Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jani Štricelj

Mobilna aplikacija za udeležence poletne šole

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, september 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00222/2012
Datum: 03.04.2012

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

Kandidat: **JANI ŠTRICELJ**

Naslov: **MOBILNA APLIKACIJA ZA UDELEŽENCE POLETNE ŠOLE**
MOBILE APPLICATION FOR PARTICIPANTS OF SUMMER SCHOOL

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Organizacija poletne šole zahteva komunikacijo z udeleženci. Za obveščanje in dostop do programa poletne šole bi bila dobrodošla mobilna aplikacija, ki bi udeležencem omogočala dostop do obvestil, pregled programa poletne šole in pregled podatkov o poletni šoli. Zasnujte funkcionalnosti za mobilno aplikacijo za organizacijo poletne šole. Mobilno aplikacijo nato razvijte na PhoneGap platformi. Razvoj na PhoneGap platformi bo omogočil uporabo mobilne aplikacije na različnih operacijskih sistemih mobilnih naprav.

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 Jani Štricelj, z vpisno številko **63080346**, sem avtor diplomskega dela z naslovom:

Mobilna aplikacija za udeležence poletne šole

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika,
- 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 19. septembra 2012 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju doc. dr. Roku Rupniku za pomoč, usmerjanje in nasvete pri izdelavi diplomskega dela. Prav tako pa bi se zahvalil tudi staršem, prijateljem in sošolcem za vso pomoč, spodbujanje in podporo, tako med študijem, kot tudi med pisanjem diplomskega dela.

Kazalo

Seznam uporabljenih kratic in simbolov

AJAX - *(angl.) Asynchronous JavaScript and XML*; asinhroni JavaScript in XML.
API - *(angl.) Application Programming Interface*; programski vmesnik.
ASCII - *(angl.) American Standard Code for Information Interchange*; ameriški standardni nabor za izmenjavo informacij.
CSS - *(angl.) Cascading Style Sheets*; kaskadne stilske podloge.
DOM - *(angl.) Document Object Model*; standard, ki opisuje objektni model za predstavitev dokumentov.
GPS - *(angl.) Global Positioning System*; sistem globalnega določanja lege.
HTML - *(angl.) Hyper Text Markup Language*; označevalni jezik za oblikovanje večpredstavnostnih dokumentov.
HTTP - *(angl.) HyperText Transfer Protocol*; protokol za prenos informacij na spletu.
HTTPS - *(angl.) Hypertext Transfer Protocol Secure*; protokol za varen prenos informacij na spletu.
IDE - *(angl.) Integrated Development Environment*; integrirano razvojno okolje.
JSON - *(angl.) JavaScript Object Notation*; zapis JavaScript objekta.
MVC - *(angl.) Model–View–Controller*; model-pogled-nadzornik.
SQL - *(angl.) Structured Query Language*; strukturirani povpraševalni jezik.

URI - *(angl.) Uniform Resource Identifier*; protokol za varen prenos informacij na spletu.

XML - *(angl.) Extensible Markup Language*; razširljiv označevalni jezik.

Povzetek

Število pametnih mobilnih naprav je v porastu, zato je razvoj mobilnih aplikacij zelo razširjen, kar pa je s seboj prineslo tudi širok nabor mobilnih platform na katerih so mobilne aplikacije zasnovane. Razvoj in podpora mobilnih aplikacij, ki delujejo na več mobilnih platformah, sta časovno in finančno velik zalogaj. V diplomskem delu je predstavljen razvoj hibridne mobilne aplikacije, ki temelji na uporabi spletnih tehnologij a pri tem ohranja dostop do raznih funkcionalnosti mobilnega operacijskega sistema. Tako lahko z enotno programsko kodo in uporabo razvojnih ogrodij dosežemo izvajanje na več mobilnih platformah, kar občutno skrajša razvojni čas. Opisane so vse uporabljene tehnologije, razvojna ogrodja in knjižnice. Aplikacija podpira pregled ključnih informacij in obvestil Poletne šole FRI in je namenjena njenim udeležencem. Razvoj je potekal skozi razvojni cikel, ki vključuje analizo, načrtovanje, kodiranje, testiranje in prevajanje.

Ključne besede:

PhoneGap, poletna šola, hibridne mobilne aplikacije, medplatformski mobilni razvoj

Abstract

The amount of smart mobile devices is increasing and is the reason for widely spread development of mobile applications. That has brought to a wide range of mobile platforms on which mobile applications are based. Development and support of mobile applications that run on multiple mobile platforms require a lot of time and financial resources. This thesis presents the development of hybrid mobile application, which is based on the use of web technologies, but at the same time maintains access to different functionalities of mobile operating system. With a single codebase and with use of the development frameworks, it is possible to achieve implementation on multiple mobile platforms, which significantly reduces time of development. In dissertation are described all used technologies, frameworks and libraries. Application supports overview of the key informations and announcements of the summer school Poletna šola FRI and it is intended for its participants. Development went through different stages of development cycle including analysis, planning, coding, testing and translating.

Keywords:

PhoneGap, summer school, hybrid mobile applications, cross platform mobile development

Poglavje 1

Uvod

Dobra informacija je tista, ki poda uporabne podatke v pravem trenutku in na pravem mestu. Prednost mobilnih tehnologij je v tem, da je mesto, kjer informacijo potrebujemo, skoraj neomejeno. Zelo pomembna dejavnika pri tem sta pravi trenutek ter kvaliteta podatkov. Vse to je moč doseči s kvalitetno mobilno aplikacijo, ki služi določenemu opravilu. Priljubljenost mobilnih aplikacij se je v zadnjih letih zelo povečala. Razloge za ta trend lahko najdemo v širši ponudbi mobilnih naprav, boljših mobilnih omrežjih ter lažji dosegljivosti mobilnih aplikacij preko distribucijskih platform. Pametne mobilne naprave so zasnovane na mobilnem operacijskem sistemu, ki združuje funkcije računalnika s strojno opremo mobilne naprave. Nabor mobilnih operacijskih sistemov je obširen, kar pa razvijalcem mobilnih aplikacij otežuje delo zaradi različnih razvojnih ogrodij in platform.

Poletna šola se na Fakulteti za računalništvo in informatiko odvija že šesto leto zapored. Program, ki ga vodi osebje fakultete, ponuja srednješolcem in študentom namenjene delavnice z zanimivimi in modernimi računalniškimi tematikami. Fakulteta za namene poletne šole pogreša informacijski sistem, preko katerega bi se udeležence poletne šole obveščalo o morebitnih spremembah, prav tako pa bi se razširjalo informacije o delavnicah.

Cilj diplomskega dela je razviti mobilno aplikacijo za udeležence poletne šole, katere programska koda mora delovati na več mobilnih operacijskih

sistemih, saj bi bila kot taka dosegljiva širši množici uporabnikov mobilnih naprav. Željeni cilj lahko dosežemo tudi s pomočjo spletnih tehnologij, ki so optimizirane za mobilne naprave. Tak pristop je bil uporabljen pri razvoju mobilne aplikacije, ki bo podpirala pregled programa, katerega prikaz bo razdeljen dnevno po sekcijah. Urnik programa bo možno urejati in ga prilagoditi osebnim željam. Omogočati mora tudi pregled podrobnosti posamezne delavnice, vnos zapiskov in nastavitev opomnika. Prav tako mora podpirati tudi prikazovanje obvestil in sponzorjev ter urejanje nastavitev.

Poglavje 2

Mobilne aplikacije

Mobilna aplikacija [3] je programska oprema, ki za uporabnika opravlja določena opravila in je namenjena delovanju na mobilni napravi. Najpogosteje je uporabljena na mobilnih telefonih in tabličnih računalnikih. Mobilne naprave zaradi lahke prenosljivosti dandanes uporablja večina prebivalstva v razvitem svetu, zaradi česar so mobilne aplikacije izjemno velik in stalno razvijajoč se trg.

Najbolj razširjene mobilne naprave so mobilni telefoni, ki podpirajo standardne mobilne storitve na širšem geografskem območju. Funkcijski mobilni telefoni so nadgradnja mobilnega telefona, saj ponujajo dodatne funkcionalnosti v obliki mobilnih aplikacij, kot so imenik, koledar in računalo. Te aplikacije so praviloma že naložene na napravo ob nakupu. Naslednja izboljšava je prišla s pametnimi mobilnimi telefoni, ki pa so zasnovani na mobilnem operacijskem sistemu in se kot naprave bolj približajo računskim zmožnostim osebnih računalnikov. Mobilni operacijski sistem razvijalcem ponuja tudi mnogo bolj napreden API za dostop do strojne opreme naprave in boljšo integracijo s sistemom. Pametni mobilni telefoni imajo vgrajenih veliko funkcionalnosti. Poleg osnovnih funkcionalnosti mobilnih telefonov imajo pogosto vgrajeno podporo za multimedijske vsebine, kamero ter povezljivost z brezžičnimi omrežji, tehnologijo Bluetooth, navigacijo GPS itd. tudi razne senzorje. Z vsemi funkcionalnostmi upravlja mobilni operacijski

sistem. Dostop do interneta z mobilno napravo je na pametnih mobilnih napravah običajen, zato mora mobilni operacijski sistem nuditi tudi dobro podporo za povezljivost, tako da so spletni brskalniki večinoma že vgrajeni v sistem. Veliko aplikacij pogosto tudi potrebuje internetno povezavo za komuniciranje s spletnimi storitvami. Tipično in najlažje je mobilno aplikacijo naložiti na mobilno napravo preko distribucijskih platform mobilnih operacijskih sistemov, kjer poznamo plačljive in zastonjske aplikacije. Zaradi dobre podpore internetne povezljivosti ima veliko mobilnih aplikacij oglase. V spodnji tabeli so prikazani trenutno najpopularnejši mobilni operacijski sistemi ter njihove distribucijske platforme.

Razvijalec se slej ko prej sreča s problematiko izbire primerne mobilne platforme. Glede na trend števila prodanih mobilnih naprav za posamezno mobilno platformo, je možno razbrati, katere so med uporabniki najbolj priljubljene. To je običajno tudi eden od razlogov izbire mobilne platforme, saj je želja potencialno čim večje število uporabnikov. Vendar se tudi trendi hitro spreminjajo, zato se pogosto dogodi, da se aplikacija razvije za več mobilnih platform posebej, kar pa je izjemno zamudno in zahtevno opravilo. To je glavni razlog, da se pojavlja vedno več rešitev, s katerimi lahko razvijemo mobilno aplikacijo, ki deluje na več mobilnih platformah. Spletne aplikacije, so na osebnih računalnikih uveljavljene že dalj časa, na mobilnih napravah pa so veliko bolj uporabljene izvirne (angl. native) aplikacije. Spletna aplikacija, ki je optimizirana za uporabo na mobilnih napravah, se lahko uporablja na več mobilnih platformah, kar pa razvijalcem ponuja dobro izbiro. Vendar je potrebno pretehtati prednosti in slabosti, ki jih ponuja razvoj izvirnih in spletnih aplikacij, kot tudi same tehnične zahteve in ciljno publiko mobilne aplikacije.

V nadaljevanju bodo opisane vrste mobilnih aplikacij, ki so prikazane na sliki 2.1.

Slika 2.1: Primerjava delovanja različnih vrst mobilnih aplikacij.

2.1 Izvirne aplikacije

Izvirna mobilna aplikacija je razvita za uporabo na določeni mobilni platformi ali mobilni napravi. Za njihov razvoj potrebujemo najprej vzpostaviti zahtevano razvojno okolje. Nekatere platforme podpirajo več programskih jezikov. Vsaka platforma uporablja drugačen API, zato ponovna uporaba programske kode na drugi platformi ni mogoča. V spodnjih točkah so bolj podrobno zapisane prednosti in slabosti.
Prednosti:

- Hitra odzivnost aplikacije in uporabniškega vmesnika.
- Izris uporabniškega vmesnika opravi mobilni operacijski sistem.
- Dostop do funkcij, informacij ter strojne opreme mobilne naprave.
- Objava aplikacije na distribucijski platformi uporabnikom daje občutek zaupanja in varnosti, saj morajo objavljene aplikacije iti čez proces

objave. Število potencialnih uporabnikov se tako poveča, saj je mobilno aplikacijo lažje najti.

- Aplikacija je naložena na napravo, običajno preko distribucijske platforme.
- Uporaba aplikacije je možna tudi brez internetne povezave, vendar lahko aplikacija upravlja le s podatki, ki so ji na voljo brez povezave.
- Razvojna orodja ponujajo hitrejši in lažji razvoj.

