UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Plaskan

Razvoj prototipa iPhone aplikacije za upravljanje z nalogami

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2010

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: 00008/2010
Datum: 01.10.2010

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

Kandidat: **BLAŽ PLASKAN**

Naslov: **RAZVOJ PROTOTIPA IPHONE APLIKACIJE ZA UPRAVLJANJE Z NALOGAMI**

DEVELOPMENT OF IPHONE PROTOTYPE APPLICATION FOR TASK MANAGEMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Predstavite delovanje prototipa poslovne aplikacije namenjene pregledovanju ter upravljanju z nalogami znotraj sistema MG.
Predstavite napravo iPhone in operacijski sistem, ki ga naprava iPhone poganja. Opišite še razvojno okolje, v katerem bo razvit prototip poslovne aplikacije, namenjene pregledovanju ter upravljanju z nalogami. Glavni del diplomskega dela naj bo namenjen razlagi delovanja prototipa. Sam postopek razvoja tega prototipa razdelite na več sklopov oziroma modulov. Sem naj spadajo omrežni, podatkovni, UI, entitetni modul ter modul orodij. Predstavite tudi pristope grajenja uporabniškega vmesnika.

Mentor:

prof. dr. Saša Divjak

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
UNIVERZA V LJUBLJANI
2

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

diplomskega dela

Spodaj podpisani **Blaž Plaskan**,

z vpisno številko **63060343**,

sem avtor diplomskega dela z naslovom:

Razvoj prototipa iPhone aplikacije za upravljanje z nalogami.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  prof. dr. Saše Divjak

- 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 1.12.2010 Podpis avtorja:

Zahvala

Sprva se iskreno zahvaljujem prof. dr. Saši Divjak, za pomoč, jasna navodila, mnenja ter strokovno svetovanje med nastajanjem diplomskega dela.

Največjo zahvalo si zaslužita starša Martina in Božidar, ki sta mi omogočila študij v Ljubljani, me sproti spodbujala ter nudila vso podporo, da sem dosegel še en življenjski cilj.

Ob tej priložnosti se moram zahvaliti tudi podjetju Agito d.o.o., ki me je tekom našega sodelovanja izšolalo v boljšega in bolj izkušenega razvijalca.

Za konec pa se moram zahvaliti še gospodu Matjažu Gorjupu, ki mi je ponudil možnost, da sem s skupnim sodelovanjem razvil prototip iPhone aplikacije in se ob tem naučil novega programskega jezika ter spoznal novo platformo.

Kazalo

SEZNAM UPORABLJENIH KRATIC IN OKRAJŠAV

iOS – Operating System (črka i je pripona, ki jo uporablja Apple pri svojih produktih)

SDK – Software Development Kit

UI – User Interface

SOAP – Simple Object Access Protocol

XML – Extensible Markup Language

HTTP – HyperText Transfer Protocol

MVC – Model-View-Controller

PDF – Portable Document Format

MP3 – MPEG-1/MPEG-2 Audio Layer 3

MPEG – Moving Picture Experts Group (standard kodiranja)

WAV – Waveform Audio

AIFF – Audio Interchange File Format

AAC – Advanced Audio Coding

MB – Mega Byte

3GS – iPhone 3rd Generation Speed

WLAN – Wireless Local Area Network

SMS – Short Message Service

MHz – Mega Hertz

eDRAM – embedded Dynamic Random Access Memory

GB – Giga Byte

GPS – Global Positioning System

2D – Two Dimensional

3D – Three Dimensional

GHz – Giga Hertz

IDE – Integrated Development Environment

USB – Universal Serial Bus

SQL – Structured Querry Language

WPF – Windows Presentation Fundation

Povzetek

V diplomskem delu je predstavljeno delovanje iPhone prototipa poslovne aplikacije, namenjene pregledovanju ter upravljanju z nalogami znotraj sistema MG. Kratica MG predstavlja ime projekta.

Outlook vtičnik MG na prvi pogled izgleda kot nekakšno orodje za upravljanje z elektronsko pošto. Vsebuje zgoraj dohodno in spodaj odhodno področje za naloge, na sredini pa še področje za naloge, ki si jih lasti uporabnik. Naloge lahko med področji po želji razporejamo. Seveda za premike nalog obstajajo tudi pravila, ki jih sistem MG uporabniku prikaže ob premikanju naloge. Na podlagi Outlook vtičnika je potekal razvoj iPhone prototipa.

Diplomsko delo na začetku opisuje napravo iPhone, operacijski sistem, ki ga naprava poganja, ter na koncu še razvojno okolje, v katerem je bil prototip razvit. Glavni del diplomskega dela je namenjen razlagi delovanja prototipa. Sam postopek razvoja tega prototipa je bil razdeljen na več sklopov oziroma modulov. Sem spadajo omrežni, podatkovni, UI, entitetni modul ter modul orodij. Razdelitev na posamezne module pripomore k lažji predstavi oziroma razumevanju delovanja prototipa. Predstavljena sta tudi dva pristopa grajenja uporabniškega vmesnika. Nekaj besed je namenjenih nastavitvam aplikacije, kakšne možnosti implementacije obstajajo ter prednosti in slabosti vsake. Proti koncu diplomskega dela so opisane tudi večje težave ter rešitev le-teh med razvojem prototipa. Nazadnje je na kratko predstavljen še strežniški del sistema MG.

Ključne besede

razvoj, poslovna aplikacija, uporabniški vmesnik, nit, servis, iPhone, iOS, Apple, entiteta

Abstract

The thesis presents how the business iPhone application prototype for managing task in the MG system works. The MG abbreviation stands for the project name.

The Outlook plug-in at first glance looks like just another e-mail manipulation tool. It has an incoming and outgoing region, as well as the middle user region for tasks. Users can freely manipulate these tasks by dragging and dropping them between regions. The MG system helps users by highlighting the available target drop positions of the task. Based on this plug-in the iPhone prototype was developed.

In the beginning the iPhone device is introduced together with the iOS operating system and the SDK for developing iPhone apps. The main part of the thesis describes the full working logic of the MG iPhone app prototype. The development of the prototype consists of the following modules: network, data, UI, entity and tools module. The modularity is introduced to help visually explain the logic behind this prototype. Described in the main part are also two ways of building UI and some samples of storing the app setting on the iPhone device. Almost at the end some problems and their solutions are described. The last thing is a description of the MG system's server configuration.

Key words

development, business application, user interface, thread, servis, iPhone, iOS, Apple, entity

1. Uvod

1.1. MATIVY MG

MATIVY Management GmbH je avstrijsko podjetje, ki se ukvarja predvsem s svetovanjem. Konec leta 2009 je podjetje MATIVY začelo razvijati univerzalno orodje, imenovano MG. Orodje MG je v prvi vrsti namenjeno vodilnim osebam v podjetju. Njegov namen je poenostaviti in s tem razbremeniti uporabnika pri vsakdanjih opravilih znotraj vodenja poslovnega procesa. Hkrati omogoča vodstvu večji nadzor nad poslovanjem. S pomočjo tega orodja lahko uporabnik lažje sledi nalogam, ki so mu bile dane, ali pa le-te razdeli naprej podrejenim osebam. Naloge se lahko razdelijo v podnaloge in s tem lahko tematsko razdrobimo neko splošno nalogo. Orodje prav tako vsebuje posebno poglavje, namenjeno izključno za izvajanje sestankov. Uporabniku tako omogoča na zelo enostaven način vključiti svoje naloge v sestanek, ki ga skliče sam ali pa je povabljenec na sestanku. Prav tako se na sklicanem sestanku lahko določijo nove naloge, ki lahko izhajajo tudi iz že obstoječih, in se razporedijo udeležencem sestanka. Prednost tega orodja v primerjavi s konkurenčnimi orodji je v tem, da se tukaj vse naloge na enostaven način povezujejo. Za povezovanje nalog ne skrbi uporabnik, ampak sistem MG. Orodje MG je v trenutni obliki razvito kot vtičnik za Microsoftovo orodje Outlook in deluje izključno preko spleta. Vzporedno se je razvijala tudi aplikacija za napravo Blackberry. Za razliko od vtičnika, aplikacija, namenjena napravi BlackBerry, vsebuje le poglavje upravljanja z nalogami, saj je za upravljanje s sestanki naprava nekoliko premajhna.

1.2. Predstavitev problema

Zaradi večanja števila iPhone naprav v poslovnem svetu se je nekoliko kasneje pojavilo tudi povpraševanje po iPhone aplikaciji, ki bi do neke mere nadomestila Outlook vtičnik na osebnem računalniku, podobno kot pri napravi BlackBerry. Medtem ko je BlackBerry aplikacija že pripravljena za uporabo, bo iPhone aplikacija zaenkrat zgolj prototip. Nekoliko kasneje pa se bo na podlagi te iPhone aplikacije razvil tudi prototip iPad aplikacije, ki bo vseboval tudi poglavje, namenjeno izvajanju sestankov. Prototip mora biti sposoben upravljati z nalogami v sistemu MG, poleg tega mora prototip delovati tudi brez omrežne povezave. Zahteva za delovanje brez omrežne povezave prihaja s strani poslovnih potovanj, takrat dostop do spleta ni vedno na voljo. Ker pa na poslovnem potovanju ni vedno časa za pisanje nalog, mora aplikacija podpirati tudi snemanje zvoka oziroma mora imeti vgrajen nekakšen diktafon.

1.3. Pričakovani rezultati

Prototip mora biti uporabniku prijazen ter odziven. Uporabniški vmesnik mora biti pregleden ter sposoben uporabniku prikazati čim več podatkov hkrati. Sinhronizacija podatkov naj bo uporabniku nemoteča. Poskrbljeno mora biti tudi za shranjevanje podatkov, v primeru, da uporabnik ostane brez povezave. Podobno velja tudi za primer, ko uporabnik nehote zapusti aplikacijo, ali pa upravljanje z aplikacijo prestreže dohoden klic. Aplikacija mora biti sposobna snemati zvočne posnetke in jih v obliki naloge s priponko shraniti na strežnik MG.

2. Apple iPhone

2.1. Apple

Apple je ameriško multinacionalno podjetje, ki se ukvarja z razvojem ter prodajo elektronskih naprav ter programske opreme. Elektronske naprave zavzemajo osebne računalnike, strežnike, mobilne telefone ter prenosne multimedijske predvajalnike. V sklopu programske opreme v ospredje postavljajo predvsem operacijske sisteme, ki jih njihove naprave poganjajo. Poleg operacijskih sistemov razvijajo tudi programe za manipulacijo in obdelavo multimedijskih vsebin ter programe, namenjene oblikovanju teksta, predstavitev ter urejanju tabel.

Leta 2007 je Apple na svetovnem trgu predstavil svoj prvi mobilni telefon imenovan »iPhone«. S to napravo je Apple tudi odprl velika vrata predvsem samostojnim razvijalcem. Zanj so pri Applu razvili namenski operacijski sistem »iPhoneOS«, kasneje preimenovan v »iOS«, ki je pisan v programskih jezikih »C«, »C++« in »Objektivni C«. Kot pripomoček razvijalcu so predstavili tudi razvojno okolje za razvoj iPhone aplikacij. Preko njihovega sistema za objavo aplikacij lahko vsak razvijalec svoje aplikacije ponudi iPhone uporabnikom ter s tem tudi nekaj zasluži. Kljub odprtosti naprave ter operacijskega sistema je potrebno pri razvoju upoštevati kar nekaj pravil ter omejitev, predpisanih s strani Appla. Po končanem postopku objave s strani razvijalca sledi testiranje aplikacije. Postopek testiranja opravi Apple. V primeru, da odkrijejo kakšno neskladnost z njihovimi zahtevami, zavrnejo aplikacijo. Aplikacijo je nato potrebno ustrezno popraviti in ponoviti postopek objave.

