb	Seznam oseb, ki so člani gospodinjstva. Med osebami je nosilec gospodinjstva.
Seznam članarin	Seznam plačevanja članarine za članstvo v posameznem odboru.
Seznam aktivnost	Seznam aktivnosti, pri katerih je gospodinjstvo ali osebe iz gospodinjstva sodelovalo.
Seznam donacij	Seznam donacij gospodinjstva ali oseb članov gospodinjstva.
Seznam pomoči	Seznam področij, na katerih je gospodinjstvo sodelovalo in pomagalo. Področja na katerih se gospodinjstvo ponuja za pomoč.
Seznam vplačil	Seznam vplačil (tako gospodinjstva, kot oseb članov gospodinjstva).
Obvestila članom	Pošiljanje obvestil o vplačilih (ob koncu leta), čestitke ob praznikih,… (V prvi vrsti je mišljeno v elektronski obliki na elektronski naslov, alternativa po navadni pošti).

Tabela 3: Funkcionalni sklopi entitete gospodinjstvo.

5.2. Članska izkaznica

Potrebno je bilo rešiti tudi problem identifikacije članov skupnosti. Odločili smo se za uvedbo članskih izkaznic. Vsak član skupnosti bo dobil svojo člansko izkaznico, ki jo bo nato uporabljal. Članska izkaznica bo sestavljena iz identifikacijskih podatkov: imena in priimka ter unikatne identifikacijske številke, ki bo podkrepljena s črtno kodo. Vsako odbor pa dobi svoj bralec črtne kode.

Črtna koda je zakodiran zapis podatkov v obliki različno širokih in razmaknjenih navpičnih črt, ki jo lahko s pomočjo optičnega bralnika hitro in zanesljivo preberemo [13]. Črtna koda omogoča hitrejše in natančnejše zajemanje podatkov s tiskanih dokumentov, raznih nalepk, embalaže, izkaznic .... Najbolj množično se črtna koda uporablja za zapis enotne številke artikla (ERAN-UCC) na embalaži artikla v maloprodaji, veliko pa se uporablja tudi na drugih področjih [12].

Slika 7: Primer članske izkaznice z osnovnimi podatki in črtno kodo.

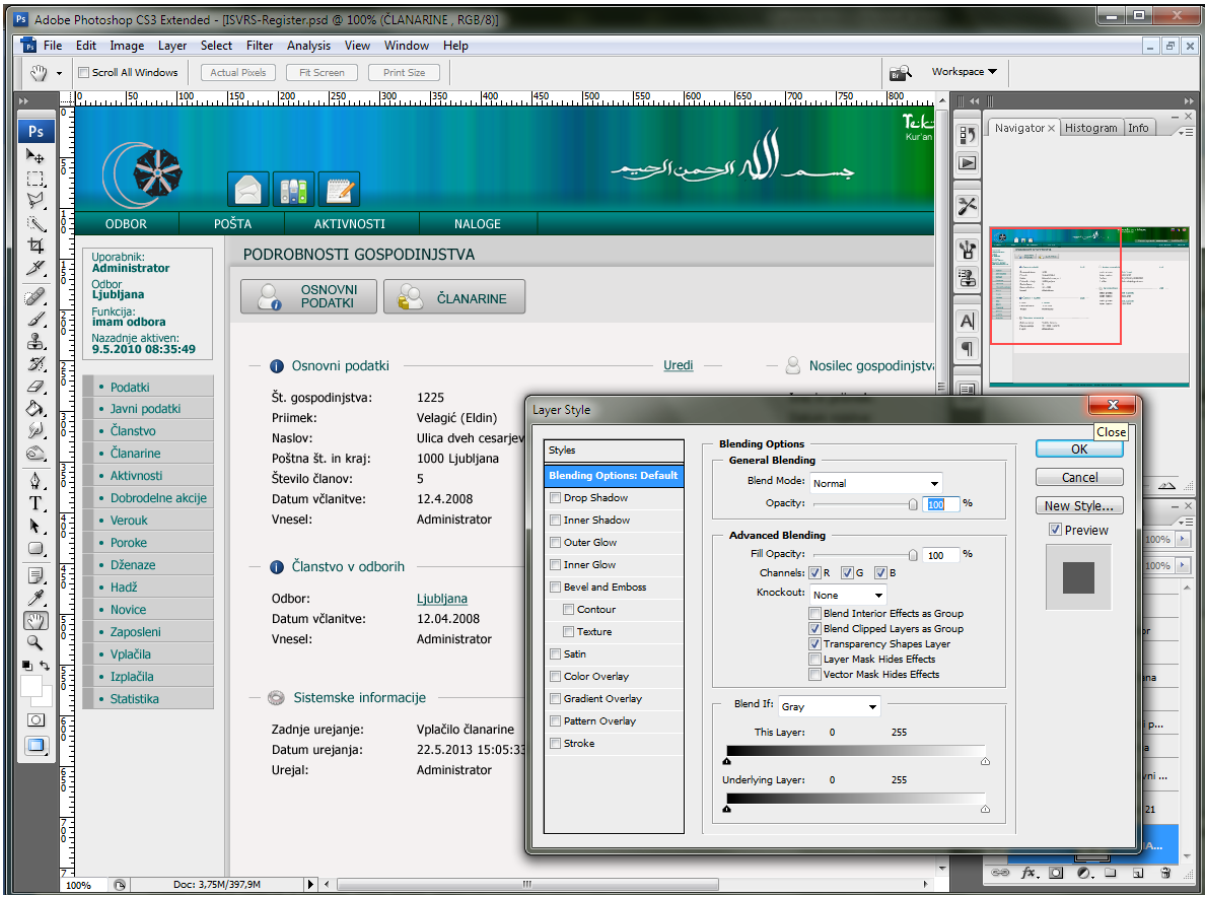
Črtna koda je čitljiva z bralniki črtne kode. Ti bralniki so lahko samostojne naprave, ki imajo vgrajen pomnilnik in program. Taki bralniki omogočajo, na primer, hitro izdelavo inventure. Največ pa se uporabljajo bralniki, ki imajo vgrajen samo program za dešifriranje črtne kode (spremembo v niz številk in črk). Ta program ugotovi, kateri tip kode je uporabljen in ga pretvori. Tak bralnik se priključi na osebni računalnik zaporedno s tipkovnico, tako da programi v računalniku ne prepoznajo ali je vnos prišel s tipkovnice ali z bralnika črtne kode. Tako se lahko (če bralnik ne more prebrati črtne kode) niz znakov vnese preko tipkovnice. Zaradi tega je pod ali nad črtno kodo vedno izpisan še niz znakov, ki je zakodiran vanjo [12].