Slabosti:

- Razvoj je potreben za vsako platformo posebej, kar privede do visokih stroškov.
- Težja podpora zaradi možnosti različnih verzij aplikacije (v primeru, da uporabnik ne posodobi na najnovejšo verzijo). Če je mobilna aplikacija objavljena na več platformah, je podpora zahtevna, saj je potrebno napako odpraviti posebej za vsako platformo, kar poveča stroške vzdrževanja.
- Proces objave aplikacije na distribucijski platformi lahko vzame veliko časa in morda zamakne načrtovan datum izida. Nekatere distribucijske platforme za objavo zahtevajo plačljiv uporabniški račun.

2.2 Spletne aplikacije

Spletna mobilna aplikacija je spletna aplikacija, ki je optimizirana za uporabo na mobilnih napravah. Razvita je z uporabo spletnih tehnologij (npr. HTML5, Javascript, CSS) ali spletnih in strežniških tehnologij (npr. PHP, ASP.NET). Za njeno distribucijo je potreben spletni strežnik, na katerem je spletna aplikacija nameščena in na voljo za uporabo preko spletnega brskalnika na mobilni napravi.
Prednosti:

- Cenejši razvoj aplikacije, saj z enkratnim razvojem pokrijemo več mobilnih platform. V primeru, da želimo aplikacijo tudi ponuditi več mobilnim platformam, to razvojni čas občutno skrajša.
- Vsem je na voljo enaka verzija aplikacije, zato je vzdrževanje lažje, saj je potrebno posodobiti le verzijo, ki je na spletnem strežniku.
- Razvoj z uporabo spletnih tehnologij, ki so razvijalcem dobro poznane.
- Obstaja veliko razvojnih ogrodij, s katerimi je mogoče ustvariti uporabniški vmesnik, ki deluje na večini mobilnih platform.

Slabosti:

- Za uporabo je potrebna internetna povezava.
- Odzivnost aplikacije ni tako dobra kot pri izvirnih mobilnih aplikacijah.
- Aplikacijo uporabniki platforme težje najdejo, saj ni objavljena na distribucijskih platformah.
- Ni dostopa do funkcij, informacij ter strojne opreme mobilne naprave.
- Tak razvoj ni primeren za grafično zahtevne aplikacije (npr. igre).

2.3 Hibridne aplikacije

Hibridne mobilne aplikacije so pravzaprav neka srednja izbira med izvirnimi in spletnimi aplikacijami. Zgrajene so na osnovi spletnih tehnologij (HTML5, Javascript, CSS), katere izvorno kodo nato zapakiramo z različnimi razvojnimi ogrodji (npr. PhoneGap) ter jo ponudimo mobilni platformi na enak način kot izvirno.
Prednosti:

- Hitra odzivnost aplikacije in uporabniškega vmesnika. Odzivnost je hitrejša kot pri spletnih mobilnih aplikacijah, saj je aplikacija naložena na mobilno napravo.

- Dostop do funkcij, informacij in strojne opreme mobilne naprave preko razvojnih ogrodij (npr. Phonegap), ki so namenjena razvijanju aplikacij za različne mobilne platforme.
- Objava aplikacije na distribucijski platformi. Število potencialnih uporabnikov se tako poveča, saj je mobilno aplikacijo lažje najti.
- Uporaba aplikacije je možna tudi brez internetne povezave, vendar lahko aplikacija upravlja le s podatki, ki so ji na voljo brez povezave.
- Cenejši razvoj aplikacije, saj z enkratnim razvojem pokrijemo več mobilnih platform. V primeru, da želimo aplikacijo tudi ponuditi več mobilnim platformam, to občutno skrajša razvojni čas.
- Razvoj z uporabo spletnih tehnologij, ki so razvijalcem dobro poznane.
- Obstaja veliko razvojnih ogrodij, s katerimi je mogoče ustvariti uporabniški vmesnik, ki deluje na večini mobilnih platform.

Slabosti:

- Proces objave aplikacije na distribucijski platformi lahko vzame veliko časa in morda zamakne načrtovan datum izida. Nekatere distribucijske platforme za objavo zahtevajo plačljiv uporabniški račun.
- Težja podpora zaradi možnosti različnih verzij aplikacije (v primeru, da je uporabnik ne posodobi na najnovejšo verzijo).
- Uporabniški vmesnik je enak na vseh mobilnih platformah, kar pa ni všeč uporabnikom, saj so vajeni videza izvirnih mobilnih aplikacij. Izdelava uporabniških vmesnikov, ki so prilagojeni mobilni platformi, vzame veliko časa.
- Tak razvoj ni primeren za grafično zahtevne aplikacije (npr. igre).

Poglavje 3

Razvojne tehnologije

V tem oddelku bodo predstavljene vse tehnologije, ki smo jih potrebovali v sklopu razvoja. Prav tako bodo opisane vse knjižnice in ogrodja, ki so bila uporabljena pri razvoju celotnega sistema (mobilna aplikacija in aplikacijski strežnik). V zadnjem delu pa so opisana uporabljena razvojna orodja.

3.1 Tehnologije

Tu bodo naštete vse uporabljene tehnologije.

3.1.1 HTML5

Je označevalni jezik, ki se uporablja za predstavitev strukture in vsebine spletne strani. Gre za peto različico standarda HTML, ki nas obkroža od leta 1990. Zasnovan je bil z namenom zagotavljanja celovite platforme za razvoj aplikacij za spletne strani, ki odpravlja potrebo po nameščanju razširitev brskalnika (npr. Adobe Flash Player). HTML5 [4] je od leta 2007 še vedno v razvoju, vendar mnogi spletni brskalniki že podpirajo številne izboljšave, ki jih prinaša. Nova verzija tako poskuša izluščiti najboljše sestavine prejšnjih verzij ter združiti različne specifikacije funkcij, brskalnikov, navad ter odpraviti vse prepogoste sintaktične napake, ki se pojavljajo na internetnih dokumentih. Dokumenti HTML so drevesna struktura elementov HTML,

ki so sestavljeni iz značk. Te pa lahko vsebujejo besedilo pa tudi druge elemente. Nov standard razširja in izboljšuje že obstoječe značke, kot tudi uvaja nove značke in API za boljšo podporo naprednim spletnim aplikacijam. Tako nam med drugim zagotavlja podporo za 2D grafiko z elementom platno (angl. Canvas element), multimedijske vsebine, funkcijo povleci in spusti, lokalno shranjevanje podatkov, upravljanje z zgodovino brskalnika in spletne aplikacije brez povezave. Številne funkcije so bile razvite z namenom, da lahko delujejo tudi na mobilnih napravah. Ravno zaradi teh razlogov je standard dobro podprt na novejših mobilnih napravah in tako postal izbira za razvoj mobilnih aplikacij.

3.1.2 CSS

CSS so stilne podloge, s katerimi določamo videz in obliko spletnih strani. V njih so zapisana pravila, kako so prikazani elementi dokumentov, ki so pisani z označevalnimi jeziki. Njihovo bistvo je ločitev strukture dokumenta označevalnega jezika od njene predstavitve. Namesto določanja stila vsakemu elementu označevalnega jezika lahko določimo pravilo, ki definira slog več elementov. Stilna podloga se lahko uporablja tudi v več dokumentih. Posledica uporabe je lažje in predvsem hitrejše urejanje in dodajanje stilov, saj so vsi zbrani na enotnem mestu. To v dokumentih odpravi ponavljanje stilov in struktura označevalnega jezika postane bolj berljiva, prav tako pa se zmanjša velikost dokumentov, kar prinese hitrejše nalaganje spletne strani. CSS prav tako podpira različne sloge glede na vrsto naprave, ki dostopa do dokumenta in tako omogoča, da se spletna stran izriše drugače na različnih napravah.

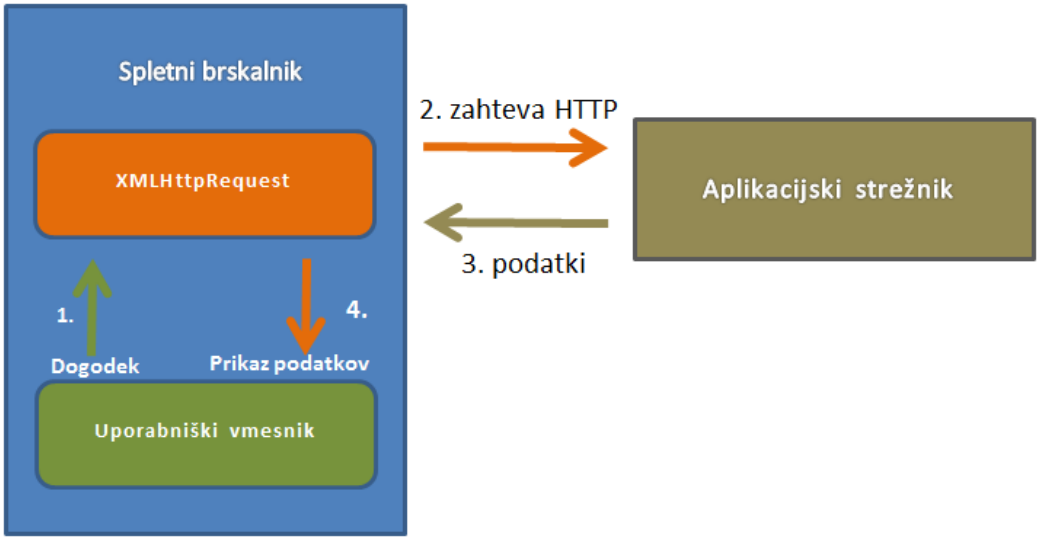
3.1.3 JavaScript

JavaScript [5] je skriptni programski jezik in je hkrati eden najbolj uporabljenih programskih jezikov. Podprt je v novejših spletnih brskalnikih in ostalih spletnih orodjih. Omogoča, da spletne strani, ki so sicer statične, oboga-

timo z interaktivnimi funkcijami. Spletni brskalnik javascript kodo prenese s strežnika in jo nato izvaja na uporabnikovi strani.

Razvit je bil pri podjetju Netscape z namenom omogočiti manipuliranje s spletno stranjo brez ponovnega osveževanja le-te. Netscape 2.0B3 je bil prvi spletni brskalnik z javascript podporo. Prvotno se je imenoval Livescript, vendar je njegovo kasnejše preimenovanje povzročilo zmedo, saj so ga mnogi povezovali s programskim jezikom Java. Njegova programska sintaksa še najbolj sledi sintaksi programskega jezika C.

Spletni brskalniki so najpogostejše okolje uporabe Javascript jezika. Koda se vključi ali doda dokumentu HTML in sodeluje z objektom DOM spletne strani. Ker se izvaja na uporabnikovi strani, je odzivnost aplikacije na uporabnikove akcije hitra.



Slika 3.1: Prenos podatkov z Ajax tehniko.

V zadnjem obdobju je Javascript jezik pridobil na uporabi predvsem na račun Ajax tehnike. Uporablja se za asinhrono izmenjavo podatkov med spletno aplikacijo in strežnikom (slika 3.1). Omogoča takojšnjo osvežitev vsebine brez potrebe po ponovnem nalaganju celotne spletne strani in uporabniku daje vtis veliko bolj tekočega delovanja uporabniškega vmesnika aplika-

cije. Prav tako se zmanjša prenos podatkov, saj s pomočjo XMLHttpRequest objekta pridobi le podatke, ki jih potrebuje. Ajax uporabljajo med drugimi tudi spletne aplikacije Gmail, Facebook in Youtube.

3.1.4 PHP

PHP [1, 6] je odprtokoden programski jezik, ki se uporablja predvsem za izgradnjo dinamičnih spletnih strani. PHP aplikacija za razliko od namiznih aplikacij, ki jo uporablja le ena oseba, običajno deluje na spletnem strežniku in je dostopna širšemu krogu ljudi.

Razvoj se je začel leta 1994, ko je Rasmus Lerdorf ustvaril zbirko skript Perl za upravljanje svoje osebne spletne strani. Zbirko, ki jo je imenoval Personal Home Page Tools, je uporabil za prikaz življenjepisa in zajemanje podatkov o številu obiskovalcev. Trenutno PHP razvija PHP Group in ga nudi kot brezplačno programsko opremo, izdano pod licenco PHP License. Mogoče ga je namestiti na večino spletnih strežnikov, operacijskih sistemov ter ga lahko uporabljamo z mnogimi relacijskimi sistemi za upravljanje podatkovnih baz.