2.2. iPhone

iPhone je skupek mobilnega telefona ter zmogljive multimedijske naprave. Naprava omogoča vse osnovne funkcije klasičnega mobilnega telefona, kot so na primer telefonski klici ter SMS sporočila, multimedijski del naprave pa omogoča predvajanje videa, gledanje slik, predvajanje glasbe ter poganjanje velike množice namenskih aplikacij. Prav tako omogoča snemanje videa ter zvoka. K snemanju videa štejemo tudi zmožnost zajemanja fotografij. Operacijski sistem lahko poganja veliko namenskih aplikacij. Te so lahko tako igre kot poslovne aplikacije ali pa različna orodja, ki uporabniku naprave olajšajo vsakdanja opravila. Zaščito proti odrgninam na zaslonu nudi vgrajena zgornja plast stekla. Vgrajeno ima tudi zelo zmogljivo baterijo. Za komunikacijo z internetom imamo več možnosti. Naprava se lahko povezuje na medmrežje preko vgrajene kartice Wi-Fi, preko vgrajenega modema, namenjenega za klice ali pa preko povezave USB. Podpira tudi BlueTooth tehnologijo, ki pa je omejena.

2.2.1. Strojna oprema (iPhone 3GS, Avgust 2010)

Ker lahko naprava iPhone poganja igre ter prikazuje multimedijske vsebine, potrebuje kar zmogljivo strojno opremo. Glede na to, da spada iPhone med mobilne naprave, ima zelo velik zaslon, ki meri po diagonali devet centimetrov, njegova resolucija pa znaša štiristo osemdeset krat tristo dvajset zaslonskih pik. Med strojno opremo štejemo tudi zmogljivo kamero za zajem slik ter videa. Njegov procesor, omejen na delovno frekvenco 600 MHz od 833 MHz maksimalne, je dodatno opremljen s 265 MB eDRAM glavnega pomnilnika. S tem je izboljšana odzivnost same naprave ter pripomore k izvajanju večopravilnosti. Za razne multimedijske vsebine ter ostale podatke ima naprava na voljo šestnajst ali dvaintrideset GB pomnilnega prostora tipa FLASH. Za prikazovanje ima na voljo tudi zmogljiv grafični procesor. Uporabnik lahko z napravo komunicira preko kapacitivnega zaslona občutljivega na dotik, ki podpira tudi več dotikov hkrati. Prav tako ima vgrajen sprejemnik signala GPS ter digitalni kompas. V primeru, ko sprejemnik GPS in digitalni kompas sodelujeta, lahko uporabniku naprava posreduje zelo natančno trenutno pozicijo ter naravnanost glede na smeri neba.

2.2.2. iOS - Programska oprema

iOS je operacijski sistem podjetja Apple, namenjen uporabi v njihovih multimedijskih napravah. Ker so pri Applu razvili tudi svoj mobilni aparat, so prepovedali uporabo operacijskega sistema iOS na vseh napravah tujega izvora. iOS izhaja iz njihovega operacijskega sistema, namenjenega za osebne računalnike, le da je prirejen za multimedijske naprave. Vsebuje vsa potrebna programska orodja za izvajanje osnovnih opravil, ki jih uporabnik pričakuje od mobilnega aparata in še veliko več. Sam uporabniški vmesnik je grajen na osnovi direktne manipulacije z uporabo gest z več hkratnimi dotiki. Rezultat tega je mobilni aparat z eno samo tipko, ki te ob pritisku vrne v glavni meni iOS. Arhitektura iOS temelji na štirih slojih. Prvi „Core OS“ ter drugi „Core Services“ sloj vsebujeta vse osnovne funkcije iOS sistema. Tretji sloj je „Media“ sloj. Ta sloj vsebuje temeljne tehnologije za podporo 2D in 3D risanja, videa ter glasbe. Zadnji sloj imenovan „Cocoa Touch“ pa vsebuje vsa osnovna ogrodja za implementacijo kolekcij, upravljanje z datotekami, upravljanje z omrežjem in tako naprej. Prav tako so na zadnjem sloju definirani vsi osnovni gradniki uporabniškega vmesnika. Sloji si sledijo od najnižjega k najvišjemu. Pri odločanju, katero dodatno tehnologijo je še potrebno dodati v aplikacijo, je najbolje začeti s čim višjim slojem, kajti velika verjetnost je, da je tam na najlažji način že implementirana tehnologija, ki jo rabi razvijalec, kar pa mu tudi omogoča hitrejši razvoj svoje aplikacije. V nižje nivoje se je pametno spuščati le, če nam ne ustreza implementacija v višjem sloju.

3. Razvojno okolje iOS SDK

Razvojno okolje iOS SDK je del paketa razvojnega okolja Xcode. Namenjeno je razvoju iPhone aplikacij. Vključuje veliko orodij, ki so v prvi vrsti namenjena temu, da razvijalcu omogočajo čim hitrejši razvoj aplikacij. Paket vsebuje tudi zelo zaželena orodja za testiranje odzivnosti aplikacij ter porabe sistemskih resursov. Razvijalec lahko s pomočjo teh orodij sproti preverja kakovost svojega dela in že na začetku pazi na pravilen pristop k razvoju aplikacije. V primeru, da razvijalec nima na voljo fizične naprave za testiranje, so v paketu prav tako prisotni razni simulatorji teh naprav. Seveda pa mora biti razvijalec pozoren tudi na odziv simulatorja, ker je ta drugačen kot sama fizična naprava. Kot primer, procesor v standardnem prenosnem računalniku deluje pri frekvenci okoli dva GHz, medtem ko na primer iPhone (iPhone 3GS, Avgust 2010) le pri šeststo MHz. Simulator v tem primeru uporabi zmožnosti prenosnega računalnika.

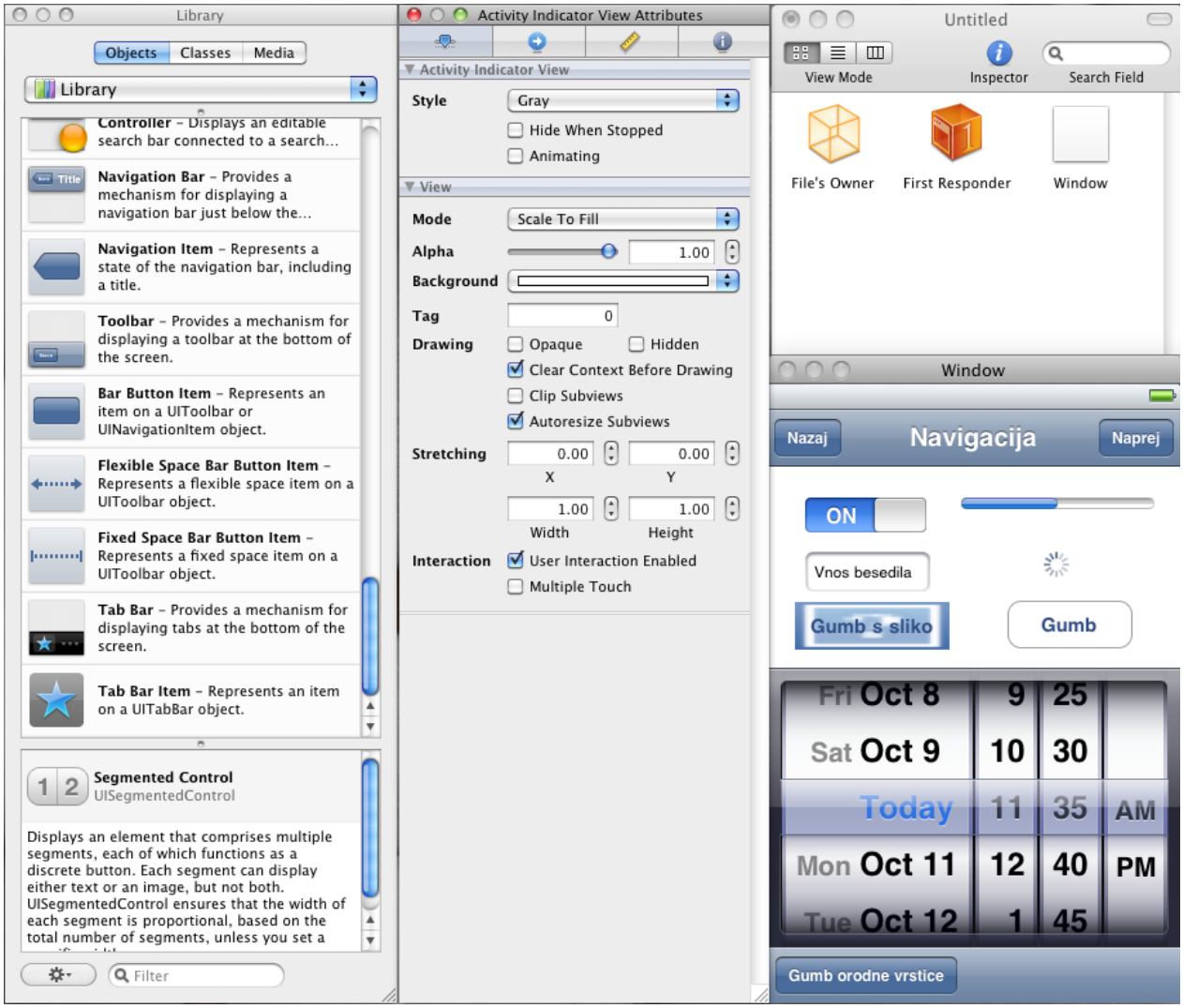
3.1. Xcode

Xcode je glavni paket s strani podjetja Apple, ki je namenjen razvijalcem, da lahko razvijajo aplikacije za operacijske sisteme Apple. Vsebuje veliko orodij, ki razvijalcu pomagajo pri razvoju. Najnovejšo (Xcode 3.2, Avgust 2010) različico tega paketa si lahko vsak uporabnik povleče z njihove uradne strani, kajti na mediju operacijskega sistema le-ta ni vključena. Seveda je potrebno poudariti tudi to, da je ta paket namenjen samo osebnim računalnikom podjetja Apple. Paket poleg orodij vsebuje tudi večino dokumentov, ki so namenjeni razvijalcem v pomoč pri razvoju. V paketu je prisoten tudi program za izdelavo uporabniškega vmesnika, imenovan „Interface Builder“. Glavni program paketa je IDE, ki podpira veliko programskih jezikov. Med programske jezike spada v prvi vrsti programski jezik „Objektivni-C“, poleg tega pa še C, C++, Fortran, Java, AppleScript, Python in Ruby. IDE vsebuje tudi zmogljiv razhroščevalnik. Z njim lahko razvijalec razhroščuje svojo aplikacijo med delovanjem, tudi če poganja aplikacijo preko iPhone naprave, priklopljene na vhod USB osebnega računalnika.

3.2. InterfaceBuilder

Interface Builder je programsko orodje za izdelavo uporabniškega vmesnika za razne aplikacije. Orodje omogoča razvijalcem, da v okolju Cocoa in Carbon na enostaven način gradijo uporabniški vmesnik s pomočjo računalniške miške. Ob zagonu programa se prikažejo okna, ki vsebujejo informacije o naboru gradnikov, zbirko trenutno uporabljenih gradnikov v projektu ter okno, kjer razvijalec vidi predogled aplikacije. Prav tako ima razvijalec tudi tukaj možnost definiranja raznih vgrajenih ali lastnih animacij ter jih uporabi nad gradnikih uporabniškega

vmesnika. Potrebno pa je poudariti, da je ne glede na to, da enostavno gradimo UI, potrebno v ozadju še vseeno vse napisati na roke.

