

# Diplomski rad

## Mobilna Web aplikacija za pretraživanje i razmenu geo-referenciranih informacija

**Zadatak:** Proučiti savremene tehnologije i standarde za razvoj mobilnih Web aplikacija za rad sa geolokacijom, lokalnim skladištenjem podataka i push notifikacijama. U praktičnom delu razviti mobilnu Web aplikaciju koja poseduje karakteristike lokaciono-zasnovanih servisa (LBS) i RIA (*Rich Internet Application*) i funkcionalnost za pretraživanje i proaktivno obaveštavanje o geo-referenciranim informacijama. Posebno ilustrovati principe razmene korisnički generisanog sadržaja kao osnove Mobile/Web 2.0 koncepta.

**Kandidat:** Mirko Borivojević, 11722

Datum prijave: \_\_\_\_\_

Datum predaje: \_\_\_\_\_

Datum odbrane: \_\_\_\_\_

**Komisija:**

1. \_\_\_\_\_

2. \_\_\_\_\_

3. \_\_\_\_\_

## Sadržaj

1.	Uvod.....	5
2.	Mobilni web .....	7
2.1.	Istorija mobilne telefonije.....	7
2.2.	Istorija mobilnih web tehnologija .....	9
2.3.	Prilagođenje web strana karektistikama mobilnog telefona.....	13
2.4.	Principi i preporuke za razvoj mobilnog weba .....	18
2.5.	Mobilni web za pametne telefone .....	22
3.	Mobilne web tehnologije i standardi .....	25
3.1.	HTML 5 .....	25
3.2.	CSS3.....	27
3.3.	JavaScript / Ajax .....	29
3.4.	Lokalni keš web aplikacije .....	31
3.5.	Geolokacija.....	33
3.6.	Grafika i multimedija.....	35
3.7.	Kratak pregled mobilnih Web aplikacionih okvira .....	39
4.	Mobilna Web aplikacija za razmenu korisnicki generisanog sadržaja .....	41
4.1.	Generalni opis i zahtevi aplikacije.....	41
4.2.	Korišćeni alati, okvir i tehnologije .....	42
4.2.1.	Jquery mobile.....	42
4.2.2.	CakePHP .....	43
4.2.3.	NodeJS.....	44
4.3.	Generalna arhitektura aplikacije.....	45
4.4.	Baza podataka.....	46
4.4.1.	Korisnici .....	47
4.4.2.	Poruke .....	47
4.4.3.	Kategorije .....	47
4.4.4.	Tačke od interesa .....	47
4.4.5.	Vezna tabela.....	47
4.5.	Serverska aplikacija za upravljanje tačkama od interesa.....	48
4.6.	Aplikacija za slanje notifikacija.....	57
4.7.	Mobilna aplikacija .....	59
5.	Funkcionalne karakteristike i evaluacija aplikacije .....	67
5.1.	Prikaz glavnih funkcionalnosti aplikacije.....	67

5.2.	Uporedni prikaz izvršenja aplikacije na različitim platformama .....	73
5.3.	Razmatranje karakteristika u odnosu na nativnu aplikaciju .....	76
6.	Zaključak.....	77



## 1. Uvod

Poslednjih godina sa prodorom pametnih telefona i tablet računara naglo povećava se korišćenje mobilnih aplikacija. Najpre native mobilne aplikacije, koje se izvršavaju na operativnom sistemu uređaja dobijaju na popularnosti zbog superiornih mogućnosti koje imaju u odnosu na mobilne internet aplikacije. U prvom redu je tu mogućnost nativnih aplikacija da koriste senzore uređaja kao što su GPS, kamera ili akcelerometar kao i funkcije operativnog sistema uređaja za pristup i smeštanje podataka na disk, rad sa grafikom i multimedijom i pristup telefonskom imeniku, listi kontakata i ostalim funkcijama telefona.

Razvojem HTML5 tehnologije smanjuje se jaz između tehnoloških mogućnosti nativnih i mobilnih internet aplikacija. Mobilne internet aplikacije dobijaju mogućnosti korišćenja hardverskih i softverskih resursa uređaja (GPS, kamera, smeštanje podataka) pa se na tržištu pojavljuje veliki broj mobilnih RIA aplikacija. Prednost koje internet aplikacije imaju u odnosu na native je izvršenje nezavisno od platforme pa je cena gotove aplikacije višestruko manja.

Iako je HTML5 standard još uvek u fazi razvoja a mobilni internet pregledači svakodnevno dodaju podršku za nove funkcionalnosti. Mogućnost mobilnih internet aplikacija svakog dana postaje sve veća pa se u bliskoj budućnosti može očekivati da one zauzmu vodeće mesto ispred nativnih aplikacija u pogledu korišćenja.



## 2. Mobilni web

### 2.1. Istorija mobilne telefonije

Pre 38 godina Martin Kuper (Martin Cooper) je šetajući ulicama Nju Jorka napravio prvi telefonski poziv na prototipu mobilnog telefona, preteći savremenih mobilnih uređaja. Trideset i osam godina kasnije skoro svaki drugi stanovnik planete ima u svom vlasništvu mobilni telefon koji nije ni nalik na prvi telefon čija je težina iznosila, sa današnje tačke gledišta, iznenađujućih 2 kilograma!

Ni najveći vizionari toga vremena nisu mogli da predvide neverovatnu brzinu kojom će se mobilne tehnologije razvijati kroz godine kao i povećanje broja mobilnih korisnika širom sveta, koje obara sve do tada postavljene rekorde. Tako je radio programu bilo potrebno 38 godina da dostigne 50 miliona korisnika, televiziji je za isto bilo potrebno 13 godina, dok je broj mobilnih korisnika u periodu od 1990 do 2010 narastao sa 12 miliona na čak preko 4.5 milijardi. [1]

Razvoj moderne mobilne telefonije od svog postanka do danas se odigrao u četiri generacije.

Prva generacija mobilnih telefona uvodi ćelijsku organizaciju pokrivenosti signalom mobilne mreže koja i danas čini osnovni način organizacije mobilnih baznih stanica. Svaka bazna stanica pokriva mobilnim signalom područje u obliku ćelije a komutacioni centri daju mogućnost transfera poziva iz jedne ćelije u drugu što omogućuje prostorno kretanje korisnika kroz ćelije u toku razgovora.

Druga generacija mobilnih telefona počinje 1990 godine uvođenjem GSM standarda. On se razlikuje od svoga predhodnika jer koristi digitalni umesto analognog signala za prenos informacija. Uvođenjem 2G tehnologije počinje era korišćenja mobilne telefonije a broj pretplatnika mobilnih usluga naglo raste.

Sa rastom broja mobilnih pretplatnika i prihvatanjem mobilnih telefona u svakodnevni način života rasla je i potreba za uvođenjem novih servisa kao što je na primer mobilni pristup internetu. Takođe, mobilna tehnologija je bila spremna za uvođenje većih brzina prenosa podataka što je rezultovalo trećom generacijom mobilne telefonije koja je pružila mobilnim korisnicima pristup internetu. Međutim veća popularnost mobilnog interneta (najpre u Americi) se nije desila sve do razvoja šire pokrivenosti dobrim 3G signalom sredinom 2000 godine.

Konačno četvrta generacija mobilne telefonije donosi poboljšanje brzine prenosa podataka i nove servise kao što su striming videa i telefoniranje putem interneta (VOIP).

U isto vreme i jednakom brzinom traje razvoj interneta, globalne mreže povezanih računara, i WWW servisa izgrađenog na osnovama interneta. Dolaskom interneta na mobilne uređaje kreće razvoj mobilne web tehnologije i mobilnog weba. Razvijaju se nove tehnologije za pristup mobilnom internetu i novi jezici za formatiranje web strana za mobilne uređaje.

Pojam mobilni internet može da označava korišćenje internet sadržaja preko internet pregledača na mobilnom telefonu ili iz nativnih aplikacija. Razlika između web aplikacija i mobilnih (nativnih) aplikacija je nastala nemogućnošću web aplikacija da koriste hardverske prednosti uređaja kao što su akcelerometar ili GPS senzor, kao i podrške nativnih aplikacija za snimanje sadržaja na memoriju

telefona, bolji korisnički interfejs i podrške za 3d grafiku. Razvojem modernih web tehnologija, koje će biti opisane u narednom poglavlju, smanjiće se jaz među tehničkim mogućnostima web i nativnih aplikacija. To će omogućiti razvoj jednako tehnički zahtevnih web aplikacija, pa će se korišćenje termina mobilni internet odnositi na prenos podataka putem interneta i njihovu obradu i korišćenje bilo u nativnim bilo u mobilnoj web aplikaciji. Iako danas native aplikacije dobijaju daleko veću pažnju javnosti istraživanje comScore [7] nalazi da web aplikacije za malo prednjače u odnosu na native aplikacije kada je u pitanju korišćenje mobilnog weba. Na primer 36 procenata mobilnih korisnika u Americi i 29 procenata evropljana je koristilo mobilni web decembra 2010 godine, dok je native aplikacije koristilo 34 procenata Amerikanaca i 28 procenata evropljana.

Ipak, kako je predmet diplomskog rada razvoj mobilnih web aplikacija na dalje ću zanemariti razmatranje nativnih mobilnih aplikacija ne umanjujući pri tom njihov značaj.

Za samo deset godina istorije mobilnih telefona oni preuzimaju vodeću kategoriju personalnih digitalnih uređaja koju su do tada zauzimali personalni računari. Nedavno je CEO firme Apple, Steve Jobs, izjavio da se nalazimo u post-PC eri kada će personalne računare zameniti prenosivi digitalni uređaji kao što su tableti i pametni telefoni. Ja se u velikoj meri slažem sa ovom tvrdnjom obzirom na superiornost mobilnih uređaja u prilagođenju i integraciji u svakodnevni život ljudi koji personalni računari nemaju.

Velika količina personalizovanog sadržaja koji se servira preko mobilnog telefona daje neuporedivo korisničko iskustvo u odnosu na bilo koji drugi digitalni uređaj tako da oni postaju deo "ličnog pribora" svakog stanovnika planete.

Neverovatni trendovi razvoja mobilnog tržišta gotovo da onemogućavaju dugoročno predviđanje njegovog razvoja. Ipak zanimljiva su sledeća kratkoročna predviđanja kompanije Cisco [8].

Do 2015 godine će postojati više od 5,6 milijardi ručnih ili personalnih mobilnih uređaja. Broj mobilnih uređaja se razvija brže nego broj mobilnih pretplatnika koji ih koriste.

Mobilni uređaji postaju sve brži pa će zato povećati količinu digitalnog sadržaja koji su u stanju da proizvedu ili da koriste. Jasan primer daju tableti, pametni telefoni i drugi mobilni uređaji koji su se pojavili u zadnjih godinu dana. U periodu 2010-2015 mobilni saobraćaj će preći količinu ostvarenog fiksnog saobraćaja 3,3 puta.

Brzina mobilnih mreža je jedan od ključnih faktora za povećanje mobilnog saobraćaja. Veće brzine znače veću konzumaciju sadržaja pa se zato projektuju nove mobilne mreže (uključujući 2G, 3G i 4G mreže) koje će ubrzati protok podataka 10 puta do 2015 godine.

Želja mobilnih pretplatnika je da dobiju najbolji mogući doživljaj mobilnih komunikacija što često znači upotrebu bogatih media kao što je slika ili video. Zato će mobilni video zauzimati dve trećine ukupnog mobilnog protoka podataka do 2015 godine.

Bilo ovo predviđanje tačno ili ne moje mišljenje je da će mobilna industrija biti jedna od dominantnih u narednoj deceniji.



## 2.2. Istorija mobilnih web tehnologija

Pristup internetu i WWW servisu bio je u početku dostupan jedino korišćenjem fiksne veze ali vremenom se tehnologija razvija i internet postaje sve više dostupan i na mobilnim uređajima. Dolaskom interneta na mobilne uređaje počinje era mobilnog interneta. Pojam mobilni internet u najširem smislu označava pristup internetu korišćenjem svih vrsta mobilnih uređaja kao što su mobilni i pametni telefoni, prenosivi računari, GSM modemi i drugi. U daljem tekstu će pojam mobilni internet označavati isključivo korišćenje interneta na mobilnim telefonima i tablet računarima koje zajedno karakterišu ograničene hardverske karakteristike.

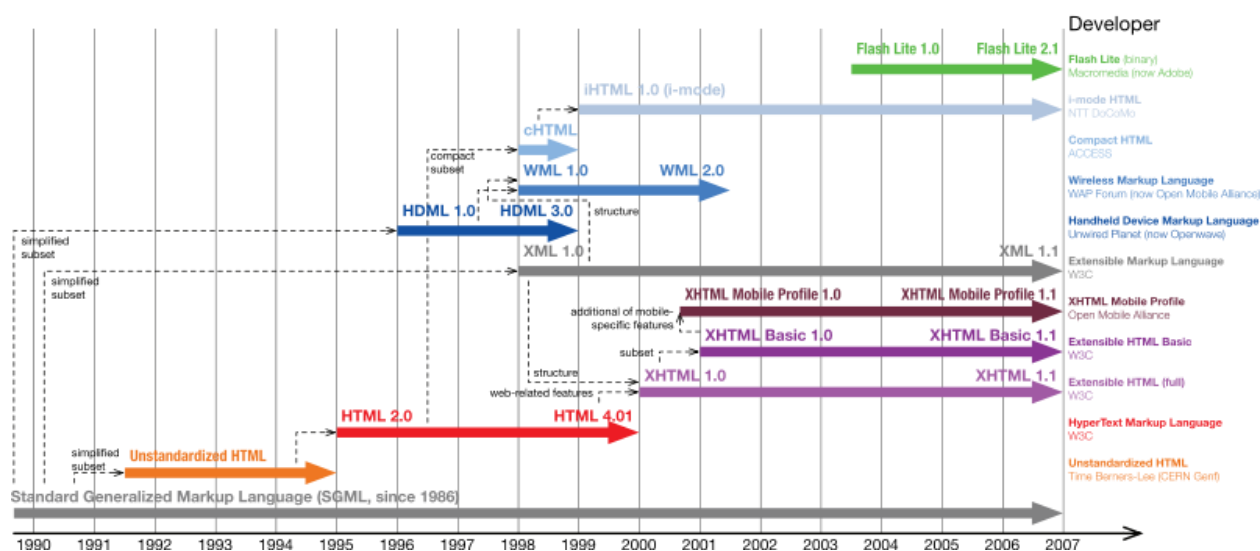
Pristup mobilnom internetu je prvi put komercijalno ponuđen u Finskoj 1996 godine na telefonu Nokia 9000 Communicator preko Sonera i Radiolinja mobilnih operatera. Prvi komercijalni mobilni web servis je pokrenut u Japanu 1999 godine kada je NTT DoCoMo ponudio i-mode. [3] Dva dominantna standarda u osnovi mobilnog weba postaju Wireless Application Protocol (WAP) i i-mode.

WAP standard opisuje skup protokola za komunikaciju koji abstrahuje arhitekturu same mobilne mreže pa time rešava problem prenosa podataka između različitih tipova mobilnih mreža kao što su GSM ili IS-95. Sa uvođenjem WAP standarda mobilni operateri su dobili mogućnost da razvijaju interaktivne servise podataka koji su dali podršku mobilnim web aplikacijama nove generacije kao što su, email na telefonu, praćenje cena na berzi, sportski rezultati, vesti i drugi.

i-mode standard ima istu namenu kao i WAP standard ali uz nešto drugačiju tehničku realizaciju. Kao što je rečeno i-mode ostaje popularan jedino u Japanu pa svetsku dominaciju uzima WAP. Danas međutim korišćenje WAP skoro da u potpunosti nestaje a njegovu ulogu zamenjuje tradicionalni TCP/IP standard. [4]

Za predstavljanje web strana na mobilnom webu razvija se niz različitih jezika markiranja. Iako je u to vreme postojao HTML kao standard za razvoj web strana prilagođenih personalnim računarima bilo je neophodno razvijati nove jezike i sintakse prilagođene mobilnim uređajima jer su oni bili jako ograničeni po pitanju pristupa internetu, veličini memorije i procesorske moći. Zbog toga je bilo neophodno razvijati mala i optimizovana rešenja za prikaz web strana na mobilnim uređajima jer mobilni internet pregledači nisu mogli da obrade nestandardne oznake i stranice sa sintaksnim greškama. Takođe je trebalo dodati i funkcionalnosti specifične mobilnim uređajima kao što su prečišćivači na tastaturi.

## Evolution of Mobile Web-Related Markup Languages



Slika 1. Evolucija jezika markiranja (markup) mobilnog web-a

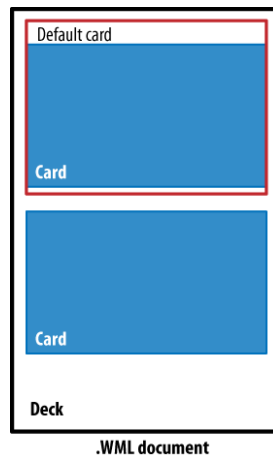
Jedan od prvih je bio jezik HDML razvijen od strane kompanije Unwired Planet (kasnije postaje Openwave). HDML jezik je bio sličan HTML-u i nikada nije postao standard za razvoj mobilnog weba ali je poslužio kao osnova pri razvoju WML standarda.

WML jezik je bio uključen u WAP 1.1 standard i postaje prvi standard objavljen za mobilni web. Objavljen je 1998 godine od strane WAP foruma (danas poznatog kao Open Mobile Alliance) koji je okupljao razne kompanije sa ciljem da razvijaju standarda za mobilno tržište. Ovaj standard je za svoju osnovu uzeo mešavinu drugih markap jezike koje su koristili različiti proizvođači mobilnih uređaja kao što su HDML firme Openwave, Tagged Text Markup Language (TTML) firme Nokia i markap jezik korišćen na Ericsson mobilnim telefonima.

WML standard je danas skoro potpuno napušten. Pametni telefoni nemaju podršku za WML dok neki stariji uređaji mogu da ga čitaju iz istorijskih razloga.

WML 1.1 standard je unapređen 2001 godine u verziji 2.0 ali ona nikada nije zaživela. Na kraju je XHTML Mobile Profile postao markap jezik korišćen u verziji WAP 2.0 protokola, a WML verzija 1.3 je zadnja sa aktivnom upotrebom. On predstavlja prvi markap jezik razvijen za WAP a kasnije su ga zamenili XHTML MP i HTML.

WML jezik je sličan HTML (HyperText Markup Language) jeziku za predstavljanje WEB strana. WML dokumenti su XML dokumenti koji su validni po WML DTD (Document type Definition) definiciji dokumenta. Kao i HTML i WML definiše elemente za navigaciju, unos podataka, hiperlinkove, prezentaciju slika i teksta i forme za unos podataka (slika 2). WML dokument se naziva dek (deck) i sastavljen je iz više kartica koje će korisnici videti kao strane. Svaka kartica predstavlja jedan entitet interakcije sa korisnikom. WML podržava mali skup proceduralnih elemenata koji se koriste za navigaciju između kartica.



Slika 2. Izgled WML dokumenta

Osnovni princip dizajna WML je optimizacija za razvoj strana prilagođenih ograničenjima koje imaju internet pregledači na mobilnim uređajima. Zbog toga on definiše mali skup elemenata koji služe za formatiranje sadržaja strane. Sledeća tabela sadrži sve elemente definisane WML jezikom raspoređene po grupama sa istom namenom.

Elementi deka i kartice	wml card template head access meta
<b>Definisanje događaja</b>	do ontimer onenterforward onenterbackward onpick onevent postfield
<b>Definisanje zadataka</b>	go prev refresh noop
<b>Definisanje promenljive</b>	setvar
<b>Unos podataka od korisnika</b>	input select option optgroup fieldset
<b>Sidra, slike i tajmeri</b>	a anchor img timer
<b>Formatiranje teksta</b>	br p table tr td

Svaki element se u dokumentu navodi se korišćenjem sledeće sintakse:

```
<element> vrednost </element>
```

Jednostavan primer WML dokumenta koji sadrži samo tekst je dat u sledećem odsečku koda:

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >

<wml>

  <card id="main" title="First Card">

    <p mode="wrap">This is a sample WML page.</p>

  </card>

</wml>
```

Kako je razvoj mobilnih uređaja tekao dalje i kako se sve više povećavala procesorska moć telefona oni su počeli da podržavaju moćnije jezike za formatiranje i prezentaciju sadržaja web strana kao što su XHTML i HTML.

XHTML je proširenje HTML jezika koje zahteva striktnu proveru sintakse XMLa, a XHTML MP predstavlja njegov podskup koji je optimizovan za ograničenja koja imanju internet pregledači na mobilnim telefonima i drugim prenosivim uređajima sa ograničenim hardverskim resursima. XHTML standard je kreiran od strane konzorcijuma za web W3C koji je nezavisan od Open Mobile Alijanse. XHTML MP je jezik koji uvodi moć CSS-a na mobilni web što ga čini moćnijim i sa boljom podrškom za dizajn u odnosu na svog vremešnog rivala WML jezik.

XHTML MP je izveden iz XHTML Basic 1.0 specifikacije dodavanjem XHTML Modula. Ipak XHTML MP ne pruža kompletnu implementaciju za sve module pa mobilni pregledači različito interpretiraju te nestandardne module. Aktuelna verzija XHTML MP 1.2 DTD standarda je objavljena marta 2008 godine. [5]

U sledećoj tabeli su predstavljeni elementi XHTML Basic 1.0 DTD definicije koji su uključeni u XHTML MP jezik:

Modul	Elementi
<b>Struktura</b>	body, head, html, title
<b>Tekst</b>	abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
<b>Hiperteks</b>	a
<b>List</b>	dl, dt, dd, ol, ul, li
<b>Osnovne forme</b>	form, input, label, select, option, textarea
<b>Osnovne tabele</b>	caption, table, td, th, tr
<b>Slike</b>	img
<b>Objekat</b>	object, param
<b>Meta informacije</b>	meta
<b>Link</b>	link
<b>Baza</b>	base

Sledeća tabela sadrži opis dodatnih XHTML Modula sa elementima

Modul	Elementi / Atributi
<b>Forme (delimično)</b>	fieldset, optgroup
<b>Legacy (delimično)</b>	start atribut on ol, value atribut on li
<b>Prezentacija (delimično)</b>	b, big, hr, i, small
<b>Stilovi</b>	style element
<b>Atributi stila</b>	style atribut

Danas, pametni telefoni i drugi moderni mobilni uređaji u potpunosti podržavaju HTML koji postaje standard za razvoj mobilnog web-a drugi tipovi oznaka kao WAP ili XHTML MP prestaju da se koriste.

HTML i drugi jezici koje koriste moderni mobilni uređaju biće detaljnije opisani u narednom odeljku koje je u potpunosti posvećeno njima i koje čini centralni deo diplomskog rada.

Kada web sadržaj napisan XHTML ili HTML jezikom treba da se prikaže na različitim tipovima mobilnih uređaja javlja se dosta problema. Na primer različiti uređaji imaju različite veličine ekrana pa sadržaj ne izgleda isto na svakom od njih. Neki uređaji imaju mogućnost prikazivanja CSS stilova dok drugi nemaju. Da bi se web sadržaj optimizovao za različite mobilne uređaje treba razviti više verzija aplikacije i na uređaje slati onu verziju koja odgovara mogućnostima uređaja. Jasno je da ovakva fragmentacija zahteva veliku kompleksnost mobilnih web aplikacija imajući u vidu broj različitih uređaja koji postoje na tržištu sa različitim karakteristikama kao što su veličina ekrana, broj boja, funkcijski tasteri telefona, veličina interne memorije i procesorska moć.

### 2.3. Prilagođenje web strana karakteristikama mobilnog telefona

U poređenju sa dizajnom web strana za personalne računare sa ekranima visoke rezolucije, dizajn mobilnih web strana ima jako puno ograničenja i specifičnosti. Na prvom mestu tu je veliki broj različitih veličina ekrana koji mobilni uređaji imaju kao i veliki broj različitih tipova tastatura i metoda unosa.

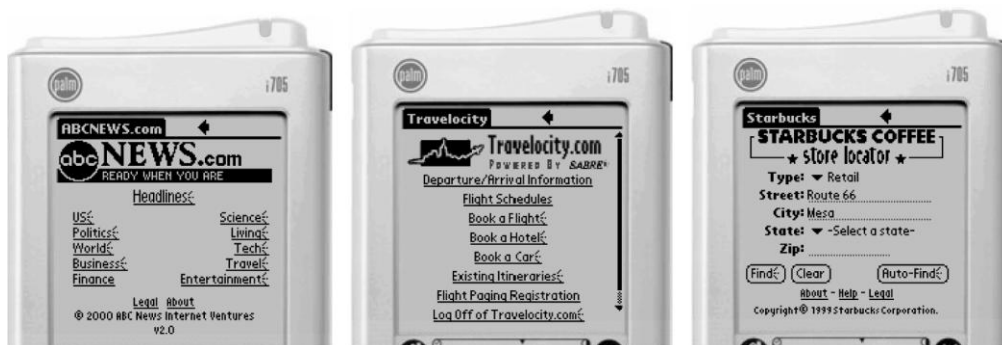
Mobilni web predstavlja optimizaciju weba sa personalnih računara namenjenju mobilnim uređajima i on uvodi nove MIME tipove, nove jezike oznaka, formate dokumenata i nove principe i preporuke za optimizaciju sadržaja za male ekrane i ograničene hardverske resurse pregledača mobilnih uređaja. U osnovi ne postoji podela interneta na desktop ili mobilni web već je to jedinstven skup sadržaja na internetu sa standardizovanim jezicima oznaka (markap jezicima), stilovima, skriptama i multimedijalnim sadržajima.

Sljedeća tabela opisuje razliku između weba na mobilnim uređajima i na personalnim računarima [9].

	<b>Mobilni Web</b>	<b>Desktop Web</b>
<b>Prosečna dužina sesije</b>	2 – 3 minuta	10 – 15 minuta
<b>Minimalna veličina ekrana</b>	90 x 60	800 x 600
<b>Maksimalna veličina ekrana</b>	240 x 400 za popularne uređaje	Neograničena
<b>Proizvođači pregledača</b>	12+ i raste	Dva sa tržišnim udelom preko 5%
<b>Bagovi pregledača</b>	Česti, Bez popravke, osim kod pametnih telefona sa nadogradnjom Osa	Retki i popravljivi zakrpama
<b>W3C standardi</b>	Nerazvijeni Ponekad ignorisani od strane mobilne industrije	Razvijeni i prihvaćeni
<b>Jezici oznaka</b>	WML CHTML XHTML Basic XHTML-MP XHTML HTML	XHTML HTML

<b>Javascript i Ajax</b>	Nema podršku na 90% mobilnih uređaka	Najčešće podržani
<b>Broj korisnika</b>	3 milijarde mobilnih korisnika širom sveta	1 milijarda laptopova, desktop računara i servera

Optimizacija web sadržaja za mobilne telefone zavisi od karakteristika mobilnog uređaja. Različite generacije uređaja imaju drugačije karakteristike i prezentacione sposobnosti. Stariji mobilni uređaji imaju monohromatske ekrane niske rezolucije i tastaturu sa jednim navigacionim tasterom (slika 3).



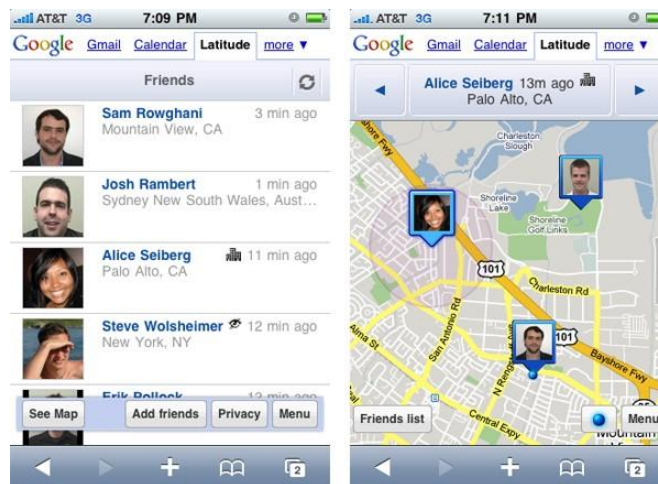
Slika 3. Izgled web strane na monohromatskom ekranu niske rezolucije

Nešto novije generacije telefona imaju ekrane u boji sa većom rezolucijom (slika 4). Tastatura ovih uređaja je najčešće istog tipa kao i kod ranijih generacija telefona mada se pojavljuju i uređaji sa QWERTY tastaturom.



Slika 4. Izgled web aplikacije na ekranu u boji veće rezolucije

Konačno najnovije generacije pametnih telefona imaju ekrane visoke rezolucije, osjetljive na dodir. Ovi telefoni imaju ugrađene senzore za detekciju položaja uređaja, GPS senzor pa pružaju mogućnost uvođenja nove generacije mobilnih web aplikacija sa bogatim sadržajem koji se može menjati u zavisnosti od konteksta uređaja (slika 5).



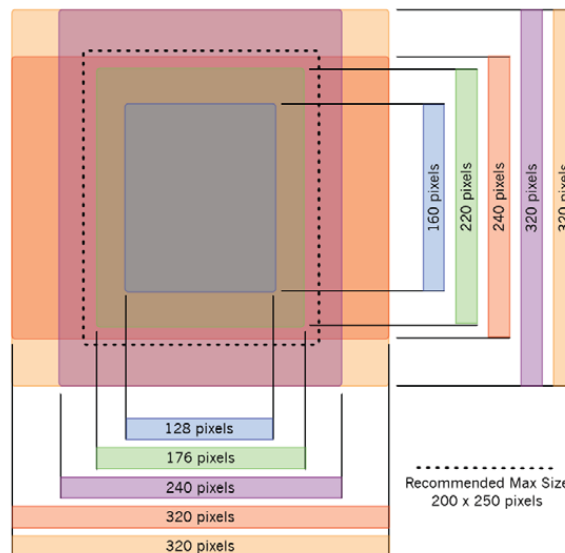
Slika 5. Izgled web aplikacije na pametnom telefonu

Da bi se razvila mobilna web aplikacija koja izgleda dobro na različitim uređajima mora se voditi računa o prilagođenju aplikacije i web sadržaja različitim karakteristikama telefona i različitim veličinama ekrana mobilnih uređaja što često uvodi potrebu za razvojem više verzija aplikacija i prikazivanju određene verzije na mobilnom uređaju.

Da bi se servirala odgovarajuća verzija mobilnog sajta potrebno je prepoznati mobilni uređaj i verziju internet pregledača i odrediti njegov skup karakteristika. To omogućava mobilnom web sajtu da prilagodi stil oznaka, skripte i izgled strane da bi se omogućilo najbolje moguće korisničko iskustvo. U svrhu prepoznavanja karakteristika mobilnog uređaja koriste se gotove biblioteke karakteristika mobilnih telefona od kojih su najpoznatije WURFL [10] i Device Atlas [11].

Odvajanjem različitih rezolucija ekrana u različite klase uređaja može se suziti broj različitih rezolucija o kojima treba voditi računa i tako uveliko smanjiti kompleksnost dizajna strane (slika 6). Takođe treba imati u vidu da je širina ekrana ta koja diktira koliko će upotrebljiv ili lep dizajn biti. Na primer slike koje izgledaju dobro na telefonima niže klase sa niskom rezolucijom ekrana, mogu da zauzimaju samo 1/3 širine ekrana visoke rezolucije pa mogu biti teško vidljive bez zumiranja.

Često se prilikom donošenja odluka o dizajnu analizira korisnička grupa i uređaji koji se pretežno koriste pa se na osnovu toga i određuje koje veličine ekrana treba optimizovati sadržaj prilikom izrade dizajna.



Slika 6. Rezolucije ekrana različitih tipova mobilnih uređaja

Danas najpopularnije mobilne web stranice kao što su na primer Facebook ili Twitter često imaju više verzija mobilnog dizajna (slika 7). Društvena mreža Facebook ima dve verzije mobilnog sajta, jednu prilagođenu starijim generacijama mobilnih telefona, i drugu prilagođenu pametnim telefonima i drugim prenosivim uređajima sa ekranima osjetljivim na dodir kao što su tablet računari.

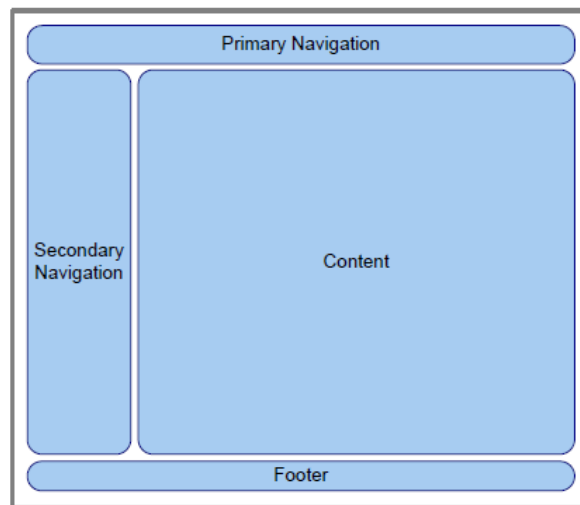


Slika 7. Izgled Facebook strane prilagođen različitim veličinama ekrana

Kod web strana razvijanih za desktop računare postoje ustaljeni dizajn obrasci i navigacione šeme među kojima je navigacija podeljena po karticama ili meni koji se nalazi sa strane glavnog sadržaja strane (slika 8). Ovakve šeme nam daju korisne vizuelne smernice gde se nalazimo trenutno na sajtu i pružaju nam referentnu tačku kako da navigiramo unutar sajta. Jasno je da je ovo teže postići u mobilnom kontekstu zbog ograničenja veličine ekrana uređaja i ograničenih navigacionih karakteristika mobilnih uređaja. Iako je izvodivo na mobilnom webu implementirati stil navigacije



svojtven desktop webu kao što su kartice, one nisu dobro rešenje zbog veličine ekrana i mogućnosti pozicioniranja fokusa naročito kod starijih uređaja. Pametni telefoni imaju daleko manja ograničenja pa je dizajn navigacije za mobilne sajtove razvijane za njih jednostavniji.



Slika 8. Raspored elemenata web strane namenjen prikazu na personalnim računarima

Preporučeni i češće korišćeni metod kreiranja navigacione šeme na mobilnim uređajima je korišćenjem proste vertikalne liste opcija, često je svakoj dodeljen redni broj u listi (0-9) koja se može brzo aktivirati pritiskom tastera na tastaturi telefona sa željenim brojem.



Slika 9. Raspored elemenata web strane namenjen prikazu na mobilnim uređajima

Prikazivanje više nivoa navigacije unutar liste ne funkcioniše dobro jer veliki broj opcija zauzima vredan prostor ekrana. Bolji način je prikazati samo jedan nivo navigacije koji se odnosi na stranu koja se trenutno pregleda. Ipak savet je da treba uključiti opcije za brzu navigaciju kroz sadržaj kao što je link ka sledećoj sekciji, ka sekciji roditelja i ka početnoj strani mobilnog web sajta. Ove linkove treba staviti na dnu strane tako da nema potrebe skrolovati ekran nakon čitanja sadržaja na vrh da bi se promenila strana (slika 9).

Ne samo da se desktop web oslanja na opisane navigacione šeme koje se danas podrazumevaju već se tekoće web strane dizajniraju za horizontalnu orijentaciju, gde su strane šire nego što su više. Mobilni dizajn zahteva promenu orijentacije dizajna u vertikalni format gde su strane više puta duže nego što su šire pa tako sadržaj i treba optimizovati.

Zbog toga je lakši deo razvoja mobilnog web sajta promena markap jezika desktop verzije sajta (npr sa XHTML na XHTML Basic) a teži deo predstavlja razvoj podrške za niz različitih tipova uređaja. Prilikom razvoja mobilne verzije sajta treba pratiti principe i preporuke prikupljene i objavljene od strane W3 konzorcijuma i dotMobi radne grupe da bi mobilni web sajt bio dobro optimizovan za rad na različitim uređajima.

## 2.4.Principi i preporuke za razvoj mobilnog weba

Zajednica programera koji razvijaju mobilne web sajtove, radna grupa W3 konzorcijuma i dotMobi su sastavili dokument u kome su opisane dobre prakse i preporuke za razvoj mobilnih web stranica. Cilj ovih preporuka je da obezbede najbolje moguć korisnički doživljaj za korisnike mobilnih telefona. Preporuke za razvoj mobilnih web sajtova date su u sledećim tačkama.

### Enkodiranje sadržaja

Ispravno enkodiranje karaktera koji se prikazuju na strani je bitno ukoliko želimo da se strana prikaže ispravno na različitim tipovima uređaja. Ukoliko se eksplicitno ne postavi odgovarajući encoding znakova na strani, može se desiti da se prilikom prikaza na mobilnom uređaju pojave čudni karakteri koji ne odgovaraju originalu. Čest slučaj je pogrešno prikazivanje ćiriličnih slova. Zbog toga treba eksplicitno postaviti encoding strane korišćenjem HTTP zaglavlja i XML zaglavlja:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

### XHTML MP 1.0 Doctype

Korišćenjem pogrešnog tipa oznaka, koji se specificira pomoću doctype zaglavlja, može dovesti da strana na mobilnom uređaju prikaže netačno. Zbog toga treba koristiti XHTML MP 1.0 kao podrazumevani jezik oznaka na mobilnim veb stranama. Ovo se postiže dodavanjem sledećeg doctype zaglavlja u kod mobilne strane:

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

### MIME tipovi

Mime tipovi koje postavlja server u okviru HTTP zaglavlja odgovora pružaju važnu informaciju mobilnom pregledaču kako da tretira sadržaj dokumenta. Ukoliko se uz sadržaj dokumenta pošalje pogrešan MIME tip može se desiti da pregledač pogrešno interpretira sadržaj dokumenta i ne uspe da ga prikaže. Za XHTML-MP i Basic sadržaje preporučeni MIME tip je `application/xhtml+xml`.

### **Naslovi strana**

Naslovi strana okruženi `<title>` oznakama su važni ali često zanemarivani elementi. Dobar naslov povećava mogućnost otkrivanja i upotrebnu vrednost web strana. Kratak i slikovit naslov strane je ključan za laku prepoznatljivost. Treba voditi računa da prilikom prikaza mobilni uređaji mogu odseći deo naslova radi boljeg prikaza.

### **Korišćenje stilova**

Treba koristiti CSS za kontrolisanje prezentacije sadržaja na strani. CSS stilovi pomažu u očuvanju konzistentnosti i centralizovanje upravljanja stilovima i snižavanje ukupne veličine web strana.

### **Objekti i skripte**

Iako veliki broj modernih mobilnih pregledača ima podršku za pogretanje skriptova, one mogu biti onemogućene podešavanjima korisnika pa treba voditi računa da strana funkcioniše i bez njih. Ukoliko je neophodno korišćenje ugrađenih objekata i skripti treba primeniti mehanizme detektovanja podešavanja mobilnog pregledača i prilagođenje sadržaja strane dostupnim funkcionalnostima, tako da uređaji sa boljom podrškom za skripte budu prosleđeni na strane koje su specijalno pravljene za te uređaje.

### **Automatsko osvežavanje**

Automatsko osvežavanje strane zahteva dodatan protok podataka pa i dodatne troškove. Zato je dobro izbegavati periodično samo-osvežavanje sadržaja strana ili obavestiti korisnika i pružiti mu mogućnost da onemogući osvežavanje ili prilagodi njegovu učestalost.

### **Prosleđivanje (Redirekcija)**

Korišćenje oznaka za redirekciju strana povećava vreme učitavanja zbog skidanja i procesiranja sadržaja nove strane. Ukoliko je neophodno koristiti redirekciju treba konfigurisati server tako da za to koristi HTTP 3xx status kodove. Treba izbegavati redirekciju što je više moguće jer povećavaju vreme i cenu koju korisnik plaća da bi mu se strana prikazala.

### **Keširanje**

Korišćenje keširanih informacija ponekad smanjuje potrebu za ponovnim učitavanjem resursa kao što su slike i stilovi, pa stoga smanjuje vreme skidanja i cenu sadržaja. Postavljanjem keš informacija u mobilne strane smanjuje se broj puta koje uređaji preuzimaju resurse strana koji se često koriste, što je naročito bitno za logo ili stilove strana. Sledeći primer pokazuje korišćenje meta direktive za postavljanje zaglavlja za kontrolu keširanja sadržaja:

`<meta http-equiv="Cache-Control" content="max-age=300"/>`

### **Struktura strane**

Dobra je praksa da se u dokumentima koristi struktura sa naslovima i podnaslovima. To podrazumeva predstavljanje redom i sa semantičkom tačnošću da bi se obezbedilo da se elementi

strane javljaju redom koji ima smisla bez potrebe za dodanom manipulacijom prezentacije. Sledeći primer pokazuje dobro semantičko kodiranje strana:

```
<h1>Top Level Heading</h1>
<h2>Second Level Heading</h2>
<p>Paragraph Body</p>
<h3>Third Level Heading</h3>
<p>Paragraph Body</p>
<h2>Second Level Heading</h2>
<p>Paragraph Body</p>
<h3>Third Level Heading</h3>
<p>Paragraph Body</p>
<h4>Fourth Level Heading</h4>
<p>Paragraph Body</p>
```

### **Tabele**

Tabele se često ne prikazuju dobro na ekranima malih veličina. Zbog toga je bitno izbegavati tabele osim ukoliko znamo da uređaj na kome se strana prikazuje ima podršku za njih. Manje tabele sa dve ili tri kolone funkcionišu dobro na većini uređaja ali to nije preporučeni pristup. Umesto toga savet je koristiti <dl> elemente listi za prikaz sadržaja.

### **Ugnježdene tabele**

Ugnježdene tabele (tabele unitar tabela) kao i tabele kojima se pokušava pozicioniranje sadržaja na web strani ne funkcionišu dobro u mobilnom dizajnu naročito jer se prikazuju veoma loše i povećavaju

### **Tabele i pozicioniranje elemenata na strani**

Tabele kojima se pokušava pozicioniranje sadržaja na web strani ne funkcionišu dobro kod mobilnog dizajna zbog nekonzistentnosti njihovog prikazivanja na različitim mobilnim uređajima. Efikasnije je elemente rasporediti korišćenjem stilova tako da oni izgledaju dobro na ekranima male širine karakterističnih za mobilne telefone.

### **Frejmovi**

Frejmovi ne funkcionišu dobro kod mobilnog dizajna jer ih većina uređaja ne podržava. Zbog toga ih treba izbegavati.

### **Broj linkova po strani**

Veliki broj linkova na strani otežava navigaciju korisniku i čitanje sadržaja. Veliki broj mobilnih pregledača zaustavlja vertikalno skrolovanje kada se naiđe na link što otežava brz pregled sadržaja. Zbog toga se treba ograničiti na maksimalno 10 linkova po strani uz korišćenje prečica na numeričkoj tastaturi kada god je to moguće tako da korisnik može brzo da ode na željeni link korišćenjem tastature.

### **Prečice za navigaciju**

Navigacija na mobilnom sajtu se može lako pretvoriti u tešku i dosadnu operaciju. Povezivanje accesskey atributa uz svaki link na strani daje korisniku lak način navigacije korišćenjem tastature

mobilnog uređaja. Naročito su korisne ukoliko se konzistentno koriste kroz ceo sajt. Povezivanje prečica tastature radi se na sledeći način:

```
<li><a href="link.html" accesskey="1">Link 1</a></li>
```

### **Izbegavati polja za unos teksta**

Korisnicima je teško da unose sadržaj u tekstualna polja korišćenjem tastature. Iako su ona ponekad neophodna treba voditi računa da se koriste što je ređe moguće a ona se mogu zameniti radio dugmadima, padajućim listama i listama linkova.

### **Podešavanje unosa teksta**

Postoji mogućnost ograničavanja karaktera koji se mogu uneti u polje korišćenjem tastature definisanjem maske unosa ili CSS stila. Ovo olakšava korisniku da unese tačnu informaciju u polje za unos. Sledeći primer ograničava unos samo na numeričke vrednosti:

```
<input type="text" style=' -wap-input-format: "*N"' />
```

Sledeći primer ograničava unos samo alfanumeričkih znakova uz obavezno prvo veliko slovo.