PHP je lahko učljiv, zato je tudi izjemno popularen med razvijalci. Po svoji sintaksi je še najbolj podoben programskima jezikoma C in Perl. Njegov interpreter izvede le kodo med določenimi razmejilniki (angl. delimiter), ostalo pa prezre. Je eden prvih strežniških skriptnih jezikov, ki so lahko vgrajeni v dokumente HTML. Kodo interpretira spletni strežnik s podporo PHP in kot rezultat običajno vrne kodo HTML, ki jo vključi v dokument HTML.

3.1.5 MySQL

MySQL [7] je odprtokodni sistem za upravljanje relacijskih podatkovnih zbirk in za delo s podatki uporablja jezik SQL. Sistem teče kot strežnik in zagotavlja večuporabniški dostop do zbirk podatkov. Napisan je v programskih jezikih C in C++ in deluje na različnih operacijskih sistemih. Zagotavlja

tudi API za veliko programskih jezikov, s katerim lahko dostopamo do podatkovnih zbirk. Veliko popularnost mu je prinesla dobra povezava z jezikom za dinamične spletne strani PHP, saj sta oba brezplačna za uporabo.

Jezik SQL je standardiziran povpraševalni jezik za izvajanje poizvedb v relacijskih podatkovnih zbirkah in je določen s standardom ANSI/ISO SQL. Standard se je razvijal od leta 1986 in je danes najbolj uporabljan povpraševalni jezik.

MySQL strežnik ne vključuje uporabniškega vmesnika za administracijo podatkovnih baz in upravljanje s podatki v podatkovnih bazah kot privzeto, ampak vključuje orodja za uporabo prek ukazne vrstice. Toda obstaja veliko namiznih (npr. MySQL Workbench) kot tudi spletnih (npr. phpMyAdmin) orodij z uporabniškim vmesnikom, ki olajšajo delo z MySQL. Uradno podprto orodje je MySQL Workbench, ki je brezplačno za uporabo in ga razvija podjetje Oracle.

3.1.6 REST

REST [8] (angl. Representational State Transfer) je stil arhitekture programske opreme za porazdeljene sisteme, kot je splet (angl. world wide web). V zadnjih letih se je uveljavil kot prevladujoč model za spletne storitve. REST arhitekturo sestavljajo odjemalci in strežniki. Odjemalec pošlje zahtevek za strežnik, ki ga obdela in vrne ustrezen odgovor. Temelji na predstavitvi virov, ki je lahko katerikoli smiseln obravnavan pojem, predstavitev pa je običajno dokument, ki zajema trenutno stanje vira.

RESTful je spletna storitev, implementirana preko protokola HTTP in principov REST. Je zbirka virov, ki je določena z naslednjimi vidiki:

- Določen mora biti osnoven URI za spletne storitve, s katerim lahko identificiramo vir.
- Vrsta podatka internetnega medija (angl. internet media type), podprta v spletni storitvi, mora biti standardna vrsta. Običajno se uporablja XML.

- Zbirko operacij spletne storitve moramo definirati s pomočjo HTTP metod (npr. GET, PUT, POST, DELETE).
- API mora biti označen z nadbesedilom, ki je način označevanja besedila ali grafičnih elementov, ki omogočajo preskok na drug del besedila ali večpredstavnostni dokument.

RESTful spletne storitve so bile uporabljene pri razvoju aplikacijskega strežnika. Uporabljene so bile kot osrednji člen med mobilno aplikacijo in podatkovno bazo. V primerjavi s spletnimi storitvami SOAP za RESTful spletne storitve ni uradnega standarda, ker je stil arhitekture in ne protokol kot SOAP. Čeprav REST ni standard, RESTful spletne storitve implementirajo ostale spletne standarde, kot so med drugimi tudi HTTP, URI in XML.

3.2 Knjižnice in ogrodja

Razvojne knjižnice in ogrodja razvijalcem zelo olajšajo delo, saj skrajšajo čas, ki bi ga sicer porabili za razvoj vseh funkcionalnosti. Običajno je uporaba knjižnic in ogrodij, ki so množično uporabljana tudi varna, saj je programska koda pod drobnogledom širše množice razvijalcev. Vendar jo je v primeru odkritja varnostne napake potrebno čim prej posodobiti na najnovejšo verzijo, ki te napake odpravlja, saj se tako učinkoviteje izognemo nezaželenim aktivnostim.

3.2.1 jQuery Mobile

jQuery Mobile [9] je spletno razvojno ogrodje z enotnim uporabniškim vmesnikom, ki temelji na HTML5 in zagotavlja podporo za vse popularne platforme mobilnih naprav. Ogrodje temelji na že dobro preizkušenima knjižnicama jQuery in jQuery UI.

jQuery je trenutno najbolj uporabna javascript knjižnica. Namenjena je poenostavitvi razvoja skriptne javascript kode, ki se izvaja na strani odjemalca. Razvoj z njo je hitrejši, saj podpira več spletnih brskalnikov in

tako odpravlja potrebo po pisanju kode, ki je prilagojena vsakemu brskalniku posebej. Njena uporaba je razširjena, saj je odprtokodna in brezplačna za uporabo. Njena sintaksa je zasnovana za navigacijo po dokumentu HTML, ravnanje z dogodki, uporabi elementov DOM, ustvarjanju animacij in razširjanju spletnih aplikacij z Ajax. Knjižnica prav tako podpira razvoj lastnih vtičnikov, ki lahko razširjajo osnovne funkcionalnosti.

Ogrodje jQuery Mobile, ki ga razvija ekipa The jQuery Project, je razvito z miselnostjo podpore vseh mobilnih naprav in mobilnih platform. Vključuje navigacijski sistem, temelječ na Ajax, ki prinaša animirane prehode med stranmi kot tudi osnovni nabor gradnikov uporabniškega vmesnika, kot so strani, pogovorna okna, orodne vrstice in gumbe. Prilagojeno je tudi napravam, ki uporabljajo zaslon na dotik. Razvoj je bil prav tako osredotočen na lahkost uporabe, kar je doseženo s preprostim označevalnim sistemom, ki opredeljuje vedenje gradnikov. Označevalni sistem uporablja HTML5 lastnosti elementov, ki pa so povsem neobvezne za uporabo. jQuery Mobile je združljiv tudi z drugimi mobilnimi ogrodji, kot je Phonegap. Vse spletne strani, ustvarjene z ogrodjem, so zgrajene na osnovi čiste in semantično pravilne kode HTML, tako da delujejo na širšem spektru mobilnih naprav. Ogrodje prav tako vključuje napreden API za upravljanje z dogodki, grafičnimi predlogami kot tudi spreminjanje globalnih nastavitev.

Glavne značilnosti ogrodja:

- Zgrajeno je na osnovi jQuery knjižnice in uporablja njegovo sintakso, ki je poznana mnogim in tako zmanjšuje potrebno učenje za razvoj.
- Podpora za vse pomembnejše mobilne ter namizne platforme.
- Majhna velikost in minimalna odvisnost od slik omogoča hitrejše nalaganje.
- Modularna arhitektura, ki omogoča kreiranje prilagojenih verzij, v katere vključimo le funkcije, ki bodo uporabljene, in tako zmanjšamo velikost knjižnice.

- Nastavitve in delovanje strani je določeno v označevalnem jeziku HTML5. Omogoča hitrejši razvoj ter zmanjšuje pisanje skriptne kode.

- Avtomatsko prilagajanje uporabniškega vmesnika velikosti zaslona naprave.

- Napreden navigacijski sistem Ajax, ki omogoča animirane prehode med stranmi, medtem pa ohranja podporo tipki za vračanje na prejšnjo stran, zaznamek in čisti URL naslov.

3.2.2 Lawnchair

Javascript knjižnica Lawnchair [10] se uporablja za preprosto shranjevanje podatkov v JSON obliki. Njena uporaba je idealna za HTML5 mobilne aplikacije, ki potrebujejo lahko, preprosto in elegantno rešitev za shranjevanje podatkov. Razvita je bila za dobro podporo na mobilnih platformah. Ponuja pregleden in lahek API za uporabo. Izdana je pod licenco MIT.

Knjižnica je bila uporabljena pri izdelavi mobilne aplikacije, saj ponuja vmesnik za shranjevanje podatkov v spletni shrambi (angl. web storage) HTML5 [11], ki pa je povečini podprt na vseh novejših mobilnih platformah. Ponuja tudi vrsto drugih vmesnikov za shranjevanje podatkov po različnih platformah. Spletna shramba HTML5 je nov način shranjevanja podatkov, ki je bil uveden v najnovejši verziji standarda. Spletnim stranem ponuja lokalno shranjevanje podatkov v uporabnikovem spletnem brskalniku. Spletna shramba je hitrejša in bolj varna od včasih uporabljenih piškotkov (angl. cookies). Brskalnik podatkov ne pošilja ob vsaki zahtevi HTTP, kot to počne s piškotki, ampak le ko jih spletna aplikacija zahteva. Možno je shranjevanje velike količine podatkov brez vpliva na zmogljivost spletne strani. Veliko se uporablja tudi za optimizacijo spletnih strani, saj lahko tako podatke, ki se ne spreminjajo, hranimo v shrambi, do njih pa dostopa spletni brskalnik brez vsakokratnega nepotrebnega nalaganja. Do podatkov, ki so shranjeni v obliki asociativnih tabel, lahko dostopa le spletna aplikacija, ki jih je shranila, prav

tako pa je možen dostop brez internetne povezave, saj so shranjeni lokalno v spletnem brskalniku. Spletna shramba se deli na dva dela:

- Lokalna shramba (angl. localStorage), v kateri podatki nimajo življenjske dobe, zato jih spletni brskalnik ne izbriše avtomatsko in so na voljo, dokler jih ne izbriše spletna aplikacija.

- Sejna shramba (angl. sessionStorage) pa hrani podatke le za čas ene seje. To pomeni, da so podatki izbrisani, ko zapremo okno spletnega brskalnika in niso več na voljo za uporabo.

3.2.3 Codeigniter

CodeIgniter [12] je odprtokodno ogrodje za razvoj dinamičnih spletnih aplikacij s programskim jezikom PHP. Cilj ogrodja je razvijalcem ponuditi orodje za mnogo hitrejši razvoj aplikacij s širokim naborom knjižnic za reševanje pogostih nalog, preprost vmesnik in logično strukturo dostopa do knjižnic. Prednost uporabe takega ogrodja je ravno nabor knjižnic, ki nam zagotavlja dostop do delujočih, varnih in dobro testiranih funkcij. Razvijalcu je prihranjeno veliko dela in se lahko osredotoči le na razvoj lastne aplikacije z veliko manjšo količino programske kode, kar prinaša večjo preglednost kode. Prva stabilna verzija ogrodja je izšla leta 2006, razvija pa ga podjetje EllisLab.

CodeIgniter je namenjen spletnim razvijalcem, ki želijo uporabiti ogrodje PHP, ki:

- Pušča zelo malo sledi (angl. footprint), kar pomeni, da za svoje delovanje porabi zelo malo pomnilnika.

- Je izjemno zmogljiv. Jedro ogrodja za svoje delovanje potrebuje zelo malo knjižnic, zato porabi malo sredstev in je zaradi tega tudi zelo hitro. CodeIgniter je med ogrodji PHP znan po svoji hitrosti.

- Ponuja široko podporo spletnim strežnikom in teče na različnih verzijah PHP in različno nastavljenimi strežniki.

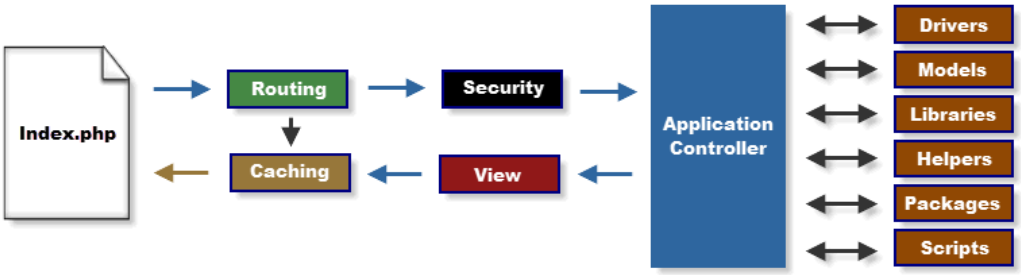
- Za svoje delovanje ne potrebuje spreminjanja množice nastavitev, preden ga razvijalec lahko začne uporabljati.
- Ne zahteva uporabe ukazne vrstice.
- Ne zahteva ravnanja po omejenih pravilih kodiranja.
- Ne zahteva učenja jezika za vizualne predloge in je le-ta na voljo izbirno.
- Daje preproste rešitve brez kompleksne arhitekture.
- Ponuja jasno, natančno in dobro opredeljeno uporabniško dokumentacijo. Programska koda je prav tako dobro dokumentirana in lahko pregledna.