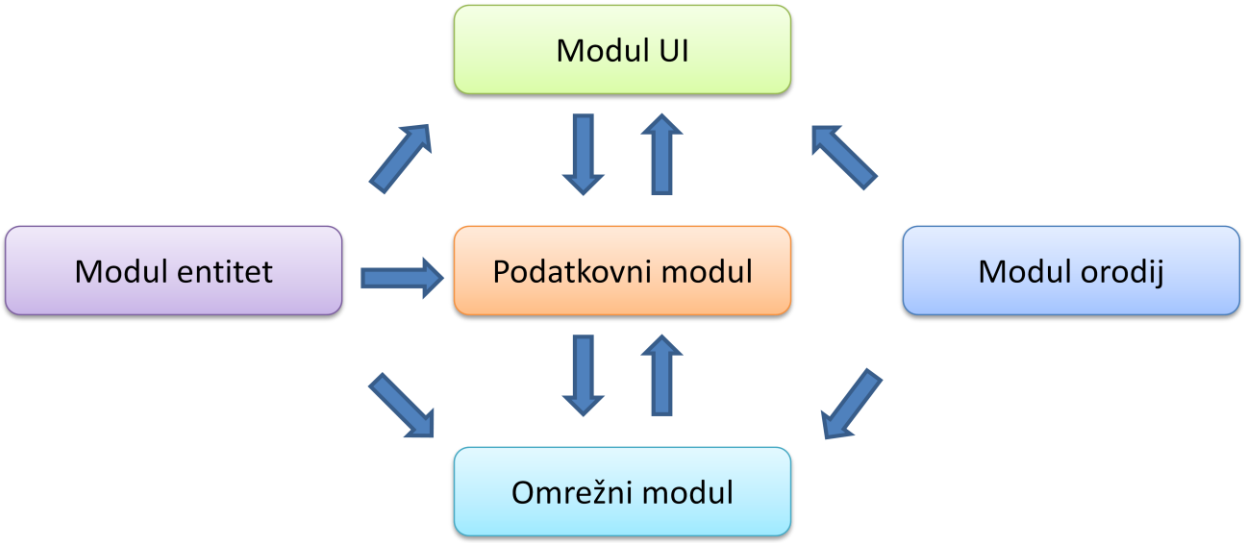


Slika 1. Interface Builder.

4. iPhone prototip MG

4.1. Razdelitev projekta na logične sklope oziroma module

Za potrebe lažjega razumevanja ter predstave je aplikacija razdeljena na posamezne module. V samem projektu je to razvidno po razvrstitvi map in podmap, ki vsebujejo objekte. Moduli so med sabo tematsko ločeni. Omrežni modul skrbi za prenos podatkov po omrežju, medtem ko ima UI modul definirano vso potrebno logiko za prikaz uporabniškega vmesnika. Podatkovni modul upravlja s podatki, modul entitet vsebuje objekte, ki jih aplikacija uporablja namesto podatkovne baze in njenih vrstic. Nazadnje je prisoten še modul orodij, ki vsebuje razne metode ter objekte, ki se rabijo v vseh ostalih modulih in jih ni bilo smiselno postaviti drugam.



Slika 2. Moduli in medsebojne povezave.

4.1.1. Omrežni modul

Kot pove že samo ime, se ta modul uporablja za izmenjavo informacij med strežnikom in napravo iPhone. Ker aplikacija za delovanje potrebuje zelo raznolike podatke z različnih servisov s strani strežnika, je tudi ta modul naprej porazdeljen na posamezne servise, ki se prekrivajo s tistimi na strežniku. Omrežni modul se deli na glavni servis ter servise seje, nalog in šifrantov. Protokol, preko katerega se informacije pretakajo, je v osnovi SOAP.

4.1.1.1. Protokol SOAP

Protokol SOAP je namenjen izmenjavi strukturiranih informacij v implementaciji omrežnih servisov v računalniških omrežjih. Kot format sporočila se uporablja format XML. Za prenos po omrežju se v tem primeru še zanaša na protokol HTTP.

Protokol SOAP sestavljajo trije deli:

- ovojnica, ki definira, kaj sporočilo vsebuje in kako ga je potrebno procesirati,
- skupek kodirnih pravil, ki opisujejo tipe podatkov, ki jih aplikacija zahteva,
- unija, ki predstavlja klice procedur in odgovorov.

Slika 3. Sestava sporočila SOAP.

4.1.1.2. Servisi znotraj omrežnega modula

4.1.1.2.1. Glavni servis

Glavni servis skrbi za zaključevanje zahtevka ter pošiljanje le-tega na strežnik. Prav tako je njegova naloga sprejemanje in preverjanje prispelih podatkov. Vsi ostali servisi znotraj omrežnega modula komunicirajo preko glavnega servisa s strežnikom. Če pogledamo zadevo z druge strani, ostali servisi pripravijo potrebne podatke in jih pošljejo glavnemu servisu, le-ta poskrbi za prenos podatkov in vrne nazaj prejete podatke. V primeru napake glavni servis ne vrne nič, obenem pa sproži postopek za prikaz napake uporabniku. Vse metode znotraj servisov so statične. Uporaba statičnih metod je razvijalcu v veliko pomoč, saj ni potrebno toliko paziti na porabo pomnilnika. Klici statičnih metod ne zahtevajo grajenja nekega objekta samo zato, da se metode tega objekta lahko uporabijo.

Glavni servis prejme podatke o telesu sporočila ter o naslovniku. S pomočjo teh informacij sestavi sporočilo in ga pošlje preko omrežne povezave na strežnik. Ob zaključevanju sporočila glavni servis sporočilu pripne standardno glavo ter v standardno ovojnico sporočila vstavi prejete podatke od ostalih servisov. Na sestavljen naslov servisa na strani strežnika nato pošlje zahtevek s sporočilom in čaka na odgovor. Po prejetem odgovoru ta servis preveri, če sporočilo vsebuje kakšno napako, vrnjeno s strani strežnika, ali pa povezave.

4.1.1.2.2. Servis seje

Servis seje skrbi za prijavo in odjavo uporabnika v sistem MG. Vsaka metoda znotraj servisa sestavi telo sporočila iz dinamičnih podatkov med izvajanjem aplikacije. Med postopkom prijave uporabnika v sistem, metoda, ki je zadolžena za prijavo, iz nastavitev naprave prebere podatke o uporabniku in sproži proces prijave. Dodatna funkcija tega servisa je tudi, da lahko sproži poizvedbo po uporabnikih v sistemu MG. Rezultat te dodatne poizvedbe je imenik znotraj aplikacije, ki nima nobene povezave s privzetim imenikom iPhone.

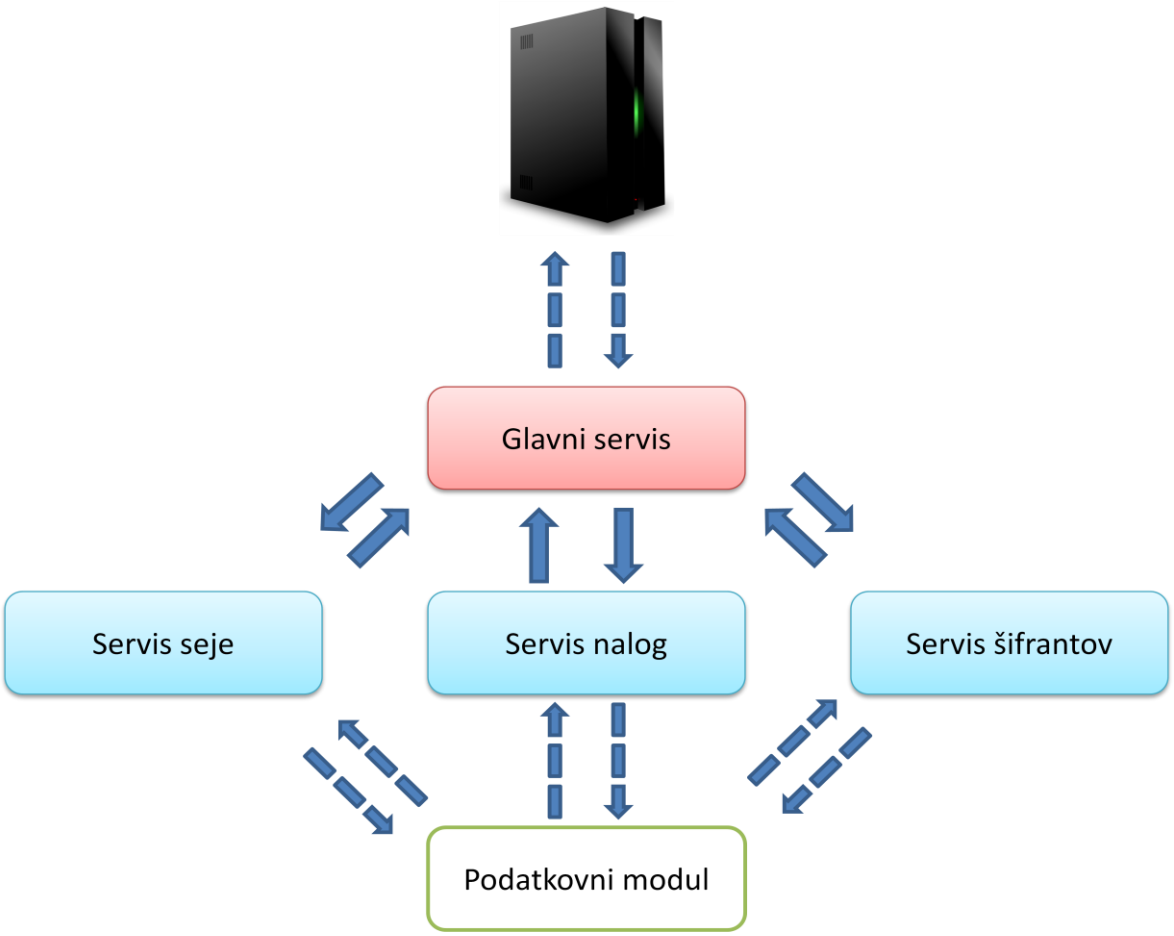
Glavna naloga servisa je pridobivanje šifre seje, ki se ob poizvedbah prenaša v sporočilu. Šifra seje se uporablja na strani strežnika za identifikacijo uporabnika. Po poteku seje je potrebno obnoviti šifro z vnovično prijavo v sistem.

4.1.1.2.3. Servis nalog

Servis nalog skrbi za poizvedbe preko glavnega servisa, ki se tičejo nalog v sistemu MG. Prav tako se preko tega servisa prenašajo priponke. Priponke se prenašajo na strežnik ločeno od sinhronizacije nalog. Ta pristop ne upočasni glavne sinhronizacije in s tem ne zadržuje uporabnika pri uporabi aplikacije.

4.1.1.2.4. Servis šifrantov

Servis šifrantov je prisoten izključno zaradi prenosa določenih šifrantov iz podatkovne baze MG, ki se potrebujejo ob delovanju aplikacije.



Slika 4. Povezava ostalih servisov z glavnim servisom.