```
<input type="text" style=' -wap-input-format: "A*a"' />
```

### **Veličina slike**

Optimizovati slike tako da budu što je manje moguće po broju piksela osim ako znamo da uređaj podržava veće slike. Većina mobilnih ekrana je širine 120 piksela a slike treba da budu uže od njegove širine da bi se izbeglo horizontalno skorlovanje i smanjila ukupna veličina strane koja je bitan faktor kada se podaci učitavaju preko sporih mobilnih mreža.

### **EksPLICITNO definisanje veličine slike**

Ukoliko slika nema eksPLICITNO definisanu visinu i širinu u pikselima mobilni uređaj mora da je računa što smanjuje brzinu prikaza strane. Ukoliko server smanjuje veličinu slike i optimizuje je za prikazivanje na uređaju to smanjuje prenos podataka i ukupno vreme potrebno da uređaj procesira i skalira sliku na željenu veličinu. Ukoliko se originalna veličina i veličina definisana na strani podudarne mobilni pregledač nema potrebe da je skalira pre prikazivanja.

### **Alt tekst**

Preuzimanje slika produžava vreme potrebno za učitavanje web strana. Kreiranje strana koje su čitljive bez slika daje mogućnost korisnicima da ih pregledaju kada je mobilni pregledač konfigurisan da prikazuje samo tekst. Kao rezultat se dobija smanjeno vreme preuzimanja sadržaja i cena. Ukoliko je korisnik omogućio prikazivanje slika njihov tekstualni opis se prikazuje do trenutka dok se slika ne preuzme sa servera na mobilni uređaj. To poboljšava upotrebnu vrednost strana tokom dugih vremena učitavanja.

### **Validna sitnaks**

Strane koje nemaju validnu sintaksu se ne mogu prikazati korektno na mobilnim uređajima. U pojedinim slučajevima, naročito na starijim telefonima, nevalidan XHTML-MP dokument se neće ni prikazati. Zbog toga je bitno korišćenje validne XHTML Basic ili MP sintakse na mobilnim web stranama radi povećanja efikasnosti prikaza.

### **Iskaćuci prozori**

Većina mobilnih uređaja ne podržava iskačuće prozore. Čak i kada ih mobilni uređaj podržava treba ih izbegavati jer promena trenutnog prozora zbunjuje korisnika.

#### **Učitavanje eksternih resursa**

Mobilni uređaj mora da odvojeno preuzme svaki eksterni resurs naveden u strani kao što su slike stilovi ili drugi objekti, što povećava vreme učitavanja i cenu prikazivanja stran. Zato treba pažljivo razmotriti broj eksternih resursa koji se koriste i ogranočiti veličinu svakog od njih tako da bude što je manji moguć uz očuvanje upotrebne vrednosti na različitim mobilnim uređajima.

#### **Ukupna veličina strane**

Velike strane zahtevaju dugo vreme učitavanja i povećavaju cenu prenesenih podataka. Zato treba voditi računa da strane, stilovi, slike i sve drugi eksterni resursi budu što manji. Dobra praksa je da mobile strane budu manje od 10kb, dok veće strane ne treba premašuju 25kb. Ove pravilo ipak nije uslovljavajuće pa se mogu praviti izuzetci u pojedinim slučajevima kada je to neophodno.

## **2.5.Mobilni web za pametne telefone**

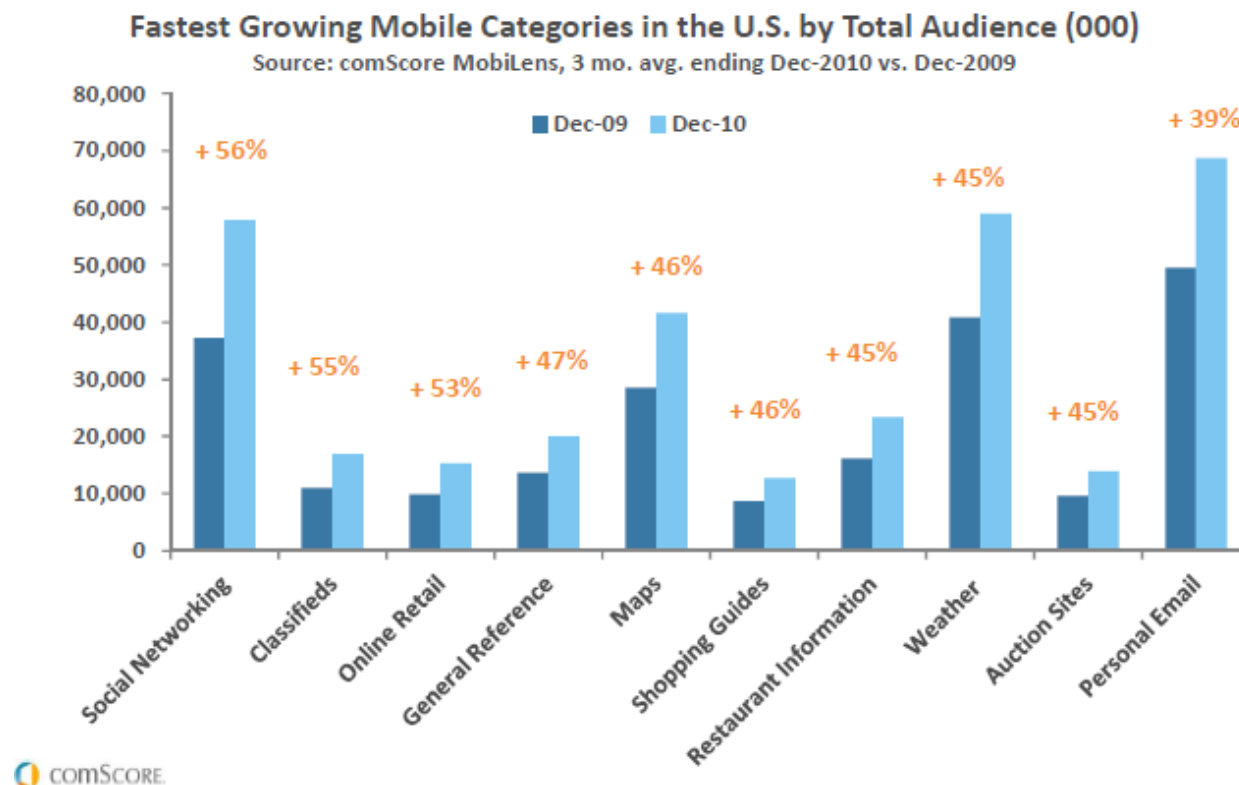
Prijem mobilnog Web pristupa kod korisnika se ubrzava od 2007 godine sa razvojem pametnih telefona (iPhone, Nokia N95, Android uređaji) i 2010 sa pojavljivanjem tablet računara sa ekranom osetljivim na dodir. Obe platforme nude bolji doživljaj mobilnog interneta (iz mobilnih aplikacija ili web pregledača) u odnosu na ranije generacije mobilnih uređaja što je doprinelo njihovoj popularnosti.

Prema izveštaju istraživačke kuće Gartner iz novembra 2010 godine, prodaja pametnih telefona na svetskom nivou se udvostručila u odnosu na 2009 godinu a njihov ukupan udeo u odnosu na prodaju svih mobilnih uređaja iznosi 19.3 procenata. [6] Prodaja pametnih telefona se u 2010 godini povećala za 72 procenta u odnosu na predhodnu godinu, dok se prodaja ostalih mobilnih telefona povećala svega 31.8 procenata.

Tokom 2010 godine dolazi do značajnog unapređenja tehnologije i pametni telefoni dobijaju niz novih mogućnosti kao što su veći ekrani, mogućnost snimanja videa visoke rezolucije, hardverske i softverske QUERTY tastature, ekrani osetljivi na dodir i slično. Sva ta tehnološka dostignuća unose poboljšanja u korisničko iskustvo i poboljšanja prezentacije i konzumiranja sadržaja na mobilnim uređajima pa dolazi do naglog povećanja trenda zamene starih modela mobilnih uređaja pametnim telefonima.

Jedan od najvažnijih trendova je prodor uređaja koji imaju punu podršku za pregledanje weba i podršku za HTML i napuštanje zastarelog WAP standarda. Punu podršku za pregledanje HTML sadržaja dobija 48 procenata telefona u Americi dok u Evropi (pet najrazvijenijih zemalja Francuska, Engleska, Nemačka, Španija i Italija) dobija 61 procenata telefona. Ako na to dodamo i procenat starijih telefona koji imaju ograničenu podršku za WEB sadržaj (kao što su WAP telefoni) dobijamo više od 90 procenata mobilnih pretplatnika u Americi i Evropi koji mogu da pregledaju neki vid web sadržaja na mobilnom telefonu.

Prema istraživanju comScore [7] mobilni sadržaj sa najvećim upotrebom u 2010 godini su socijalne mreže, čitanje elektronske pošte, vremenska prognoza i navigacija. Ovi a i ostali tipovi sadržaja doživljavaju rast u upotrebi u odnosu na 2009 godinu za skoro 50 procenata što je predstavljeno na slici 10.



Slika 10. Najbrže rastuće mobilne kategorije u SAD

Danas na pametnim telefonima i tablet računarima mobilni internet koristi većinu pogodnosti na koje smo navikli na personalnim računarima. Korisnici pametnih telefona često imaju mobilne pretplate sa neograničenim protokom podataka pa ih ne zanima optimizacija sadržaja već koriste bogat multimedijalni sadržaj kao što su slike i video u velikoj meri.

Zato mobilni web početkom ere pametnih telefona i tableta doživljava renesansu u pogledu razvoja sadržaja. Sve manje se vodi računa o optimizaciji sadržaja, a stavlja se akcenat na poboljšanje korisničkog iskustva prilikom upotrebe aplikacija. Ipak velika fragmentacija mobilnih web pregledača napravljena od strane više desetina proizvođača mobilnih uređaja povećava komplikovanost razvoja sadržaja za mobilni internet.

Tako mobilne aplikacije namenjene pametnim telefonima imaju korisnički interfejs prilagođen ekranu osjetljivom na dodir i promeni orijentacije ekrana. On više nije optimizovan samo za korišćenje u uspravnoj orijentaciji ekrana nego mora biti napravljen da podrži i horizontalnu orijentaciju. Korisnički interfejs je lak za upotrebu, sa dobrim odzivom i često je animiran. U tom smislu on je bliži web sajtovima namenjenim personalnim računarima nego mobilnim sajtovima razvijanim za ranije generacije mobilnih uređaja. U prilog tvrdnji ide i činjenica da pametni telefoni poseduju memoriju i procesorsku moć ekvivalentnu personalnim računarima pre desetak godina, pa mogu sa lakoćom da se izbore i sa prikazivanjem složenijih i većih web stranica, koje uključuju i

strane dizajnirane za prenosive računare. Ipak web sajтови namenjeni pametnim telefonima i tabletima imaju posebnu vrstu web dizajna koja je posebno prilagođena ekranima osetljivim na dodir i promeni orijentacije ekrana.

Pojavljuje se mogućnost za razvoj pametnih mobilnih web aplikacija sa promenljivim sadržajem u zavisnosti od konteksta korisnika. Tako korišćenjem geolokacijskih servisa i GPS senzora na telefonu je moguće servirati informacije na mobilnoj web aplikaciji koji su prilagođene trenutnoj lokaciji korisnika. Na primer, korisnik može dobiti informacije o restoranima koji se nalaze u neposrednoj blizini njegove trenutne lokacije kao i navigaciona uputstva kako da dođe do same lokacije restorana. Ovo nije jedini tip kontekstno zasnovanih servisa što ćemo videti u narednom poglavlju kada će biti detaljnije opisan svaki od njih.

Često se mobilni web namenjen pametnim telefonima naziva i Mobile Web 2.0, mada za ovaj pojam ne postoji nijedna definicija. Mobilni Web 2.0 je nastao 2007 godine sa pojavom prvih pametnih telefona na tržištu koji su uneli velike promene za mobilni web: podrška za WiFi, 3G, podrška za Ajax i Flash, striming videa. Mobilni web 2.0 sajтови često imaju mnoge od sledećih karakteristika:

- Ajax i bogat korisnički interfejs
- Geolokacija
- Offline pristup podacima i lokalni keš
- Aktivnosti socijalnog umrežavanja
- Striming videa na zahtev ili uživo
- Kontekstualne reklame
- HTML 4/5, CSS 2/3, Javascript
- Podrška za ekrane osetljive dodir

Mnoge od ovih karakteristika ću detaljno objasniti u narednom poglavlju, a kroz praktični deo diplomskog rada biće demonstrirana njihova upotreba u izradi mobilne web aplikacije novije generacije optimizovane za rad na pametnim telefonima i tablet računarima.



### 3. Mobilne web tehnologije i standardi

#### 3.1.HTML 5

Internet pregledačima na personalnim računarima i pametnim telefonima dugo vremena fale napredne funkcionalnosti koje bi ih približile nativnim aplikacijama koje imaju mogućnost da koriste hardverske resurse i funkcije operativnog sistema uređaja kao što su 3D grafika, lokalna memorija uređaja ili geolocijske funkcije GPS senzora. Ta dugo željena nadgradnja weba pojavljuje se kroz uvođenje HTML 5 standarda koji definiše novi jezik za formatiranje sadržaja web strana i skup novih funkcionalnosti koje će biti implementirane u internet pregledačima i stojati na raspolaganju programerima u vidu APIa.

HTML 5 predstavlja novu generaciju HTML standarda koja će zauzeti mesto HTML 4.01, XHTML 1.0 i XHTML 1.1 standarda. HTML 5 uvodi nove funkcionalnosti potrebne modernim web aplikacijama i standardizuje mnoge postojeće koje su bile u upotrebi godinama na web platformama, ali nikada nisu ušle u sastav ni jednog standarda. Nove funkcionalnosti imaju za cilj da zamene korišćenje dodataka kao što su Flash ili Micorsoft Silverlight ili komplekse skripte i da se mesto njih koriste ugrađene funkcionalnosti pregledača. U tom pogledu novi standard ima daleko bolju podršku za razvoj dinamičnih aplikacija nego prethodne verzije HTMLa koje su originalno bile predviđene za kreiranje statičkog sadržaja.

Kao nadgradnja ranijih standarda HTML 5 uglavno podržava postojeće elemente HTMLa. Nove oznake se koriste na isti način kao i stare ali je semantika elemenata donekle promenjena. U poređenju sa HTML4 standardom HTML5 sintaksa uvodi dosta uprošćenja, a najbolji primeri su novi format DOCTYPE Character Set zaglavlja:

	HTML4	HTML5
<b>DOCTYPE</b>	<code>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"&gt;</code>	<code>&lt;!DOCTYPE html&gt;</code>
<b>CHARSET</b>	<code>&lt;meta http-equiv="Content-Type" content="text/html; charset=utf-8"&gt;</code>	<code>&lt;meta charset="utf-8"&gt;</code>

HTML4 sadrži veliki broj semantičkih elemenata koji označavaju različite delove web stranice kao što su forme, liste paragrafi, tabele itd. HTML5 uvodi i nove semantičke elemente koji služe za označavanje delova strane sa različitom namenom kao što je navigacija, zaglavlje ili dno strane i slično. Uvedeni semantički elementi su:

Element	Namena
<code>&lt;header&gt;</code>	Zaglavlje strane
<code>&lt;footer&gt;</code>	Dno strane
<code>&lt;nav&gt;</code>	Navigaciona funkcionalnost strane
<code>&lt;article&gt;</code>	Deo sadržaja strane koji bi imao smisla kada bi bio izdvojen kao pojedinačna vest
<code>&lt;section&gt;</code>	Grupisanje članaka po temama ili podela članka u različite sekcije
<code>&lt;time&gt;</code>	Prikazivanje datuma i vremena
<code>&lt;aside&gt;</code>	Blok sadržaja koji se prikazuje sa strane glavnog sadržaja
<code>&lt;hgroup&gt;</code>	Grupisanje više zaglavlja u jedno

**<figure> and <figcaption>** Označavanje slike i njenog opisa kao jedinstven element

U HTML4 ovi elementi su najčešće predstavljani korišćenjem div elemenata sa različitim vrednostima id-a i class atributa. Iako bi stranica vizuelno izgledala isto semantička poboljšanja koje uvode novi HTML5 elementi olakšavaju formatiranje sadržaja web strane u strukturu koju lako mogu da prepoznaju internet pregledači pa je lakša manipulacija sadržajem strane i na primer izdvajanje sadržaja u RSS bez dodatnog programiranja (slika 11).



Slika 11. Izgled web strane formatiran HTML5 oznakama

Za promenu stila i izgleda web strane koristi se CSS3 jezik koji nije deo HTML5 specifikacije već čini poseban standard. CSS3 uvodi poboljšanja u odnosu na ranije verzije CSS-ova koja daju mogućnost implementacije lepših i funkcionalnijih korisničkih interfejsa web aplikacija koje sada dolaze korak bliže po mogućnosti prezentacije sadržaja desktop aplikacijama. Neke od karakteristika CSS standarda će biti predstavljene u delu koji se bavi grafikom i multimedijom kod HTML5 aplikacija.

Iako ne čini deo HTML5 specifikacije Javascript se podrazumeva kao njegov sastavni deo. On se koristi za programiranje dinamičkog ponašanja web strane kao i za pristup interfejsima za programiranje aplikacija koje novi standard propisuje. Skup APIa koji se trenutno nalaze u HTML5 specifikaciji može se podeliti po sledećim grupama:

- Offline i Web Storage
  - Web Storage
  - IndexedDb
  - Lokalni keš aplikacije (offline pristup)
- Grafika
  - Podrška za audio i video sadržaje
  - 2D crtanje u Canvas elementu
  - 3D crtanje WebGL
  - SVG grafika

- Pristup hardveru i funkcijama operativnog sistema
  - Drag and drop
  - Pristup fajl sistemu
  - Geolokacija
  - Detekcija promene orijentacije uređaja
  - Prepoznavanje govora i glasovni unos teksta
- Obrada podataka i komunikacija
  - Web Workers
  - Dvosmerna komunikacija sa serverom korišćenjem Web Socket-a
  - Notifikacije

HTML 5 specifikacija se trenutno nalazi u fazi razvoja od strane W3C konzorcijuma i WHATWG (Web Hypertext Application Technology Working Group ) grupe. WHATWG i W3C predviđaju da će prvo izdanje HTML 5 specifikacije biti završeno 2012 godine (konačna verzija se planira za 2020 godinu) kada može početi zvanična podrška HTML5 standarda u mobilnim i desktop internet pregledačima. Bez obzira na to veliki deo postojeće HTML 5 specifikacije je već danas podržan u zadnjim verzijama internet pregledača na personalnim računarima i pametnim telefonima kao što su iPhone, Android i Windows Phone 7 uređaji.

### 3.2.CSS3

CSS je jezik stilova i koristi se za opisivanja načina prezentacije sadržaja – izgleda i formatiranja dokumenta koji je napisan u jeziku oznaka kao što je HTML. CSS je zamišljen inicijalno da omogući razdvajanje sadržaja dokumenta od načina prezentacije sadržaja kao što su raspored elemenata, boje i font. Razdvajanje stila od sadržaja poboljšava fleksibilnost i kontrolu nad opisom prezentacije sadržaja – više strana mogu deliti isti stil prezentacije, smanjuje se kompleksnost prezentacije i stila.

CSS specifikaciju održava konzorcijum za web (W3C). Kako ona svojim opsegom prevazilazi potrebe ovog rada opisaću samo specifičnosti uvedene u verziji 3 standarda koje se tiču prikaza sadržaja na prenosivim uređajima.

Stil se na stranu može dodati na jedan od tri načina:

- Pomoću `<style>` oznake unutar XHTML ili HTML sadržaja
- Spoljnim stilom kao .css fajl
- Atributom `style` unutar oznaka

Kod mobilnih web aplikacija, koje koriste AJAX za učitavanje sadržaja, ili običnih mobilnih dokumenta preporuka je da se prezentacija navede unutar `<style>` oznaka. Time se postiže ubrzanje jer nema potrebe za slanjem dodatnih zahteva i nema kašnjenja zbog formatiranja prikaza.

Jasno je da se isti stil prezentacije ne može koristiti za sve tipove uređaja. Na primer za mobilni telefon treba razviti drugačiji stil u odnosu za računare sa velikom dijagonalom ekrana zbog specifičnih potreba za formatiranje i prikaz sadržaja. Ukoliko se ista verzija sajta prikazuje i na mobilnim i na personalnim računarima jedini način za promenu prikaza i izgleda strane je kroz CSS

fajl. CSS3 uvodi media filtere za upravljanje stilovima u zavisnosti od uređaja na kojem se strana prikazuje. U zavisnosti od vrednosti media atributa uključuju se različiti stil za prikaz strane. Tako se media atribut sa vrednošću *screen* koristi za uključivanje stila za računare visoke rezolucije a sa vrednošću *handheld* za mobilne uređaje.

```
<link rel="stylesheet" type="text/css" media="screen" href="desktop.css" />
<link rel="stylesheet" type="text/css" media="handheld" href="mobile.css" />
```

U praksi međutim neki moderni mobilni pregledači koriste *screen* verzije CSSa zato što su u mogućnosti da ih prikažu. Neki drugi koriste *screen* verziju CSSa kada misle da se radi o punoj verziji sajta ili koriste *handheld* kada misle da se radi o mobilnoj verziji sajta u zavisnosti od DOCTYPE meta oznake ili korisnikovih podešavanja.

Pregledač / platforma	Korišćen <i>media</i> atribut
<b>Safari</b>	screen
<b>Android</b>	screen
<b>Symbian/S60</b>	screen
<b>Nokia Serija 40</b>	screen
<b>webOS</b>	screen
<b>BlackBerry</b>	screen (handheld ako postoji meta)
<b>NetFort</b>	handheld
<b>Openwave</b>	handheld
<b>Internet Explorer</b>	screen
<b>Motorola Internet Browser</b>	handheld
<b>Opera Mobile</b>	screen
<b>Opera Mini</b>	screen

U predhodnoj tabeli je prikazano korišćenje media atributa u zavisnosti od platforme mobilnog uređaja – vidimo da se nije moguće oslanjati samo na *media="handheld"* atribut.

CSS3 uvodi media upite kojima se može prevazići pomenuti problem. Ove kompleksne definicije uvode uslove o veličini ekrana i vrednosti media atributa koji im odgovaraju. Na primer moguće je definisati upit za primenu posebnog stila na uređajima koji podržavaju jedino *screen* i imaju maksimalnu širinu od 480px. Ova definicija bi odgovarala iPhone uređajima. Ovaj media upit se piše kao:

```
<link type="text/css" rel="stylesheet" media="only screen and (max-device-width:480px)"
href="iphone.css" />
```

Na primer mogu se ciljati uređaji veće rezolucije koje nisu iPhone uzimajući za minimalnu širinu uređaja 481px.

```
<link media="screen and (min-device-width: 481px)" href="notiphone.css" type="text/css"
rel="stylesheet" />
```

Neki pregledači razumeju media upite i unutar samog CSS fajla. Na primer sledeći kod menja boju pozadine strane kada se prikazuje na iPhone uređaju.

```
@media only screen and (max-device-width: 480px) {
body {
background-color: red;
}
}
```

Ekstenzija media upita za ciljanje orijentacije uređaja *orientation* dozvoljava definisanje različitih stilova za portret i landscape orijentaciju telefona:

```
<link rel="stylesheet" media="all and (orientation:landscape)" href="land.css" />
<link rel="stylesheet" media="all and (orientation:portrait)" href="port.css" />
```

Sljedeća tabela prikazuje listu kompatibilnosti mobilnih uređaja sa CSS media upitima:

Pregledač / platforma	Uslovni media upiti	Upiti za orijentaciju
<b>Safari</b>	Da	Da (od OS 3.2)
<b>Android</b>	Da	Da (od 2.0)
<b>Symbian/S50</b>	Da od verzije 5	Ne
<b>Nokia Serija 40</b>	Da posle edicije 6	Ne
<b>webOS</b>	Da	Ne
<b>BlackBerry</b>	Ne	Ne
<b>NetFront</b>	Ne	Ne
<b>Openwave</b>	Ne	Ne
<b>Internet Explorer</b>	Ne	Ne
<b>Motorola Internet Browser</b>	Ne	Ne
<b>Opera Mobile</b>	Yes	No
<b>Opera Mini</b>	Yes	No

Veliki deo modernih mobilnih pregledača implementira WebKit (Safari, Android, webOS, Symbian). WebKit projekat je dodao mnogo ekstenzija CSS-u a neke od ovih su razmatrane da se uključe u CSS3 standard:

- -webkit-border-radius definiše zaobljene ivice elementa
- -webkit-box-shadow definiše senku elementa
- -webkit-columns definiše širinu i broj kolona
- -webkit-border-image definiše sliku koja se koristi kao ivica elementa
- -webkit-text-stroke definiše boju okvira teksta
- -webkit-text-fill-color definiše boju punjenja teksta

### 3.3.JavaScript / Ajax

Pravljenje aplikacija za Web zahteva više od sadržaja i prezentacije. Korisnici očekuju da web stranice budu interaktivne i da odgovaraju na njihove akcije. JavaScript omogućava dodavanje dinamike web stranicama koje mogu da imaju karakteristike nativnih aplikacije.

Par godina nakon izlaska standarda mobilni pregledači su počeli da dodaju podršku za JavaScript. Postoji mnogo dostupnih verzija (u trenutku pisanja od 1.0 do 1.8). Najstabilnija verzija za sve

pregledače je 1.3. Kod pametnih telefona srednje klase verzija 1.5 je najstabilnija. Novije verzije rade jedino na najnovijim edicijama Firefox i Safari (uključujući i Safari na iOS) pregledača. Programeri često nisu svesni o verziji JavaScripta za koju pišu kod već vode računa o kompatibilnosti u odnosu na svojstva. Postoje puno tehnologija (ili APIa) uključenih u JavaScript koje nisu obavezne i koje neće raditi na svim uređajima.

DOM čini skup metoda za manipulisanje, pregledanje i editovanje XML i HTML dokumenata korišćenjem APIa. Korišćenjem DOMa može se menjati struktura strane na dinamičan način bez potrebe za osvežavanjem. Mobilni internet pregledači mogu biti JavaScript kompatibilni ali da nemaju podršku za DOM funkcionalnosti. Takođe neki pregledači dozvoljavaju pregledanje DOM stabla dokumenta ali ne i menjanje.

Ajax, originalno akronim za Asinhroni JavaScript i XML je tehnika koja uključuje pravljenje asinhronih poziva ka serveru bez osvežavanja strane, prekida korisnikove aktivnosti, menjanja istorije pregledača ili gubljenja globalnih promenljivih. Ajax je naročito bitan kod mobilnih uređaja jer omogućuje skidanje samo sadržaja koji treba osvežiti bez nepotrebnih dodatnih učitavanja cele strane. Kako Ajax nije deo oficijalnog standarda podrška za njega se razlikuje od uređaja do uređaja.

JavaScript Objektna Notacija (poznata kao JSON) je lagani format za razmenu podataka koji je kompatibilan sa skoro svakim programskim jezikom u upotrebi danas. Koristi se u JavaScriptu za prenos sadržaja AJAX zahteva. Ranije je naročito bio popularan XML format za prenos sadržaja ali on naglo gubi na popularnosti zbog veće robusnosti.

Sa dolaskom HTML 5 standarda JavaScript će podržavati nove APIe. Mobilni pregledači već usvajaju neke od njih iako je standard još uvek u fazi razvoja. Neki od podržanih APIa su:

- Offline pristup
- Local Storage
- Geolocation API
- Canvas crtanje
- Push notifikacije (WebSocket)

U sledećoj tabeli je dat pregled podrške internet pregledača za JavaScript:

Pregledač / platforma	Podrška za JavaScript
<b>Safari</b>	Da
<b>Android</b>	Da
<b>Symbian/S60</b>	Da
<b>Nokia Serija 40</b>	Da
<b>webOS</b>	Da
<b>BlackBerry</b>	Da od 3.8
<b>NetFront</b>	Da
<b>Openwave</b>	Ne
<b>Internet Explored</b>	Da
<b>Motorola Internet Browser</b>	Da
<b>Opera Mobile</b>	Da
<b>Opera Mini</b>	Da

### 3.4.Lokalni keš web aplikacije

Karakteristika HTML5 koja naročito privlači pažnju proizvođača internet pregledača, pogotovu za mobilne uređaja, je keš web aplikacije. Keš web aplikacije koji se često naziva i „offline pristup“ pruža mogućnost korisnicima da rade sa aplikacijom čak i kada nema internet konekcije. HTML5 će veoma uvećati upotrebnu vrednost mobilnog weba u tipičnom scenariju kada korisnici napuste pokrivenost mobilne mreže. Offline keš omogućava web aplikaciji da definiše koji podaci (skripte, stilovi, fajlovi) su potrebni aplikaciji da bi ona mogla da se izvršava bez pristupa mreži. Internet pregledač pravi lokalnu kopiju neophodnih fajlova iz koje aplikacija čita podatke sve dok se ponovo ne uspostavi internet konekcija. Postoje brojni primeri aplikacija za koje je offline pristup bitan:

- Čitanje i pisanje elektronske pošte
- Čitanje i pisanje dokumenata
- Prikazivanje prezentacija
- Pravljenje to-do listi

Korišćenjem offline podataka mogu se izbeći uobičajni mrežni zahtevi potrebni za učitavanje aplikacije. Ukoliko je manifest keša ažuriran internet pregledač zna da nema potrebe da li su ostali resursi strane izmenjeni pa se strana učitava veoma brzo iz lokalnog keša. Učitavanje resursa iz keša umesto slanja više HTTP zahteva radi provere da li su resursi izmenjeni štedi količinu prenetih podataka kroz mrežu što je naročito bitno za mobilni web. Spora vremena učitavanja je razlog zašto native aplikacije prednjače u odnosu na mobilni web pa keširanje sadržaja pomaže u prevazilaženju problema. Keš aplikacije daje programerima eksplicitnu kontrolu nad keširanjem sadržaja. Fajl manifesta keša sadrži listu resursa, grupisanih u logičke celine, koje internet pregledači pamte na disku uređaja.

Proizvođači internet pregledača namenjenih pametnim telefonima se trkaju u usvajanju HTML5 karakteristika. Popularni internet pregledači na pametnim telefonima već sada podržavaju offline keširanje, nove multimedijalne elemente i napredne formate za unos podataka. Ipak ne postoji garancija da internet pregledač implementira funkcionalnost keširanja sadržaja pa to mora ispitati pre njegove upotrebe sledećim delom JavaScript koda:

```
if(window.applicationCache) {  
  
    // postoji podrška za keširanje sadržaja  
  
}
```

Jednostavan primer korišćenja keša aplikacije može se demonstrirati na aplikaciji koja sadrži jednu web stranu sastavljenu od HTML dokumenta, CSS fajla i JavaScript fajla. Da bi se aktivirala podrška za keš u HTML5 aplikaciji mora se uključiti manifest atribut u okviru HTML zaglavlja:

```
<!DOCTYPE html>  
  
<html manifest="application.manifest">
```

```
.  
.   
.   
  
</html>
```

U okviru manifest fajla treba navesti listu imena resursa koje treba keširati. Manifest predstavlja standardni tekstuelni fajl sa UTF-8 kodiranjem. Manifest fajl mora imati „text/cache-manifest“ MIME tip koji se postavlja od strane web servera. Manifest fajl ima sledeći format:

```
# komentari počinju hash simbolom  
  
CACHE MANIFEST  
  
# spisak fajlova koje treba keširati  
  
about.html  
  
html5.css  
  
index.html  
  
happy-trails-rc.gif  
  
lake-tahoe.JPG  
  
# fajl koji ne treba keširati  
  
NETWORK  
  
signup.html  
  
FALLBACK  
  
signup.html offline.html  
  
/app/ajax/ default.html
```

Uključivanjem imena fajla u CACHE MANIFEST sekciju internet pregledaču se nalaže da pribavlja fajl iz keša čak i u slučajevima kada postoji aktivna internet konekcija. Fajlovi u FALLBACK sekciji daju alternativne putanje koje zamenjuju resurse koji ne mogu biti učitani. Zahtev ka /app/ajax/ ili ostalim fajlovima čija adresa počinje sa /app/ajax/ će se preusmeriti na default.html stranu ukoliko fajl app/ajax/\* nije dostupan. NETWORK sekcija označava resurse koji se uvek pribavljaju sa mreže, čak i kada postoji keširana verzija resursa.

Ovde se završavaju neophodni uslovi za pokretanje aplikacije u režimu bez internet konekcije, i ukoliko internet pregledač podržava lokalni keš aplikacija će biti dostupna i van područja pokrivenosti mobilnim signalom.



### 3.5.Geolokacija

Jedna od najboljih osobina mobilnih uređaja je da ih možemo nositi uvek sa sobom bilo gde da idemo. Zbog toga treba obratiti pažnju na prostorni kontekst korisnika web aplikacije. Ukoliko web sajt zna tačnu lokaciju na kojoj se posetilac nalazi može mu ponuditi korisne kontekstualne informacije. Na primer ukoliko sam na odmoru u Grčkoj i pretražujem reč pica želim da dobijem informaciju gde je najbliža picerija i kako mogu doći do nje. Ili ukoliko živim u Nišu ne želim da na web strani vidim reklame za fitnes centar koji se nalazi u Beogradu.

Lokacijsko zasnovani servisi su jedni od osnovnih karakteristika modernih mobilnih aplikacija. Mobilna web aplikacija može saznati lokaciju posetioca sajta korišćenjem raznih metoda. Najtačnija pozicija se može odrediti korišćenjem GPS senzora na uređaju ali je pozicioniranje moguće i trijangulacijom pomoću baznih stanica ili putem IP adrese uređaja.

W3C konzorcijum radi na donošenju standarda za određivanje korisnikove lokacije iz web aplikacija korišćenjem JavaScript jezika. Ovaj standard ima za cilj razvoj podrške za geolokacioni API koji će biti implementiran od strane internet pregledača i nuditi standardan interfejs web aplikacijama koje žele da saznaju trenutnu lokaciju posetioca sajta . Iako je standard još uvek u fazi nacрта proizvođači internet pregledača su već implementirali geolokacioni API u svoje proizvode. Na primer mobilni Safari ima podršku za geolokacioni API od verzije iOS verzije 3.0 dok Android pregledači imaju podršku od verzije 2.0. Geolokacioni API se ne oslanja na jednu od tehnologija za određivanje lokacije već omogućava internet pregledaču da odabere jedan od metoda. Uređaj može saznati poziciju korisnika korišćenjem jedne ili kombinacije više sledećih izvora:

- IP adresa
- Trijangulacija koordinata
- GPS senzor
- Lokacija Wi-Fi pristupne tačke
- Lokacija GSM ili CDMA ćelije u kojoj se nalazi mobilni uređaj
- Definisan od strane korisnika

Ispitivanje lokacije korisnika je asinhron proces koji može trajati neko vreme dok se ne odredi tačna lokacija (kao kod GPS). Da bi web aplikacija saznala lokaciju korisnika on mora da da aplikaciji privilegije za čitanje njegove lokacije (slika 12).



Slika 12. Dijalog za potvrdu korišćenja korisnikove trenutne geografske lokacije

Implementacija APIa u mobilnom internet pregledaču uvodi JavaScript objekat „navigator“ sa atributom „geolocation“ preko koga se vrši pristup APIu. Pošto ne postoji garancija da internet pregledač implementira funkcionalnost geolociranja mora se ispitati da li postoji podrška jednostavnom proverom da li postoji navigator.geolocation JavaScript objekat kao u sledećem kodu:

```
if (navigator.geolocation==undefined) {  
    alert("Geolocation API is not present");  
}
```

Interakcija sa geolokacionim APIem se vrši putem getCurrentPosition funkcije geolocation objekta. Metod prima dva callback parametra od kojih je prvi funkcija koja prima poziciju korisnika, a drugi funkcija za upravljanje greškama. Određivanje lokacije korisnika je asinhron proces pa se zato koriste callback metode za upravljanje lokacijom.

```
navigator.geolocation.getCurrentPosition(userLocated, locationError);
```

Prvi callback metod sadrži kod za upravljanje pronađenom lokacijom korisnika.

```
function userLocated(position) {  
  
    var latitude = position.coords.latitude;  
  
    var longitude = position.coords.longitude;  
  
    var timeOfLocation = position.timestamp;  
  
}
```

Metod za obradu greški sadrži kod za upravljanje greškama ukoliko pozicija korisnika nije dostupna ili ne može biti određena ili aplikacija nema odgovarajuće dozvole korisnika.

```
function locationError(error) {  
  
    alert(error.code);  
  
}
```

Nađena lokacija korisnika definiše sledeće atribute po W3C standardnu:

- Geografska širina
- Geografska dužina
- Nadmorska visina (opciono)
- Tačnost
- Tačnost za nadmorsku visinu (opciono)
- Pravac kretanja (opciono) u stepenima u smeru obrtanja kazaljke na satu
- Brzina (opciono) u metrima u sekundi

Određena lokacija korisnika se dalje može koristiti na primer za prikazivanje njegove trenutne lokacije na mapi ili za serviranje sadržaja zasnovanog na njegovoj lokaciji.

### 3.6.Grafika i multimedija

HTML5 donosi dugo godina očekivanu podršku za reprodukciju slika i videa iz internet pregledača, koja je do sada bila dostupna korišćenjem dodatka kao što su Adobe Flash i Microsoft Silverlight, kao i podršku za 2D crtanje i 3D animaciju u okviru web stane.

Za ugrađivanje multimedijalnog sadržaja u kod stranice koriste se audio i video oznake koje reprodukuju sadržaj definisan u src atributu. Ovi multimedijalni objekti mogu biti upravljani kroz JavaScript kod pozivanjem funkcija za puštanje, pauziranje, stopiranje i upravljanje jačinom zvuka. Ubacivanje drugih oznaka između oznaka za reprodukciju sadržaja predstavlja sadržaj koji će se prikazati u slučaju da internet pregledač ne podržava API za rad sa multimedijom što poboljšava kompatibilnost strane za rad na starijim uređajima.

```
<video src="video.avi" controls>
```

```
<!--Oznake koje se nalaze ovde biće prikazane na uređajima koji ne podržavaju video -->  
<object data="player.swf" type="application/x-shockwave-flash">  
<param value="video.flv" name="movie"/>  
</object>  
</video>
```

Audio sadržaj se umeće u web stranu korišćenjem audio oznaka.

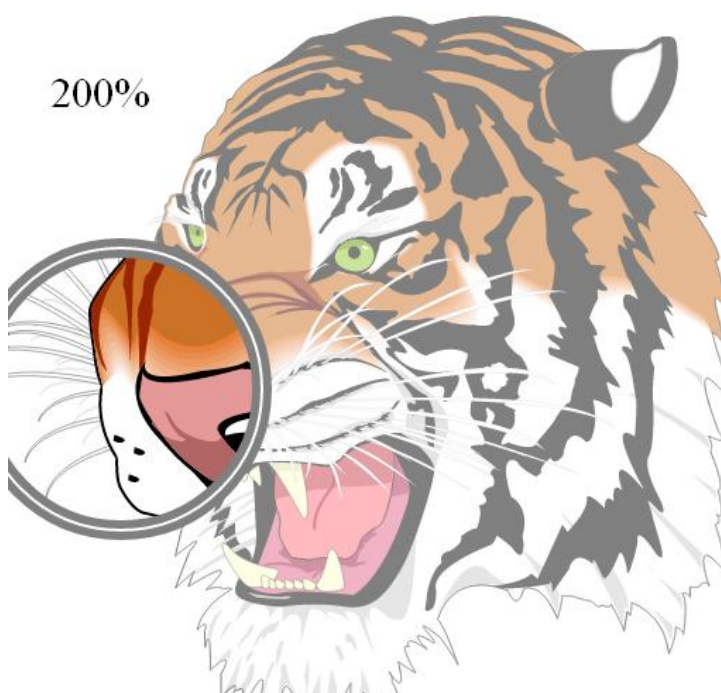
```
<audio controls src="johann_sebastian_bach_air.ogg">  
Kompozicija Johann Sebastian Bach-a.  
</audio>
```

Najlakši način za proveru da li je video ili audio sadržaj podržan od strane pregledača je dinamičko kreiranje oznaka korišćenjem JavaScript funkcije na sledeći način:

```
var hasVideo = !(document.createElement('video').canPlayType);
```

Ukoliko video ili audio sadržaj nije podržan, može se pozvati skripta koja ubacuje multimedijalni sadržaj korišćenjem dodataka kao što je Adobe Flash.

Osim multimedijalnog sadržaja API za crtanje na web strani je dugo vremena bio san svakog web dizajnera. HTML5 specifikacija uvodi korišćenje canvas elementa koji sadri skup funkcija za 2D crtanje. Canvas definiše dvodimenzijalnu površinu na kojoj se može crtati korišćenjem JavaScript APIa. Nacrtna slika nije je u skalarnom formatu. Za predstavljanje vektorske grafike HTML5 uvodi SVG grafiku koja ne gubi na kvalitetu prilikom uveličavanja slike (slika 13).



Slika 13. Primer vektorske SVG grafike

Prilikom definisanja canvas elementa moraju se obavezno navesti atribut „id“ i dimenzije „height“ i „width“.

```
<canvas width="300" height="300" id="canvas">
<!--Oznake koje se nalaze ovde biće prikazane na uređajima koji ne podržavaju canvas -->
</canvas>
```

Za crtanje po canvas elementu koristi se JavaScript objekat 2D context. Pribavljanje pokazivača na 2D context objekat može se obaviti sledećim kodom.

```
var canvas = document.getElementById('canvas');

if (canvas.getContext) {
    // canvas je podržan
    var context = canvas.getContext('2d');
}
```

U isto vreme treba proveriti da li internet pregledač podržava API za 2D crtanje. Objekat context sadrži niz metoda za crtanje. Pre početka crtanja treba podesiti sledeće parametre:

- Boja linije
- Stil punjenja
- Stil linije
- Debljina linije

U sledećoj tabeli su navedeni metodi za 2D crtanje zajedno sa objašnjenjima za svaki od njih:

Metod	Opis
<b>fillRect(x, y, width, height)</b>	Crtanje popunjenog pravougaonika
<b>strokeRect(x, y, width, height)</b>	Crtanje pravougaonika bez unutrašnjeg punjenja
<b>clearRect(x, y, width, height)</b>	Brisanje pravougaone površine
<b>beginPath()</b>	Početak crtanja putanje
<b>closePath()</b>	Zatvaranje putanje
<b>moveTo(x, y)</b>	Pomeranje olovke na zadate koordinate
<b>lineTo(x, y)</b>	Crtanje linije od trenutne pozicije olovke do zdatih koordinata
<b>arc(x, y, radius, startAngle, endAngle, anticlockwise)</b>	Crtanje luka sa centrom u x,y i poluprečnikom radius. Ugao se zadaje u radijanima a anticlockwise definiše smer crtanja luka.+
<b>quadraticCurveTo(controlx, controly, x, y)</b>	Crtanje kvadratne Bezierove krive
<b>bezierCurveTo( control1x, control1y, control2x, control2y, x, y)</b>	Crtanje kubne bezierove krive
<b>stroke()</b>	Crtanje putanje definisane između zadnje beginPath() naredbe

<b>fill()</b>	Zatvaranje putanje definisane između zadnje beginPath() naredbe i punjenje oblika
<b>drawImage(x, y)</b>	Crtanje slike (JavaScript image objekat)
<b>createImageData(width, height)</b>	Kreiranje ImageData objekta sa sadržajem koji je zadat matricom celobrojnih vrednosti koji predstavljaju piksele
<b>getImageData(x, y, w, h)</b>	Pribavljanje ImageData objekta iz trenutne slike
<b>putImageData(image_data, x, y)</b>	Smeštanje ImageData objekta u sliku
<b>strokeText(string, x, y)</b>	Crtanje zadebljanog teksta
<b>fillText(string, x, y)</b>	Punjenj teksta

Napredne opcije za crtanje uključuje senčenje teksta, gradijente, skaliranje slika, transparentnost, fontove, stilove linija, paterne i druge ali one nisu sve podržane od strane svih internet pregledača.

Korišćenjem osnovnih grafičkih primitiva u sledećem primeru je demonstrirano crtanje prostog oblika na 2D canvas elementu.