5.3. Oblikovanje in priprava uporabniškega vmesnika

Ko smo zaključili z zbiranjem zahtev naročnika in se prepričali, da smo natančno zbrali vse kar naročnik želi, smo lahko pričeli s postavitvijo zasnove uporabniških mask. Potrebno je bilo zasnovati in oblikovati vse zaslonske maske uporabniškega vmesnika in dialoge za vnos in urejanje podatkov.

Najprej smo pripravili predloge zaslonskih mask. Pri pripravi smo si pomagali z dokumentom, ki je nastal v prejšnji fazi razvoja ob zbiranju zahtev. Za samo oblikovanje pa smo uporabili program Adobe Photoshop.

Ko pripravljamo zaslonske maske moramo natančno zajeti vsa podatkovna polja, ki bodo na zaslonski maski, definirati vrsto podatkovnih polj in način dostopa do podatkovnega polja. Na podlagi teh definicij pa izbrati primeren gradnik za posamezno podatkovno polje.

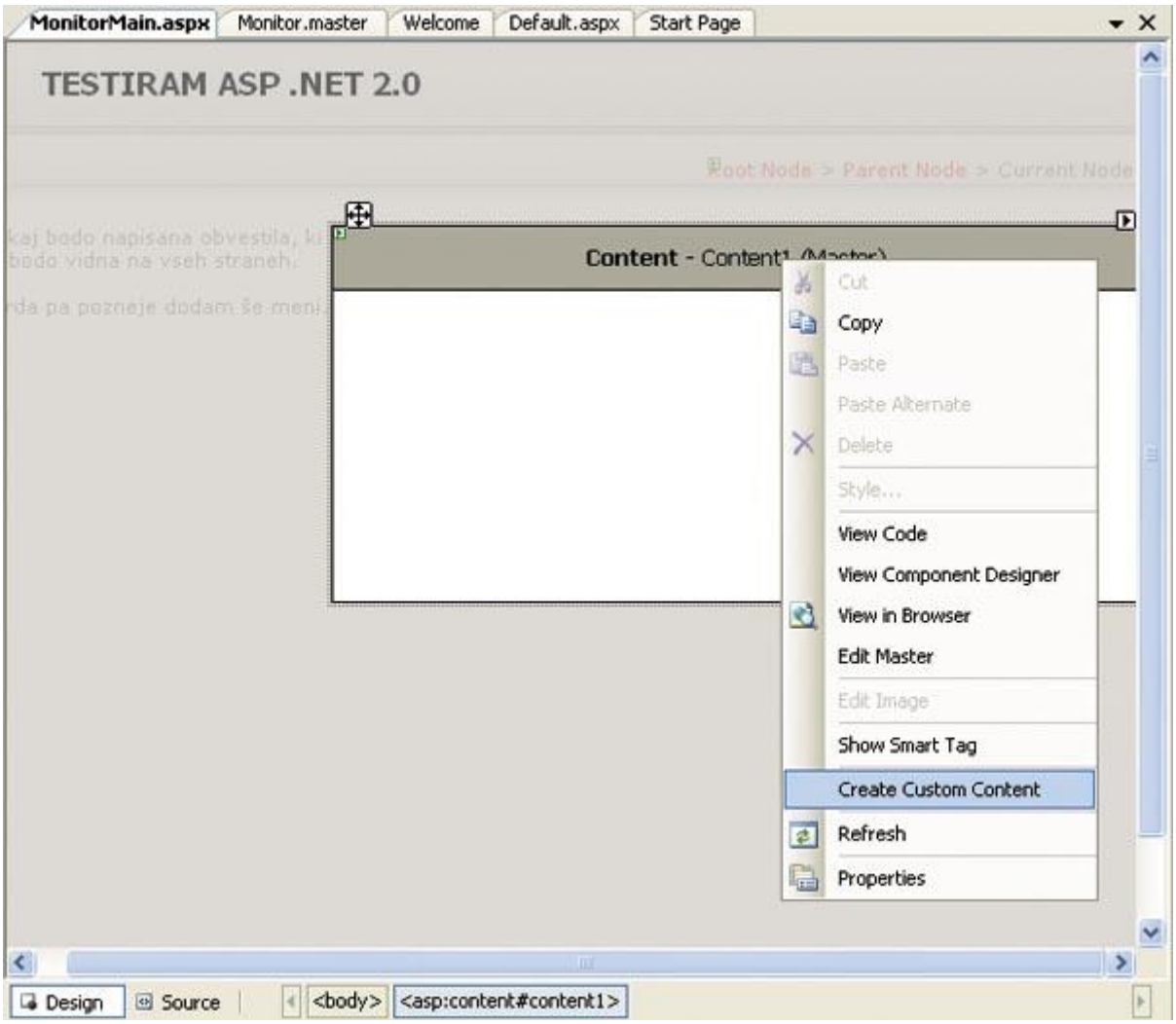


Slika 8: Oblikovanje zaslonske maske podrobnosti gospodinjstva.

Pripravljene osnutke zaslonskih mask smo nato pokazali naročniku aplikacije. Skupaj z naročnikom smo pregledali podrobnosti posameznih mask in odpravili morebitne pomanjkljivosti.

Po potrjenih osnutkih zaslonskih mask in potrjeni grafični podobi uporabniškega vmesnika smo se lotili razreza in postavitve spletne aplikacije. Grafično predlogo smo razbili na manjše dele, jo s pomočjo Adobe Photoshopa razrezali na posamezne grafične elemente, ki smo jih nato shranili kot GIF, PNG, JPEG datoteke. Razrezane dele pa smo nato ponovno sestavili s pomočjo jezika HTML. Z uporabo CSS, JavaScripta in knjižnice JQuery pa je uporabniški vmesnik dobil svojo končno podobo. Uporabniški vmesnik spletne aplikacije pa smo zasnovali z uporabo Microsoftovega ogrodja predlog strani (Master pages).