4.1.2. UI modul

Modul uporabniškega vmesnika vsebuje vse potrebne objekte, ki sestavljajo aplikacijo z vidika uporabnika, oziroma vse tisto, kar uporabnik vidi na zaslonu naprave iPhone. Ker gre za poslovno aplikacijo, se na ekranu v večini primerov prikazujejo zgolj podatki. Zaradi tega je velika večina uporabniškega vmesnika predstavljenega s tabelami.

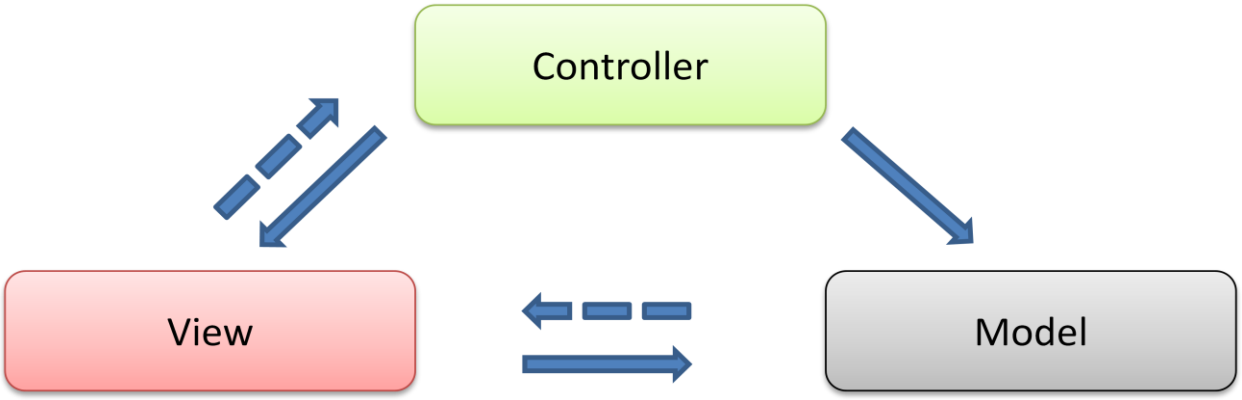
Kot je bilo že omenjeno, imamo dva načina gradnje uporabniškega vmesnika. V prvem primeru se vsi gradniki ustvarijo programsko oziroma s programsko kodo, v drugem primeru pa si lahko pomagamo z namenskim orodjem. Drugi primer nam omogoča, da gradnike enostavno povlečemo na zaslon in poljubno manipuliramo z njimi, ter jim nastavljamo lastnosti. Slaba stran drugega načina je še dodaten objekt, ki vsebuje podatke od gradnikov, zapisan v obliki XML, ki na koncu vpliva na velikost same aplikacije. Zapis XML, ki ga namensko orodje generira, je zelo neprijazen do razvijalca. (V tem primeru je za primerjavo vzeto podobno orodje imenovano Microsoft Expression Blend. Čeprav se v osnovi tehnologiji razlikujeta, sta obe orodji namenjeni gradnji UI.) Neodvisno od načina pristopa, moramo še vseeno v ozadju vse povezati programsko. Rezultat teh spoznanj je razvoj aplikacije usmeril v prvi način sestavljanja uporabniškega vmesnika, predvsem zato, ker ima programer veliko večjo kontrolo nad obnašanjem uporabniškega vmesnika.

Kot so pokazale že izkušnje v preteklosti, v večini primerov gradnja uporabniškega vmesnika glede na implementacijo ostale logike časovno porabi največ časa ter razvijalčevega potrpljenja.

4.1.2.1. Splošno

Gradnja uporabniškega vmesnika za iPhone temelji na vzorcu MVC. Vzorec MVC vsebuje tri glavne komponente:

- Model (podatki, ki jih želimo prikazati),
- View (vizualni elementi aplikacije),
- Controller (povezava zgornjih dveh komponent).



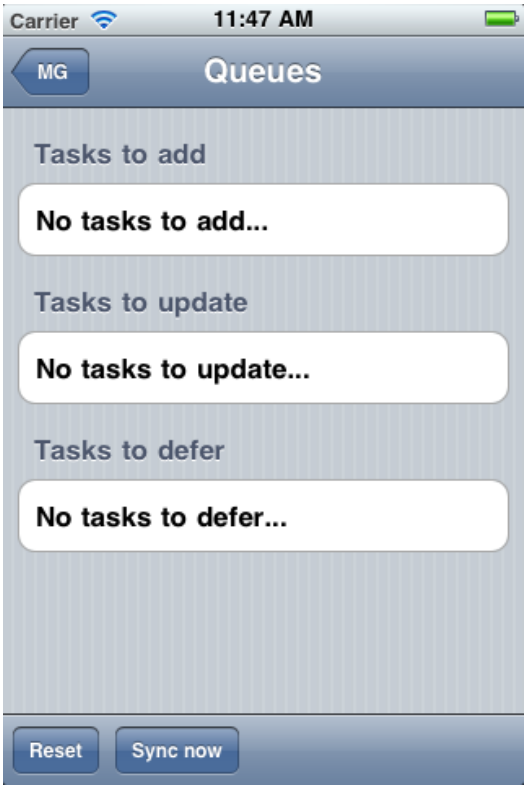
Slika 5. Komponente vzorca MVC.

Iz kratic je tudi razvidno, od kod izvira ime vzorca. Model nosi vse potrebne podatke, ki jih View rabi za prikaz. Controller pa služi kot povezava med Model ter View komponento, kajti View in Model komponenti ne vesta nič ena o drugi. Prav tako Controller skrbi za odzive na uporabnikove ukaze ter za prikaz komponent zaslona.

4.1.2.2. Gradniki

4.1.2.2.1. Osnovni gradniki

V tej aplikaciji je uporabljenih večina standardnih gradnikov, ki jih samo ogrodje od iPhone ponuja. Sem spadajo razni gumbi ter tabele, drsniki in tako dalje. Vsak osnovni gradnik ima veliko lastnosti, ki se jih lahko poljubno nastavi. Vzemimo za primer gumb. Gumbu se lahko nastavi barva, napis, velikost, oblika, vizualni učinek aktivnega stanja, lahko se mu določi poljubna slika namesto standardne oblike in tako dalje. Drugi primer je lahko tabela. Tabeli se lahko nastavi tip tabele, navaden ali tematsko grupiran tip, videz posamezne vrstice in tako naprej.



Slika 6. Primer grupirane tabele.

Da prikažemo gradnik na zaslonu, je v najkrajšem in najbolj preglednem primeru potrebno napisati tri vrstice programske kode. Prvo vrstico predstavlja konstruktor, ki gradnik ustvari. Nato gradniku nastavimo okvir, ki vsebuje podatke o velikosti gradnika in njegovi poziciji na zaslonu. Zadnja vrstica predstavlja klic metode, ki postavi gumb na zaslon.

```
UIButton helpButton = [UIButton buttonWithType:UIButtonTypeCustom];
[helpButton setFrame:CGRectMake(50.0, 294.0, 220.0, 49.0)];
[self.view insertSubview:helpButton belowSubview:highScoresButton];
```

Slika 7. Primer programske kode za ustvarjanje gumba.

4.1.2.2.2. Izpeljani oziroma prirejeni gradniki

Pri razvoju aplikacij pride včasih do primera, ko osnovni gradniki niso dovolj vsestranski, da zadostijo razvijalčevim potrebam. V tem primeru je potrebno vzeti gradnik, ki je v osnovi najbližji razvijalčevim zahtevam in ga primerno prirediti. Pri tej aplikaciji je bilo to potrebno narediti v primeru vrstice tabele. Vzet je bil osnovni gradnik vrstice tabele in se mu je dodalo

nekaj novih komponent za prikaz besedila ter nekaj lastnosti in metod, ki omogočajo uporabo teh novosti.

Slika 8. Zgoraj prirejena in spodaj osnovna vrstica tabele.

4.1.2.3. Prikazovanje podatkov

Povsod, kjer se srečamo z napravo, ki ima majhen zaslon, se vedno poraja vprašanje, na kakšen način prikazati uporabniku čim več podatkov. Hkrati moramo gledati tudi na to, da ni potrebno podatkov z zaslona brati s povečevalnim steklom.

Glede na to, da gre tukaj za poslovno aplikacijo, si lahko v veliki večini pomagamo in prikazujemo podatke tabelarično. V ogrodju iPhone se nahaja osnoven gradnik, ki predstavlja tabelo. Dobra stran omenjenega gradnika je njegova raznolikost in neobčutljivost na podane podatke.

4.1.2.3.1. Tabelarično prikazovanje podatkov

Ko je enkrat gradnik tabele dodan na zaslon, je le-temu potrebno v osnovi nastaviti dve vrednosti. Nastaviti je potrebno od tabele izvor podatkov ter objekt, ki pove, kakšni so podatki v zbirki. Ta objekt vsebuje vnaprej definirane metode, ki jih je potrebno prepisati, da je na koncu dosežen želeni rezultat. V teh metodah je definirano, koliko razdelkov ima tabela, koliko je vrstic v posameznem razdelku, kakšna je po videzu ena vrstica, kaj se zgodi v primeru, ko uporabnik izbere vrstico, in tako dalje.

Kot je bilo že omenjeno, nekatere tabele znotraj aplikacije vsebujejo prirejene gradnike vrstic. To je doseženo v metodi, ki je zadolžena za definicijo prikaza vrstice v tabeli. Namesto osnovnega gradnike tabele je na njegovem mestu ustvarjen nov izpeljani gradnik, prejšnji pa je zavržen.

4.1.2.3.2. Prikaz priponk

Za prikaz vseh podatkov pa ni zadostoval gradnik tabele. Kot primer vzemimo priponko, ki predstavlja sliko ali pa dokument PDF. V tem primeru je bil uporabljen gradnik, imenovan UIWebView, ki ima zmožnost prikazovati takšno vsebino. Kot dodatek ima že vgrajen tudi mehanizem za povečevanje in premikanje same vsebine.

4.1.2.4. Prehodi in animacije

Ogrodje iPhone vsebuje ogromno možnosti za prehode med pogledi ter animacije za večino gradnikov uporabniškega vmesnika. Po potrebi lahko razvijalec definira tudi svoje. V tej aplikaciji je uporabljen najbolj standarden prehod med pogledi. Definiran je pri navigaciji med dodajanjem pogleda na zaslon in aktiviranjem le-tega. Rezultat je podoben pomiku starega pogleda v levo, za njim pa se z desne strani pripelje novi.

Kot primer definiranja poljubne animacije lahko vzamemo navaden gumb, ki ga pomikamo po zaslonu. Gumb ima neko začetno točko. Nad pogledom nato sprožimo postopek animacije, kjer definiramo, koliko časa traja animacija, kakšno pot ima premik objekta ter kje je končna točka premika. Po osnovnih nastavitvah še samo poženemo animacijo in za vse ostalo poskrbi ogrodje iPhone.

4.1.2.5. Upravljanje z zvokom

Ker se v današnjem svetu vse stvari dogajajo z visokim tempom, je tudi pri tej aplikaciji prišla zahteva za vgradnjo nekakšne vrste diktafona. Rezultat te funkcije je višja hitrost zapisa neke naloge. Na mobilni napravi lahko oseba veliko hitreje posname kakšno nalogo v obliki zvočnega zapisa, kot pa vse podrobno natipka v tekstovni obliki. Posneto se lahko potem naknadno spremeni v tekstovno obliko, ko in če si posameznik vzame čas za to.

Ker je iPhone v osnovi namenjen telefonskim pogovorom, ima že vgrajen mikrofon in zvočnik. Temu sledi tudi ogrodje za upravljanje s to strojno opremo, ki je razvijalcu tudi na voljo. Ogrodje ponuja dve možnosti, predvajanje in snemanje zvočnega zapisa.