Ogrodje je v večji meri zasnovano na programski arhitekturi MVC. Ta nam omogoča boljšo strukturo programske kode, saj ločuje programsko logiko (v modelu) od predstavitve (v pogledu) in obdelave (v krmilniku). Glavna ideja je učinkovita ločitev kode, kar izboljša ponovno uporabnost kode in tako v večji meri odstrani njeno podvajanje. To pomeni, da sta razvoj in vzdrževanje aplikacije lažja in bolj prijazna razvijalcu, saj se lahko pogled (angl. view) spremeni brez posega v model in obratno. CodeIgniter MVC arhitektura je implementirana le pasivno, saj model ni zahtevan, ampak se ga lahko uporabi po potrebi. Tak pristop sicer ni priporočljiv, vendar ga ogrodje ponuja kot možnost za razvoj manjših aplikacij, ki ne potrebujejo ločene programske logike v modelu.

Model predstavlja osnovno logično strukturo podatkov domene aplikacije. V njem se nahaja programska logika, ki dostopa do podatkov, jih obdela in vrne. Model ni odvisen od nadzornika in pogleda.

Pogled (angl. view) ustvari dokument, ki je vrnjen in prikazan uporabniku. Uporablja se za predstavitvene namene in tako vsebuje kodo za izris uporabniškega vmesnika. Pogled ne spreminja podatkov, vendar jih le umesti v vsebino in prikaže. Zaradi boljše razdrobljenosti so lahko pogledi tudi delni (npr. vsebujejo le glavo spletne strani).

Nadzornik (angl. controller) je povezovalni člen med zahtevami, modelom in pogledom ter ostalimi viri. Tolmači zahteve uporabnika, zahteva in spreminja stanje modela ter skrbi za prikaz pogleda. Običajno vhodne podatke pošlje modelu, ki na podlagi le-teh (če obstajajo) pridobi željene informacije, katere nadzornik nato pošlje ustreznim pogledom.



Slika 3.2: Tokovi podatkov znotraj ogrodja CodeIgniter (vir: [12]).

Tokovi podatkov znotraj sistema ogrodja CodeIgniter potekajo, kot prikazuje slika 3.2, v naslednjem vrstnem redu:

1. Datoteka index.php služi za inicializacijo potrebnih sredstev za izvajanja ogrodja.
2. Usmerjevalnik (angl. router) obravnava zahtevo http, da ugotovi njen namen.
3. Preveritev, ali predpomnilniška datoteka obstaja, – v primeru obstoja vrne neposredno brskalniku ter prekine nadaljnje izvajanje. Kadar je predpomnjenje izklopljeno, se korak preskoči.
4. Preden se zahteva pošlje nadzorniku, se vsi vhodni podatki varnostno pregledajo in filtrirajo.
5. Nadzornik naloži ustrezen model, potrebovane knjižnice, pomočnike in druge vire, ki so potrebni za obdelavo zahteve.
6. Dokument, ki je poslan brskalniku, ustvari pogled. Kadar je predpomnjenje vklopljeno, se dokument shrani pred pošiljanjem.

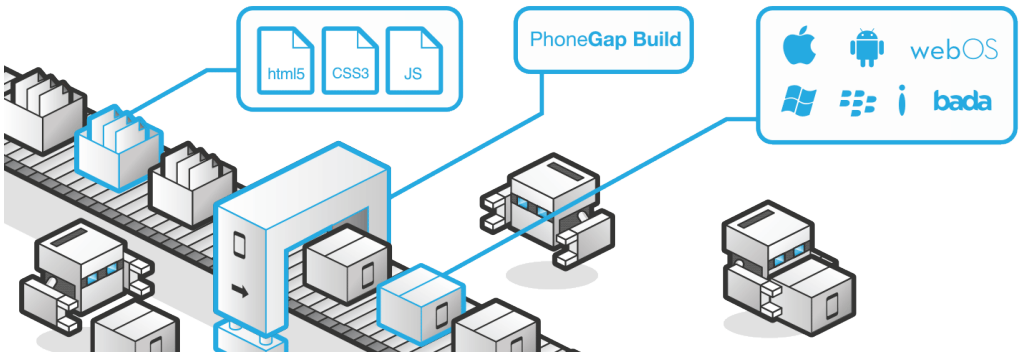
Ogrodje CodeIgniter je bilo v diplomskem delu uporabljeno kot osnova za aplikacijski strežnik. Ogrodju je bila dodana tudi podpora za RESTful spletne storitve, katere so bile implementirane za interakcijo med aplikacijo in podatkovnim strežnikom.

3.2.4 Phonegap

Odprtokodno razvojno ogrodje Phonegap [2, 13, 14] je namenjeno hitremu razvoju mobilnih aplikacij s pomočjo HTML5, Javascript in CSS, ki delujejo na različnih mobilnih platformah. Razvoj medplatformskih aplikacij zahteva uporabo različnih ogrodij in programskih jezikov. Phonegap to rešuje z uporabo standardnih sodobnih spletnih tehnologij, ki so podprte tudi na mobilnih napravah. Tak način razvoja je zelo popularen med razvijalci, saj omogoča razvoj hibridnih mobilnih aplikacij, kot prikazuje slika 3.3. Ogrodje Phonegap so začeli razvijati leta 2008 pri podjetju Nitobi. Trenutno je ogrodje odprtokodno in povsem brezplačno za uporabo. Podpira naslednje mobilne platforme:

- Apple iOS,
- Google Android,
- RIM BlackBerry OS,
- HP webOS,
- Microsoft Windows Phone 7,
- Symbian Symbian OS,
- Samsung Bada.

Ogrodje deluje tako, da spletno aplikacijo, razvito s tehnologijami HTML5, Javascript in CSS, zapakira v ovoj, preko katerega ima spletna aplikacija dostop do raznih funkcij naprave s pomočjo PhoneGap API. Prevedena aplikacija vsebuje dodatno plast uporabniškega vmesnika, saj vključi pogled privzetega spletnega brskalnika mobilne platforme in ga raztegne na 100 %



Slika 3.3: Proces razvoja hibridne mobilne aplikacije s PhoneGap (vir: [23]).

širine naprave in 100 % višine naprave brez kakršnih koli robov. V pogledu teče zapakirana spletna aplikacija ter običajnemu uporabniku deluje kot običajna aplikacija (če je platformi prilagojen tudi uporabniški vmesnik). Zaradi različnih pogonov, ki jih mobilni brskalniki uporabljajo na mobilnih platformah za izris, se lahko na različnih platformah uporabniški vmesnik izriše drugače na kar mora biti razvijalec pozoren. Prevedeno aplikacijo je možno objaviti na distribucijskih sistemih vseh podprtih mobilnih platform.

API nudi dostop do funkcionalnosti operacijskega sistema z uporabo jezika Javascript. Logika aplikacije je pisana z jezikom Javascript za uporabo Phonegap API, ta pa komunicira z operacijskim sistemom za izvršitev želenih funkcij. PhoneGap prav tako podpira izvorne vtičnike, s katerimi lahko razvijemo programsko kodo za mobilno platformo in ustrezen Javascript vmesnik, do katerega lahko dostopamo iz spletne aplikacije. Tako lahko razširimo osnovne funkcionalnosti ogrodja, vendar moramo biti pazljivi, saj običajno razviti vtičniki delujejo le na eni platformi.

PhoneGap ogrodje je le eno izmed mnogih orodij na tržišču, namenjenih razvoju mobilnih aplikacij za več mobilnih platform. Podobna razvojna ogrodja so med drugimi tudi Appcelerator Titanium (iOS in Android), appMobi XDK (iOS in Android), Corona SDK (iOS, Android, Kindle Fire in Nook), MoSync SDK(iOS, Android in Windows Phone) in Rhodes (iOS, Android, BlackBerry OS, Windows Mobile in Windows Phone 7).

3.3 Razvojna orodja

Vsa orodja, ki so bila uporabljena pri razvoju, s katerimi smo si pomagali za čim lažje in čim bolj učinkovito doseganje ciljev.

3.3.1 SAP Sybase PowerDesigner

PowerDesigner [15] je vodilno orodje v industriji za podatkovno modeliranje in rešitev za upravljanje metapodatkov za arhitekturne podatke, informacijsko arhitekturo in podjetniško arhitekturo. Aplikacija deluje v operacijskem sistemu Microsoft Windows ter v razvojnem okolju Eclipse kot vtičnik. Orodje je plačljivo in podpira modelno usmerjen razvoj programske opreme, ki zagotavlja zbirko navodil za strukturiranje specifikacij, ki je kasneje v razvoju zelo uporabna. Modeliranje je vizualno ter nam tako, da boljšo predstavo, kako načrtovati novosti in spremembe programske opreme.

Program je bil uporabljen za podatkovno in aplikacijsko modeliranje, natančneje točno za načrtovanje podatkovnega modela podatkovne baze in modela primerov uporabe s pomočjo jezika UML. UML je standardiziran modelirni jezik za splošne namene na področju objektno usmerjenega razvoja programske opreme. Standard je bil ustvarjen in je upravljan s strani konzorcija Object Management Group. UML vključuje zbirko grafičnih elementov za zapis modelov za načrtovanje.

Program podpira naslednje tehnike modeliranja:

- Podatkovno; podpira konceptualne, logične in fizične modele.
- Aplikacijsko; podpira vse diagrame UML ter nudi napredne objektno relacijske preslikave.
- Poslovno procesno; podpira intuitiven netehničen opis poslovnih procesov ter definicijske diagrame.
- Strukture podjetja; modeliranje od poslovnih ciljev do implementacije.

3.3.2 NetBeans IDE

NetBeans [16] je brezplačno in odprtokodno integrirano razvojno okolje (angl. integrated development environment). Vsebuje vsa potrebna orodja za razvoj namiznih, spletnih in mobilnih aplikacij v Java platformi, prav tako pa podpira tudi razvoj s programskimi jeziki C, C++, Javascript, PHP in Groovy. Orodje je zelo modularno, saj so vse funkcije razvojnega okolja ločene v posamezne module. To orodju omogoča veliko razširljivost, saj mora biti razvit le modul za podporo posameznega programskega jezika. Ponujene module lahko uporabnik prosto vklaplja in izklaplja. Ločitev na module prinaša tudi optimizacijske prednosti, saj se privzeto vklopijo le moduli, ki jih potrebujemo, kar omogoča hitrejše nalaganje razvojnega okolja. Vsak modul ima vključene predloge, ki nas usmerjajo ob začetku razvoja. NetBeans IDE je možno prenesti v različnih paketih, ki so razdeljeni glede na smiselne skupke modulov, pri čemer je jedro razvojnega okolja vedno enako. Okolje podpira tudi vtičnike, ki jih lahko razvijajo zunanji razvijalci ter tako še obogatijo osnovne funkcionalnosti, ki jih ponuja razvojno okolje. Omogočen je direkten dostop do podatkovnih zbirk in spletnih storitev.

Z NetBeans razvojnim okoljem je bila razvita mobilna aplikacija kot tudi aplikacijski strežnik. Zaradi njegove obširne podpore programskim jezikom to olajša razvoj, saj med razvojnimi orodji ni potrebno menjavati, ampak je uporabljeno le eno. Dobra podpora jezikom PHP, Javascript, HTML in CSS vse to omogoča. Razvoj je poenostavljen zaradi avtomatskega obarvanja in dopolnjevanja kode, preverjanja napak, prestrukturiranja kode, namigov ter raznih ostalih funkcij.

3.3.3 MySQL Workbench

Je vizualno orodje, namenjeno skrbnikom, razvijalcem in načrtovalcem podatkovnih baz za uporabo v MySQL sistemu. Orodje MySQL Workbench [17] je razvito za več računalniških platform. Na voljo je v dveh izdajah, ki sta Community Edition (odprtokodna in brezplačna za uporabo, pri njenem ra-

zvoju pa aktivno sodelujejo razvijalci iz celega sveta) ter Standard Edition (podpira vse funkcije Community Edition ter dodaja podporo za preverjanje pravilnosti modelov in izgradnjo dokumentacije). Za potrebe diplomskega dela je bilo uporabljeno za ustvarjanje podatkovnih zbirk ter kasneje za vnašanje, urejanje in pregledovanje podatkov v njej. Grajenje stavkov SQL je s tem orodjem enostavnejše, saj lahko pravilnost rezultatov takoj primerjamo s podatki iz podatkovne zbirke.