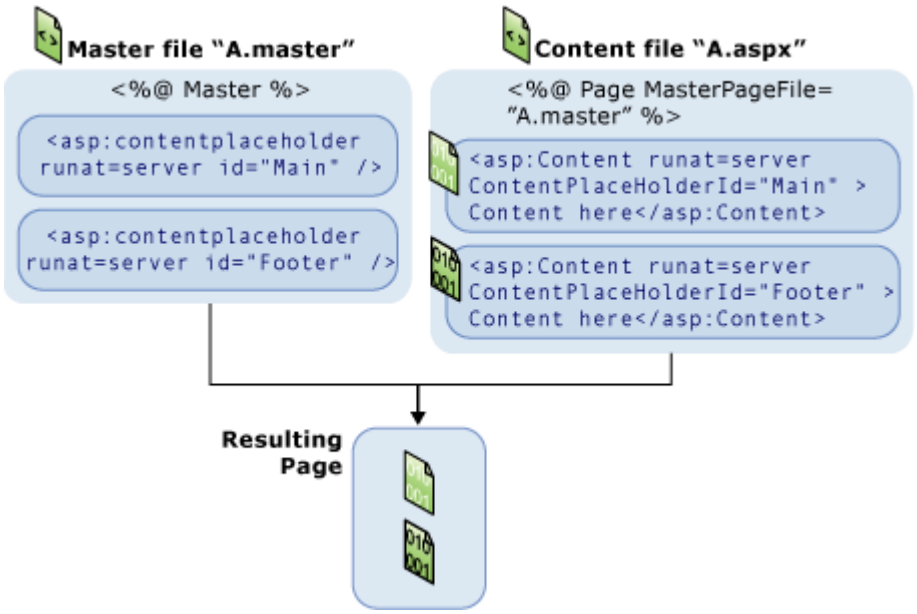
Pri vsakem malo obsežnejšem spletnem projektu naletimo na problem poenotenega videza spletnega mesta in navigacije. Včasih smo to reševali z vključevanjem datotek v vsako spletno stran ali pa s pomočjo uporabniških spletnih kontrolnikov (ASCX v ASP.NET). Predloge strani (Master pages) omogočajo, da določimo videz osnovne strani in s kontrolnikom ContentPlaceholder določimo prostor, na katerem bo vsaka stran dodala svojo vsebino (vizualno dedovanje). Tako je osnovni videz vsebovan na enem mestu in je poznejše spreminjanje kar najbolj poenostavljeno [14].