4.1.2.5.1. Predvajanje

Prvo vprašanje, ki se pojavi pri predvajanju posnetkov je, kakšne vse formate zvočnega zapisa naprava podpira. Ogrodje je tukaj do razvijalca zelo prijazno in ponuja cel spekter standardiziranih zvočnih zapisov, kot so MP3, WAV, AIFF, AAC in tako dalje. Potrebno je tudi omeniti, da je zvočni zapis zelo enostavno predvajati. Če želimo v aplikacijo vključiti predvajalnik zvoka, moramo v projekt vključiti knjižnice, kjer se nahajajo objekti, potrebni za izkoriščanje strojne opreme za predvajanje zvoka. Nato je potrebno ustvariti objekt predvajalnika. Predvajalniku moramo nato povedati, kakšnega tipa je zvočni zapis in le-tega podati kot vir podatkov. Na koncu le še preprosto sledi klic metode predvajalniku za začetek predvajanja. Sam predvajalnik vsebuje tudi vse standardne metode za upravljanje z zvokom, kot so na primer premor, stop in tako dalje.

4.1.2.5.2. Snemanje

Enako kot v primeru predvajanja zvočnega zapisa se tudi pri snemanju zvočnega zapisa poraja vprašanje formatov, ki jih naprava podpira. Pri snemanju zvočnega zapisa je ogrodje iPhone zelo omejeno. Potrebno je pripomniti, da za tem stojijo patenti in razne tožbe v primeru zlorabe ali uporabe brez licence, kajti zvočni zapisi se lahko prosto dekodirajo, kodirati zvočni zapis brez licence pa je prepovedano. Na koncu ni preostala druga možnost, kot da se v aplikaciji uporabi eden izmed kodirnih algoritmov iz ogrodja iPhone. Posnetek se po prenosu na strežnik spremeni v ustrezen končni format, preden se zapiše v podatkovno bazo. Posneti zvočni zapisi se do trenutka sinhronizacije s strežnikom hranijo na iPhone napravi in se naknadno pobrišejo ob uspešni sinhronizaciji.

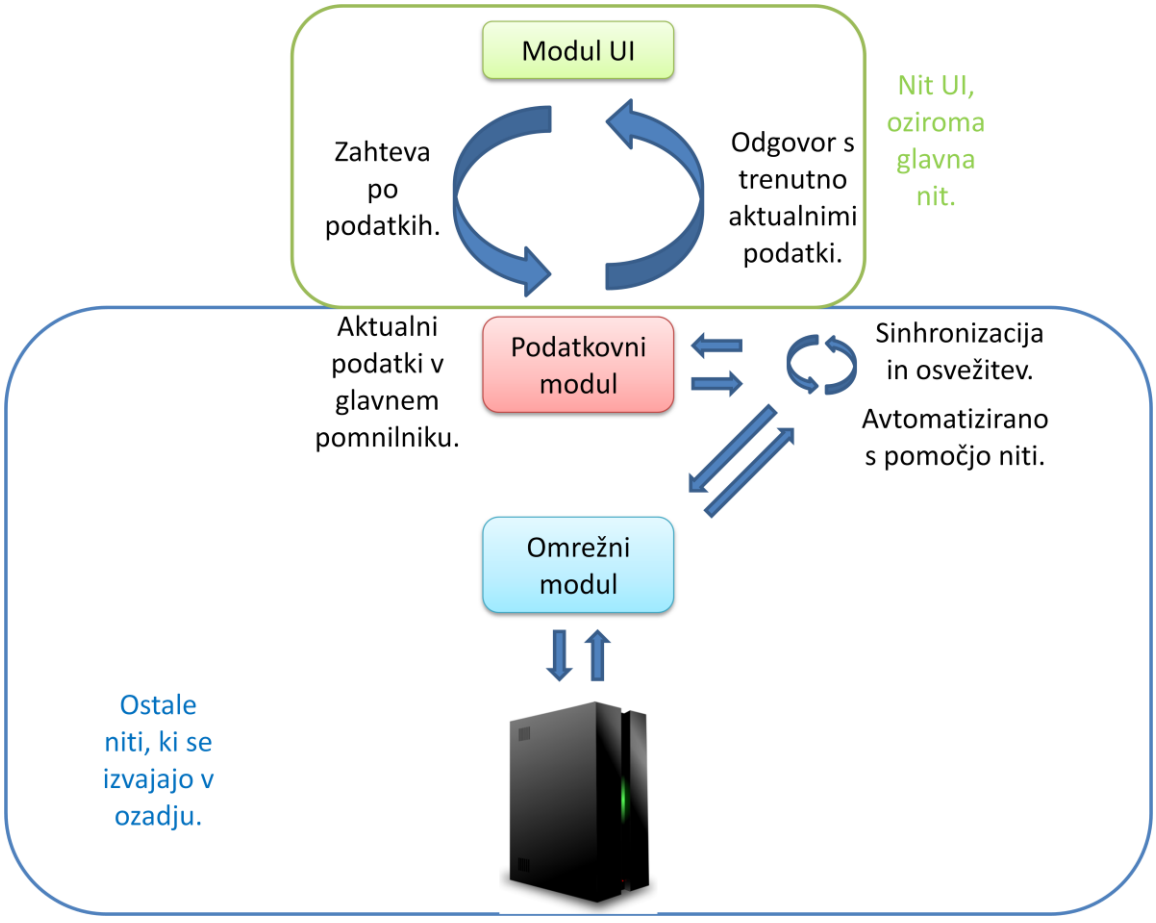
Med potekom snemanja se zvočni zapis shranjuje direktno v pomnilnik naprave tipa FLASH. Takšen postopek zagotovi manjšo verjetnost, da se zapolni ves prosti pomnilnik, kot če bi to shranjevali v glavni pomnilnik. Po končanem snemanju se ponovno prebere zvočni zapis in se uporabniku ponudi možnost, da posnetek predvaja. V primeru, da se uporabnik ne strinja s posnetkom, se avtomatsko izbriše iz naprave.

4.1.3. Podatkovni modul

Ker na UI rabimo prikazati določene podatke, jih moramo tudi od nekje pridobiti. Temu služi podatkovni modul. V tem primeru gre za objekt, ki ima življenjsko dobo enako dolgo kot glavna nit aplikacije. Glavna nit ta objekt ustvari in tudi uniči ob koncu delovanja. Prav tako je v aplikaciji ta objekt ustvarjen le enkrat in se po potrebi prenaša po aplikaciji le s kazalcem na ta objekt.

4.1.3.1. Povezava UI ter omrežnega modula

Podatkovni modul je glavni povezovalni člen UI in omrežnega modula. Kot primer vzamemo uporabnika, ki zahteva prikaz nekih podatkov. Modul UI to sporoči podatkovnemu modulu. Če podatkovni modul nima podatkov, jih gre iskat na strežnik preko omrežnega modula.



Slika 9. Funkcija podatkovnega modula.

4.1.3.2. Zagotavljanje pripravljenosti podatkov na prikaz

Zaradi konstantne potrebe po različnih podatkih je potrebno imeti podatke vedno na voljo in pripravljene na uporabo. S tem zagotovimo, da je uporabnik zadovoljen z odzivnostjo in rezultati aplikacije. S pomočjo glavnega pomnilnika naprave iPhone, ki ga je na voljo 256 MB (Avgust 2010, iPhone 3GS), lahko zgornje zahteve enostavno dosežemo. Vsi podatki, ki jih omrežni servis prejme, se po procesiranju v obliki entitet zadržujejo v glavnem pomnilniku. Kaj so entitete, bo opisano v nadaljevanju. Ker je oblika podatkov v večini primerov niz besed, nam primanjkovanje pomnilnika na tej napravi naj ne bi delalo preglavic. Seveda pa ne smemo pozabiti na pravilno sproščanje objektov, ki jih ne potrebujemo več. Po prenehanju delovanja aplikacije se ves zaseden pomnilnik kaskadno sprosti. Za to skrbijo metode v objektih, ki jih mora razvijalec sam definirati. Te metode sproščajo vse ostale objekte, ki jih je razvijalec dodatno ustvaril.

Ker med samim delovanjem aplikacije večkrat pride do sprožitve sinhronizacije s strežnikom, se morajo podatki v ozadju obdelati in nato nadomestiti aktualne podatke. Stare podatke, ki jih ima uporabnik trenutno na voljo, je potrebno nato nadomestiti z novimi, ne da bi uporabnik to opazil. Podatkovni modul v celoti deluje s pomočjo niti. Zaradi tega včasih tudi pride do situacije, ko se glavna nit za hip ustavi, v tem primeru se uporabniku prikaže sporočilo, kaj se dogaja v ozadju.

Ob zagonu aplikacije se na začetku preveri, če so se v nastavitvah spremenili podatki za prijavo uporabnika. V tem primeru se pobriše vsa zgodovina prejšnjega uporabnika. Če podatki za prijavo ostanejo nespremenjeni, se le-ti obdelajo in začasno shranijo v glavni pomnilnik v obliki entitet. Na tem mestu se naloži glavni meni in uporabnik lahko začne uporabljati aplikacijo. Medtem se v ozadju z drugo nitjo požene proces sinhronizacije s strežnikom. Sinhronizacija zavzema vse operacije, ki so potrebne, da se podatki pridobijo s strani strežnika in se pripravijo na uporabo. To vključuje tudi test povezljivosti na strežnik. V primeru, da v začetni fazi sinhronizacije prijava spodleti, aplikacija počaka trideset sekund in nato ponovno poizkusi, enako velja za poskus povezljivosti. V primeru, da strežnik pošlje napako o napačni prijavi, pa se to prikaže uporabniku v obliki opozorila.