```
context.beginPath();
context.moveTo(75,25);
context.quadraticCurveTo(25,25,25,62.5);
context.quadraticCurveTo(25,100,50,100);
context.quadraticCurveTo(50,120,30,125);
context.quadraticCurveTo(60,120,65,100);
context.quadraticCurveTo(125,100,125,62.5);
context.quadraticCurveTo(125,25,75,25);
context.stroke();
```

Oblik koji je nacrtan ovim kodom je prikazan na slici 14:



Slika 14. Izgled prostog oblika nacrtanog u 2D canvas elementu

### 3.7. Kratak pregled mobilnih Web aplikacionih okvira

Programiranje mobilnih web aplikacija i sajova nosi niz specifičnosti kao što su različite veličine ekrana uređaja, optimizacija korisničkog interfejsa za navigaciju dodirum umesto kursom miša, različite podržane verzije JavaScripta na uređajima, optimizacija sadržaja za ograničenu brzinu veze koju skoro svi mobilni uređaji imaju.

Srećom mnogi JavaScript aplikacioni okviri implementiraju ove i mnoge druge paradigme za razvoj mobilnih web aplikacija pa programeri ne moraju mnogo da vode računa o specifičnostima već razvijaju aplikacioni kod koji je nezavisan od platforme.

Neke od karakteristika JavaScript mobilnih web aplikacionih okvira su:

- **Optimizacija za ekrane osjetljive na dodir:** Navigacija prstima umesto kursora miša uvodi dodatne izazove pri dizajniranju korisničkog interfejsa. Mobilni web aplikacioni okviri pružaju standardnu skup UI elemenata i procedura za upravljanja događajima nad korisničkim interfejsom koji su nezavisni od platforme i veličine ekrana uređaja.
- **Nezavisnost od platforme:** Napisani kod se izvršava na bilo kom tipu mobilnog uređaja
- **Mala veličina:** Zbog ograničene brzine protoka podataka mobilnih uređaja i zbog visoke cene prenosa podataka aplikacioni okviru optimizuju količinu prenetih podataka a i sama njihova veličina ne prevazilazi desetinu kilobajta.
- **Koriste HTML5 i CSS3 standarde:** Moderni mobilni pregledači podržavaju ove standarde a mobilni aplikacioni okviri prevazilaze specifičnosti njihove implementacije na različitim uređajima.

U trenutku pisanja dostupno je na desetinu mobilnih web aplikacionih okvira koji zadovoljavaju pomenute karakteristike.

**jQuery Mobile** aplikacioni okvir je najpopularniji razvojni okvir danas i pomaže pri kreiranju korisničkog interfejsa prilagođenog najpopularnijim mobilnim platformama kao što su iOS i Android.

Aplikacioni okvir ima malu bazu koda od svega 20KB i ogroman broj elemenata za građenje korisničkog interfejsa koji imitiraju nativne elemente na koje su korisnici navikli kao što su switch kontrole i slajderi.

**Titanium mobile** je moćan i robustan mobilni aplikacioni okvir koji omogućava da se korišćenjem HTML, CSS i JavaScripta razviju nativne mobilne aplikacije za iOS i Android.

Kao jedan od najvećih web razvojnih okvira koji postoje sa preko 300 APIa ima veliku zajednicu programera koji ga koriste pa je lako naći pomoć.

Titanium Mobile podržava nativne UI elemente za iOS i Android kao što su tabele, tabovi i switch kontrole. Takođe ima i APIe za pristup kameri uređaja i nativnom fajl sistemu uređaja.

**The-M-Project** je još jedan JavaScript aplikacioni okvir koji implementira prednosti novih HTML5 funkcionalnosti za razvoj boljih mobilnih aplikacija.

Aplikacionu okvir podržava popularni model-view-controler (MVC) projektni obrazac za razvoj softvera.

Implementira podršku za rad aplikacije u offline režimu pa korisnici mogu da nastavu sa radom u aplikaciji čak i kada izgube konekciju sa internetom na mobilom uređaju.

The-M-Projekat ima odličnu dokumentaciju i vodiče za početnike pa je naročito preporučljiv za programere koji se prvi put sreću sa razvojem mobilnih web aplikacija.

**Sencha Touch** je prvi HTML5 mobilni web aplikacioni okvir. On pruža razvoj aplikacija koje izgledaju kao native iPhone ili Android aplikacije. Naročito je preporučiv za razvoj kompleksnih poslovnih aplikacija sa puno interaktivnosti. Dobro je dokumentovan i ima dobru tehničku podršku obzirom da se njegova komercijalna upotreba naplaćuje.



## 4. Mobilna Web aplikacija za razmenu korisnički generisanog sadržaja

### 4.1. Generalni opis i zahtevi aplikacije

U praktičnom delu je trebalo razviti mobilnu Web aplikaciju koja poseduje karakteristike RIA, funkcionalnost lokaciono-zasnovanih servisa (LBS) i zasnovana je na tehnologijama opisanim u 3. poglavlju: HTML5, CSS, Javascript + AJAX, Local Storage, Geolocation, kontekstualna svesnost (pristup senzorima na uređaju i lokalnom profilu korisnika) i push notifikacijama korišćenjem WebSocket APIa. Cilj je ilustrovati principe razmene korisnički generisanog sadržaja kao osnove tzv. Mobile/Web 2.0 koncepta.

Aplikacija je namenjena pretraživanju geo-referenciranih informacija i razmeni korisnički-generisanog tekstualnog sadržaja. Generalni opis zahteva je:

- Prikaz sopstvene lokacije na mapi i ažuriranje u skladu sa kretanjem
- Kreiranje sopstvenog profila, smeštanje i ažuriranje podataka o profilu lokalno na uređaju i eventualno na serveru
- Generisanje tekstualnog sadržaja i njegovo tag-ovanje za trenutnu lokaciju dobijenu sa GPS, kao i slanje i smeštanje ovih podataka na serveru.
- Pregled tekstualnog sadržaja tag-ovanog za određene lokacije od strane drugih korisnika
- Pregled objekata od interesa u skladu sa izborom korisnika
- Pretraživanje tag-ovanog sadržaja, objekata od interesa po zadatim (prostornim i atributskim) kriterijumima
- Notifikacija o nastalom događaju putem push notifikacije (npr. korisnički generisani sadržaj određenog tipa ili POI je blizu trenutne lokacije korisnika, ili se poklapa sa njegovim profilom)

Kao odgovor na zahteve razvijen je mobilni servis za pretragu restorana na teritoriji grada. Na mapi grada prikazane su lokacije svih restorana u korisnikovoj neposrednoj okolini. Mapa se ažurira u skladu sa kretanjem. Tačna lokacija korisnika se određuje na osnovu GPS senzora mobilnog uređaja ili na osnovu lokacije WiFi pristupne tačke ili lokacije mobilne bazne stanice. Moguće je pretraživanje tačaka od interesa – u ovom slučaju restorana – na osnovu vrste kuhinje i maksimalne udaljenosti od trenutne lokacije korisnika.

Korisnici mogu razmenjivati kratak tekstualni sadržaj (kratka poruka, komentar, preporuka) tagovan za trenutnu lokaciju dobijenu sa GPS koji se smešta na serveru. Na mapi grada kao dodatni sloj prikazuju se georeferencirane poruke. Mobilni korisnici bi pored restorana (objekata) mogli da pregledaju i ove kratke poruke/komentare (npr. prikazati poruke koje su do 200m od moje lokacije a generisao ih je Slavko) i da dobijaju notifikacije o njima kad se nađu u njihovoj blizini u skladu sa svojim profilom.

U svakom trenutku korisnici mogu dobiti notifikaciju o nastalom događaju koji se poklapa sa njegovim profilom. Profil korisnika definiše tip notifikacije koju korisnik želi da prima – na primer obaveštenja o popustima u restoranima nacionalne kuhinje - kao i maksimalnu udaljenost lokacije

događaja od trenutne lokacije. Obaveštenje se prima u vidu push notifikacije u realnom vremenu i moguće je prikazati na mapi tačku od interesa za koju je obaveštenje generisano (restoran, poruka).

## **4.2.Korišćeni alati, okvir i tehnologije**

Za potrebe implementacije korišćene su različiti programski okviri i tehnologije odabrane u skladu sa zahtevima projekta da bi se smanjilo potrebno vreme za razvoj i poboljšao opšti kvalitet programskog koda.

### **4.2.1. JQuery mobile**

Za klijentski deo aplikacije koji se izvršava na mobilnim uređajima korišćen aplikacioni okvir jQuery Mobile koji je prethodno opisan u poglavlju 3.

jQuery Mobile aplikacioni okvir podržava razvoj JavaScript koda i jedinstvenog korisničkog interfejsa koji radi na svim najpopularnijim internet pregledačima za pametne telefone i tablet uređaje. Da bi se postigla podrška za veliki broj različitih tipove uređaja jQuery Mobile gradi strane koristeći semantički čist HTML primenjujući CSS i JavaScript da bi se dobio bogat, interaktivan korisnički interfejs.

Osnovne karakteristike aplikacionog okvira su:

- Izrađen je na osnovu jQuery core biblioteke pa obezbeđuje lako učenje jer je najveći broj web programera nekada koristilo jQuery
- Kompatibilnost sa svim poznatim mobilnim platformama – iOS, Android, Blackberry, Palm WebOS, Nokia/Symbian, Windows Mobile, bada, MeeGo i ostalim uređajima koji interpretiraju HTML
- Mala veličina biblioteke (12k kompresovanog koda) i mali broj grafičkih resursa radi povećanja brzine učitavanja
- Izrada strana korišćenjem HTML5 oznaka radi ubrzanja vremena razvoja i smanjenja količine neophodnog JavaScript koda
- Automatska inicijalizacija korisničkog interfejsa korišćenjem HTML5 data-role atributa koji prevode elemente strane u jQuery Mobile komponente korisničkog interfejsa
- Podrška za različite tipove unosa – dodir, miš, navigacioni kursor korišćenjem jednostavnog APIa
- Moćan okvir za definisanje tema pa se izgled korisničkog interfejsa aplikacije može lako promeniti

Aplikacioni okvir je deteljno testiran na velikom broju različitih mobilnih platformi. Trenutna verziji aplikacionog okvira provereno radi na sledećim platformama:

- Apple iOS (3.1-4.2): testirano na iPhone, iPod Touch, iPad
- Android (1.6-2.3): testirano na HTC Incredible, Motorola Droid, Google G1 i Nook Color
- Blackberry 6: testirano a Torch i Style

- Windows Phone 7: testirano na HTC Surround
- Palm WebOS (1.4): testirano na Pre, Pixi
- Opera Mobile (10.1): Android
- OperaMini (5.02): iOS, Android
- Firefox Mobile (beta): Android

Obzirom da je jQuery Mobile izrađen nad jQuery core bibliotekom aplikacije takođe rade i na svim verzijama internet pregledača za personalne računare – Firefox, Chrome, Safari, Internet Explorer, Opera itd.

#### **4.2.2. CakePHP**

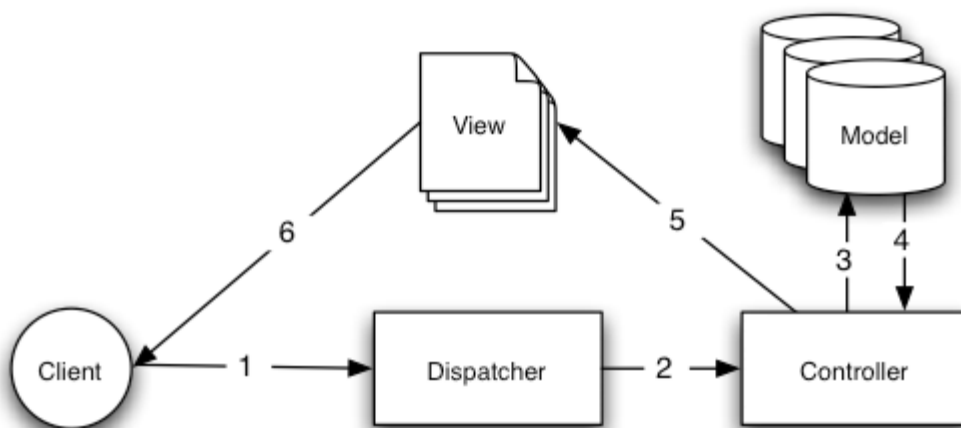
Serverski deo koji implementira API za pretraživanje i pribavljanje podataka o tačkama od interesa i služi za komunikacijom sa bazom podataka je rađen u programskom jeziku PHP u aplikacionom okviru CakePHP. Odlučio sam se za CakePHP zbog višegodišnjeg prethodnog iskustva koje imam u radu sa ovim okvirom.

CakePHP je besplatan, open-source aplikacioni okvir namenjen ubrzanom razvoju web aplikacija. Osnovne karakteristike aplikacionog okvira su:

- Model, View, Controller arhitektura
- Automatsko generisanje koda
- Biblioteke za rad sa HTML, formama, paginacijom, AJAX-om, JavaScript-om, XML-om
- Podrška za autentikaciju i liste za kontrolu pristupa
- Jednostavan i proširiv model za validaciju podataka
- Komponente za upravljanje sigurnošću, sesijama i HTTP zahtevima
- Dodatne klase za rad sa datotekama, direktorijumima, nizovima i slično

CakePHP prati MVC projektni obrazac za razvoj softvera. Programiranje korišćenjem MVC obrasca odvaja aplikacioni kod u tri glavna dela:

- Model koji reprezentuje podatke aplikacije
- View koji vrši prikaz podataka
- Controller koji upravlja i prosleđuje korisničke zahteve



Slika 15. Redosled poziva MVC zahteva u CakePHP aplikacionom okviru

Slika 15 pokazuje primer najprostijeg MVC zahteva u CakePHP. Primera radi razmotrimo šta se dešava ako klijent „Slavko“ klikne na link „Kupi kolač!“ na glavnoj strani aplikacije.

- Slavko je kliknuo na link koji pokazuje na <http://www.example.com/cakes/buy> a internet pregledač pravi zahtev ka web serveru
- Komponenta dispičer proverava URL zahteva (/cakes/buy) i prosleđuje zahtev ka odgovarajućem kontroleru
- Kontroler obavlja akciju definisanu programskim kodom. Na primer proverava da li je Slavko prijavljen korisnik
- Kontroler takođe koristi model da bi pribavio podatke. Modeli obično reprezentuju tabele u bazi podataka. U ovom primeru kontroler koristi model da bi pribavio iz baze listu zadnjih kupovina korisnika Slavko
- Kada kontroler obavi svoj posao on predaje podatke dalje u pogled. Pogled uzima ove podatke i sprema ih za prezentaciju na klijentu. Pogledi u CakePHP najčešće prikazuju HTML format ali se lako mogu prikazati i PDF, XML, JSON i slično u zavisnosti od potrebe
- Kada pogled formira konačnu stranu sadržaj se dalje vraća i prikazuje u Slavkovom internet pregledaču

Skoro svaki zahtev u aplikaciji prati ovaj jednostavan model. Ovakvim pristupom aplikacija se razbija u modularne pakete koje je lako održavati, a i smanjuje se ukupna količina programskog koda zbog ponovne upotrebljivosti.

#### 4.2.3. NodeJS

Za potrebe slanja notifikacija ka klijentima u realnom vremenu trebalo je izabrati tehnologiju koja ima podršku za asinhronu obradu događaja. Server za notifikacije održava perzistentne TCP konekcije sa svim trenutno aktivnim klijentima. Svaki put kada se generiše događaj server notifikacija sekvencijalno šalje obaveštenje svim registrovanim klijentima.

Node.js je svega godinu dana star aplikacioni okvir koji omogućava razvoj web aplikacija visokih performansi koje daju odgovore u realnom vremenu korišćenjem JavaScript koda koji se izvršava na serveru. JavaScript se tradicionalno izvršavao samo u okviru internet pregledača, ali nedavno se pojavio veliki interes za dovođenje JavaScript koda i u serversko programiranje.

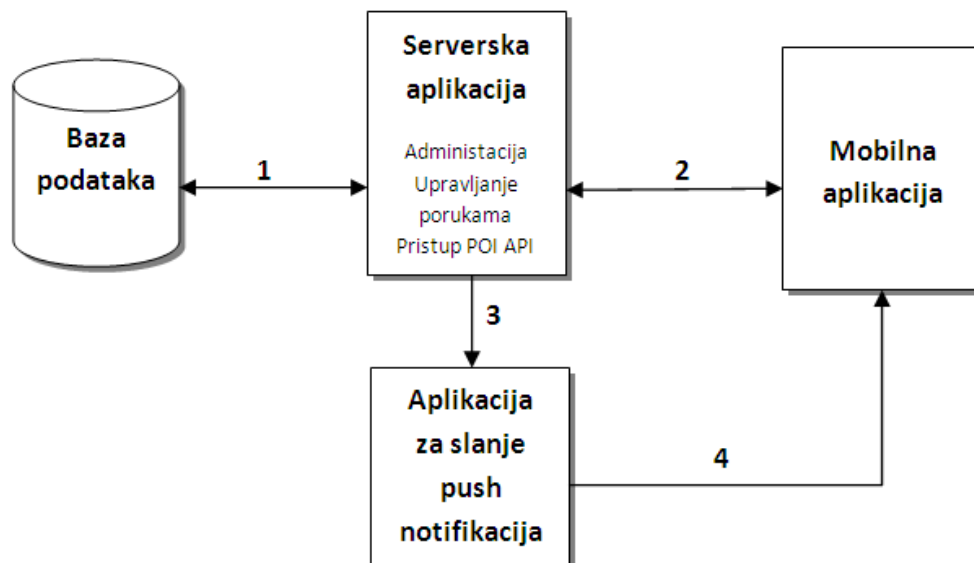
Node.js uvodi novu paradigmu u serverskom programiranju jer se kod izvršava kada se desi neki unapred definisani događaj za razliku od tradicionalnih rešenja koja se izvršavaju sekvencionalno u programskim nitima. Web serveri, kao što je Apache, koji se koriste za obradu PHP i drugih CGI skripti su zasnovani na nitima i oni pokreću novu programsku nit za svaku dolazeću konekciju od klijenata. Iako ova paradigma funkcioniše za veliku većinu internet aplikacija, kod aplikacija koje se izvršavaju u realnom vremenu i koje imaju veliki broj dolazećih konekcija, kao što je program za časkanje, ona dobro ne funkcioniše.

Node.js koristi petlju koja obrađuje događaje umesto programskih niti koja ima mogućnost da obradi milione konkurentnih konekcija. Kao prednost koristi činjenicu da serveri potroše većinu vremena čekajući na I/O operacije kao što su čitanje datoteke sa hard diska, pristup udaljenom web servisu ili čekanje da datoteka bude poslata na server, jer su ove operacije znatno sporije od memorijskih operacija. Svaka I/O operacija u Node.js je asinhrona, što znači da server može da nastavi da obrađuje dolazeće zahteve dok čeka na I/O. JavaScript kod pruža odličnu podršku za programiranje zasnovano na događajima zbog anonimnih funkcija koje se koriste kao odgovori na definisani događaj.

Za konkretni slučaj servera za slanje push notifikacija, Node.js je doneo prednost da kompletna aplikacija može da se razvije u 50-ak linija JavaScript koda, bez potrebe za bazom podataka i blokirajućim I/O jer se konekcije od klijenata pamte u memoriji, a perzistentne WebSocket konekcije se nalaze u pasivnom čekaju dok se ne desi događaj koji aktivira slanje poruke ka svim trenutno konektovanim klijentima.

### **4.3.Generalna arhitektura aplikacije**

Arhitektura aplikacije je troslojna sa bazom podataka na najnižem sloju, aplikacijom za slanje notifikacija i aplikacijom za smeštanje i pretraživanje tačaka od interesa na srednjem sloju i klijentom za mobilne uređaje na najvišem sloju. Diagram visokog nivoa arhitekture aplikacije je prikazan na slici 16:

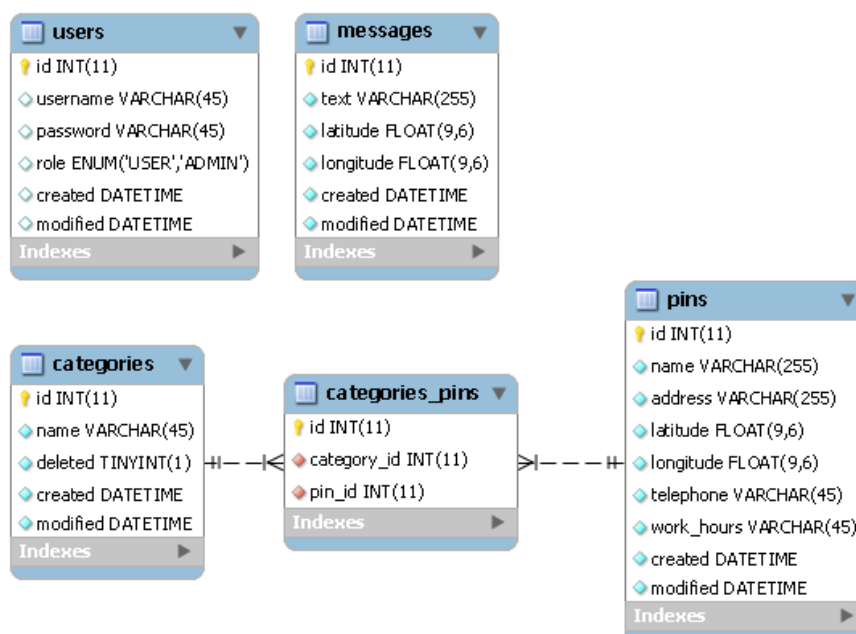


Slika 16. Dijagram visokog nivoa arhitekture aplikacije

Veze između pojedinih aplikativnih komponenti definišu normalne tokove podataka među komponentama.

#### 4.4. Baza podataka

Šema baze podataka aplikacije data je na slici 17. U nastavku će biti opisana uloga svake od tabela:



Slika 17. Šema baze podataka

#### 4.4.1. Korisnici

Tabela Users pamti registrovane korisnike. Iako mobilna aplikacija ne zahteva nikakvu prijavljivanje i obezbeđuje potpunu anonimnost korisnika registracija se koristi za administratore koji imaju mogućnost uređivanja sadržaja kao što su Kategorije (vrsta kuhinje restorana) i dodavanje Pinova (objekat restorana).

- username – korisničko ime
- password – lozinka u enkriptovanom obliku
- role – nivo privilegija korisnika

#### 4.4.2. Poruke

Tabela Messages pamti geo-tagovane poruke koje se šalju od strane mobilnih korisnika i sadrže njegovu trenutnu lokaciju dobijenu od GPS senzora.

- text – tekst poruke
- latitude – geografska širina na kojoj je poruka dodata
- longitude – geografska dužina na kojoj je poruka dodata

#### 4.4.3. Kategorije

Tabela Categories pamti tip kuhinje restorana na osnovu kojih su oni razvrstani. Kategorija restorana može biti na primer pekara, riblji restoran, picerija i slično. Kategorije su uvedene sa razlogom da mobilni korisnici mogu pretraživati i dobijati notifikacije za određenu kategoriju restorana koja ih zanima

- name – ime kategorije restorana
- deleted – marker koji označava da li je kategorija obrisana (soft delete)

#### 4.4.4. Tačke od interesa

Tabela Pins služi za pamćenje tačaka od interesa – u konkretnom slučaju restorana. Tabela sadrži osnovne informacije o restoranu kao i njegovu tačnu geografsku lokaciju.

- name – Ime restorana
- address – Adresa restorana
- latitude – Geografska širina restorana
- longitude – Geografska dužina restorana
- telephone – Broj telefona za rezervacije restorana
- work\_hours – Radno vreme restorana

#### 4.4.5. Vezna tabela između kategorija i tačaka od interesa

Tabela categories\_pins služi kao vezna tabela između tabela Categories i Pins tipa n:m. Svaki restoran može pripadati u više kategorija i svaka kategorija sadrži neograničen skup restorana.

- category\_id – Strani ključ ka tabeli Kategorijes
- pin\_id – Strani ključ ka tabeli Pins

## 4.5. Serverska aplikacija za upravljanje tačkama od interesa

Serverska aplikacija čini centralnu tačku koja vrši koordinaciju i upravljanje zahtevima između drugih delova aplikacije. Uloge aplikacije su:

- Implementira administrativni portal za dodavanje tačaka od interesa (restorana u bazu podataka)
- Slanje notifikacije klijentima o popustima u restoranima.
- Implementira API za pristup i pretraživanje tačaka od interesa (restorani, poruke) koje koristi mobilna aplikacija i prikazuje ih na mapi.
- Implementira API za prijem korisnički generisanog saržaja

Administrativni portal je razvijen kao web aplikacija namenjena personalnim računarima i služi za moderiranje informacija koje se pamte u aplikativnu bazu podataka. Administrativni portal zahteva autentifikaciju identiteta administratora (slika 18).

Username\*

Password\*

Login

Autor: Mirko Borivojević <borivojevic@gmail.com>

Slika 18. Izgled strane za autentifikaciju korisnika



Administrator ima mogućnost ažuriranja liste restorana i liste kategorija restorana (slika 19).

#### Actions

New Pin

List Categories

New Category

#### Pins

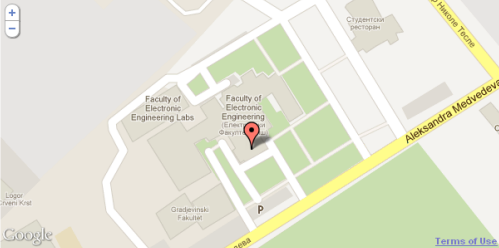
Id	Name	Address	Latitude	Longitude	Telephone	Work Hours	Created	Modified	Actions
1	Kluzo		43.317791	21.900000		08-24	2011-05-22 16:35:19	2011-05-22 16:35:19	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
2	Casa di Pasta		43.317894	21.899561		08-24	2011-05-22 16:36:43	2011-05-22 16:36:43	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
3	Stara Srbija		43.317348	21.896610		08-24	2011-05-22 16:37:01	2011-05-22 16:37:01	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
4	Mali Vikend		43.317924	21.899902		08-24	2011-05-22 16:37:20	2011-05-22 16:37:20	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
5	MBN		43.318020	21.894861		08-24	2011-05-22 16:37:38	2011-05-22 16:37:38	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
6	Pekara Branković		43.317440	21.893391		08-24	2011-05-22 16:38:03	2011-05-22 16:38:03	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
7	Mc Donald's		43.321079	21.895279		08-24	2011-05-22 16:38:38	2011-05-22 16:38:38	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
8	Laf		43.320267	21.901962		08-24	2011-05-22 16:39:19	2011-05-22 16:39:19	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
9	Micko		43.320194	21.902533		08-24	2011-05-22 16:39:39	2011-05-22 16:39:39	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
10	Hleb i Pecivo		43.318542	21.892094		08-24	2011-05-22 16:40:23	2011-05-22 16:40:23	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
11	Nišlijska Mehana		43.325722	21.898659		08-24	2011-05-22 16:41:55	2011-05-22 16:41:55	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
12	Gusar		43.323669	21.896898		08-24	2011-05-22 16:42:11	2011-05-22 16:42:11	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
13	Picerija Gogo		43.320206	21.893831		08-24	2011-05-22 16:42:59	2011-05-22 16:42:59	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
14	Mr King		43.320999	21.894817		08-24	2011-05-22 16:43:39	2011-05-22 16:43:39	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
15	Bata Bane		43.317783	21.889893		08-24	2011-05-22 16:44:03	2011-05-22 16:44:03	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
16	Pekara Cair 2		43.314732	21.899076		08-24	2011-05-22 16:45:18	2011-05-22 16:45:18	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
17	Pekara Cair 1		43.311710	21.911598		08-24	2011-05-22 16:45:48	2011-05-22 16:45:48	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
18	MBN		43.318863	21.909559		08-24	2011-05-22 16:46:25	2011-05-22 16:46:25	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Page 1 of 1, showing 18 records out of 18 total, starting on record 1, ending on 18

<< previous | next >>

Slika 19. Izgled strane za uređivanje sadržaja

Prilikom dodavanja novog restorana treba popuniti sve neophodne informacije i označiti lokaciju restorana na mapi (slika 20).



Name\*

Klub Elektronskog Fakulteta

Address

Aleksandra Medvedeva 14

Telephone

N/A

Cellphone

N/A

Email

N/A

Website

N/A

Work Hours

08-17

Category

Restoran  
Riblji restoran  
Italijanska kuhinja  
Kineska kuhinja  
Palacinkarnica  
Pekara  
Cevapdzinica  
Kafana  
Brza hrana  
Picerija

Submit

Slika 20. Izgled strane za dodavanje restorana

Administrator ima mogućnost slanja obaveštenja mobilnim korisnicima o akcijama u restoranima (slika 21).

## Pošalji obavestenje

Ime restorana

Mc Donald's ▼

Tekst obaveštenja

U McDonald's-Restoranu Tvrđava narednih pola sata  
važi 50% popusta na sve obroke.

- I'm loving it!

Pošalji

Autor: Mirko Borivojević <borivojevic@gmail.com>

Slika 21. Izgled strane za slanje obaveštenja

Za slanje notifikacija ka samim klijentima je zadužena Aplikacija za slanje push notifikacija kojoj serverska aplikacija izdaje zahtev za slanje.

Serverska aplikacija je implementirana korišćenjem MVC projektnog obrasca pa su tako sve akcije razbijene u tri celine:

- Model – implementira logiku za snimanje i pristup podacima iz baze. Korisni ORM za mapiranje šeme baze podataka u objekte pa se bazi pristupa na objektni način i nema potrebe za pisanjem SQL upita
- View – implementira prikazivanje sadržaja HTML strane. Koristi standardni template engine za generisanje strana koji razbija strane u nezavisne elemente.
- Controller – implementira vezu između pogleda i modela – rutira zahteve za snimanje podataka od pogleda ka modelima i prosleđuje neophodne podatke za prikaz iz modela ka pogledima

Za demonstraciju tipičnog toka podataka koristiću akciju za pamćenje novog restorana (slika 20)

U pogledu je implemenirana forma za dodavanje novog restorana (slika 22).

```

100644 | 33 lines (28 sloc) | 0.866 kb
raw | blame | history

1 <?php
2 $javascript->link('http://maps.google.com/maps/api/js?sensor=true', false);
3 $javascript->link('addPin', false);
4 ?>
5 <div id="addPinMap" style="width: 600px; height: 300px;"></div>
6 <?php
7 echo $this->Form->create('Pin');
8
9 echo $this->Form->hidden('latitude');
10 echo $this->Form->hidden('longitude');
11 echo $this->Form->hidden('zoom');
12 echo $this->Form->hidden('close');
13
14 echo $this->Form->error('latitude');
15 echo $this->Form->error('longitude');
16
17 echo $this->Form->input('name');
18 echo $this->Form->input('address');
19 echo $this->Form->input('telephone');
20 echo $this->Form->input('cellphone');
21 echo $this->Form->input('email');
22 echo $this->Form->input('website');
23 echo $this->Form->input('work_hours');
24 echo $this->Form->input('Category');
25
26 echo $this->Form->end('Submit');
27 ?>
28
29 <script type="text/javascript">
30 $(document).ready(function() {
31     setupMap();
32 });
33 </script>

```

Slika 22. Isečak koda – forma za dodavanje restorana

U kontroleru je implementirana akcija za smeštanje podataka u bazu (slika 23). Koristi se objektni pristup i nema pisanja SQL upita.

```

20
21
22 function admin_add() {
23     if(true == $this->params['isAjax']) {
24         $this->layout = 'modal';
25     }
26     $success = false;
27     if($this->RequestHandler->isPost()) {
28         $success = $this->Pin->save($this->data);
29     }
30     $category_options = $this->Pin->Category->find('list', array(
31         'conditions' => array('Category.deleted' => false)
32     ));
33     $this->set('categories', $category_options);
34     $this->data['Pin']['close'] = (false == $success) ? 0 : 1;
35 }

```

Slika 23. Isečak koda – akcija za smeštanje u bazu podataka

U modelu su definisana pravila za validaciju podataka prilikom snimanja u bazu podataka (slika 24). Ukoliko forma ne ispunjava pravila validacije, podaci neće biti zapamćeni, a administratoru će biti prikazana greška na formi za dodavanje restorana.

```

100644 | 28 lines (25 sloc) | 0.878 kb
raw | blame | history

1 <?php
2 class Pin extends AppModel {
3
4     var $hasAndBelongsToMany = array('Category');
5
6     /**
7      * Constructor. Binds the model's database table to the object. Specify model validation array.
8      *
9      * @param integer $id Set this ID for this model on startup
10     * @param string $table Name of database table to use.
11     * @param object $ds DataSource connection object.
12     */
13     function __construct($id = false, $table = null, $ds = null) {
14         parent::__construct($id, $table, $ds);
15         $this->validate = array(
16             'name' => array(
17                 'notEmpty' => array('rule' => 'notEmpty', 'message' => __('Name is required.', true)),
18             ),
19             'latitude' => array(
20                 'notEmpty' => array('rule' => 'notEmpty', 'message' => __('Latitude is required.', true))
21             ),
22             'longitude' => array(
23                 'notEmpty' => array('rule' => 'notEmpty', 'message' => __('Longitude is required.', true)),
24             ),
25         );
26     }
27 }
28

```

Slika 24. Isečak koda – definisana pravila validacije podataka

Za slanje notifikacija implementiran je interfejs ka Aplikaciji za slanje push notifikacija. Kada god je u programskom kodu potrebno slati notifikaciju, na primer kada se snimi nova poruka koja je stigla od mobilnog portala putem APIa ili je poslata poruka od restorana (slika 25) poziva se ovaj interfejs koji prosleđuje dalje zahtev ka Aplikaciji za slanje push notifikacija.

```

38 function sendPushNotification($message, $latitude, $longitude, $categories = false) {
39     // Send notification
40     $url = 'http://localhost:7777';
41     $fields = array(
42         'message'=>urlencode($message),
43         'latitude'=>urlencode($latitude),
44         'longitude'=>urlencode($longitude)
45     );
46     if(false != $categories) {
47         $fields['categories'] = $categories;
48     }
49     //url-ify the data for the POST
50     $fields_string = '';
51     foreach($fields as $key=>$value) { $fields_string .= $key.'='.$value.'&'; }
52     $fields_string = rtrim($fields_string, '&');
53
54     //open connection
55     $ch = curl_init();
56
57     //set the url, number of POST vars, POST data
58     curl_setopt($ch, CURLOPT_URL, $url);
59     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
60     curl_setopt($ch, CURLOPT_PORT, 7777);
61     curl_setopt($ch, CURLOPT_POST, count($fields));
62     curl_setopt($ch, CURLOPT_POSTFIELDS, $fields_string);
63
64     //execute post
65     $status = curl_exec($ch);
66     //close connection
67     curl_close($ch);
68
69     return $status;
70 }

```

Slika 25. Isečak koda – metod za prosleđivanje poruke ka aplikaciji za slanje push notifikacija

Za pribavljanje i pretraživanje tačaka od interese implementiran je API REST tipa kojem se zahtev šalje korišćenjem standardnog HTTP-a.

Za pretraživanje tačaka od interesa u određenom geografskom prostoru potrebno je pozvati API na sledećoj adresi:

- [http://46.21.104.5/pins/list\\_markers](http://46.21.104.5/pins/list_markers)

Parametri poziva se prosleđuju kao standardi GET ili POST podaci. Metod za pretraživanje liste objekata (restorani, poruke) prima sledeće parametre

- sw\_lat – Jugozapadna geografska širina oblasti u kojoj se objekat traži
- sw\_lon – Jugozapadna geografska dužina oblasti u kojoj se objekat traži
- ne\_lat – Severoistočna geografska širina oblasti u kojoj se objekat traži
- ne\_lon – Severoistočna geografska dužina oblasti u kojoj se objekat traži
- name – Ime traženog objekta (nije obavezno)
- category – Kategorija kojoj traženi objekat pripada (nije obavezno)
- area – Maksimalna udaljenost od trenutne lokacije korisnika u kojoj se objekat traži (nije obavezno)
- latitude – Trenutna geografska širina korisnika (nije obavezno)
- longitude – Trenutna geografska dužina korisnika (nije obavezno)

Metod za pretraživanje tačaka je implemeniran u kontroleru i on uzima u obzir poruke i restorane (slika 26).

```

36 function list_markers() {
37     $this->layout = 'ajax';
38     $sw_lat = isset($_REQUEST['sw_lat']) ? $_REQUEST['sw_lat'] : null;
39     $sw_lon = isset($_REQUEST['sw_lon']) ? $_REQUEST['sw_lon'] : null;
40     $ne_lat = isset($_REQUEST['ne_lat']) ? $_REQUEST['ne_lat'] : null;
41     $ne_lon = isset($_REQUEST['ne_lon']) ? $_REQUEST['ne_lon'] : null;
42
43     $markers = array('markers' => array());
44     if(null != $sw_lat && null != $sw_lon && null != $ne_lat && null != $ne_lon) {
45
46         $conditions = array(
47             'NOT' => array('Pin.latitude' => null),
48             'NOT' => array('Pin.longitude' => null),
49             'Pin.latitude >' => $sw_lat,
50             'Pin.latitude <' => $ne_lat,
51             'Pin.longitude >' => $sw_lon,
52             'Pin.longitude <' => $ne_lon
53         );
54         if(isset($_REQUEST['name'])) {
55             $conditions['Pin.name LIKE'] = "%{$_REQUEST['name']}%";
56         }
57         if(isset($_REQUEST['category'])) {
58             $categories = explode(' ', $_REQUEST['category']);
59             $this->Pin->bindModel(array('hasOne' => array('CategoriesPin')));
60             $conditions['CategoriesPin.category_id'] = $categories;
61         }
62         $data = $this->Pin->find('all', array(
63             'conditions' => $conditions,
64             'fields' => array(
65                 'Pin.id',
66                 'Pin.name',
67                 'Pin.latitude',
68                 'Pin.longitude',
69                 'Pin.address',
70                 'Pin.telephone',
71                 'Pin.work_hours',
72             ),
73             'recursive' => 2
74         ));
75         foreach($data as $pin) {
76             $categories = array();
77             $category_ids = array();
78             foreach($pin['Category'] as $category) {
79                 $categories[] = $category['name'];
80                 $category_ids[] = $category['id'];
81             }
82             $pin['Pin']['category'] = implode(' ', $categories);
83             $pin['Pin']['category_ids'] = implode(' ', $category_ids);
84             unset($pin['Category']);
85             $pin['Pin']['type'] = 'pin';
86             $markers['markers'][]['marker'] = $pin['Pin'];
87         }
88     }
89
90     // Add messages data
91     $conditions = array(
92         'NOT' => array('Message.latitude' => null),
93         'NOT' => array('Message.longitude' => null),
94         'Message.latitude >' => $sw_lat,
95         'Message.latitude <' => $ne_lat,
96         'Message.longitude >' => $sw_lon,
97         'Message.longitude <' => $ne_lon
98     );
99     $data = $this->Message->find('all', array(
100         'conditions' => $conditions,
101         'fields' => array(
102             'Message.id',
103             'Message.text',
104             'Message.latitude',
105             'Message.longitude',
106         ),
107         'recursive' => -1
108     ));
109     foreach($data as $pin) {
110         $pin['Message']['type'] = 'message';
111         $pin['Message']['id'] = 'mes_' . $pin['Message']['id'];
112         $markers['markers'][]['marker'] = $pin['Message'];
113     }
114     // Filter markers by area
115     if(isset($_REQUEST['area']) && isset($_REQUEST['latitude']) && isset($_REQUEST['longitude'])) {
116         foreach($markers['markers'] as $key => $marker) {
117             if($this->distance($_REQUEST['latitude'], $_REQUEST['longitude'], $marker['marker']['latitude',
118                 $marker['marker']['longitude']) > $_REQUEST['radius']) {
119                 unset($markers['markers'][$key]);
120             }
121         }
122     }
123     $this->set('markers', $markers);
124 }

```

Slika 26. Isečak koda – metod za pretragu tačaka od interesa

Pronađeni rezultati se prosleđuju pogledu koji ih formatira kao JSON objekat i prosleđuje odgovor klijentu (slika 27).

```
{
  - markers: [
    - {
      - marker: {
        id: "7",
        name: "Mc Donald's",
        latitude: "43.321079",
        longitude: "21.895279",
        address: "",
        telephone: "",
        work_hours: "08-24",
        category: "Restoran",
        category_ids: "1",
        type: "pin"
      }
    },
    - {
      - marker: {
        id: "13",
        name: "Picerija Gogo",
        latitude: "43.320206",
        longitude: "21.893831",
        address: "",
        telephone: "",
        work_hours: "08-24",
        category: "Picerija",
        category_ids: "10",
        type: "pin"
      }
    },
    - {
      - marker: {
        id: "14",
        name: "Mr King",
        latitude: "43.320999",
        longitude: "21.894817",
        address: "",
        telephone: "",
        work_hours: "08-24",
        category: "Brza hrana",
        category_ids: "9",
        type: "pin"
      }
    }
  ]
}
```

Slika 27. Isečak koda – JSON sadržaj odgovora metoda za pretragu tačaka od interesa

Za snimanje korisnički generisanog sadržaja implementiran je poseban REST API metod kojem se pristupa sledećom adresom:

- [http://46.21.104.5/messages/save\\_message.json](http://46.21.104.5/messages/save_message.json)

Parametri ovog api poziva su:



- latitude – Geografska širina poruke
- longitude – Geografska dužina poruke
- message – Tekst poruke generisane od strane korisnika

U slučaju uspešnog snimanja metod vraća potvrđan odgovor sa JSON enkodiranim sadržajem

- {"Status":"OK"}

## 4.6. Aplikacija za slanje notifikacija

Mobilni klijent uspostavlja perzistentnu WebSocket konekciju sa aplikacijom za slanje notifikacija. Korišćenjem ove veze server može poslati poruku o nastalom događaju klijentu potpuno asinhrono. Ovakva paradigma je potpuno nova u domenu web programiranja gde je tradicionalno jedini mogući tok interakcije bio razmena zahteva inicirana od strane klijenta – request/response.

Kako PHP ne podržava programiranje asinhronog koda koji bi reagovao na događaje u realnom vremenu morao sam da izaberem drugu tehnologiju. Odlučio sam se za NodeJS koji predstavlja JavaScript koji se izvršava na serveru kao što je opisano u poglavlju 4.2.3. NodeJS kod odgovara na događaje u realnom vremenu – na primer pristigla nova konekcija od strane klijenta ili generisana poruka koja se treba poslati svim klijentima.

Aplikacija za slanje notifikacija „osluškuje“ sledeće dolazeće konekcije:

- Web Socket konekcije na portu 7777
- HTTP konekcije na portu 80

Preko HTTP konekcije aplikacija komunicira sa serverskom aplikacijom za rad sa POI koja joj prosleđuje zahtev za slanje poruka svim klijentima.

Na portu 7777 aplikacija prima dolazeće WebSocket konekcije od strane mobilnih klijenata. Aplikacija ažurira listu aktivnih konekcija ka klijentima. Nakon primanja zahteva za slanje poruke aplikacija vrši broadcast svim aktivnim klijentima. Lista aktivnih klijenata se pamti u programskoj memoriji i nema potrebe za korišćenjem baze podataka (slika 28).

```

100644 | 61 lines (50 sloc) | 1.461 kb
raw | blame | history

1  var sys = require("sys")
2  , http = require("http")
3  , fs = require("fs")
4  , path = require("path")
5  , ws = require('./lib/ws/server')
6  , qs = require('querystring');
7
8  var connections = Array();
9  var message = null;
10
11  var httpServer = http.createServer(function(req, res){
12    if (req.method == 'POST') {
13      var body = '';
14      req.on('data', function (data) {
15        body += data;
16      });
17      req.on('end', function () {
18        var post = qs.parse(body);
19        message = JSON.stringify(post);
20        for (var i = 0; i < connections.length; i++) {
21          var connection = connections[i];
22          server.send(connection.id, message);
23        }
24      });
25    }
26    res.end('OK');
27  });
28
29  var server = ws.createServer({
30    server: httpServer
31  });
32
33  server.addListener("listening", function(){
34    sys.log("Listening for connections.");
35  });
36
37  // Handle WebSocket Requests
38  server.addListener("connection", function(conn) {
39    console.log('[*] open');
40
41    connections.push(conn);
42
43    conn.addListener("message", function(message){
44      if (message == 'close') {
45        console.log('[-] close requested')
46        conn.close();
47      } else {
48        console.log('[+] ', (new Buffer(message)).inspect());
49        server.broadcast("update");
50      }
51    });
52
53    conn.addListener("close", function(){
54      console.log('[*] close');
55    });
56  });
57
58  server.addListener("disconnect", function(conn){ });
59
60  server.listen(7777);
61

```

Slika 28. Isečak koda – aplikacija za slanje push notifikacija

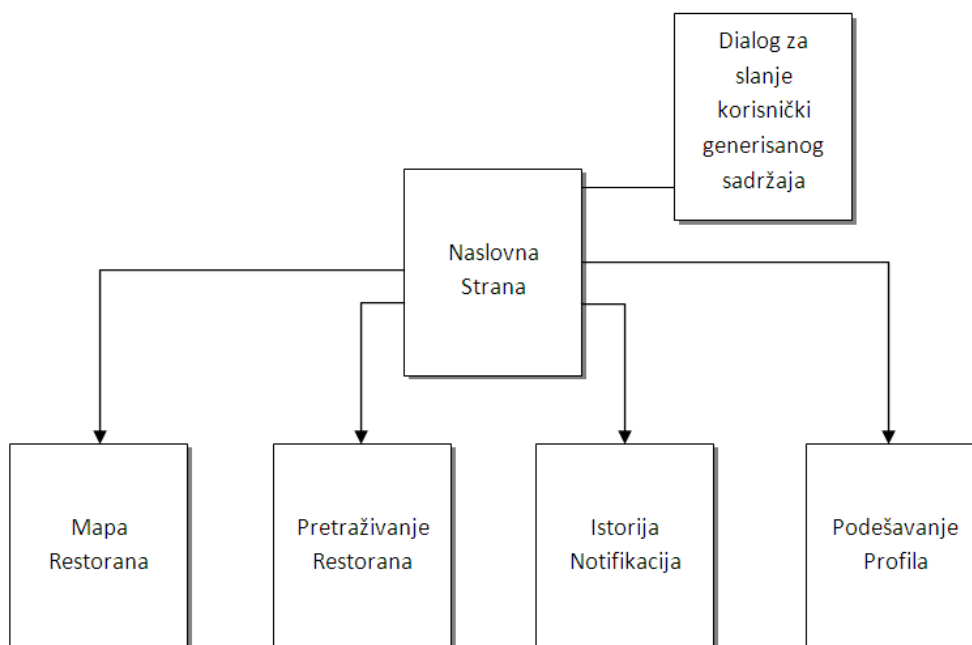
Za pokretanje klijenta na serveru neophodno je instalirati NodeJS okruženje. Pokretanje se vrši sledećom komandom

- `nohup node push-server.js > output.out &`

Server ostaje aktivan u pozadini sve vreme „oslušujući“ dolazne konekcije i reagujući na događaje potpuno asinhrono.

## 4.7. Mobilna aplikacija

Kompletan prikaz mobilne aplikacije je dat u poglavlju 5.1. U ovom poglavlju će biti opisane samo najbitnije funkcionalnosti kroz prikaz izvornog koda. Na slici 29 predstavljena je mapa mobilne aplikacije:



Slika 29. Mapa mobilne aplikacije

Mobilna aplikacija je implemenirana u jQueryMobile aplikacionom okviru. Korišćenjem gotovih komponenti koje imitiraju native kontrole operativnog sistema postignut je nativni izgled aplikacije. Tako na primer naslovna strana sadrži komponentu koja imitira list komponentu kod nativnih aplikacija. Lista sadrži linkove ka drugim stranama. Na slici 30 prikazan je izvorni kod strane:

final-thesis / application / app / views / pages / homepage.ctp [Edit this file](#)

```
100644 | 18 lines (12 sloc) | 0.573 kb raw blame history
1 <div class="page" data-role="page" data-theme="c" style="width:100%; height:100%">
2
3     <?php echo $this->element('header'); ?>
4
5     <div data-role="content">
6
7         <ul data-role="listview" data-inset="true" data-theme="c" data-dividertheme="b">
8             <li data-role="list-divider">Meni</li>
9             <li><a href="pins_map" rel="external">Mapa restorana</a></li>
10            <li><a href="pins/filter_pins_form">Pretraži restorane</a></li>
11            <li><a href="pages/notifications">Obaveštenja</a></li>
12            <li><a href="pages/profile_settings">Podešavanja profila</a></li>
13        </ul>
14
15    </div>
16
17 </div>
18
```

Slika 30. Isečak koda – izvorni kod naslovne strane

U zaglavlju strane se nalazi dugme koje otvara dijalog za slanje tekstualnog sadržaja na server kome se pridružuje trenutna lokacija korisnika. Kako sve strane dele isto zaglavlje, dijalog je dostupan na svim stranama (slika 31).

final-thesis / application / app / views / elements / header.ctp [Edit this file](#)

100644 | 10 lines (7 sloc) | 0.304 kb [raw](#) [blame](#) [history](#)

```
1 <div data-role="header">
2     <?php if($this->params['url']['url'] != '/'): ?>
3         <a href="/" data-role="button" data-icon="home" rel="external">Početna</a>
4     <?php endif; ?>
5
6     <h1>Vodič kroz restorane</h1>
7
8     <a href="/pages/send_message_dialog" data-rel="dialog" data-icon="plus">Pošalji poruku</a>
9
10 </div>
```

Slika 31. Isečak koda – izvorni kod zaglavlja strane

Strana sa mapom implemenira Google Maps v3 JavaScript komponentu na koju je dodat sloj tačaka od interesa koje se učitavaju korišćenjem APIa opisanog u prethodnom poglavlju. Funkcija za interakciju sa APIem i učitavanje tačaka od interesa data je u isečku koda na slici 32.

```

33 var updateMarkers = function() {
34     var bounds = map.getBounds();
35     var southWest = bounds.getSouthWest();
36     var northEast = bounds.getNorthEast();
37     var url = 'pins/list_markers'
38         + '?sw_lat=' + southWest.lat()
39         + '&sw_lon=' + southWest.lng()
40         + '&ne_lat=' + northEast.lat()
41         + '&ne_lon=' + northEast.lng();
42
43     var name = getParameterByName('name');
44     if(name != "") {
45         url += '&name=' + name;
46     }
47     var category = getParameterByName('category');
48     if(category != "") {
49         url += '&category=' + category;
50     }
51     var area = getParameterByName('area');
52     if(area != "") {
53         url += '&area=' + area;
54     }
55     var latitude = getParameterByName('latitude');
56     if(latitude != "") {
57         url += '&latitude=' + latitude;
58     }
59     var longitude = getParameterByName('longitude');
60     if(longitude != "") {
61         url += '&longitude=' + longitude;
62     }
63
64     $.getJSON(url, function(data) {
65         $.each(data.markers, function(index, data) {
66             pin = data.marker;
67             if($.inArray(pin.id, markers) != -1) {
68                 return;
69             }
70             markers.push(pin.id);
71
72             var latLng = new google.maps.LatLng(
73                 parseFloat(pin.latitude),
74                 parseFloat(pin.longitude));
75
76             if('pin' == pin.type) {
77                 var category_ids = pin.category_ids.split('.');
78                 var category_id = parseInt(category_ids[category_ids.length-1]);
79                 var image = new google.maps.MarkerImage(get_marker_icon(category_id));
80
81                 var marker = new google.maps.Marker({
82                     position : latLng,
83                     map : map,
84                     flat : true,
85                     cursor : 'pointer',
86                     pin_id : pin.id,
87                     title : pin.name,
88                     icon: image
89                 });
90
91                 var contentString = '<div id="content">'+
92                     '<b>' + pin.name + '</b>'+
93                     '<div id="bodyContent">'+
94                         '<p><b>kategorija:</b>' + pin.category + '</p>'+
95                         '<p><b>Adresa:</b>' + pin.address + '</p>'+
96                         '<p><b>Telefon:</b>' + pin.telephone + '</p>'+
97                         '<p><b>Radno vreme:</b>' + pin.work_hours + '</p>'+
98                     '</div>'+
99                     '</div>';
100
101             }
102
103             if('message' == pin.type) {
104                 var image = new google.maps.MarkerImage('/img/markers/comment.png');
105
106                 var marker = new google.maps.Marker({
107                     position : latLng,
108                     map : map,
109                     flat : true,
110                     cursor : 'pointer',
111                     pin_id : pin.id,
112                     title : 'Poruka',
113                     icon: image
114                 });
115
116                 var contentString = '<div id="content">'+
117                     '<b>Poruka</b>'+
118                     '<div id="bodyContent">' + pin.text + '</div>'+
119                     '</div>';
120
121             }
122
123             google.maps.event.addListener(marker, 'click', function() {
124                 if (infowindow) infowindow.close();
125                 infowindow = new google.maps.InfoWindow({
126                     content: contentString,
127                     maxWidth: 480,
128                     maxHeight: 600
129                 });
130                 infowindow.open(map, marker);
131             });
132         });
133     });
134 }

```

Slika 32. Isečak koda – Funkcija za interakciju za APIem i učitavanje tačaka od interesa

Funkcija uzima u obzir trenutno vidljivu površinu ekrana i na osnovu nje zateva od servera podatke o tačkama od interesa u određenoj oblasti definisanoj geografskim koordinatama vidljive površi.

Podešavanja profila korisnika se pamte lokalno korišćenjem LocalStorage APIa implementiranog od strane internet pregledača. Pre korišćenja LocalStorage potrebno je proveriti da li je on podržan od strane internet pregledača (slika 33).

```
1 function supportsLocalStorage() {  
2     try {  
3         return 'localStorage' in window && window['localStorage'] !== null;  
4     } catch (e) {  
5         return false;  
6     }  
7 }
```

Slika 33. Isečak koda – funkcija za proveru podržanosti LocalStorage API-a

Ukoliko jeste vrednostima se lako pristupa i lako se snimaju korišćenjem localStorage objekta (slika 34). Format LocalStorage baze podataka je key-value tipa i u nju se mogu smeštati jednostavne tekstualne vrednosti. Ne postoji potreba za definisanjem šeme baze podataka niti je potrebno prethodno kreiranje iste. Dovoljno je da internet pregledač podržava API.

```
73     $('#profile-settings-page').bind("pageshow", function() {  
74         loadFormData();  
75         var profileSettingsForm = $('#profile-settings-form');  
76         profileSettingsForm.submit(function() {  
77             if(!supportsLocalStorage()) {  
78                 apprise('Nije moguće snimiti podešavanja jer uređaj ne podržava lokalnu memoriju');  
79                 return;  
80             }  
81             var notificationEnabled = $('#notification').val();  
82             var notificationArea = $('#notification-area').val();  
83             var notificationCategories = new Array();  
84             $.each($('input[name=category[]]:checked'), function() {  
85                 notificationCategories.push($(this).val());  
86             });  
87             localStorage["notificationEnabled"] = notificationEnabled;  
88             localStorage["notificationArea"] = notificationArea;  
89             localStorage["notificationCategories"] = notificationCategories;  
90             apprise('Podešavanja su snimljena');  
91         });  
92     });
```

Slika 34. Isečak koda – funkcija za lokalno snimanje korisničkog profila

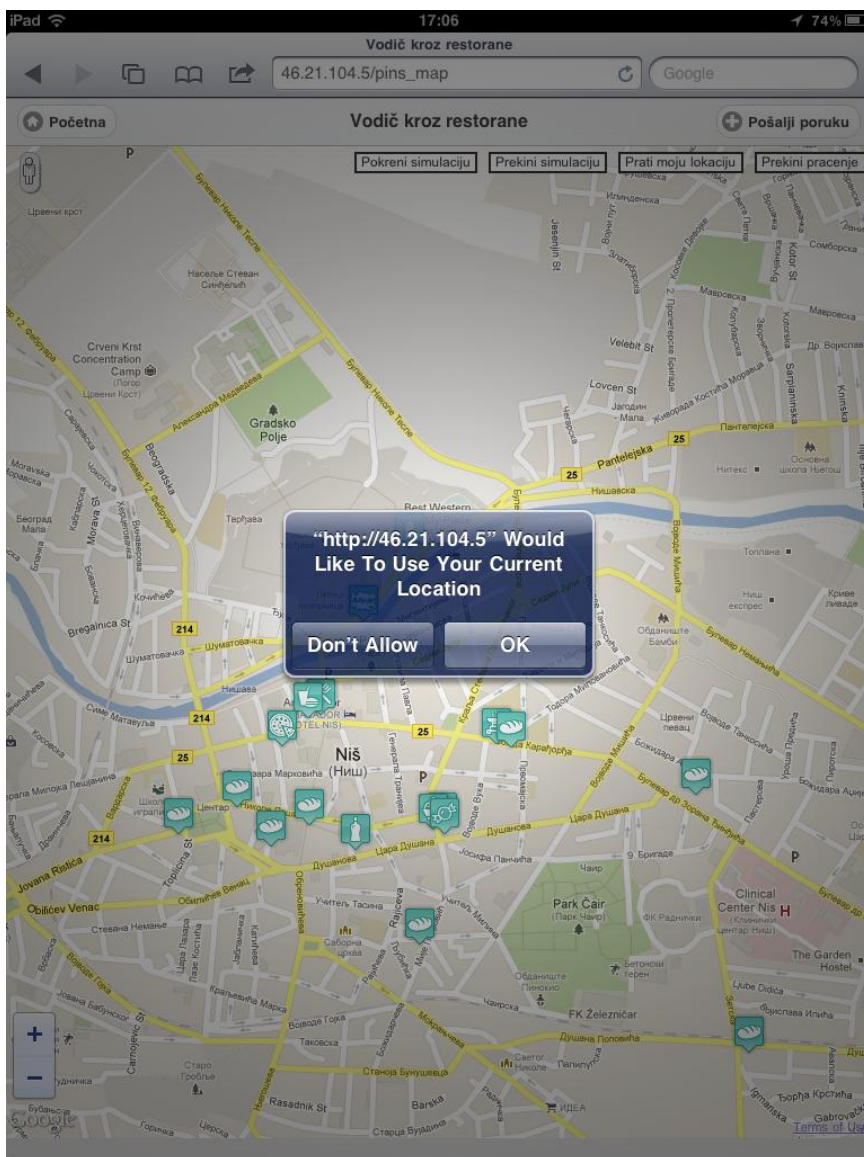
Slanje poruka na server implementirano je korišćenjem APIa objašnjenog u prethodnom poglavlju. Nakon što korisnik unese tekst poruke i pritisne dugme "Pošalji" određuje se trenutna lokacija korisnika korišćenjem geolokacijskog APIa implementiranog u internet pregledaču (slika 35).

```
77  
78 var updateLocation = function() {  
79     geo_position_js.getCurrentPosition(showPosition, console.log("Couldn't get location"));  
80 }  
81  
82 var showPosition = function(p) {  
83     latitude = p.coords.latitude;  
84     longitude = p.coords.longitude;  
85 }  
86
```

Slika 35. Isečak koda – funkcija za određivanje korisnikove trenutne lokacije

Funkcija updateLocation poziva geolokacijski API koji vraća GPS koordinate korisnika. Geolokacijski API može da odredi trenutnu lokaciju korisnika i ako GPS senzor nije dostupan korišćenjem IP adrese korisnika ili mac adrese WIFI pristupne tačke na koju je korisnik konektovan. Da bi se dobila

geografska lokacija korisnika najpre je potrebno da on to eksplicitno odobri kada ga internet pregledač bude pitao da li želi da podeli svoju trenutnu lokaciju (slika 36).



Slika 36. Dijalog za potvrdu korišćenja korisnikove trenutne geografske lokacije

Nakon dobijanja lokacije korisnika poruka se šalje na server HTTP pozivom ka prethodno opisanom API-u za smeštanje poruka (slika 37). U POST zaglavlju poziva se smeštaju koordinate geografske lokacije korisnika i tekst generisane poruke.

```

34 var saveGeoMessageOnServer = function() {
35     if(false == latitude || false == longitude) {
36         apprise('Lokacija nije dostupna. Pokušajte ponovo kroz nekoliko sekundi');
37         return false;
38     }
39     var text = $('#messageTextarea').val();
40     if('' == text) {
41         apprise('Morate uneti tekst poruke');
42         return false;
43     }
44     var data = {
45         latitude: latitude,
46         longitude: longitude,
47         message: text
48     };
49     $.post('/messages/save_message.json', data, function(data) {
50         $('#ui-dialog').dialog('close');
51     });
52 }

```

Slika 37. Isečak koda – funkcija za snimanje korisnički generisanog sadržaja na server

Mobilni klijenti primaju notifikacije o nastalim događajima u realnom vremenu preko WebSocket konekcije koju uspostavljaju sa serverskom aplikacijom za slanje notifikacija. WebSocket API je implementiran od strane mobilnog internet pregledača i mora biti podržan da bi mobilni klijenti mogli da primaju push notifikacije. Trenutno WebSocket API podržavaju jedino iOS uređaji sa verzijom operativnog sistema 4.2 ili novijom. U bliskoj budućnosti očekuje se podrška za WebSocket i ostalih proizvođača mobilnih operativnih sistema u prvom redu Android i Windows Phone 7.

```

5  $(document).ready(function(){
6      //Start tracking users location
7      if (geo_position_js.init()) {
8          geolocationTask = setInterval(updateLocation, 10000);
9      }
10     if ("WebSocket" in window) {
11         host = "46.21.104.5:7777";
12         conn = new WebSocket("ws://" + host + "/");
13         conn.onmessage = function(evt) {
14             postMessage(evt.data);
15         };
16
17         conn.onerror = function() {
18             console.log('error');
19         };
20
21         conn.onclose = function() {
22             console.log('close');
23             conn = false;
24         };
25
26         conn.onopen = function() {
27             console.log('opened');
28             //alert("You are connected");
29         };
30     }
31 });
32
33

```

Slika 38. Isečak koda – funkcija za uspostavljanje perzistentne WebSocket konekcije

Klijent otvara perzistentnu konekciju sa serverom na soku 7777 (slika 38) i definiše funkciju koja će odgovoriti na događaj svaki put kada bude primljena poruka sa servera. Server šalje podatke o tekstu poruke, kategoriji restorana za koju je događaj definisan i geografskoj lokaciju poruke u JSON enkodiranom obliku.

Funkcija koja prima podatke o generisanom događaju proverava da li obaveštenje odgovara definisanom profilu korisnika i ukoliko odgovara prikazuje je na ekranu uređaja (slika 39). Sve primljene poruke se smeštaju u LocalStorage internet pregledača.

Provera profila korisnika se izvodi u sledećim koracima:



- Provera postojanja lokalnog profila korisnika
- Provera da li su obaveštenja dozvoljena definisana lokalnim profilom
- Provera da li je geografska lokacija obaveštenja unutar maksimalne dozvoljene udaljenosti od trenutne lokacije korisnika definisane lokalnim profilom
- Ukoliko je poruka od strane restorana proverava se da li je kategorija restorana definisana u lokalnom profilu korisnika

```

87 // Returns true if notification location is within users defined area range
88 var notificationCheck = function(lat, lon, categories) {
89     if(false == supportsLocalStorage()) {
90         return true;
91     }
92
93     if(undefined == localStorage.notificationEnabled || !localStorage.notificationEnabled) {
94         return false;
95     }
96
97     // Check notification distance
98     if(undefined != localStorage.notificationArea && false != latitude && false != longitude) {
99         var notificationArea = localStorage.notificationArea;
100         var areaDistance = distVincenty(latitude, longitude, lat, lon);
101         if(areaDistance > notificationArea) {
102             return false;
103         }
104     }
105
106     // Check notification categories
107     if(undefined == categories) {
108         return true;
109     }
110     var notificationCategories = localStorage.notificationCategories.split(',');
111     var inArray = false;
112     $.each(categories, function(index, value) {
113         if($.inArray('"' + value + '"', notificationCategories)) {
114             inArray = true;
115         }
116     });
117     return inArray;
118 }

```

Slika 39. Isečak koda – funkcija za proveru da li obaveštenje odgovara definisanom profilu korisnika

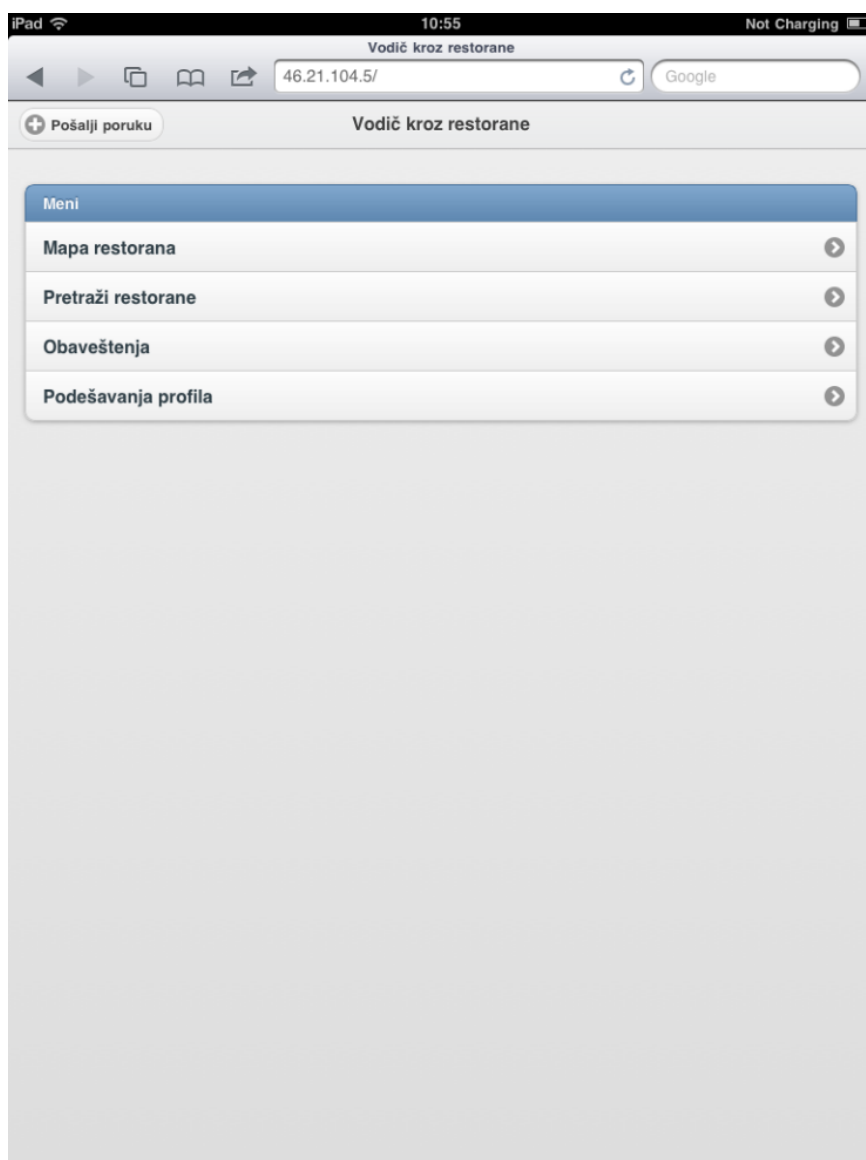
Tek ukoliko su zadovoljeni svi definisani uslovi korisnik dobija vidljivu notifikaciju na telefonu. Lokaciju notifikacije može pregledati i na mapi što će biti detaljno demonstrirano u narednom poglavlju.



## 5. Funkcionalne karakteristike i evaluacija aplikacije

### 5.1.Prikaz glavnih funkcionalnosti aplikacije

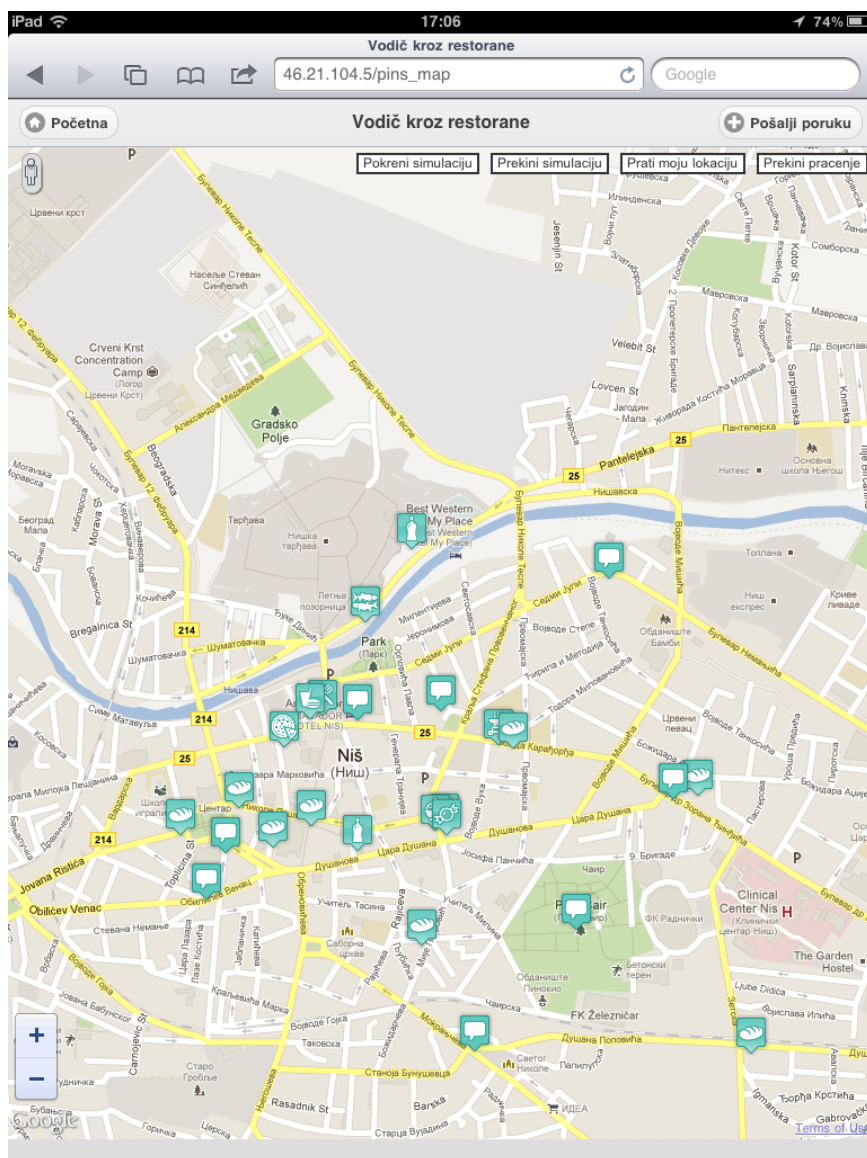
Na slici 29 data je mapa sajta mobile aplikacije. Na glavnoj strani se nalazi lista sa linkovima za navigaciju ka drugim stranama aplikacije (slika 40).



Slika 40. Izgled glavne strane mobilne aplikacije

Sve strane imaju istu strukturu sa zaglavljem na vrhu i sadržajem koji se širi u zavisnosti od visine ekrana. Zaglavlje sadrži naslov aplikacije, dugme za povratak na naslovnu stranu i dugme za pokretanje dijaloga za dodavanje tekstualnog sadržaja tagovanog za trenutnu geografsku lokaciju korisnika.






Strana sa mapom restorana prikazuje trenutnu geografsku lokaciju korisnika (slika 41).



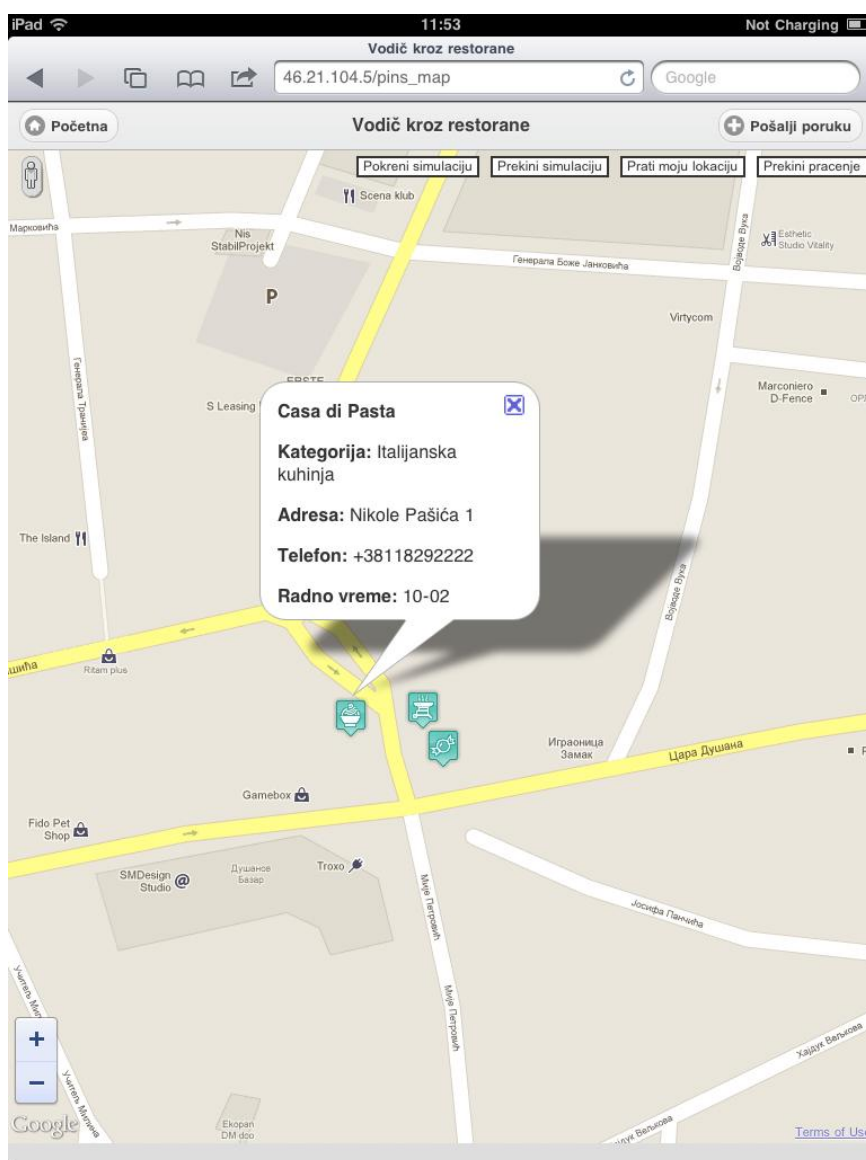
Slika 41. Izgled strane sa mapom restorana

Na mapi su restorani označeni različitim slikama u zavisnosti od vrste kuhinje radi lakšeg prepoznavanja. Korisnički generisani sadržaj takođe ima posebnu sliku.

Vrsta tačke	Slika
Restoran	
Riblji restoran	
Italijanska kuhinja	
Kineska kuhinja	
Palačinkarnica	
Pekara	

Ćevapdžinica	
Kafana	
Brza hrana	
Picerija	
Korisnički generisan sadržaj	

Dodirom na sličicu restorana otvara se „balončić“ sa informacijama o restoranu (slika 42).



Slika 42. Prikaz informacija o restoranu

Na mapi se nalaze i akcije za pokretanje / prekidanje praćenja i pokretanje / prekidanje simulacije kretanja.

Aktiviranjem praćenja mapa se ažurira u skladu sa korisnikovom trenutnom lokacijom i na mapi se prati njegovo kretanje. Prilikom pokretanja praćenja neophodno je da korisnik eksplicitno da dozvolu aplikaciji za korišćenje njegove lokacije iz sigurnosnih razloga kao što je to ranije objašnjeno (slika 36).

Simulacija praćenja služi samo u demonstrativne svrhe i njenim pokretanjem mapa prati unapred definisano kretanje.

Na strani za pretragu restorana se u formi mogu definisati parametri za filtriranje liste restorana. Pretraga je moguća po imenu restorana, vrsti kuhinje i maksimalnoj udaljenosti restorana od korisnikove trenutne lokacije (slika 43). Pronađene lokacije se prikazuju na mapi.

iPad 17:09 73%

Vodič kroz restorane

46.21.104.5/#pins/filter\_pins\_form Google

Početna Vodič kroz restorane Pošalji poruku

### Pretraživanje liste restorana

Pretraživanje je moguće prema imenu restoran, vrsti kuhinje i maksimalnoj udaljenosti restorana od trenutne lokacije. Pronađeni rezultati će biti prikazane na mapi.

Ime restorana:

Maksimalna udaljenost od objekta (km):

Kategorija restorana:

- ☐ Restoran
- ☐ Riblji restoran
- ☐ Italijanska kuhinja
- ☐ Kineska kuhinja
- ☐ Palačinkarnica
- ☒ Pekara
- ☐ Čevapdžinica
- ☐ Kafana
- ☐ Brza hrana
- ☐ Picerija

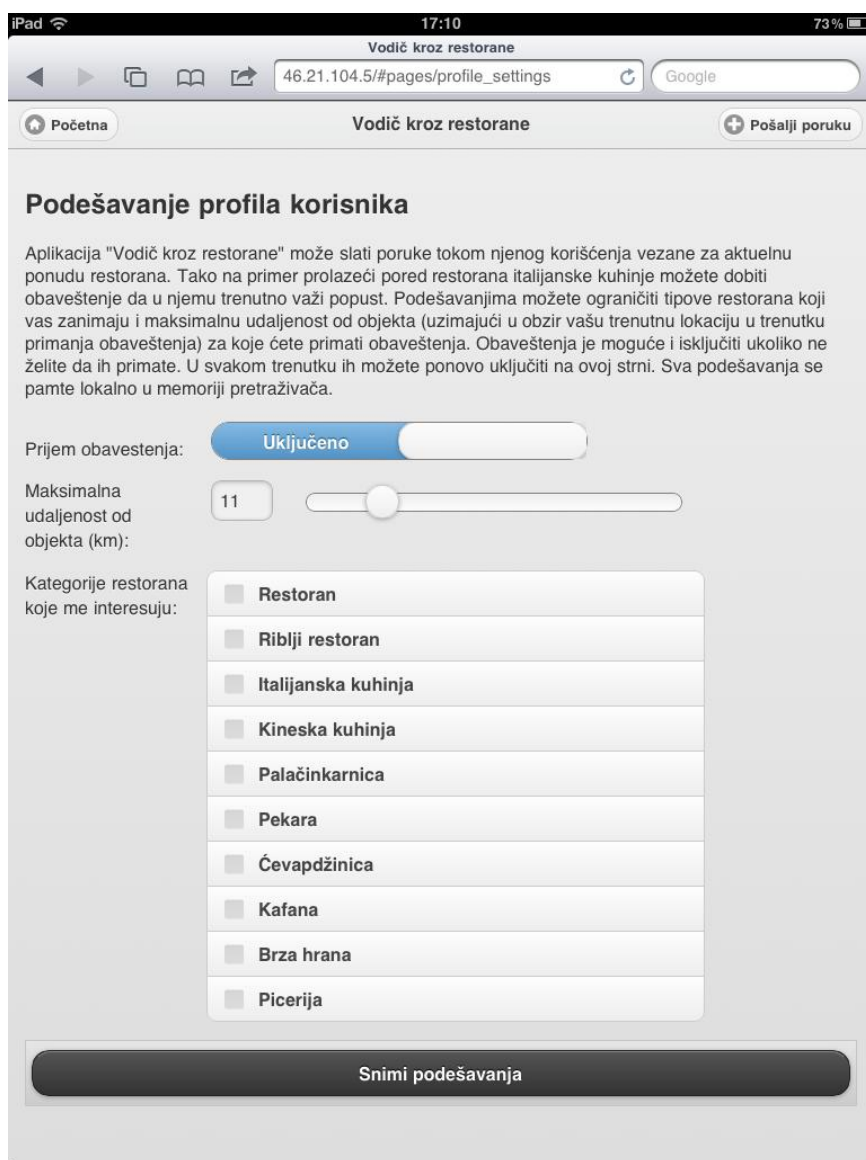
Pronadji

Slika 43. Izgled strane za unos kriterijuma za pretragu restorana

Aplikacija "Vodič kroz restorane" može slati poruke tokom njenog korišćenja vezane za aktuelnu ponudu restorana ili to može biti korisnički definisan sadržaj u neposrednoj blizini. Tako na primer prolazeći pored restorana italijanske kuhinje korisnik može dobiti obaveštenje da u njemu trenutno

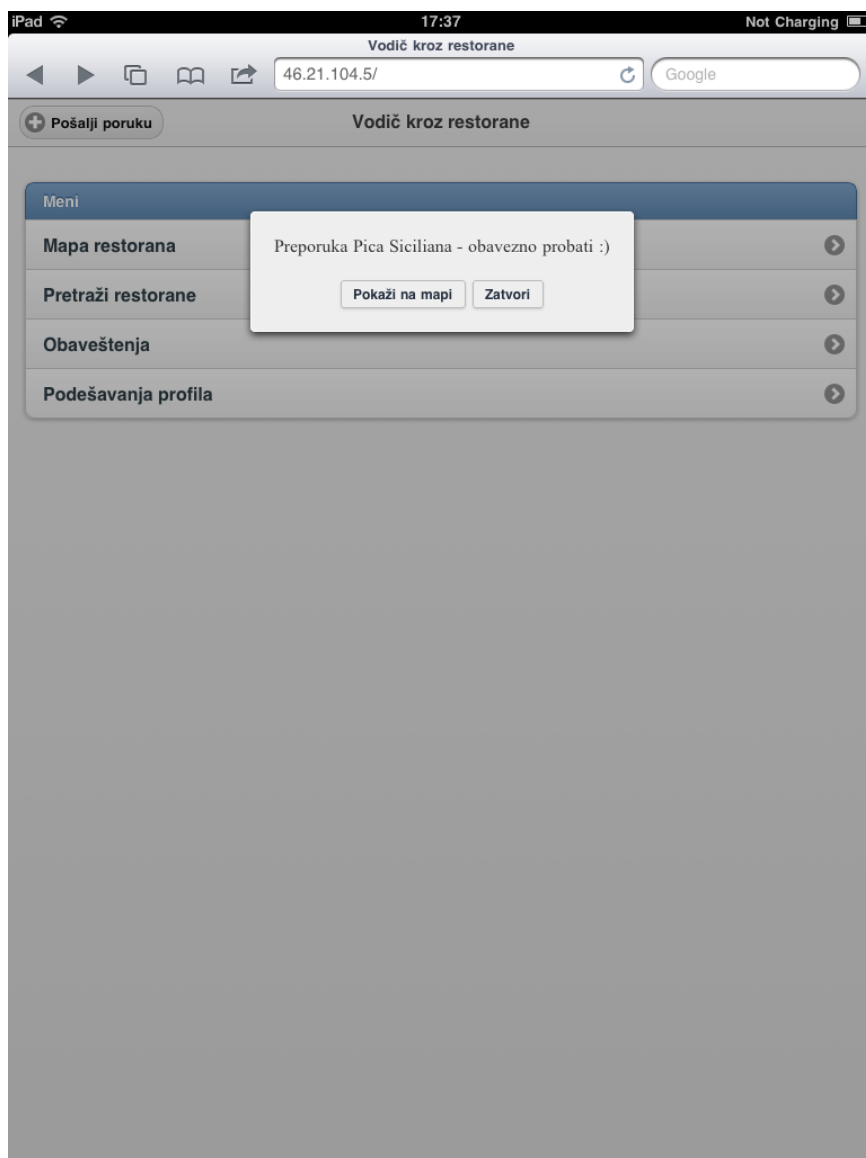
važi popust. Prijem obaveštenja zavisi od trenutnog profila korisnika kojim se može definisati na strani za podešavanje profila.

Podešavanjima može se ograničiti tip restorana za koji korisnik želi da prima obaveštenja i maksimalnu udaljenost objekta restorana od trenutne lokacije korisnika (slika 44). Obaveštenja je moguće i isključiti. Sva podešavanja se pamte lokalno u memoriji internet pregledača pa je obezbeđena potpuna anonimnost i privatnost korisnika.



Slika 44. Izgled strane za podešavanje lokalnog profila korisnika

Dugme u zaglavlju strane „Pošalji poruku“ otvara dijalog za slanje geotagovanog tekstualnog sadržaja. Nakon snimanja, poruka se prosleđuje serveru gde se permanentno smešta u bazu podataka. Prijem poruke aktivira i slanje obaveštenja svim mobilnim klijentima o generisanom događaju.

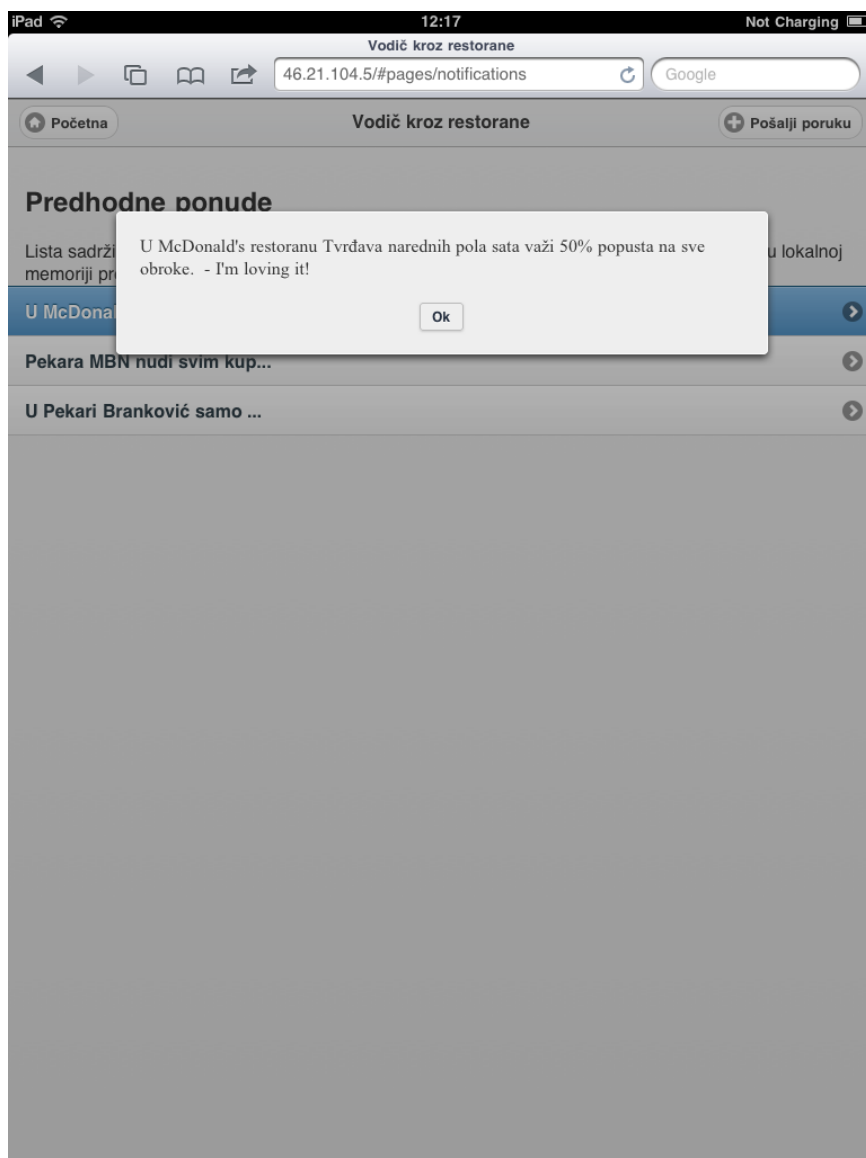


Slika 45. Izgled pristiglog obaveštenja u iskaćućem prozoru

Pristigla obaveštenja se prikazuju korisniku u vidu iskaćućeg prozora nakon čega ih on može, ukoliko želi, videti na mapi koja se pozicionira na tačnu geografsku lokaciju poruke (slika 45).

Poruke od servera se šalju svim trenutno aktivnim klijentima. Da li će se poruka prikazati ili ne odlučuje se tek na strani klijenta u zavisnosti od definisanog profila. Sve primljene poruke, bez obzira da li odgovaraju profilu korisnika ili ne se pamte u memoriji internet pregledača. Kasnije korisnik može pregledati listu svih pristiglih poruka na strani Obaveštenja (slika 46).





Slika 46. Izgled strane sa listom prethodno pristiglih obaveštenja

## 5.2. Uporedni prikaz izvršenja aplikacije na različitim platformama

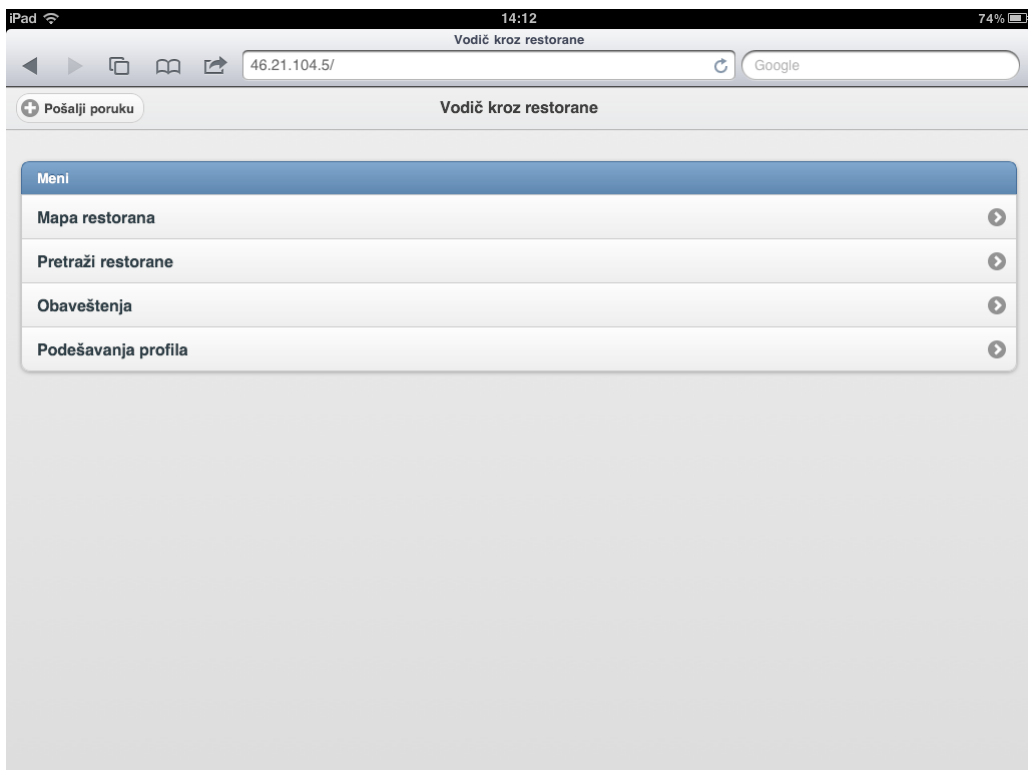
Bitan segment aplikacije predstavlja nezavisnost od platforme i veličine ekrana uređaja sa koga se pristupa aplikaciji. Za testiranje i validaciju platforme nezavisnosti koristio sam uređaje koji su mi bili dostupni a to su HTC Desire mobilni telefon sa Android OS v 2.2 verzijom operativnog sistema i dijagonalom ekrana 3.7 inča rezolucije 800x480 piksela i Apple iPad tablet računar sa iOS 4.3 verzijom operativnog sistema i dijagonalom ekrana 9.7 inča rezolucije 1024x768 piksela.

Korisnički interfejs aplikacije je prilagodljiv veličini i rezoluciji ekrana pa je obezbeđena jednaka preglednost. Na slici 47 je dat izgled aplikacije na HTC Desire telefonu.

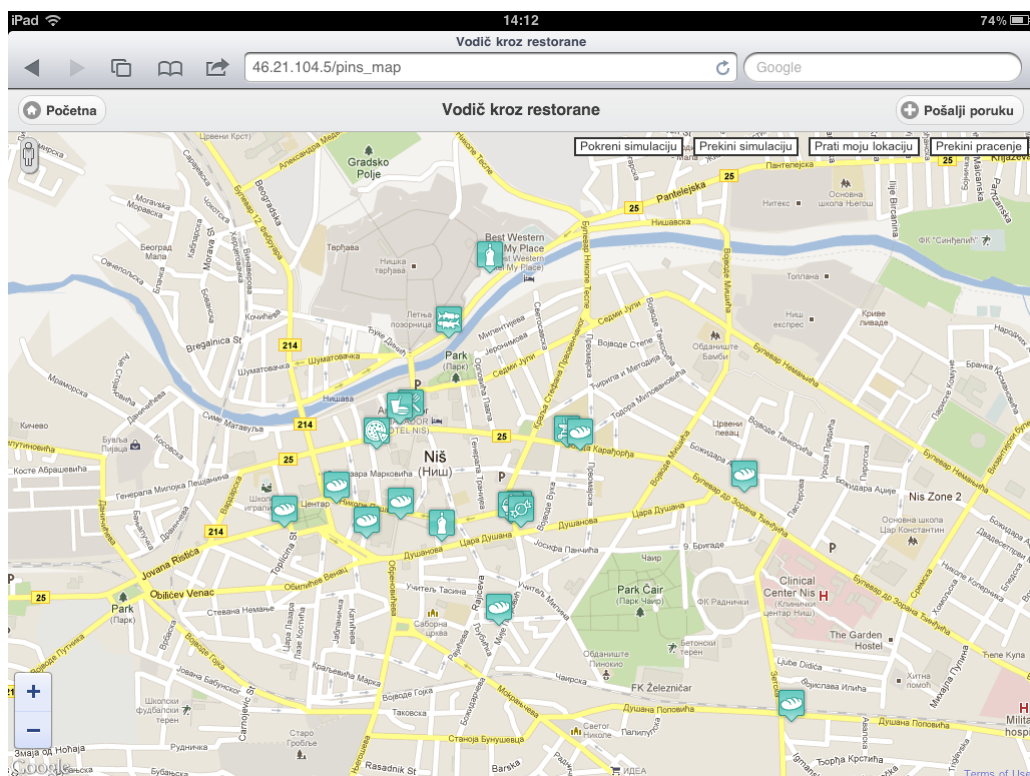


Slika 47. Izgled aplikacije na telefonu HTC Desire

Korisnički interfejs je prilagodljiv i orijentaciji uređaja (slika 48 i 49)



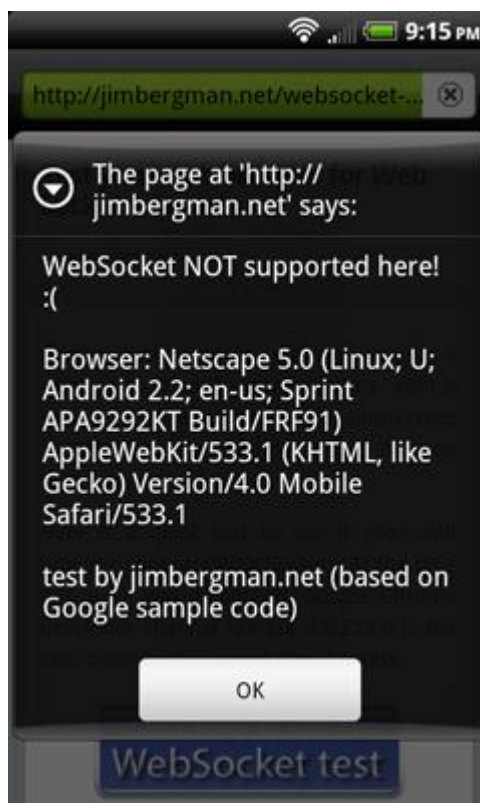
Slika 48. Izgled naslovne strane prilagođen orijentaciji uređaja



Slika 49. Izgled strane sa mapom prilagođen orijentaciji uređaja

Platformska nezavisnost aplikacije obezbeđuje rad i na uređajima koji nemaju podržane sve neophodne HTML5 API-e potrebne za normalno funkcionisanje aplikacije. Ukoliko određeni API nije dostupan aplikacija će nastaviti sa izvršavanjem ali će korisniku biti prikazana poruka da određena funkcionalnost nije dostupna iz tehničkih razloga.

Prilikom validacionog testiranja aplikacije primetio sam da prijem notifikacija nije moguć na prethodno pomenutom HTC Desire telefonu. Razlog je jer verzija Android OS 2.2 ne podržava implementaciju WebSockets API-a na koji se push notifikacije oslanjaju a što je potvrđeno WebSocket testom [12] (slika 50).



Slika 50. Rezultat web socket testa na Android 2.2 operativnom sistemu

Ostale funkcionalnosti aplikacije funkcionišu nesmetano čime je obezbeđen mehanizam za rad bez potrebe za ograničavanjem minimalnog skupa funkcionalnosti koji uređaj treba da zadovolji.

### 5.3. Razmatranje karakteristika u odnosu na nativnu aplikaciju

Korišćenjem HTML5 APIa funkcionalnost aplikacije sa približila mogućnostima nativnih aplikacija. Naročito su za to zaslužni Geolokacijski, LocalStorage i WebSocket API jer su pre njih ove funkcionalnosti bile dostupne jedino nativnim aplikacijama.

I pored unapređenih funkcionalnosti web aplikacije i dalje zaostaju za nativnim aplikacijama u pogledu performansi i izgledu korisničkog interfejsa. Nativne aplikacije još uvek izgledaju lepše zbog velike količine UI efekata pa se u isto vreme stiče utisak boljeg odziva na korisnikove akcije.

Sa druge strane mobilne aplikacije smanjuju potrebno vreme razvoja jer rade nezavisno od platforme i programskog jezika podržanog od strane uređaja pa je konačna cena aplikacije višestruko niža od nativnih.

Na primeru aplikacije za pretragu restorana i razmenu korisnički generisanog sadržaja nativna aplikacija bi prednjačila u mogućnosti da push notifikacije budu primljene čak i kada aplikacija trenutno nije aktivna. Što se tiče korisničkog interfejsa i performansi aplikacije HTML5 rešenje daje zadovoljavajuće rezultate koji mogu da pariraju nativnoj aplikaciji.

## 6. Zaključak

U okviru diplomskog rada teorijski i praktično je obrađena tema razvoja RIA mobilnih internet aplikacija namenjenih prenosivim uređajima. U drugom poglavlju dat je kompletan osvrt na istoriju mobilne internet tehnologije koja je prethodila razvoju savremenih tehnologija. U trećem poglavlju su obrađene sve najznačajnije mobilne web tehnologije i standardi koji se koriste u razvoju takozvanih mobilnih Web 2.0 aplikacija.

U praktičnom delu je razvijena mobilna aplikacija za razmenu korisnički generisanog sadržaja radi demonstracije mogućnosti prethodno opisanih tehnologija i standarda. Aplikacija omogućava rad nezavistan od platforme i operativnog sistema uređaja i koristi hardverske i softverske resurse koji je po funkcionalnosti približavaju nativnim mobilnim aplikacijama.

Ovakva vrsta mobilnih internet aplikacija je tradicionalno nepoznata korisnicima mobilnih uređaja jer koristi najnovije tehnologije koje tek u trenutnu pisanja nalaze svoju primenu u razvoju komercijalnih aplikacija. Ipak u bliskoj budućnosti očekuje se da ove tehnologije dobiju širu podršku od strane proizvođača mobilnih internet pregledača što bi dovelo po mišljenju autora do preuzimanja vodeće pozicije mobilnih internet aplikacija u pogledu korišćenja ispred nativnih aplikacija.

# Literatura

---

1. United Nations Cyberschool Paper  
<http://www.un.org/cyberschoolbus/briefing/technology/tech.pdf>
2. Internet World Statistics <http://internetworldstats.com/stats.htm>
3. Mobile Web Wikipedia članak [http://en.wikipedia.org/wiki/Mobile\\_Web](http://en.wikipedia.org/wiki/Mobile_Web)
4. WAP Wikipedia članak [http://en.wikipedia.org/wiki/Wireless\\_Application\\_Protocol](http://en.wikipedia.org/wiki/Wireless_Application_Protocol)
5. XHTML Mobile Profile Specifikacija  
<http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf>
6. Istraživnje Gartnera <http://www.gartner.com/it/page.jsp?id=1543014>
7. The comScore 2010 Mobile Year in Review
8. Cisco Visual Networking Index (VNI) Global Mobile Data Traffic Forecast, 2010–2015
9. Gail Rahn Frederick with Rajesh Lal, „Beginning Smartphone Web Development Building JavaScript, CSS, HTML and Ajax-based Applications for iPhone, Android, Palm Pre, BlackBerry, Windows Mobile, and Nokia S60“, Apress, 2010.
10. WURFL <http://wurfl.sourceforge.net/>
11. Device Atlas <http://deviceatlas.com/>
12. WebSocket API test <http://jimbergman.net/android-2-3-websocket-support/>