Prednosti uporabe:

- Barvanje in avtomatsko dopolnjevanje sintakse kode SQL.
- Izvedbo in prikaz rezultata za več stavkov SQL hkrati, pri čemer lahko vzpostavimo več hkratnih povezav na strežnik MySQL.
- Poenostavlja načrtovanje in vzdrževanje podatkovnih zbirk. Podpira modelno usmerjeno načrtovanje podatkovnih zbirk.
- Omogoča pretvarjanja podatkovnih modelov v predhodne ali naslednje podatkovne modele.
- Izgradnja podatkovnih modelov preko že obstoječih skript SQL.
- Omogoča vizualno primerjavo med podatkovnima zbirkama. Obe sta lahko produkcijski podatkovni zbirki, možna pa je tudi primerjava s podatkovnim modelom.
- Izgradnja dokumentacije podatkovnega modela v dokument HTML.

Poglavje 4

Mobilna aplikacija za izvajanje poletne šole

Poletna šola se na Fakulteti za računalništvo in informatiko odvija že šesto leto zapored. Program, ki ga vodi osebje fakultete, ponuja srednješolcem in študentom namenjene delavnice z zanimivimi in modernimi računalniškimi tematikami. Fakulteta za namene poletne šole pogreša informacijski sistem, preko katerega bi udeležence poletne šole obveščali o morebitnih spremembah. Preko sistema pa bi prav tako potekala distribucija informacij o delavnicah.

Postavitev takega sistema je primerna naloga za diplomsko delo. Za prikaz informacij je bila izbrana mobilna aplikacija, ki mora delovati na različnih mobilnih napravah. Sam sistem je bil razdeljen na dva dela. Vnos in urejanje podatkov poletne šole sta bila razvita v sklopu drugega diplomskega dela, to diplomsko delo pa prikazuje razvoj mobilne aplikacije. Zaradi enotnega sistema je pred začetkom razvoja moral biti načrtovan podatkovni model. Prav tako pa potrebujemo med mobilno aplikacijo in podatkovno zbirko nek vmesni člen za prenos podatkov.

Razviti je bilo potrebno:

- podatkovni model,
- aplikacijski strežnik,

- mobilno aplikacijo.

4.1 Analiza

4.1.1 Zajem zahtev

Zajem zahtev je bil prvi izmed korakov razvoja. Potrebno je bilo zajeti čim več podatkov o delovanju mobilne aplikacije. Običajno jih zajame analitik s poznavalcem problematike. Priporočljivo je, da so prisotni tudi morebitni uporabniki aplikacije. Zajete zahteve naslednjim korakom razvoja zagotavljajo boljše razumevanje problematike. Zajem zahtev smo razdelili na dva sklopa z vidika razvoja mobilne aplikacije.

Funkcionalne zahteve:

- Uporabnik se lahko prijavi v sistem ali odjavi iz njega. V primeru, da še nima uporabniškega računa, mu mora biti omogočena njegova pridobitev.
- Omogočen mora biti pregled programa, kar pomeni prikaz delavnic.
- Delavnice so ločene glede na sekcije. Vsaka sekcija ima določeno svojo prepoznavno barvo. To mora biti upoštevano pri prikazu programa.
- Uporabnik lahko ureja urnik in ga prilagodi svojim potrebam. V tem sklopu uporabnik določa, katere delavnice bodo prikazane v programu.
- Omogočen mora biti pregled obvestil.
- Omogočen mora biti pregled podrobnosti delavnice. Podrobnosti morajo vsebovati ime in časovno trajanje delavnice, ime in priimek predavatelja, ime sekcije, opis delavnice in ime dvorane. Prikazana morajo biti tudi kratka navodila dostopa do dvorane.
- Uporabnik si lahko za posamezne delavnice piše zapiske.

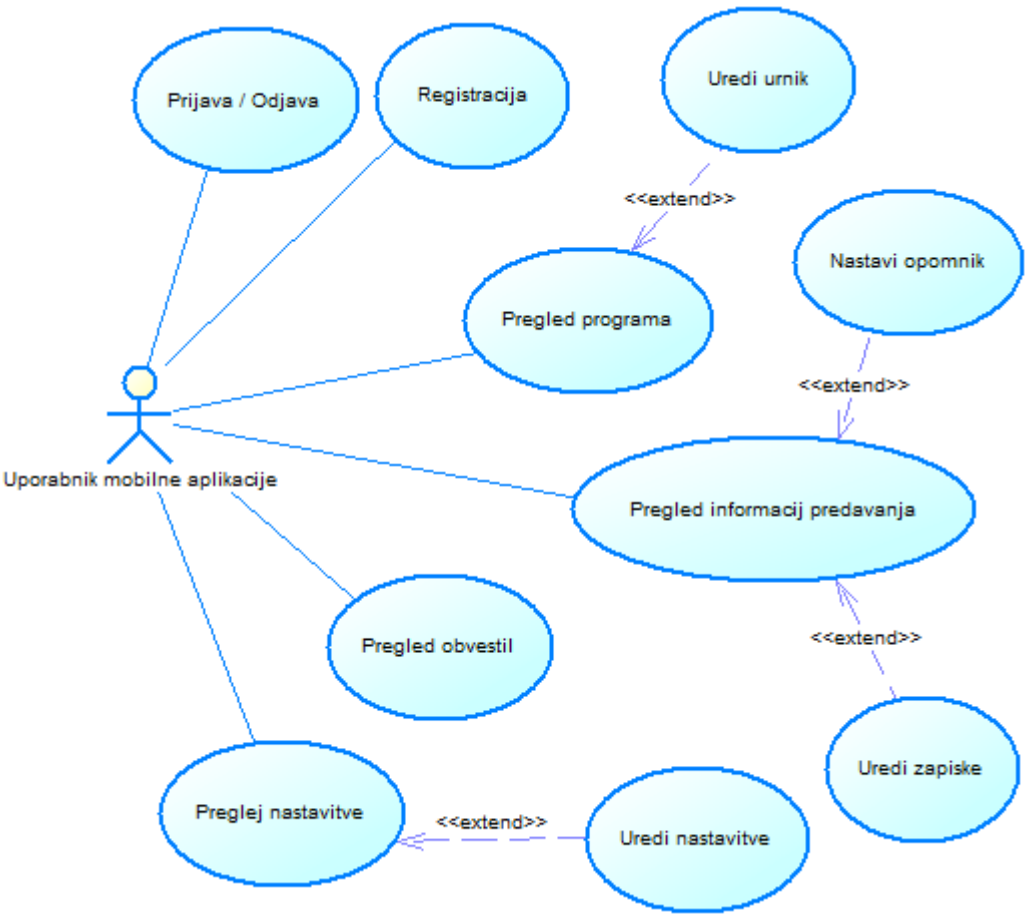
- Uporabnik si lahko za posamezno delavnico nastavi opomnik.
- Omogočen mora biti prikaz obvestil. Obvestila so ločena glede na tipe obvestil.
- Aplikacija mora vsebovati nastavitve, kjer uporabnik sam določi, kateri tip obvestil naj bo prikazan. Prijavljeni uporabniki lahko nastavitev za pošiljanje obvestil na elektronski naslov, ki je privzeto vklopljena, poljubno spremenijo.
- V začetnem oknu morata biti prikazana ime dogodka ter njegov logotip.
- Mobilna aplikacija mora prikazovati sponzorje na vseh glavnih oknih aplikacije. Sponzorji morajo biti razdeljeni po pomembnosti. Bolj pomembni morajo imeti večjo verjetnost, da se prikažejo v mobilni aplikaciji.

Tehnični kriteriji:

- mobilna aplikacija mora delovati na več mobilnih platformah,
- uporabniški vmesnik mora biti prilagojen za uporabo v napravah z zaslonom na dotik,
- podatki morajo biti na voljo za uporabo tudi brez omrežne povezave,
- vsa gesla morajo biti pred pošiljanjem v omrežje šifrirana,
- sporazumevanje med mobilno aplikacijo in podatkovno zbirko mora potekati preko spletnih storitev.

4.1.2 Primer uporabe

Diagram primera uporabe je najpreprostejša vizualna predstavitev medsebojnega delovanja sistema z uporabnikom in zunanjimi sistemi. Opisuje, kako bi akter uporabljal sistem za izpolnitev določenega cilja oziroma opravila. Na sliki 4.1 je prikazan primer uporabe mobilne aplikacije glede na zajete zahteve.



Slika 4.1: Diagram primera uporabe mobilne aplikacije.

4.2 Načrtovanje

4.2.1 Arhitektura

Mobilna aplikacija je namenjena prikazovanju informacij, ki pa jih mora pridobiti iz zunanjega vira. Informacije so shranjene v podatkovni zbirki na nekem oddaljenem strežniku, do katerega lahko dostopamo iz mobilne naprave preko interneta. Mobilne naprave imajo pogosto različno strojno opremo in niso tako zmogljive kot osebni računalniki. Da bi samo napravo čim manj obremenjevali, je pomembno, da programsko logiko poizkusimo ločiti od prikaza. Zato uvedemo vmesni člen, ki bo ob zahtevi dostopal do podatkov in jih po obdelavi vrnil odjemalcu. Arhitektura med odjemalcem in strežnikom, ki vsebuje vmesni člen, se imenuje trinivojska arhitektura [18] (slika 4.2) in se deli na:

- Odjemalca, ki se nahaja na prvem nivoju (predstavitveni nivo). Uporabniški vmesnik se uporablja za izris in pridobivanje informacije, ki so potrebne za izvršitev zahtevanih opravil. V našem primeru je odjemalec mobilna aplikacija.

- Aplikacijski strežnik se nahaja na drugem nivoju (aplikacijski nivo) in procesira zahteve, pridobiva podatke, nad njimi izvrši vso poslovno in procesno logiko ter jih vrne v obliki informacij. Krajše povedano; podatke usmerja in procesira med ostalima nivojema.

- Podatkovni strežnik deluje na tretjem nivoju (podatkovni nivo), na katerem se nahaja podatkovna zbirka, ki hrani podatke.

Za sistem Poletne šole FRI je bila uporabljena tri nivojska arhitektura, pri čemer se na prvem nivoju nahaja mobilna aplikacija, na drugem nivoju spletni strežnik, ki ponuja spletne storitve, in na tretjem nivoju strežnik MySQL.

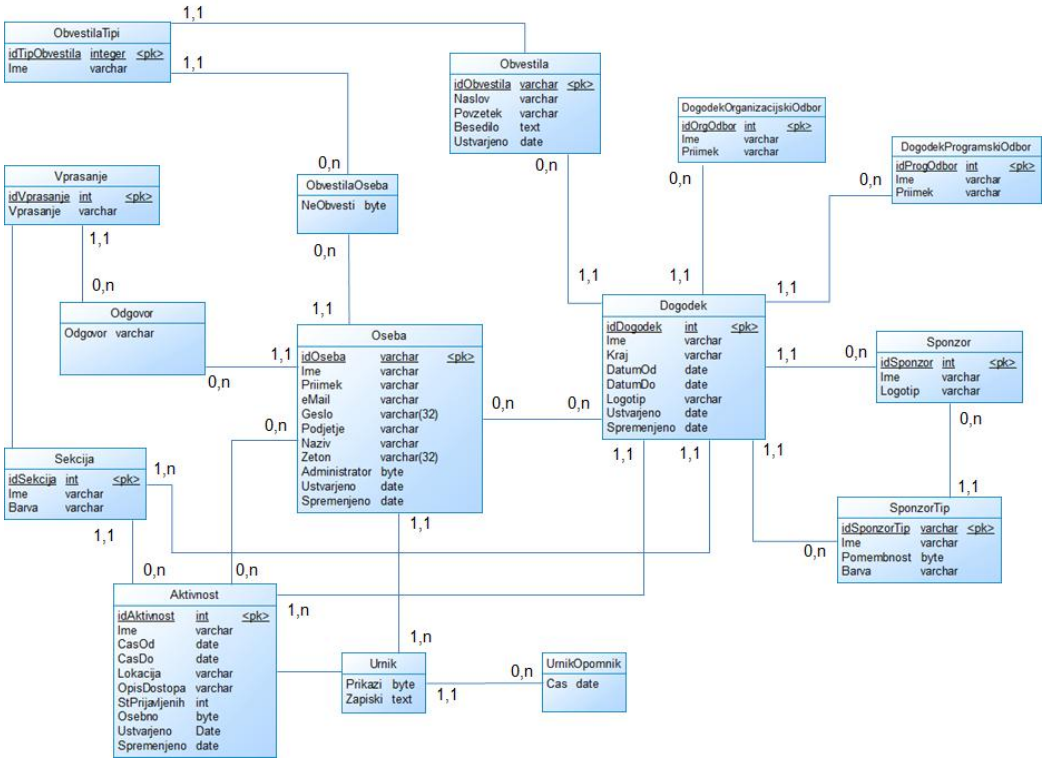
Slika 4.2: Trinivojska arhitektura.