4.1.3.3. Lokalna shramba podatkov za delo brez omrežne povezave

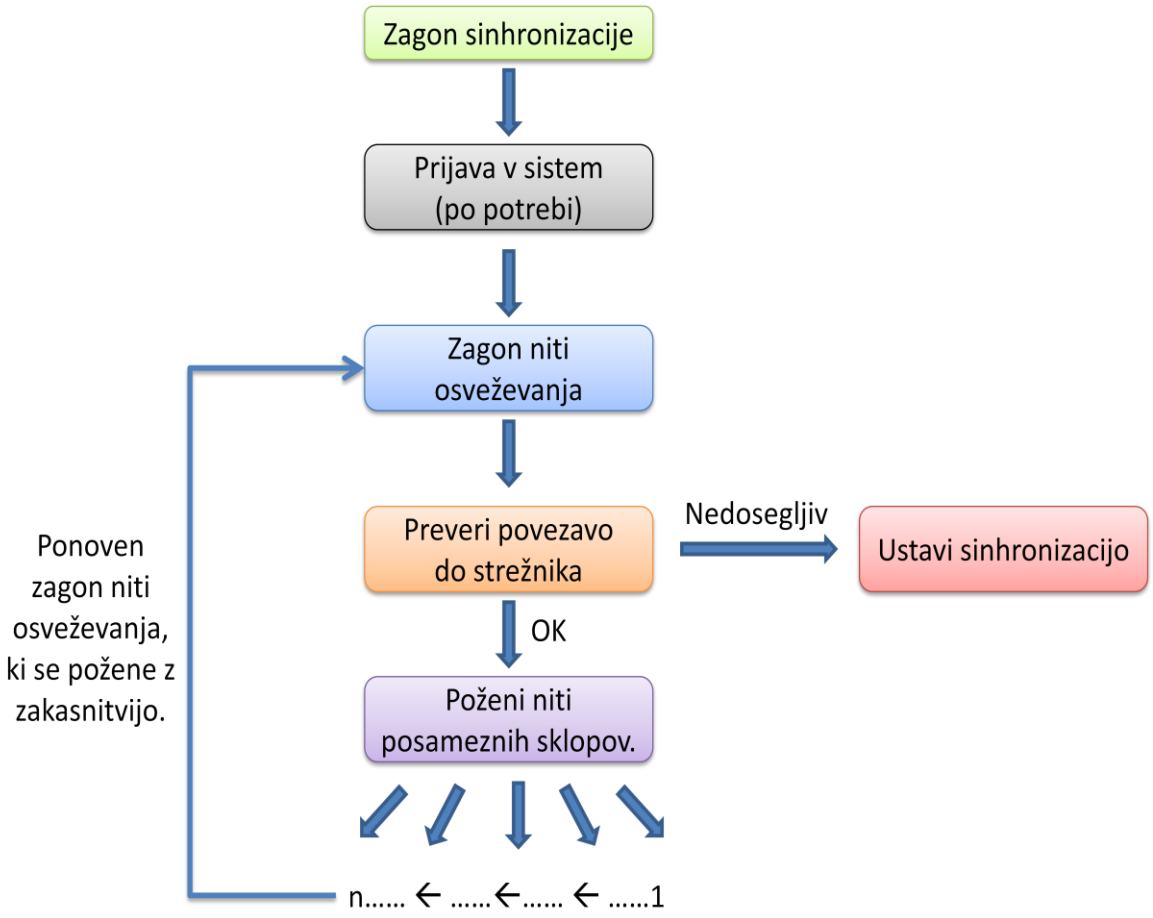
Ker gre v tem primeru za mobilno poslovno aplikacijo, je potrebno zagotoviti tudi delovanje brez omrežne povezave, bodisi WLAN bodisi podatkoven prenos podatkov od ponudnika mobilnih storitev. Zato se zadnji aktualni podatki sproti shranjujejo na napravo v obliki XML, kot jo prejme aplikacija od strežnika. Takšen pristop razvijalcu tudi olajša delo, kajti tako za podatke s strežnika kot za podatke z naprave rabi samo eno metodo, da se podatki obdelajo. Lokalno se prav tako shranjujejo tudi zvočni zapisi, dokler le-ti niso preneseni na strežnik, nato se ob

uspešnem prenosu samodejno izbrišejo. Kljub zelo majhni verjetnosti, da se v napravi pojavijo pokvarjeni oziroma napačni podatki, je na voljo tudi zmožnost čiščenja aplikacije. Drugače rečeno – obstaja funkcija, ki povrne aplikacijo v prvotno stanje in očisti celoten prostor, v katerem aplikacija zadržuje podatke.

4.1.3.4. Sinhronizacija iPhone in strežnikom MATIVY MG

Sinhronizacija s strežnikom poteka v treh fazah. Prva faza preveri stanje seje, druga faza požene klice proti strežniku in zahteva podatke, zadnja pa požene novo nit z zakasnitvijo, ki si jo uporabnik nastavi poljubno v nastavitvah. Ta nova nit je nov zahtevek za sinhronizacijo. V primeru, da v prvi fazi ali med drugo fazo seja spodleti, se celoten postopek prekine in se začne vse od začetka, skupaj s prijavo uporabnika v sistem in pridobivanjem šifre seje. V drugi fazi se prav tako za vsak klic na strežnik ustvari nova nit. To omogoča večji izkoristek tako omrežja kot tudi same naprave, pri delovanju pa ne ovira uporabnika pri upravljanju z aplikacijo. Vsaka taka nit je tudi dodatno zaposlena s spreminjanjem podatkov iz oblike XML v obliko entitet. Po končanem obdelovanju podatkov se zamenja kazalec na podatke, ki jih uporabnik uporablja, stari podatki pa se sprostijo. Ker je zamenjava kazalcev hitra operacija, uporabnik ne zazna spremembe podatkov.

Zaradi uporabe več niti za upravljanje s podatki je potrebno zagotoviti tudi varnost podatkov, da jih ne moreta hkrati uporabljati dve niti. V tem primeru se uporablja način zaklepanja podatkov s ključavnicami. V glavnem je to zastavica, ki pove, ali so podatki v uporabi ali ne. Vsaka metoda, ki upravlja ali bere podatke, ima v sebi implementacijo pregleda zasedenosti ključavnic ter možnost zaklepanja podatkov. V primeru, da so podatki zaklenjeni, mora nit počakati določen interval in poskusiti ponovno.

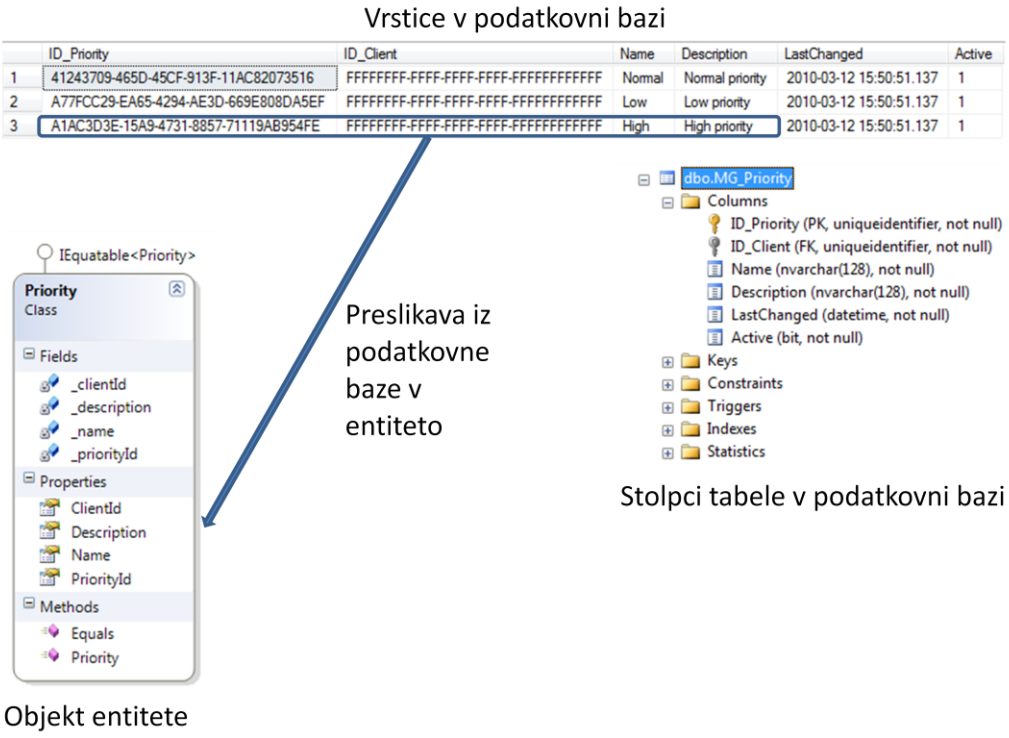


Slika 10. Primer poteka sinhronizacije.

4.1.4. Modul entitet

4.1.4.1. Kaj so entitete?

Entitete si lahko predstavljamo kot objekte, ki vsebujejo lastnosti za vsak stolpec tabele iz podatkovne baze. En objekt predstavlja eno vrstico v bazi. Celo tabelo v tem primeru predstavlja polje teh objektov v obliki entitet.



Slika 11. Primerjava entitete in vrstice v podatkovni bazi.

4.1.4.2. Kaj omogočajo?

Entiteta kot objekt vsebuje določene metode za manipuliranje s podatki. V primeru te aplikacije ima vsaka entiteta definirane statične metode, ki znajo iz niza besed oziroma drevesne strukture XML sestaviti nov objekt enakega tipa. Definirane so tudi metode, ki znajo iz lastnosti objekta sestaviti zapis oziroma drevo XML. Dodatno so implementirane tudi metode za primerjavo ter kopiranje entitet.

4.1.4.3. Zakaj jih rabimo?

Glavni namen entitet v tem primeru je poenostavitev ter povečanje hitrosti operacij, ki jih mora aparat izvršiti. Da aparatu ni potrebno vedno znova in znova prebirati zapisa formata XML in iskati v njem podatkov, se raje uporabijo entitete. Več časa procesor zapravi pri prebiranju in iskanju nekega podatka v zapisu XML, kot pa da se sprehodi po pomnilniku preko nekaj kazalcev.

4.1.5. Modul orodij

V modulu orodij se nahajajo vsi objekti, ki se uporabljajo po celotni aplikaciji. Njihov namen je v večini primerov manipuliranje s podatki. Sem spadajo na primer orodje za spreminjanje niza besed v datumsko obliko in obratno, orodje za prevajanje aplikacije ter objekt, ki vsebuje globalne vrednosti. Glavni dve orodji sta objekta, ki izvajata operacije pretvorbe zapisa XML. Prvi skrbi za grajenje objektov iz zapisa XML, drugi pa obratno. V drugem primeru objekt skrbi predvsem za zapis podatkov na napravo, kajti zapis XML znajo zgraditi entitete same.

4.1.5.1. Orodje za prevode znotraj aplikacije

Kot velika večina današnjih orodij na poslovnem področju tudi ta aplikacija podpira večjezičnost. Obstaja veliko načinov prevajanja aplikacije.

V tej aplikaciji se nahaja datoteka, ki vsebuje v formatu zapisa XML vse potrebne prevode, ki se rabijo na UI. Da pa lahko to datoteko uspešno preslikamo na UI, potrebujemo nek pomožni objekt, ki za to skrbi.

4.1.5.1.1. Kako prevaja?

Ta objekt, ki se uporablja za prevajanje, vsebuje slovar besed kot lastnost ter nekaj metod, potrebnih za prevajanje. Tako kot podatkovni modul se tudi ta objekt ustvari samo enkrat in se preko kazalca prenaša po objektih UI. Ob konstrukciji objekta se prebere datoteka XML in se ustvari slovar prevodov. Med gradnjo slovarja se že sproti filtrirajo prevodi in se uporabijo samo prevodi za jezik, ki je izbran v nastavitvah. S tem postopkom tudi nekaj privarčujemo na pomnilniku. Ta pristop omogoča hitro menjavo ter popravljanje ali dodajanje prevodov v drugih jezikih. Enostavno projektu zamenjamo datoteko XML ter v nastavitve dodamo dodaten jezik, kot dodatno možnost pri izboru.

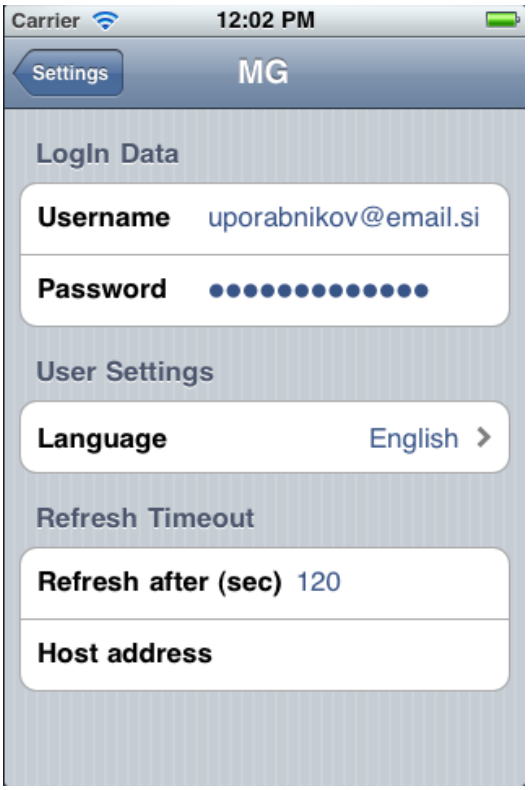
V primeru, da je potrebno nekaj prevesti, enostavno pokličemo metodo tega objekta, ki služi za prevajanje. Metodi kot argumentu pa podamo ključ prevoda, da ga zna objekt poiskati v slovarju prevodov, in še osnovni prevod v angleščini. Slednji argument služi v primeru, če slovar ne vsebuje prevoda. V tem primeru objekt vrne drugi argument kot prevod.