Slika 9: Primer nove strani na podlagi predloge strani.

K poenotenemu videzu strani spadajo tudi definicije, ki jih zapisujemo v datoteke CSS. Zaradi enostavnejšega preklapljanja med različnimi nabori datotek CSS so v ASP .NET 2.0 uvedene teme, ki so izvedene kot podimeniki v okviru imenika "App_Themes". Ime takega podimenika je hkrati ime teme, ki ga lahko pozneje določimo posamezni strani, celotni rešitvi, omogočeno pa je seveda tudi dinamično preklapljanje med temami v času izvajanja.

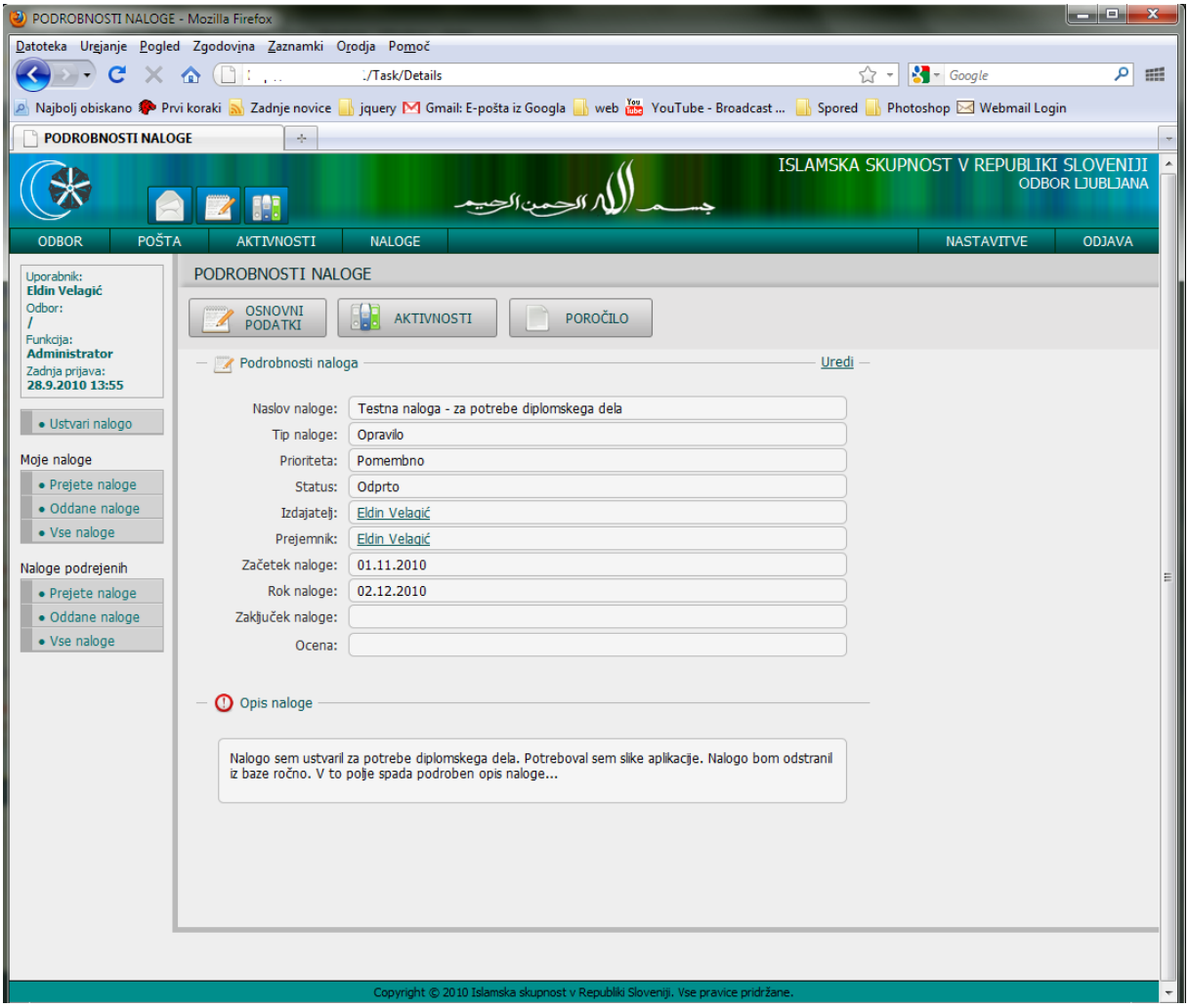
```
<%@ Page Language="C#" MasterPageFile="~/Default.master" Theme="White" %>
```