4.2.2 Podatkovni model

Za opis podatkovnega modela je bil uporabljen entitetno-relacijski diagram. Velikokrat je to eden izmed prvih korakov v načrtovanju aplikacije. Ustvari jih načrtovalec, ki pozna domeno problematike. Diagram nam ponazarja relacije med posameznimi entitetami v podatkovnem modelu, pri čemer je entiteta lahko oseba, prostor, dogodek ali objekt obravnavane domene, medtem ko relacija med dvema entitetama pomensko opisuje povezavo. Uporabljen podatkovni model je prikazan na sliki 4.3.

4.3 Aplikacijski strežnik

Aplikacijski strežnik smo uporabili za prenos podatkov med mobilno aplikacijo in podatkovno zbirko. Zasnovan je na ogrodju CodeIgniter, kateremu smo dodali knjižnico (codeigniter-restserver) za podporo spletnih storitev RESTful. Aplikacijski strežnik za svoje delovanje potrebuje spletni strežnik s podporo PHP. Osnovna struktura je bila postavljena pred razvojem mobilne aplikacije, vendar je razvoj virov spletnih storitev potekal vzporedno z razvojem mobilne aplikacije.



Slika 4.3: Podatkovni model.

Implementacija spletnih storitev podpira izmenjavo podatkov v obliki XML, JSON, CSV in HTML. Izmenjava podatkov poteka privzeto v obliki JSON, ki je skriptni jezik za preprost opis podatkov. JSON izhaja iz jezika Javascript in predstavlja zapis objekta. Ogrodje CodeIgniter razvijalce spodbuja k uporabi arhitekturnega vzorca MVC, kateremu z uporabo knjižnice codeigniter-restserver [19] ne sledimo, saj izpisa podatkov ne opravi preko pogleda. Knjižnica vsebuje razred, v katerem je implementirana podpora storitvam RESTful, razred za spreminjanje oblik podatkov ter datoteko z nastavitvami. Knjižnico moramo tudi naložiti v ogrodje pred uporabo. Razred v nadzorniku v ogrodju CodeIgniter privzeto razširja razred *CI_Controller*, toda zaradi uporabe knjižnice mora razširjati razred *REST_Controller* (implementiran v dodani knjižnici). Metode, ki jih kasneje ustvarimo, morajo slediti vzorcu *vir_(zahteva HTTP)*. Primer dejanske uporabe je prikazan v kodi 4.1, kjer so vir aktivnosti in zahtevana metoda HTTP POST. Poleg POST knjižnica podpira tudi GET, PUT in DELETE. Za dostop do podatkov, v zahtevah HTTP, so na voljo metode *get()*, *post()*, *put()*, *delete()*, za vračanje informacij pa metoda *response()*. Pri aplikacijskem strežniku smo v nadzorniku upravljali le sprejem in preverjanje podatkov ter vračanje informacij, programska logika in dostop do podatkov pa se nahajata v modelih.

Koda 4.1: Vir aktivnosti v spletnih storitvah RESTful.

```
function aktivnosti_post()
{
        $dogodek = $this->_postDogodek();

        $eMail = $this->post('email');

        $this->load->model('aktivnost');
        $this->_response($this->aktivnost->
            dobiAktivnostiDogodka($dogodek));
}
```

4.4 Mobilna aplikacija

Mobilna aplikacija je namenjena vsem udeležencem poletne šole, za katero mora prikazovati ključne informacije. Zasnovana je na hibridni vrsti mobilne aplikacije, ki se sklada z opisanimi tehničnimi kriteriji iz poglavja 4.1.1. Izdelana je s pomočjo HTML5, Javascript in CSS tehnologij. Za uporabniški vmesnik smo uporabili razvojno ogrodje jQuery Mobile. Razvoj uporabniškega vmesnika je voden preko označevalnega jezika HTML5. Potrebno je ustvariti dokumente HTML5, v katerih z elementi HTML in njihovimi atributi ogrodju dajemo navodila, kako naj se izriše uporabniški vmesnik. V dokumentu, ki je prikazan prvi (glavni meni), je potrebno vključiti vse potrebne knjižnice, ogrodja in stilne predloge, uporabljene v aplikaciji. Ker jQuery Mobile temelji na knjižnici jQuery, mora biti le-ta vključena v dokument pred prvim. jQuery Mobile s pomočjo knjižnic in stilskih predlog na podlagi gradnikov, uporabljenih v dokumentu HTML, ustvari uporabniški vmesnik prilagojen zmogljivostim mobilne naprave. Vsaki zaslonski maski smo ustvarili tudi ločeno Javascript datoteko, v kateri obravnavamo dogodke, pridobivamo informacije, katere dinamično prikažemo v uporabniškem vmesniku, in vso ostalo programsko logiko. Za prehod med zaslonskimi maskami poskrbi ogrodje sámo s pomočjo Ajax (v primeru, da mobilna naprava to tehniko podpira) in jih obogati z animacijami. V enem dokumentu imamo lahko tudi več zaslonskih mask programa, vendar smo jih ločili na posamezne dokumente zaradi preglednosti.

Ogrodju jQuery Mobile smo morali spremeniti tudi nekaj privzetih nastavitev ter prevesti nekaj ukazov oziroma sporočil v slovenščino. To nam je uspelo storiti s kodo 4.2, ki pa jo je potrebno vključiti v dokument, preden se vključi ogrodje jQuery Mobile. Ob uspešni vključitvi ogrodja ta sproži dogodek *mobileinit* v objektu *document*, koda 4.2 pa ta dogodek ujame in prepiše privzete nastavitve.

Koda 4.2: Sprememba privzetih nastavitev ogrodja jQuery Mobile.

```
$(document).bind("mobileinit", function () {
```

```
    $.mobile.loadingMessage = "nalagam";
    $.mobile.loadingMessageTextVisible = true;
    $.mobile.pageLoadErrorMessage = "Napaka pri
        nalaganju";
    $.mobile.addBackBtn = true;
    $.mobile.page.prototype.options.backBtnText = "
        Nazaj";
});
```

Glavni jezik uporabniškega vmesnik je slovenščina, saj jo bodo v veliki večini uporabljali slovenski uporabniki glede na dejstvo da je Poletna šola FRI organizirana v Sloveniji. Za bolj celovito podobo mobilne aplikacije je izredno pomembna uporaba enotnega jezika uporabniškega vmesnika v celotni aplikaciji. Potrebno je bilo preimenovati sporočilo o nalaganju strani, sporočilo o napaki ter gumb za vrnitev, ki se pojavi na podstraneh. Poleg jezikovnih sprememb je nastavljeno, da se sporočilo o nalaganju vedno prikaže, prav tako pa mora biti na vsaki podstrani prikazan navigacijski gumb *Nazaj*.

4.4.1 Glavni meni

Je prva zaslonska maska mobilne aplikacije in predstavlja izhodišče za vse podprte funkcionalnosti. Za lažje in bolj učinkovito upravljanje z aplikacijo je uporabniški vmesnik preprost in pregleden (slika 4.4). Omogočena je tudi dobra podpora za mobilne naprave z zasloni na dotik, saj so gradniki veliki in jasni. Zaslonska maska je razdeljena na glavo in vsebino. Glava je del vseh zaslonskih mask in se loči od ostalega vmesnika s črno barvo. V zaslonski maski glavnega menija vsebuje ime dogodka in povezavo, ki odpre obrazec za prijavo uporabnika v sistem. Na vseh zaslonskih maskah, razen v glavnem meniju, glava vsebuje tudi gumb *Nazaj*, ki uporabnika vrne za eno stopnjo nazaj proti glavnemu meniju. Vsebina zaslonske maske prikazuje logotip dogodka, povezavo do programa, povezavo do obvestil, povezavo do nastavitev in logotip sponzorja. Povezave so realizirane preko elementov A

jezika HTML in vsebujejo veliko atributov, ki so navodila za ogrodje jQuery Mobile. Z atributom *data-role*, ki mu nastavimo vrednost *button*, ogrodju sporočimo, naj povezavo izriše v obliki gradnika gumb. Povezavam so nastavljene različne ikone za lažje prepoznavanje. Sliki obeh logotipov podpirata različne ločljivosti zaslona mobilne naprave in se za pravilen prikaz avtomatsko prilagodita.

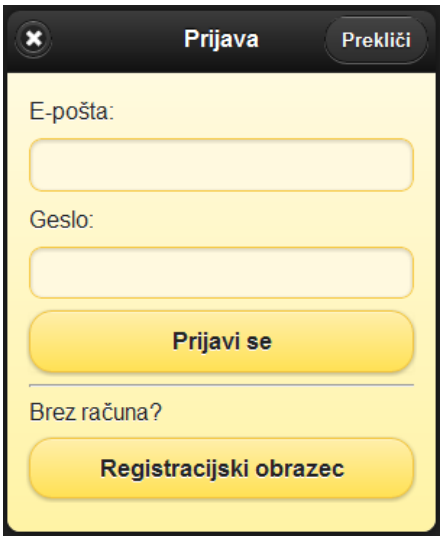
Slika 4.4: Glavni meni mobilne aplikacije.

4.4.2 Prijava in registracija

Registracija v sistem in uporaba kot prijavljeni uporabnik je ponujena izbirno. Vse funkcionalnosti mobilne aplikacije delujejo enako prijavljenim in neprijavljenim uporabnikom. Prijavljeni uporabniki lahko prejemajo obvestila na elektronski naslov. Prijava in registracija je bila implementirana zaradi mogočih bodočih funkcionalnih nadgradenj mobilne aplikacije in možnih sinhronizacij, ki bi preprečevale izgubo shranjenih podatkov in nastavitev.

Klik na gumb *Prijava*, ki se nahaja v glavnem meniju, odpre novo okno (slika 4.5), v katerem se prikaže obrazec, katerega mora uporabnik za prijavo v sistem izpolniti. Obrazec vsebuje dve vnosni polji in sicer za vnos

elektronskega naslova in gesla. Uporabniki so identificirani z elektronskim naslovom. Ko uporabnik z vnosom podatkov konča mora pritisniti na gumb *Prijavi se*, ki v ozadju izvrši proces preverjanja pristnosti uporabnika. Podatki se pošljejo na spletno storitev, ki preveri podatke in vrne rezultat. V primeru uspešnega rezultata mobilna aplikacija prejme žeton, ki ga skupaj z uporabnikovim elektronskim naslovom shrani v lokalno shrambo HTML5, kot prikazuje koda 4.3, s pomočjo knjižnice Lawnchair. Žeton predstavlja začasno identifikacijo pristnosti uporabnika in se uporablja za nadaljnje izkazovanje pristnosti ob klicu spletnih storitev.



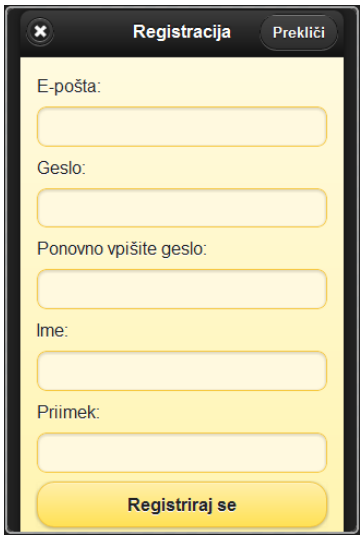
Slika 4.5: Prijavno okno mobilne aplikacije, ki omogoča prijavo v sistem.

Koda 4.3: Shramba podatkov s pomočjo knjižnice Lawnchair.

```
var uporabnik = new Lawnchair({name:'uporabnik'},
    function(uporabnik) {
        uporabnik.save({key:'email',data:email});
        uporabnik.save({key:'zeton',data:data.zeton});
});
```

Ob kliku na gumb *Registracijski obrazec* nam mobilna aplikacija prijavno

okno zapre in samodejno odpre novo okno (slika 4.6), v katerem se nahaja obrazec, ki uporabniku omogoča ustvariti svoj račun. Potreben je vnos elektronskega naslova, gesla, ponovni vpis gesla ter imena in priimka uporabnika. Ko so vsi podatki vnešeni, mora uporabnik pritisniti gumb *Registriraj se*, zatem pa mobilna aplikacija pošlje podatke na spletno storitev, ki poizkuša ustvariti novega uporabnika. Ob uspešni registraciji spletna storitev vrne uporabnikov žeton in uporabnika prijavi. V spodnjem delu okna se nahaja tudi gumb, ki nas preusmeri nazaj na prijavni ekran.

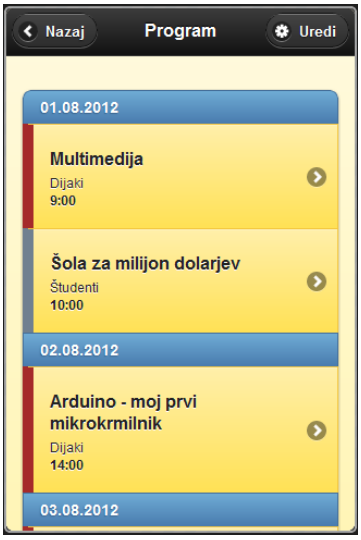


Slika 4.6: Obrazec za vzpostavitev novega uporabniškega računa.

4.4.3 Program

V tej zaslonski maski (slika 4.7) se izpiše seznam vseh delavnic poletne šole. V glavi uporabniškega vmesnika se nahaja gumb *Uredi*, ki nas preusmeri k urejanju urnika. Informacije o programu mobilna aplikacija dobi preko spletne storitve s pomočjo razvojne tehnike Ajax in dinamično ustvari seznam. Vsi podatki o delavnicah se shranijo lokalno na mobilni napravi za možen kasnejši dostop, ko mobilna naprava ne bi imela internetne povezave. Seznam se loči na dnevne skupine začetka posamezne delavnice in vsebuje povezave

do podrobnosti posamezne delavnice. Vsaka delavnica v seznamu vsebuje tudi kratek opis. Na skrajno levem predelu posamezne povezave se nahaja barvni stolpec, ki prikazuje barvo sekcije posamezne delavnice in uporabniku omogoča lažji pregled, kateri sekciji pripada delavnica. Povezava vsebuje tudi ime delavnice, ime sekcije in uro začetka. Puščica na desni strani povezave uporabniku simbolizira, da gre dejansko za povezavo do podrobnosti posamezne delavnice in da pritisk nanjo nekaj izvede. V seznamu se prikazujejo le delavnice, katerih prikaz je omogočen. Privzeto so prikazane vse delavnice. Seznam je zgrajen s pomočjo gradnika *listview* (jQuery Mobile).

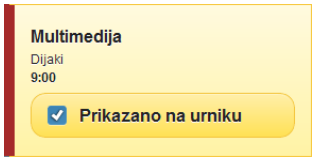


Slika 4.7: Prikaz programa poletne šole.

4.4.4 Urejanje urnika

Prikazano in zgrajeno je na podoben način kot program. Loči se le v tem, da seznam ne vsebuje povezav, ampak elemente, namenjene le prikazovanju informacij posameznih delavnic, ter da je dodan nov gumb, s katerim lahko spreminjamo prikaz delavnice na seznamu. Spremenjena podoba posamezne delavnice v seznamu urejanja urnika, je prikazana v sliki 4.8. Gumb, s katerim

spreminjamo prikaz na programu, vsebuje tudi kljukico, ki prikazuje trenutno stanje prikaza.

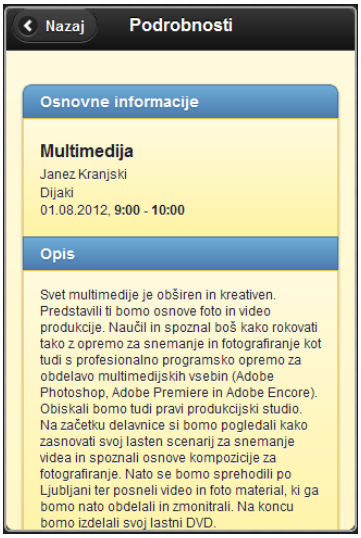


Slika 4.8: Urejanje prikaza delavnice v programu.

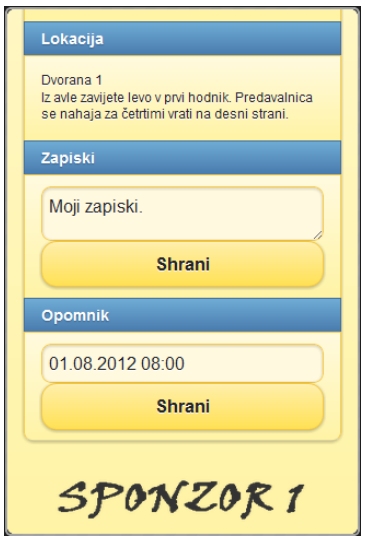
4.4.5 Podrobnosti delavnice

Vse informacije povezave s posamezno delavnico se nahajajo v zaslonski maski *Podrobnosti* (slika 4.9 in slika 4.10). Zaslonska maska vsebuje seznam, ki je zaradi preglednosti ločen na enote:

- Osnovne informacije; vsebuje ime delavnice, ime in priimek predavatelja, ime sekcije ter datum začetka delavnice, ki vsebuje tudi uro začetka in konca delavnice. V tej enoti so torej zbrane najbolj pomembne informacije delavnice.
- Opis; vsebuje besedilo, ki opisuje program in cilje delavnice.
- Lokacija; vsebuje ime dvorane ter krajša navodila za dostop do dvorane.
- Zapiski; vsebuje vnosno polje in gumb Shrani. Vnosno polje je namenjeno zapiskom udeleženca delavnice. Vanj lahko vnaša poljubno besedilo in ga shrani s pritiskom na gumb. Zapiski so vidni tudi v naslednjih prikazih zaslonske maske *Podrobnosti*.
- Opomnik; vsebuje vnosno polje in gumb Shrani. Ob kliku na vnosno polje se nam odpre novo okno, prilagojeno napravam na dotik, preko katerega lahko preprosto izberemo datum in čas opomnika. Opomnik je potrebno potrditi na gumb *Shrani*.



Slika 4.9: Prikaz podrobnosti delavnice 1. del.



Slika 4.10: Prikaz podrobnosti delavnice 2. del.

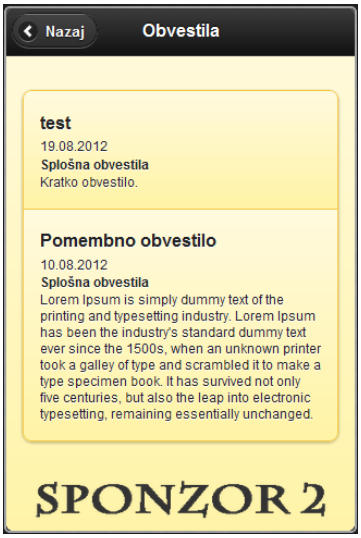
Opomnik je narejen s pomočjo knjižnice mobiscroll, katero smo uporabili za časovno nastavitev opomnika. Odlikuje jo dobra podpora za naprave z zaslonom, različne mobilne platforme, ogrodje jQuery Mobile in razne teme uporabniškega vmesnika. Opomnik se prikaže v novem oknu (slika 4.11) in vsebuje pomikalna kolesca za vsako enoto posebej ter gumba za potrditev oziroma preklic. Knjižnica, ki zasede zelo malo prostora, je dobro dokumentirana in ponuja veliko nastavitev, s katerimi jo lahko prilagodimo lastnim potrebam. V oknu so uporabljeni le elementi za izbiro datuma in časa, ki pa smo jih morali tudi prevesti v slovenščino.

Slika 4.11: Okno za izbiranje časa opomnika.

4.4.6 Pregled obvestil

Zaslonska maska *Obvestila*, ki je prikazana na sliki 4.12, udeležencem omogoča pregled vseh obvestil poletne šole. Obvestila so v seznamu prikazana dinamično, saj jih aplikacija pridobi preko spletne storitve. Podprto je tudi pregledovanje obvestil brez internetne povezave, vendar brez sinhronizacije. Spletna storitev informacije vrne v obliki JSON. Primer je prikazan v kodi 4.4. Posamezno obvestilo vsebuje naslov obvestila, datum vnosa, ime tipa obvestila ter besedilo obvestila. V seznamu so obvestila razvrščena od najno-

vejšega proti najstarejšemu. Seznam obvestil lahko vsebuje le obvestila, ki pripadajo tipu obvestila, katerega prikaz je dovoljen v nastavitvah mobilne aplikacije. Tu pa se pojavi problem, saj spletna storitev za neprijavljene uporabnike nima informacij, kateri tipi obvestil ne smejo biti prikazani. Rešitev problema dosežemo z enotno identifikacijo posameznega tipa obvestil v mobilni aplikaciji in podatkovni zbirki. V zahtevku HTTP spletne storitve je poslan tudi parameter filter, ki vsebuje tabelo vseh identifikacij tipov obvestil, katerih prikaz ni dovoljen. Glede na vhodne podatke spletna storitev vrne vsa primerna obvestila.



Slika 4.12: Prikaz obvestil poletne šole.

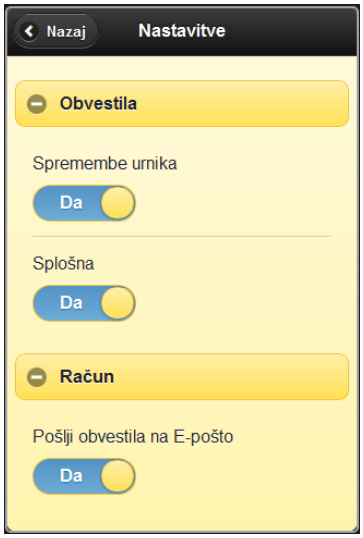
Koda 4.4: Primer izmenjave podatkov v obliki JSON.

```
{"item":{"id":"4","Naslov":"test","Povzetek":"Kratek
    povzetek","Besedilo":"Kratko obvestilo.","Ime":"Splo
    \u0161na obvestila","Ustvarjeno":"19.08.2012"}}
```

4.4.7 Nastavitve

Zadnja povezava iz glavnega menija vodi do nastavitev aplikacije. Nastavitve se delijo na naslednja zložljiva sklopa:

- Obvestila; sklop je viden prijavljenim in neprijavljenim uporabnikom. Vsebuje drsnike posameznih tipov obvestil, s katerimi lahko uporabnik vklopi ali izklopi prikaz tipa obvestila med obvestili mobilne aplikacije.
- Račun; sklop je viden le prijavljenim uporabnikom. Vsebuje drsnik *Pošlji obvestila na E-Pošto*, s katerim lahko uporabnik vklopi ali izklopi pošiljanje obvestil poletne šole na uporabnikov elektronski naslov.



Slika 4.13: Zaslonska maska za urejanje nastavitev.

4.4.8 Prikaz sponzorjev

Prikaz logotipa sponzorjev je omogočen na vseh zaslonskih maskah. Sponzorji so ločeni s tipi sponzorjev. Vsak tip sponzorja ima določeno pomembnost na lestvici od 1 do 5, pri čemer 1 pomeni najmanjšo in 5 največjo pomembnost. Logotipi sponzorjev morajo biti v mobilni aplikaciji prikazani tudi takrat, ko

le-ta nima internetne povezave. Mobilna aplikacija ob vsakem zagonu preveri informacije o sponzorjih poletne šole, saj se lahko podatki spreminjajo. Spletna storitev ob zahtevi najprej pridobi spisek vseh sponzorjev, ki pripadajo poletni šoli, nato pa vsak logotip posebej kodira v ASCII tekst s pomočjo metode Base64. Metoda se uporablja za pretvorbo binarnih podatkov v ASCII tekst in obratno. Ko je obdelava končana, spletna storitev vrne informacije o sponzorjih mobilni aplikaciji, ta pa jih shrani v lokalno shrambo HTML5, da bodo na voljo tudi brez internetne povezave. Mobilna aplikacija zatem ustvari vrstni red 50 naključnih sponzorjev glede na pomembnost. Bolj pomembni sponzorji imajo za prikaz logotipa v mobilni aplikaciji veliko večjo verjetnost kot manj pomembni sponzorji. Ob vsaki menjavi zaslonske maske mobilna aplikacija vzame logotip prvega sponzorja v vrsti, ga dinamično prikaže ter odstrani iz vrste. Ko se vrsta izprazni, se ponovi postopek izgradnje nove vrste.

4.5 Varnost

Vsa uporabnikova gesla so pred pošiljanjem iz mobilne aplikacije v aplikacijski strežnik kodirana s kodirnim algoritmom SHA-256 [21], ki izhaja iz zbirke zgoščevalnih funkcij SHA-2 in je le eno izmed štirih različic (SHA-224, SHA-256, SHA-384 in SHA-512) algoritma SHA-2. Različice se razlikujejo glede na uporabljeno število bitov v ključu, ki jo ponazarja številka poleg imena. Kodirni algoritem ni tako množično uporabljan kot njegov predhodnik SHA-1, toda ponuja boljšo varnost. Napadi, ki so bili uspešno izvedeni nad algoritmom SHA-1, so bili neuspešni z uporabo SHA-2. Uporaba SHA-256 je bila v mobilni aplikaciji izvedena s pomočjo Javascript knjižnice CryptoJS [20], ki je razvita z uporabo najboljših vzorcev in praks. Podpira veliko kodirnih algoritmov, vendar smo za naše potrebe vključili le algoritem SHA-256. Njena uporaba je preprosta, saj geslo, ki ga podamo kot parameter, metoda *SHA256* vrne kodirano z algoritmom SHA-256.

Varnost bi lahko izboljšali tudi z uporabo komunikacijskega protokola

HTTPS. Omogoča zaščiten promet na podlagi protokola HTTP, kateremu doda podporo za TLS in SSL, ki sta kodirna protokola. Za njegovo uporabo bi moral spletni strežnik, na katerem je postavljen aplikacijski strežnik, podpirati protokol HTTPS. Vsi podatki, izmenjani med mobilno aplikacijo in aplikacijskim strežnikom, bi bili med prenosom kodirani, ko bi tudi v mobilni aplikaciji nastavili, naj izmenjava podatkov poteka preko HTTPS.

4.6 Testiranje

Testiranje mobilne aplikacije je potekalo z orodjem PhoneGap Emulator [22]. Orodje omogoča testiranje mobilnih aplikacij, razvitih z ogrodjem PhoneGap. Za uporabo orodja potrebujemo:

- spletni brskalnik Google Chrome,
- vtičnik Ripple Emulator za spletni brskalnik Chrome.

PhoneGap Emulator uporablja posnemovalnik (angl. emulator) mobilnega okolja Ripple Emulator, ki podpira več mobilnih platform. Z orodjem lahko posnemamo uporabo mobilne aplikacije v različnih mobilnih napravah. Mobilne platforme uporabljajo različne pogone spletnih brskalnikov, na katerih delujejo hibridne mobilne aplikacije. Zato je pomembno, da razvijalec aplikacijo testira na različnih mobilnih napravah, saj lahko prihaja do razlik v delovanju. Poleg različnih mobilnih naprav lahko posnemamo tudi omrežje naprave in senzor pospeškomer (angl. accelerometer) ter senzor GPS.

4.7 Prevajanje

PhoneGap Build [23] je oblačna spletna storitev, ki omogoča prevajanje izvorne kode hibridne mobilne aplikacije, izdelane s HTML5, CSS in Javascript, v izvirne mobilne aplikacije. Podpira prevajanje aplikacij za mobilne platforme iOS, Android, Windows Phone, BlackBerry OS, webOS, Symbian OS in Bada. Storitev bo brezplačna za uporabo, dokler bo v razvojni različici

beta, zatem pa bo brezplačna le še za odprtokodne projekte. Razvijalcem hibridnih mobilnih aplikacij prihrani veliko dela, saj razvojnega okolja ni potrebno vzpostaviti za vsako platformo posebej, ampak izvorno kodo prenesemo v storitev, ki nato izdela prevode za vse podprte mobilne platforme. Prevedeno aplikacijo lahko objavimo tudi v distribucijski platformi podprtih mobilnih platform, toda za iOS, Android in BlackBerry OS je potrebno predhodno uvoziti ključe oziroma certifikate, ki objavo dovoljujejo. Storitev ponuja tudi obilico uporabnih funkcij, izmed katerih je bolj zanimiva podpora razhroščevanju (angl. debugging), ki omogoča pregled informacij aplikacije, ki deluje na mobilni napravi kar preko spletnega brskalnika. Aplikacije, zgrajene z ogrodjem PhoneGap, morajo v izvorni kodi le vključiti ogrodje, storitev pa datoteke ogrodja avtomatsko doda k aplikaciji. Storitev je bila uporabljena v zadnji fazi razvoja za dejansko testiranje mobilne aplikacije na mobilnih napravah z uporabo ustvarjenih prevodov.

Poglavje 5

Sklepne ugotovitve

Glavni cilj diplomskega dela je dosežen. Razvita hibridna mobilna aplikacija uspešno realizira vse zajete zahteve in je pripravljena za uporabo na široki množici mobilnih naprav. Udeležencem poletne šole torej omogoča enostaven pregled ključnih informacij in obvestil. Prvotni cilj se je zaradi oddaljene hrambe podatkov razširil v mali informacijski sistem, katerega razvoj je potekal z uporabo dobrih razvojnih ogrodij in knjižnic, ki temeljijo na najboljših vzorcih in praksah. Morali smo biti pazljivi predvsem na varnost, modularnost in vzdržljivost celotne arhitekture.

Razvoj s pomočjo spletnih tehnologij HTML5, CSS in JavaScript ima zagotovo dobro prihodnost tudi v svetu mobilnih naprav. Na tem področju je veliko spremembo pripravil še vedno razvijajoč se standard HTML5, ki zagotavlja celovito razvojno platformo in dokazuje, da bo z uporabo spletnih tehnologij mogoče ustvariti kakršnokoli vrsto aplikacije. Podpora se stalno izboljšuje, saj nekatere platforme že uvajajo JavaScript API za dostop do funkcionalnosti operacijskega sistema. Trenutno stanje podpore v mobilnih napravah je dobro, vendar ne najboljše. Izdelan uporabniški vmesnik ne ponuja tako dobre izkušnje kot uporabniški vmesnik izvirne mobilne aplikacije, saj je odzivnost počasnejša, prav tako pa so opazna rahla zatikanja še posebej na manj zmogljivih mobilnih napravah. Ogrodje jQuery Mobile ravno iz teh razlogov svoj razvoj še vedno usmerja k izboljšanju uporabniške izkušnje in

jo z vsako izdano različico tudi izboljšuje.

Med izdelavo mobilne aplikacije so se pojavile nove ideje o mogočih izboljšavah oziroma popravkih delovanja sistema:

- Trenutno razvita mobilna aplikacija ponuja pregled programa le enega dogodka (Poletna šola FRI). Njeno delovanje bi lahko razširili na več dogodkov in tako izdelali univerzalno mobilno aplikacijo za udeležence raznih konferenc. Uporabniki bi na mobilno napravo namestili le eno mobilno aplikacijo in ne vsake posebej za vsak dogodek.

- Izdelava sinhronizacijskega sistema, ki bi občutno izboljšal sinhronizacijo podatkov med mobilno aplikacijo in aplikacijskim strežnikom. Sinhronizacijski sistem bi shranjeval uporabnikove spremembe, dokler mobilna naprava ne bi vzpostavila povezave z internetom, nato pa poslal spremembe v aplikacijski strežnik. S tem bi se izognili morebitnim izgubam podatkov, prav tako pa bi udeleženci aplikacijo uporabljali na različnih mobilnih napravah z vsemi svojimi spremembami.

- Urejanje urnika po sekcijah. Poletna šola je razdeljena na delavnice za dijake in študente, pri čemer študentje v svojem programu ne želijo videti delavnic, namenjenih dijakom in obratno. Trenutno med urejanjem urnika lahko izklopimo prikaz vsake delavnice posebej, s to izboljšavo pa bi lahko izklopili prikaz vseh delavnic posamezne sekcije in tako uporabnikom prihranili nekaj časa.

- Omogočiti izvoz vseh shranjenih zapiskov uporabnika in mu tako omogočiti njihovo uporabo tudi zunaj sistema. Izvoz bi uporabniki lahko izvedli na pomnilnik mobilne naprave v obliki podatkovne datoteke ali pa po elektronski pošti.

- Izvoz programa v koledar mobilne naprave za mobilni platformi iOS in Android. API ogrodja PhoneGap trenutno ne ponuja dostopa do koledarja mobilne naprave, vendar so na voljo vtičniki, ki ogrodju to podporo dodajajo za omenjeni mobilni platformi. Za njeno izvedbo bi

mobilna aplikacija morala preveriti, na kateri mobilni platformi deluje, ter nato izvesti kodo, ki bi bila prilagojena vsakemu vtičniku posebej.

- Izboljšanje komunikacije med predavatelji in udeleženci. Predavatelj dogodka bi preko sistema vsem udeležencem delavnice zastavil vprašanje, ti pa bi nanj lahko odgovorili kar preko mobilne aplikacije. Predavatelji bi na ta način prejeli uporabne in zanimive povratne informacije.

V diplomskem delu smo prikazali, da je za informativno mobilno aplikacijo, kot je poletna šola, hibriden razvoj aplikacij povsem primerna izbira, prav tako pa omogoča hiter razvoj in posledično manjšo porabo finančnih sredstev. Razvita aplikacija sloni na uporabi popularnih brezplačnih ogrodij in knjižnic, kar zagotavlja dobro podporo in bodoče izboljšave. V kolikor se bo razvit sistem tudi dejansko uporabljal, bi bilo z razvojem smiselno nadaljevati.

Slike

Literatura

[1] David Sklar *Learning PHP 5*, Sebastopol: O'Reilly Media, Združene države Amerike, 2004, pogl. 1.

[2] John M. Wargo *PhoneGap Essentials: Building Cross-Platform Mobile Apps*, Boston: Addison-Wesley Professional, Združene države Amerike, 2012, pogl. 1.

[3] (2012) Mobile app. Dostopno na: http://en.wikipedia.org/wiki/Mobile_app.

[4] (2012) HTML5 Tutorial. Dostopno na: http://www.w3schools.com/html/html5_intro.asp.

[5] (2012) JavaScript. Dostopno na: http://en.wikipedia.org/wiki/JavaScript.

[6] (2012) PHP. Dostopno na: http://en.wikipedia.org/wiki/PHP.

[7] (2012) MySQL. Dostopno na: http://gradiva.txt.si/racunalnistvo/racunalniska-omrezja/70_strezniki/mysql/.

[8] (2012) Representational state transfer. Dostopno na: http://en.wikipedia.org/wiki/Representational_state_transfer.

[9] (2012) jQuery Mobile. Dostopno na: http://jquerymobile.com/.

[10] (2012) Lawnchair. Dostopno na: http://brian.io/lawnchair/.

[11] (2012) HTML5 Web Storage. Dostopno na: http://www.w3schools.com/html/html5_webstorage.asp.

[12] (2012) CodeIgniter. Dostopno na: http://codeigniter.com/.

[13] (2012) PhoneGap. Dostopno na: http://phonegap.com/.

[14] (2012) PhoneGap Development. Dostopno na: http://www.metaltoad.com/services/phonegap-application-development.

[15] (2012) SAP Sybase PowerDesigner. Dostopno na: http://www.sybase.com/products/modelingdevelopment/powerdesigner/

[16] (2012) NetBeans. Dostopno na: http://netbeans.org/

[17] (2012) MySQL Workbench. Dostopno na: http://www.mysql.com/products/workbench/

[18] (2012) Trinivojska arhitektura. Dostopno na: http://colos1.fri.uni-lj.si/ERI/RACUNALNISTVO/PODATKOVNE_BAZE/trinivojska_arhitektura.html

[19] (2012) codeigniter-restserver. Dostopno na: https://github.com/philsturgeon/codeigniter-restserver

[20] (2012) CryptoJS. Dostopno na: http://code.google.com/p/crypto-js/

[21] (2012) SHA-1 and SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512) Encryption Algorithm. Dostopno na: http://www.vocal.com/cryptography/sha-algorithm/

[22] (2012) PhoneGap Emulation. Dostopno na: http://emulate.phonegap.com/

[23] (2012) PhoneGap Build. Dostopno na: https://build.phonegap.com/