4.1.5.2. Globalne nastavitve projekta

Globalne nastavitve projekta v tem primeru ne predstavljajo nastavitve projekta v opcijskem meniju razvojnega okolja, ampak nastavitve aplikacije, ki so v pomoč razvoju aplikacije. Da ni potrebno na N mestih znotraj programske kode iskati in popravljati enako nastavitev, je tukaj objekt z globalnimi nastavitvami, ki vsebuje metode, ki vračajo vse potrebne podatke. S tem pristopom lahko po celotni aplikaciji izvedemo spremembe le s popravkom na enem mestu. Ta pristop tudi poenostavi programsko kodo ter pripomore k preglednosti.

4.2. Nastavitve aplikacije

Pri razvoju aplikacije, namenjene iPhone napravi, imamo dve možnosti pri izbiri lokacije nastavitev. Prva možnost je, da se preprosto v aplikacijo dogradi meni oziroma zaslon z nastavitvami. Drugo možnost pa ponuja naprava oziroma operacijski sistem iOS sam. S pomočjo razvojnega okolja lahko, na zelo enostaven način, skupaj z aplikacijo dodamo na napravo tudi nastavitveni meni za aplikacijo znotraj nastavitvenega menija naprave. Slabost prvega načina je čas trajanja implementacije. Veliko odvečnega časa porabimo za implementacijo celotne podobe nastavitev ter umeščanje novega menija v aplikacijo. Po drugi strani pa je dobra stran tega pristopa prosta pot glede uporabe raznih gradnikov ter dizajna uporabniškega vmesnika. Pri drugem načinu implementacije nastavitev pa se razmere obrnejo. Veliko lažje je ustvariti meni, smo pa omejeni glede dizajna ter gradnikov. V primeru prototipa rabimo preproste nastavitve, kot so uporabniško ime ter geslo, po potrebi tudi naslov strežnika, kjer se nahajajo servisi, in še nastavitev razmika med sinhronizacijami s strežnikom. Zaradi teh preprostih zahtev je razvoj nastavitev prototipa usmeril drugi način implementacije.



Slika 12. Nastavitve aplikacije.

4.3. Težave pri razvoju aplikacije

Pri večini primerov razvoja aplikacij razvijalec prej ali slej pride do težav. Nekatere težave se lahko odpravijo brez večjih muk, spet druge lahko razvijalcu delajo velike preglavice. Med razvojem te aplikacije se je prav tako porajalo nekaj težav oziroma vprašanj. Upravljanje s pomnilnikom je bila tukaj največja težava. Naslednjo težavo je povzročilo dejstvo, da mora biti aplikacija sposobna teči brez omrežne povezave, kar pa je tudi vplivalo na hitrost in odzivnost aplikacije. Za namene testiranja pa je bilo za konec potrebno napravo še kupiti, kar pa je prej kot težave pri razvoju aplikacije, povzročilo finančne težave. Kljub temu da ponuja razvojno okolje simulator naprave, ki je kar tesen približek prave naprave, to še ne pomeni, da se obnaša enako kot naprava sama. To se je v končni fazi poznalo najbolj na področju upravljanja s pomnilnikom. Do tistega trenutka, ko je bila aplikacija prvič testirana na napravi, se ni toliko gledalo na upravljanje s pomnilnikom. Po nekajkratnih sesutjih aplikacije pa je to kaj hitro postalo glavni hrošč v aplikaciji ter prioriteta pri razvoju.

4.3.1. Upravljanje s pomnilnikom

iOS operacijski sistem, ki teče na napravi iPhone, ne podpira tehnologije »Garbage Collection«, zato je za upravljanje s pomnilnikom izključno zadolžen razvijalec sam. Pri tehnologiji »Garbage Collection« ogrodje samo skrbi za čiščenje objektov, ki se ne rabijo več in s tem sprošča pomnilnik. Procedura se požene v določenih intervalih ali pa ko začne napravi primanjkovati pomnilnika. Tukaj pa temu ni tako. Vsak objekt, ki ga razvijalec ustvari, mora za sabo tudi sprostiti. Objekti imajo v sebi vgrajen števec, ki pove, kolikokrat se ta objekt uporablja. V primeru, da po končani obdelavi objekta ne rabimo več in je ta števec večji od nič, pride do tako imenovanega puščanja pomnilnika. V tem primeru ostane pomnilnik rezerviran, čeprav se ne rabi več. V nasprotnem primeru, če je števec manjši od nič, pa pride do napake, ker ne moremo objekta večkrat sprostiti, kot smo ga uporabili. V tem primeru se aplikacija preprosto sesuje.

4.3.2. Hitrost aplikacije

Vsak uporabnik pričakuje od katerekoli aplikacije, da se le-ta hitro odziva na njegove poteze. To razvijalcu predstavlja največje breme pri razvoju, ker poleg razvoja UI, ta del prav tako zahteva veliko časa, da se izpopolni. Za poslovno aplikacijo je to še posebej pomembno. Poslovne aplikacije v večini primerov obdelajo v svojem življenjskem obdobju veliko podatkov. Veliko pridobitev v poslovnem svetu lahko doseže aplikacija, ki pospeši sam poslovni proces. Razvijalčeva naloga tukaj je razviti algoritme, ki v najkrajšem času izvedejo in predstavijo nek

rezultat. Drugi problem nastane, ko želimo te obdelane podatke predstaviti uporabniku. V našem primeru se je ta problem pojavil pri prikazu oziroma osveževanju tabele nalog. Vsakokrat, ko je uporabnik preklopil na kakšen drug meni znotraj aplikacije, je bilo potrebno ob naslednjem prehodu na prvi meni osvežiti podatke v tabeli. Ob vsaki menjavi menijev se je lahko v ozadju osvežil seznam nalog, zato pa je tukaj potrebna osvežitev. Uporabniku je potrebno vedno kazati najbolj aktualne podatke. Če bi kazali stare podatke, lahko uporabnika na neki točki zmedemo ali pa povzročimo nepotrebne napake pri sinhronizaciji, v obeh primerih dosežemo nezadovoljstvo s strani uporabnika. Velik delež povečanja hitrosti naše aplikacije je prispevala uvedba entitet namesto zapisa XML. Drugo povečanje hitrosti znotraj aplikacije je bila uporaba več hkratnih niti za obdelavo podatkov. Določen del povečanja hitrosti komunikacije je prispeval tudi postopek omrežne izmenjave podatkov. Aplikacija podatke prenesene s strežnika takoj shrani brez pretvorb na pomnilnik naprave. Šele potem, ko ima naprava čas za obdelavo podatkov, se sproži proces, ki pretvarja podatke. Ta pristop reši še naslednji problem. Recimo, da med sinhronizacijo uporabnik zapusti oziroma ugasne aplikacijo. V primeru, da bi v tem trenutku potekala pretvorba, še preden bi se podatki shranili na napravo, bi uporabnik v tem trenutku izgubil aktualne podatke in bi ob naslednjem zagonu moral počakati, da bi se podatki znova prenesli s strežnika. Veliko takšnih problemov je mogoče rešiti že pred začetkom razvoja z ustrezno postavljenim načrtom ter analizo uporabe aplikacije.

4.3.3. Postopek shranjevanja podatkov brez omrežne povezave

Shranjevanja podatkov se lahko lotimo na več načinov. Na začetku projekta je bilo določeno, da se bodo podatki shranjevali v binarni obliki. Tukaj bi enostavno shranili objekte v datoteko. Kasneje pa je bil zaradi konsistentnosti z ostalimi aplikacijami, ki so se razvijale v enakem obdobju, način shranjevanja spremenjen in so se podatki shranjevali v obliki XML. Kot primer vzemimo tudi BlackBerry aplikacijo, razvito v okolju Java. Aplikacija sproti shranjuje vse prejete podatke s strežnika in tudi spremembe nad podatki, ki jih naredi uporabnik. To omogoči aplikaciji, da ima vedno shranjene najbolj aktualne podatke, prav tako pa skrbi tudi za to, da aplikacija deluje tudi brez omrežne povezave.

Vsaka aplikacija znotraj operacijskega sistema iOS ima svojo mapo, kamor lahko razvijalec shranjuje podatke med delovanjem aplikacije. V to mapo lahko shranjujemo poljubno vsebino.

4.3.4. Omejenost kodiranja zvočnega zapisa

Kot je bilo že omenjeno, ponuja ogrodje iOS kar nekaj standardnih zvočnih zapisov. Ker pa so v ozadju teh zvočnih zapisov zakoreninjeni tudi patenti, je v večini primerov razvijalcu, prej kot

karkoli drugega, zagrenjen razvoj z uporabo zvočnih zapisov. Predvsem je tukaj mišljeno snemanje zvočnega zapisa. Drugi problem predstavlja tudi izbor pravilnega zvočnega zapisa. Med razvojem je bil vsak zvočni zapis preizkušen nad istem zvočnem posnetku. Na koncu je bil zvočni zapis izbran glede na kriterija kompresije ter kvalitete zapisa. Tukaj je velikost končnega zapisa še posebej pomembna, kajti moramo se zavedati, da se podatki prenašajo preko mobilnega interneta, ki je počasen ter trenutno še precenjen.

V končni fazi se na strani strežnika podatki dokončno pretvorijo v bolj splošen zvočni zapis MP3 in se le-ta shrani v podatkovno bazo. Ob naslednjim zahtevku, ki ga uporabnik sproži, da bi predvajal posnetek, se prenese zvočni posnetek v obliki zvočnega zapisa MP3 s strežnika na napravo iPhone.

4.3.5. Simulator iPhone proti napravi iPhone

Med razvojem aplikacije je potrebno velikokrat napredek tudi testirati. V primeru, ko razvijalec nima na voljo fizične naprave, se lahko zanese na simulator naprave, ki je vključen v Xcode razvojno okolje. Simulator vsebuje veliko večino funkcij fizične naprave, nekatere funkcije se seveda morajo vključiti ročno. Primer funkcije je obrat naprave za devetdeset stopinj. Ta funkcija služi zasuku uporabniškega vmesnika za devetdeset stopinj v smeri obrata naprave. Po končanem razvoju oziroma v obdobju testiranja je aplikacijo potrebno testirati na fizični napravi. Aplikacija se vede nekoliko drugače, če jo poganjamo na simulatorju, kot pa če jo poganjamo na fizični napravi. Glavna razlika med testiranjem te aplikacije je bila pri upravljanju s pomnilnikom. S pomočjo orodij za profiliranje aplikacij ter s fizično napravo se je v tem obdobju dokončno popravila programska koda na vseh mestih, kjer je aplikacija še puščala objekte v pomnilniku. Naslednja razlika med simulatorjem in fizično napravo je sama hitrost procesiranja. Kot že vemo, naprava iPhone vsebuje procesor z šeststo MHz urinega takta. Simulator na drugi strani pa vzame procesno moč od računalnika, ki ga poganja. Današnji računalniki imajo v večini primerov procesorje s taktom nad dva GHz. S pomočjo predpostavk lahko sklepamo, da bo aplikacija na računalniku tekla hitreje, in tudi praksa je to dokazala. Na to temo so bile naknadno izboljšave narejene na področju upravljanja s podatki.