Slika 10: Nadomestitev kontrolnika ContentPlaceholder z vsebino.

V času izvajanja se predloge strani (Master pages) ravnajo sledeče:

- Uporabnik zahteva vsebino spletne strani (Content page) z vnosom URL naslova v brskalnik.
- Ko strežnik dobi zahtevek po spletni strani pogleda direktivo @Page. V kolikor se ta direktiva sklicuje na datoteko, ki vsebuje predlogo strani, bo prebrana tudi ta predloga. Če je to prvi zahtevek po omenjenih straneh, bosta prevedeni tako vsebina kot predloga.
- Predloga strani s posodobljeno vsebino je združena v kontrolno drevo (controll tree) strani z vsebino.
- Vsebine iz zahtevane spletne strani (Content page) je zlita v pripadajoči kontrolnik ContentPlaceHolder znotraj predloge strani.
- Vsebina sestavljene strani se vrne spletnemu brskalniku.



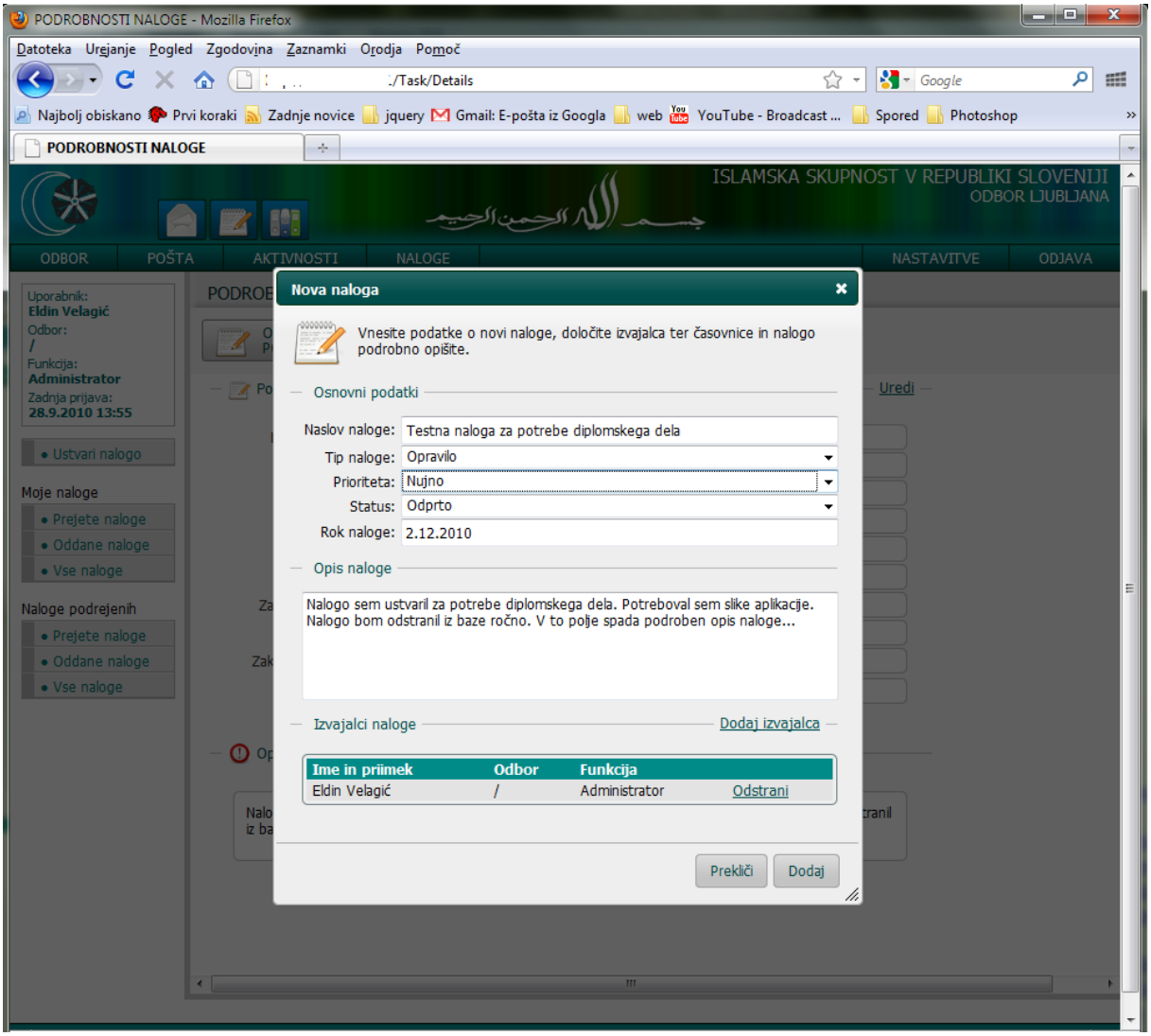
Slika 11: Zaslonska maska za podrobnosti naloge, prikazana v spletnem brskalniku.

Vnašanje in spreminjanje podatkov poteka preko za to posebej pripravljenih dialogov. Cilj je bil striktno ločiti prikaz podatkov in informacij od vnosa in urejanja vsebin. S tem smo hoteli doseči, da bo spletna aplikacija bolj pregledna ter, da se bodo uporabniki lažje in hitreje znašli. Tako kot zaslonske maske smo tudi dialoge najprej oblikovali s programom Adobe Photoshop in jih kasneje oblikovali s HTML-jem in CSS-om. Za prikazovanje dialogov pa smo uporabili JavaScript knjižnici JQuery in JQuery UI. Podatkovna polja znotraj dialogov so združena v formo, ki se nato, po kliku uporabnika na ustrezen gumb, preko AJAX-a pošlje na strežnik.
Uporaba knjižnice JQuery nam definicijo dialogov poenostavi:

```
////////////////////////////////////////////////
//    Add new task button click function
////////////////////////////////////////////////
$('#add-new-task').click(function(e) {
    e.preventDefault();
    $('#newTaskDialog').dialog('open');
});
////////////////////////////////////////////////////////////
//   Add new task dialog
```

```
    ////////////////////////////////////////////////////////
    $('#newTaskDialog').dialog({
        autoOpen: false,
        modal: true,
        draggable: true,
        width: 500,
        buttons: {
            "Prekliči":
                function() {
                    $(this).dialog('close');
                },
            "Dodaj":
                function() {
                    $(this).find('form').submit();
                }
        }

    });
```



Slika 12: Dialog za vnos nove naloge.

5.4. Testiranje delovanja spletne aplikacije

Testiranje aplikacije je zelo pomemben proces razvoja aplikacije, kateremu razvijalci velikokrat posvetijo premalo pozornosti. Posledica pomanjkanja testiranja je navadno nestabilna in nezanesljiva rešitev. Nerealno je pričakovati, da bo razvijalec ob prvem poizkusu pripravil rešitev, ki bo primerna za produkcijsko okolje. Pri velikih projektih in sodobnih metodologijah razvoja informacijskih sistemov je testiranje sistematizirano in strogo definirano.

Testiranje spletne aplikacije ISVRS smo izvajali na več načinov. Prvo fazo testiranja smo izvedli ob samem razvoju aplikacije. Novo napisan modul aplikacije je bilo potrebno ročno testirati in preveriti ali deluje tako kot mora. Takšen način testiranja je dolgotrajen in razvijalcu vzame veliko dragocenega časa, namreč ob vsaki spremembi kode je dotični modul potrebno ponovno testirati. Zato smo na delih, kjer se je to dalo, uporabili enotske teste oziroma teste enot.

Testi enot so majhni delčki kode znotraj naše aplikacije ali samostojni programčki, ki preverjajo pravilnost delovanja posameznih funkcij znotraj aplikacije. Z njimi preverimo obnašanje posameznih funkcij, ko jih podvržemo različnim scenarijem.

Slika 13: Z dobrim testiranjem lahko odkrijemo še tako skrite napake in pomanjkljivosti.

Spletne aplikacije oziroma obličje sistema je zelo težko ali skoraj nemogoče testirati z enotskimi testi. Zato smo v ta namen izdelali seznam vnaprej pripravljenih primerov uporabe (testcase). Nastal je dokument, ki je vseboval testni scenarij za testiranje zaslonskih mask in dialogov uporabniškega vmesnika.

Primer testnega scenarija dialoga za vnos novega gospodinjstva.

ID	Opis testa	Pričakovan razplet	Dejanski razplet	Izvajalec	Datum
1	Ne vnesemo nobenega podatka v dialog in pritisnemo gumb dodaj.	Aplikacija nas opozori, da nismo vnesli vseh potrebnih podatkov.	Pričakovan razplet.	Eldin Velagić	1.12.2010
2	Ne vnesemo nobenih podatkov in pritisnemo gumb prekliči.	Dialog se zapre.	Pričakovan razplet.	Eldin Velagić	1.12.2010
3	Vnesemo vse potrebne podatke a ne izberemo nosilca gospodinjstva. Pritisnemo gumb shrani.	Aplikacija nas opozori, da nismo izbrali nosilca gospodinjstva.	Pričakovan razplet.	Eldin Velagić	1.12.2010
4	Vnesemo vse potrebne podatke in za nosilca gospodinjstva izberemo osebo, ki je že nosilec nekega gospodinjstva.	Aplikacija nas opozori, da je izbrana oseba že nosilec drugega gospodinjstva in da je dotična oseba lahko nosilec samo enega gospodinjstva.	Napaka. Kljub temu, da je oseba že nosilec drugega gospodinjstva se vseeno doda kot nosilec novega gospodinjstva.	Eldin Velagić	1.12.2010
5	Vnesemo vse potrebne podatke in za nosilca gospodinjstva izberemo osebo, ki je član drugega gospodinjstva, a ni nosilec.	Aplikacija nas vpraša ali res želimo člana prenesti iz obstoječega gospodinjstva v novo gospodinjstvo.	Pričakovan razplet.	Eldin Velagić	1.12.2010
6	Vnesemo vse potrebne podatke in za nosilca gospodinjstva izberemo osebo, ki je član drugega gospodinjstva, a ni nosilec. Ko nas aplikacija vpraša ali res želimo člana prenesti iz obstoječega	Oseba se prenese v novo gospodinjstvo in postane nosilec novega gospodinjstva. Odpremo podrobnosti starega gospodinjstva in oseba tam več ni vidna.	Pričakovan razplet.	Eldin Velagić	1.12.2010

	gospodinjstva v novo gospodinjstvo, pritisnemo DA.				
7	Vnesemo vse potrebne podatke in za nosilca gospodinjstva izberemo osebo, ki je član drugega gospodinjstv,a a ni nosilec. Ko nas aplikacija vpraša ali res želimo člana prenesti iz obstoječega gospodinjstva v novo gospodinjstvo, pritisnemo NE.	Dialog z vprašanjem se zapre in izberemo lahko novo osebo.	Pričakovan razplet.	Eldin Velagić	1.12.2010
8	Vnesemo vse potrebne podatke in za vnos nosilca gospodinjstva kliknemo na gumb dodaj novo osebo.	Odpre se dialog za vnos nove osebe.	Pričakovan razplet.	Eldin Velagić	1.12.2010
9	Vnesemo vse potrebne podatke in izberemo nosilca gospodinjstva, ki še ni nosilec. Kliknemo na gumb dodaj.	Ustvari se novo gospodinjstvo. Dialog se zapre in v seznamu je vidno tudi novo gospodinjstvo.	Pričakovan razplet.	Eldin Velagić	1.12.2010

Tabela 4: Testni scenarij dialoga za vnos novega gospodinjstva.

6. Sklepne ugotovitve

S pisanjem diplomske naloge sem želel predstaviti postopek izdelave informacijskega sistema. Cilj je bil izdelati aplikacijo, ki se po zagovoru diplomskega dela ne bo zavrgla, ampak se bo dejansko uporabljala. Ker je bila spletna aplikacija obsežnejše narave, je razvoj potekal v projektni skupini in bil razdeljen na razvoj obličja in razvoj zaledja sistema. Sam sem razvijal obličje sistema in se pri tem veliko naučil. Podrobneje sem uspel spoznati JavaScript, knjižnico JQuery in arhitekturo MVC, ter se praktično srečati s celotnim procesom razvoja informacijskega sistema. Spoznal sem, da je pri razvoju IS zelo pomembno, kakšna je komunikacija z naročnikom. Začetnim fazam razvoja je tako potrebno posvetiti več pozornosti in podrobno prisluhniti željam naročnika, namreč dobra začetna analiza lahko prihrani veliko časa in nepotrebnega napora pri kasnejšem programiranju.

Sam postopek razvoja je potekal brez večjih težav. Naročnik je natanko vedel kaj želi imeti, tako, da smo že po nekaj srečanjih naredili natančen načrt. Na žalost, zaradi pomanjkanja časa, v načrt nismo mogli vključiti vseh funkcionalnih zahtev, smo pa aplikacijo definirali tako, da je odprta za morebitne posodobitve in razširitve.

Največja težava je bila pripraviti uporabniški vmesnik tako, da bo deloval enako v vseh najbolj priljubljenih spletnih brskalnikih. Težave so se pojavljale predvsem v brskalniku Internet Explorer. Rešili smo jih tako, da smo uporabili ločene CSS in JavaScript datoteke za Internet Explorer. Za podporo starejših brskalnikov se nismo odločili.

Z izdelavo diplomske naloge sem pridobil praktično znanje o postopku razvoja informacijskega sistema in prepričan sem, da mi bo pridobljeno znanje koristilo tudi v prihodnje.

7. Priloge

7.1. Kazalo slik

7.2. Kazalo tabel

8. Viri

[1] Statut Islamske skupnosti v Republiki Sloveniji, Ljubljana, Islamska skupnost v Republiki Sloveniji, 2010, str: 5, 35.

[2] (2010) Microsoft Visual Studio 2010. Dostopno na: http://www.microsoft.com/business/smb/sl-SI/strezniki-in-orodja/visual-studio-pro.mspx

[3] (2010) Adobe Photoshop - Wikipedija, prosta enciklopedija. Dostopno na: http://sl.wikipedia.org/wiki/Adobe_Photoshop

[4] (2010) ASP.NET MVC Overview: The Official Microsoft ASP.NET Site. Dostopno na: http://www.asp.net/mvc/tutorials/asp-net-mvc-overview-cs

[5] (2010) Mvc Design Pattern. Dostopno na: http://www.infosum.net/programming/mvc-design-pattern.html

[6] (2009) Razlaga MVC (Model View Controller) vzorca. Dostopno na: http://odkrivanjetoplevode.blogspot.com/2009/07/razlaga-mvc-model-view-controller.html

[7] (2010) Ajax (programiranje) - Wikipedija, prosta enciklopedija. Dostopno na: http://sl.wikipedia.org/wiki/Ajax_%28programiranje%29

[8] (2010) Programski jezik JavaScript. Dostopno na: http://zaversnik.fmf.uni-lj.si/Gradiva/JavaScript/

[9] (2010) JavaScript - Wikipedija, prosta enciklopedija. Dostopno na: http://sl.wikipedia.org/wiki/JavaScript

[10] (2010) JQuery. Dostopno na: http://ussi.feri.uni-mb.si/index.php/vaje/vaja08/123-uvod-v-jquery

[11] (2010) CSS. Dostopno na: http://zaversnik.fmf.uni-lj.si/Gradiva/CSS/

[12] (2010) Črtna koda - Wikipedija, prosta enciklopedija. Dostopno na: http://sl.wikipedia.org/wiki/%C4%8Crtna_koda

[13] (2010) Čitalec kartic in čitalec črtnih kod - MaFiraWiki. Dostopno na: http://wiki.fmf.uni-lj.si/wiki/%C4%8Citalci_kartic_in_%C4%8Ditalci_%C4%8Drtne_kode

[14] (2006) Microsoftov ASP.NET, drugič. Dostopno na: http://www.monitor.si/clanek/microsoftov-asp-net-drugic/

[15] (2010) ASP.NET Master Pages. Dostopno na: http://msdn.microsoft.com/en-us/library/wtxbf3hh.aspx