5. Strežnik MG

5.1. Strežnik

Glavni strežnik MG poganja aplikacijski servis. Ta servis je aplikacija, ki poganja vso poslovno logiko aplikacije MG, ter skrbi za komunikacijo s podatkovno bazo MG. Podatkovna baza MG je zaradi boljše odzivnosti postavljena na ločenem strežniku. Drugi razlog za ločevanje aplikacijskega ter podatkovnega strežnika je sama požrešnost pomnilnika SQL Server servisa. Servis SQL med delovanjem zasede po potrebi ves razpoložljiv pomnilnik.

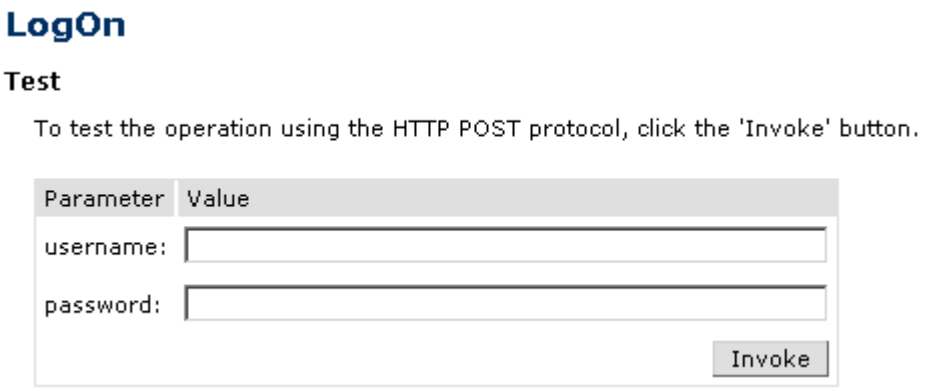
5.2. Aplikacijski servis

Aplikacijski servis je glavni posrednik med podatkovno bazo ter klientom. Poslovna logika implementirana v tem servisu skrbi za pravilno obdelavo podatkovne baze. Prav tako implementira vsa potrebna pravila in postopke za upravljanje z nalogami znotraj sistema MG. Metode, definirane v poslovni logiki, uporabljajo vsi ostali servisi, ki jih uporabljajo klienti. Med ostale servise štejemo Web servise in posrednik HTTP, namenjen Outlook vtičniku ter administraciji.

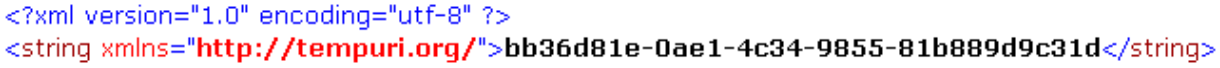
5.3. Web servisi

Web servisi so namenjeni predvsem komuniciranju sistema MG s produkti tretjih oseb. Sem spadata tudi iPhone in BlackBerry aplikaciji. Vse metode znotraj servisa so univerzalne vsem končnim klientom. Glavna razlika med Web servisi ter posrednikom HTTP so implementacije nekaterih metod, ki so do neke mere poenostavljene proti tistim, ki se kličejo preko posrednika HTTP. Servisi komunicirajo preko aplikacijskega servisa in poslovne logike do podatkovne baze MG. Klienti komunicirajo s servisi preko protokola SOAP.

Spodaj je prikazan primer klica LogOn metode Web servisa na strežniku MG. Primeri so se izvajali v brskalniku na strani strežnika. V polja je potrebno vnesti uporabniško ime ter geslo za uporabniški račun. Kot rezultat metoda vrne šifro seje, ki predstavlja uporabnika na strani strežnika. Vse ostale zahteve oziroma klici na strežnik zahtevajo šifro seje kot parameter pri klicu.



Slika 13. Primer uporabe Web servisa, metoda LogOn.



Slika 14. Primer uporabe Web servisa, rezultat metode LogOn .

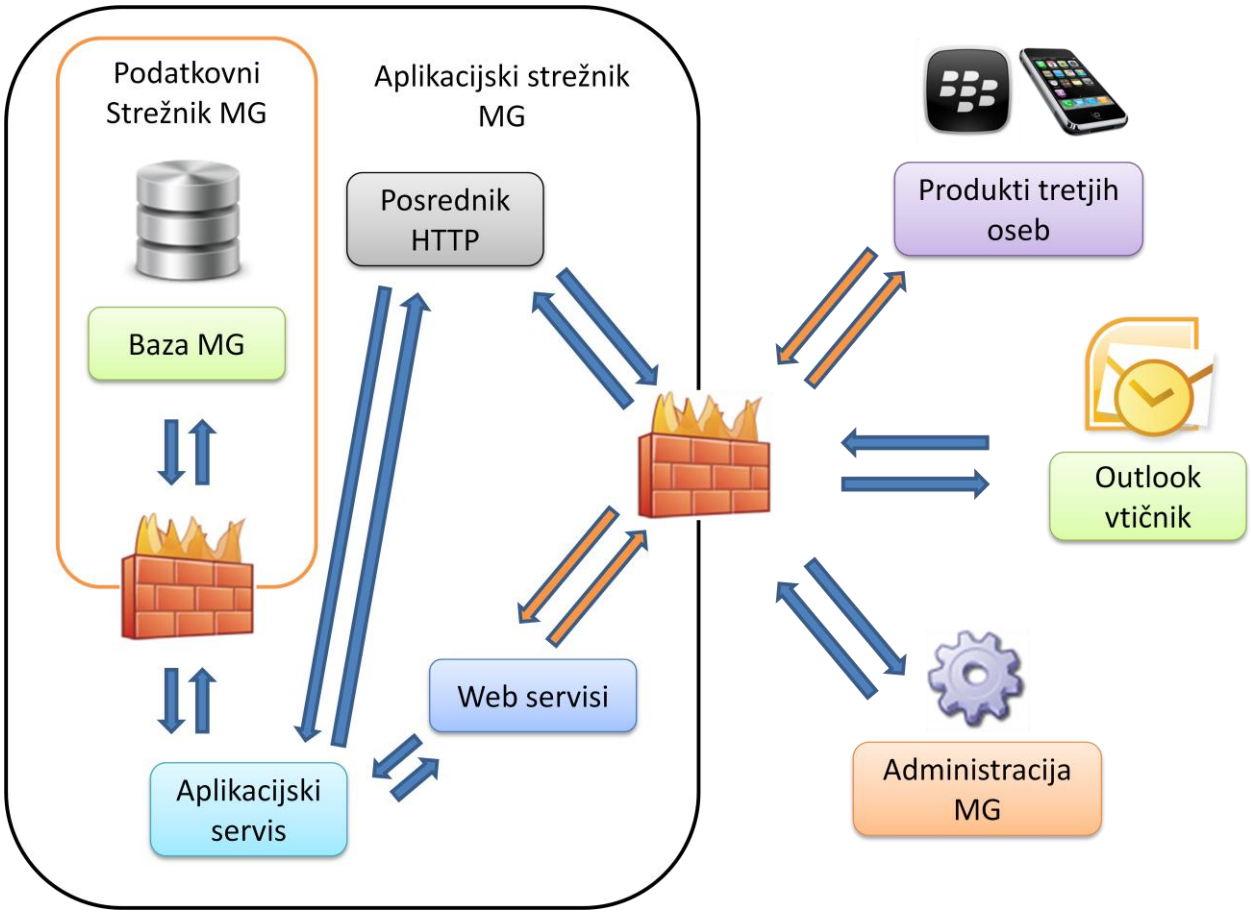
5.4. Posrednik HTTP

HTTP posrednik potrebujeta Outlook vtičnik ter administracija sistema MG. Prototip aplikacije MG je bil v prvotni različici implementiran v okno internetnega brskalnika kot internetna aplikacija. Implementacija je temeljila izključno na Microsoftovi tehnologiji WPF. Ena izmed zahtev delovanja te aplikacije je bila zmožnost dostopa do sistema MG tudi znotraj javnega omrežja. Problem, ki se pojavi v javnih omrežjih, je požarni zid. Javna omrežja imajo zaradi varnostnih zahtev v večini primerov odprta samo omrežna vrata s številko 80. Zaradi tega se je pri sistemu MG poleg aplikacijskega servisa uporabil še posrednik HTTP, ki preusmeri promet na strani strežnika iz vrat 80 na vrata, kjer posluša promet aplikacijski servis. Obratno velja v drugo smer, proti klientu.

5.5. Podatkovna baza MG

Vsi podatki, ki jih sistem MG rabi za delovanje, se nahajajo v podatkovni bazi SQL. Programski jezik SQL je namenjen upravljanju s podatki znotraj sistema za upravljanje relacijskih podatkovnih baz. S pomočjo ukazov SQL lahko dodajamo (INSERT) podatke v podatkovno bazo, jih posodabljamo (UPDATE) ali pa brišemo (DELETE). Najbolj pogoste operacije, ki jih SQL omogoča, pa so poizvedbe (SELECT).

5.6. Povezava posameznih razdelkov med seboj



Slika 15. Povezava posameznih razdelkov med seboj.

6. Sklepne ugotovitve

Končni rezultat diplomskega dela je funkcionalen prototip iPhone aplikacije za upravljanje z nalogami v sistemu MG. Aplikacija vsebuje vse funkcionalnosti, ki so bile na začetku definirane. Uporabniški vmesnik je prijazen do uporabnika ter odziven. S pomočjo zaslonov, ki vsebujejo tabele, uporabniku na enostaven način prikazuje kopico podatkov. Aplikacija se je preko omrežja sama sposobna sinhronizirati z strežnikom MG. Sinhronizacija le v redkih primerih zmoti uporabnika, aktualni podatki pa se shranjujejo lokalno v pomnilniku naprave iPhone. Za konec pa aplikacija prav tako vsebuje diktafon, ki je v obliki naloge sposoben shraniti zvočni posnetek na strežnik MG.

Med razvojem prototipa se je pojavila marsikatera težava, ki pa je bila z nekaj truda na koncu tudi odpravljena. Prehod iz programskega jezika C# na programski jezik »Objektivni-C« tudi ni potekal brez težav. Kot prvotno .NET razvijalec sem se moral odvaditi veliko starih navad ter na novo spoznati določene dele novega programskega jezika, ki jih prejšnji ni vseboval. Po mojem mnenju ter izkušnjah je prvi veliko bolj enostaven ter pregleden od slednjega. Veliko sem se naučil predvsem glede porabe pomnilnika pri razvoju aplikacij. V tem primeru je to še toliko bolj pomembno, ker naprava iPhone ne vsebuje toliko glavnega pomnilnika kot nek osebni računalnik.

Kazalo slik

Viri

[1] (2010) MATIVY Management GmbH uradna stran. Dostopno na: http://www.mativy.com.

[2] (2010) iPhone Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/IPhone.

[3] (2010) Apple inc. Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Apple_Inc..

[4] (2010) Microsoft SQL Server 2008 Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Microsoft_SQL_Server#SQL_Server_2008.

[5] (2010) Xcode uradna stran. Dostopno na: http://developer.apple.com/technologies/tools/xcode.html.

[6] (2010) SOAP Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/SOAP.

[7] (2010) iOS uradna stran. Dostopno na: http://www.apple.com/iphone/ios4/.

[8] (2010) MVC Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller.