Izrada multiplatformskih aplikacija korištenjam programske zbirke Cordova

Dizdar, Jurica

Undergraduate thesis / Završni rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University North / Sveučilište Sjever**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:122:699841

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**



Repository / Repozitorij:

University North Digital Repository







Sveučilište
Sjever

Završni rad br. 445/MM/2015

Izrada multiplatformskih aplikacija korištenjem programske zbirke Cordova

Jurica Dizdar, 4680/601

Varaždin, rujan 2015. godine



Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 445/MM/2015

Izrada multiplatformskih aplikacija korištenjem programske zbirke Cordova

Student

Jurica Dizdar, 4680/601

Mentor

Vladimir Stanisavljević, mr.sc.

Varaždin, rujan 2015. godine

Sažetak

Živimo u modernom vremenu u kojem su mobilne aplikacije naša svakodnevice. Statistički gledano, gotovo da ne postoji osoba u zapadnom društvu koja ne posjeduje mobitel odnosno pametni telefon i svakim danom se taj broj povećava. Kao što sam istaknuo, mobiteli su naša svakodnevica a, broj mobilnih aplikacija koje koristimo se iz dana u dan povećava. Mobilne aplikacije dijelimo na nativne i hibridne. Nativne koje se razvijaju na uobičajenim programskim jezicima poput C++-a i Jave te hibridne čiji se razvoj temelji na web tehnologijama(HTML5, CSS, Javascript). Uz svoje prednosti i nedostatke, hibridne aplikacije polagano zauzimaju sve veći udio na tržištu a, po nekim statističkim pokazateljima(Yahoo) u narednim godinama, broj hibridnih aplikacija na tržištu će biti veći od nativnih aplikacija. S tim u umu ovaj rad ima namjeru prikazati postupak izrade višeplatformske aplikacije za studente Sveučilišta Sjever u jednom od multiplatformskih integriranih razvojnih okruženja (engl. IDE). Stoga će u ovom radu biti opisani programski okviri za razvoj hibridnih mobilnih aplikacija. Apache Cordove i jQuery Mobile su programski okviri na kojima se temelji ovaj rad i čije će funkcionalnosti biti opisane kako kroz teorijski tako i kroz praktični dio. Teorijski dio rada je podijeljen u pet poglavlja kroz koje će biti opisani navedeni radni okviri uz naglasak na programskoj zbirci Cordova, tehnologijama koje se koriste za izradu i mogućnostima koje ona pruža za razvoj višeplatformskih mobilnih aplikacija. Isto tako, u radu je opisana i usporedba Cordove s konkurencijom odnosno usporedba sa sličnim hibrdinim razvojnim platformama koje se mogu naći na tržištu. Osim Cordove, temelj ovog rada je i jQuery Mobile radni okvir. jQM je radni okvir za oblikovanje i izradu korisničkog sučelja koji se koristi kako bi se poboljšalo mobilno korisničko iskustvo. Stoga će u radu biti opisani pojedini jQuery Mobile elementi, navigacija stranica te funkcionalnosti koje se mogu ostvariti implementiranjem radnog okvira. jQuery je okosnica jQuery Mobile, stoga će rad obuvatiti i kratki opis jQuerya te HTML5 elemenata koji se koriste za rad sa navedenim radnim okvirom. U praktičnom dijelu, izradit će se aplikacije ”Sjeveroljubac” koja će biti namijenjena za sve studente Sveučilišta Sjever. Aplikacija koristi standardni izgled i organizaciju aplikacije za određenu platformu. Osim mobilnog izgleda, aplikacija će koristiti nativne mogućnosti mobilnih uređaja kao što je navigacija za prikazivanje mapa, prikazivati će podatke koji će biti pribavljeni iz udaljenih izvora kao što su RSS i druge web stranice te će koristiti lokalnu pohranu za pohranjivanje određenih podataka. Osim klijentske strane, aplikacija će se pomoću Node.js spajati sa Uninovim serverom kako bi se ostvarila autentikaciija studenata pomoću LDAP protokola.

Ključne riječi:

Cordova, jQuery, jQuery Mobile, HTML5, CSS, Javascript, Ajax, LDAP, JSON, XML, mobilne platforme, Command-Line

Popis korištenih kratica i akronima

HTML - opisni internet jezik koji služi za izradu web stranica, služi za oblikovanje i povezivanje internet sadržaja na internet stranicama

jQuery Mobile - razvojni okvir prilagođen za ekrane osjetljive na dodir ili preciznije JavaScript biblioteka usmjeren na razvoj okvira koji je kompatibilan sa širokim spektrom pametnih telefona i tableta

JavaScript - objektni skriptni jezik osmišljen od strane Netscape-a, ovaj skriptni jezik se izvršava na web pregledniku na strani korisnika

API (application programing interface)- skup programskih pravila koji se programeri moraju držati da bi ostvarili željene rezultate kod programa

XML (exstensible markup language)- jezik za označavanje podatak stvoren s ciljem da bude čitljiv svima

XHR (XMLHttpRequest)- skup pravila koja su dostupna skriptnim jezicima poput Javascripta, koja se koriste za slanje HTTP ili HTTPS zahtjeva prema serveru i učitavanje povratnih podataka

Sadržaj

1. Uvod 1
2. Programska zbirka Cordova 3
   2.1. Komponente 3
   2.2. Dodaci i podržane mobilne platforme 5
   2.3. Programska zbirka PhoneGap 6
   2.4. Programska zbirka jQuery Mobile 8
   2.5. Izgradnja Cordova aplikacije 10
      2.5.1. Konfiguracija Cordove 12
      2.5.2. Web prikazi za različite platforme 13
      2.5.3. Node.js i npm 14
      2.5.4. Osnove rada s cordova naredbama 14
      2.5.5. Instalacija programske zbirke Cordova 15
   2.6. Zaključak Cordove 16
3. Alternative Cordovi 18
   3.1. Programski okvir Ionic 18
   3.2. Programski okvir Kendo UI 19
   3.3. Programski okvir Sencha Touch 20
   3.4. Programski okvir Appcelerator Titanium 21
   3.5. Zaključak alternativa Cordove 22
4. Rad s jQuery Mobileom 23
   4.1. Osnove programske zbirke jQuery 23
      4.1.1. Pristup DOM elementima 24
      4.1.2. Upravljanje događajima i funkcijama 25
      4.1.3. AJAX 27
   4.2. Osnove HTML 5 29
      4.2.1. Viewport meta oznaka 29
      4.2.2. Data-attributes 30
   4.3. Izrada osnovne aplikacije korištenjem jQM razvojnog okvira 31
      4.3.1. Stranice i alatne trake 32
      4.3.2. Struktura mobilnih stranica 32
      4.3.3. Vrste poveznica kod jQM-a 33
      4.3.4. Tranzicije stranica 34
      4.3.5. Dijaloški prozor 35
      4.3.6. Gumbi 36
      4.3.7. Liste 38
      4.3.8. Teme jQuery Mobilea 38
      4.3.9. Themeroller 39
      4.3.10. Zaključak jQM-a 40
5. Izrada aplikacije ”Sjeveroljubac” 41
   5.1. Struktura projekta 41
   5.2. Struktura aplikacije 42
   5.3. Višestranični predložak 43

5.4. Stranica index.html 43
5.4.1. Izrada dijaloškog prozor 45
5.5. Stranica student.html 46
5.6. Izrada stranice popisa odjela 47
5.6.1. Klik i each funkcije 48
5.6.2. Pojedinačne Stranice odjela 49
5.6.3. Dijaloški prozor za kreiranje poveznica 51
5.6.4. Lokalna pohrana podataka 52
5.7. Stranica s popisom nastavnika 54
5.8. Stranice vijesti i obavijesti 56
5.9. Stranica postani_sjeveroljubac.html 59
5.9.1. Implementacija Google mapsa 62
5.10. Autorizacija kroz LDAP 64
5.10.1. Izrada stranice LDAP_login.html 67
5.11. Izrada tema u ThemeRolleru 69
5.12. Izrada vizualnog identiteta aplikacije 70
5.13. Prilagođeni stilovi 71
5.14. Konfiguracijska datoteka config.xml 72
5.15. Testiranje aplikacije 75
6. Zaključak 79
7. Literaturne reference 81

1. Uvod

Prošlo je gotovo 30-ak godina od kad je započela era mobilne tehnologije. 1988. g. mobilni uređaji su izgledali poput torbica i organizatora koji su podsjećali na kalkulatore (PSION) a, bežični internet i rasprostranjena dostupnost resursima poput interneta bili su daleki san. Jedino ako ste bili SF pisac. Gledajući u budućnost, pretpostavljam da sa sigurnošću mogu reći da je nemoguće predvidjeti napredak tehnologije. Kreativnost programera u svijetu jamči da nećemo moći predvidjeti koje će se sve funkcionalnosti pojaviti na našim mobilnim uređajima.

Kada govorimo o mobilnim aplikacijama, glavno pitanje koje se postavlja je, nativne ili hibridne? Iako su prve mobilne aplikacije za iOS bile temeljene na web tehnologijama one nisu mogle uspostaviti vezu sa nativnim funkcionalnostima uređaja kao što su kamera, kontakti itd. Hibriden mobilne aplikacije se baziraju na HTML5, CSS i Javascript tehnologijama dok su nativne aplikacije razvijene korištenjem programskih jezika Java, C++, s alatima poput Xcode, Eclipse i sl. Najveća prednost hibridnih mobilnih aplikacija u odnosu na nativne je ta što su one kompatibilne za sve mobilne platforme dok se nativne aplikacije moraju razvijati za određenu platformu i pisati na određeni način u određenom programskom jeziku. Najveći nedostatak hibridnih aplikacija je taj što one ne mogu u potpunosti koristiti sve ugrađene mogućnosti pojedine mobilne platforme. Daljnjim razvojem tehnologije i mobitela, hibridne aplikacije će polagano početi istiskivati nativne sa tržišta. Prema nekim procjenama (Yahoo) 2017. više od 50% aplikacija koje će se naći na tržištu, biti će hibridne aplikacije.

Apache Cordova i jQuery Mobile (slika 1.1) su radni okviri za izradu hibridnih mobilnih aplikacija. Ovi radni okviri omogućavaju izradu aplikacija koji će biti podržani na gotovo svim mobilnim platformama i u svim internet preglednicima. Stoga će se u ovom radu koristiti navedeni radni okvir te će on obuhvaćati pojmove koji su vezane uz razvoj HTML5 hibridnih mobilnih aplikacija i tehnologije koje se koriste prilikom rada sa Cordovom i jQuery Mobileom. U teorijskom dijelu rada biti će opisana funkcionalnsot Cordove, komponente od kojih se sastoji Cordova projekt, karakteristika koje Cordova podržava i konfiguracija Cordova projekta. Isto tako, u radu će biti opisan postupak instalacije Cordove okruženja i neke osnovne naredbe koje se mogu realizirati preko naredbenog retka (Command-Linea). JQuery Mobile je javascript programska zbirka čija je osnova jQuery. Stoga ću u rad obuvatiti jQuery Mobile elemenate koji olakšavaju razvoj aplikacije, a budući da se temelji na HTML5 i jQuery-ju ukratko će biti objašnjeno kako dvije tehnologije funkcioniraju te će biti opisana uloga jQM u svrhu poboljšanja mobilnog doživljaja. Nakon

teorijskog djela slijedit će opis izrade mobilne aplikacije ”Sjeveroljubac” koja će biti namijenjena za sve studente Sveučilišta Sjever. Glavna svrha aplikacije je da studentima omogući što brži pristup informacijama koje su vezane uz samo Sveučilište. Aplikacija će se sastojati od poveznica na službenu stranica Sveučilišta, službeni mail Sveučilišta, moodle i studomat jer su upravo te poveznice koje studenti najčešće koriste kroz svoje studentsko razdoblje. Isto tako, budući da je Sveučilište Sjever okosnica odnosno temelj same aplikacije, unutar aplikacije su stavljene i poveznice pomoću kojih se student može povezati sa Sveučilištem na društvenim mrežama. Aplikacija će biti izgrađena za Android i iOS mobilne platforme.



Slika 1.1 JQM i Cordova

2. Programska zbirka Cordova

Apache Cordova je besplatna, programska zbirka za razvoj višeplatformskih nativnih aplikacija koja koristi isključivo HTML5 tehnologiju. Tvorci Apache Cordove htjeli su jednostavniji način za izgradnju višeplatformskih mobilnih aplikacija i odlučili su se da ga implementiraju kao kombinaciju nativnih programskih priključaka i web tehnologije. Ova vrsta mobilne aplikacije se naziva hibridna aplikacija.[1]

Početna korist Cordova programske zbirke je njezina nativna sposobnosti koje je podržana u pregledniku. U vrijeme prvih mobilnih aplikacija, najbolji način za izgradnju mobilnih aplikacija koje su funkcionirale na mnogim mobilnim uređajima je bio taj, da ga se izgradi pomoću HTML-a. Nažalost, za programere, mobilne aplikacije su trebale učiniti mnogo više od onoga što je podržano u web preglednicima. Izgradnja web aplikacija koje će biti u interakciji sa fotoaparatom na uređaju ili sa lokalnim kontaktima (imenikom) nije bila moguća. Da bi se zaobišla ova ograničenja, Cordova implementira paket dodataka koji se protežu na nativne mogućnosti uređaja (kao što su kamere, akcelerometar, kontakti) pokrećući web aplikaciju unutar nativnog kontejner.

Cordova ne prevodi web aplikaciju u nativni jezik za svaku podržanu mobilnu platformu (npr, C za iOS ili Java za Android), kod Cordove web aplikacija se pokreću nemodificirane unutar nativnog aplikacijskog okvira.[2]

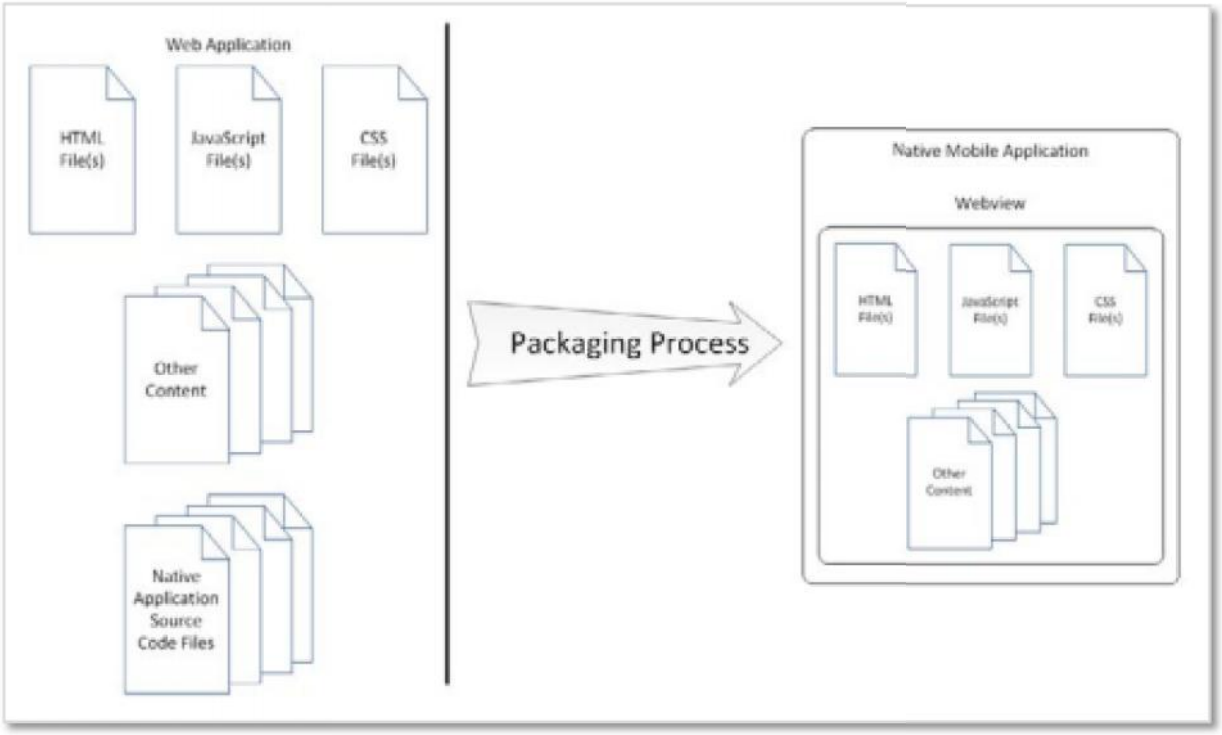
2.1. Komponente

Komponente (slika 2.1) od kojih se sastoji Cordova aplikacija:

1. Izvorni kod (source code) za nativni aplikacijski kontejner za svaku podržanu mobilnu platformu. kontejner renderira HTML5 aplikaciju na uređaj.

2. Paketi dodataka(API) koji pružaju pokretanje web aplikacije unutar kontejnera s kojima se omogućava pristup nativnim mogućnostima uređaja

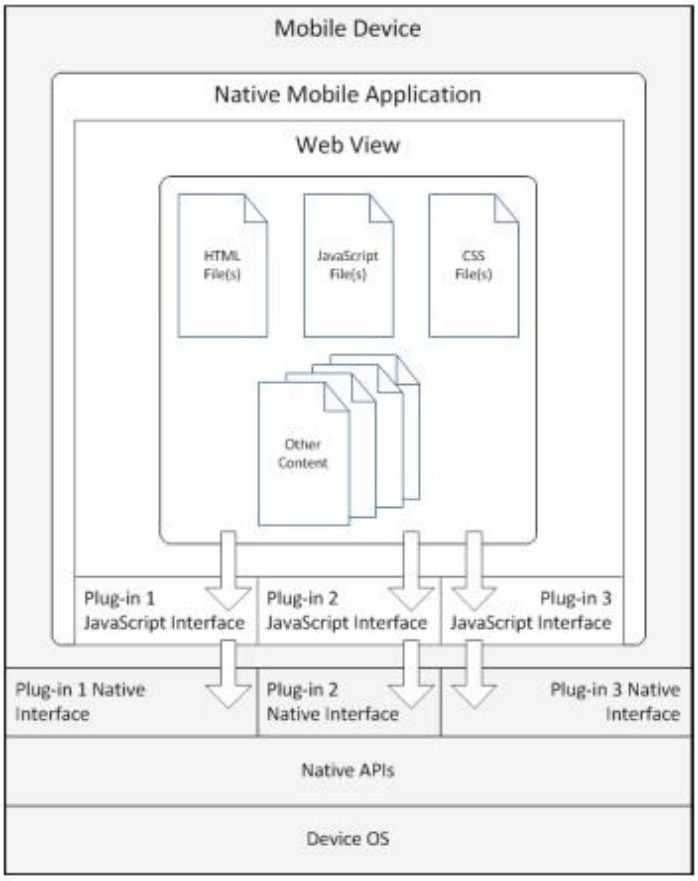
3. Set alata (tools) koji se koriste za upravljanje procesom stvaranja aplikacijskih projekata, upravljanje raznim pluginovima, izgradnja nativnih aplikacija (koristeći SDK) i testiranje aplikacije na mobilnim simulatorima i emulatorima.



Slika 2.1 Cordova komponente

Korisničko sučelje aplikacije sastoji se od jednog zaslona koji sadrži web pogled (web view) koji upotrebljava dostupan prostor na zaslonu uređaja. Kad se aplikacija pokrene, učitava se ta početna web stranica (index.html ili neka druga) unutar web prikaza. Tada aplikacija predaje kontrolu web prikaua i omogućava korisniku interakciju sa web aplikacijom. U interakciji sa sadržajem web aplikacije, poveznicama (linkovima) ili Javascript kodom unutar aplikacije, korisnik može očitavati i drugi sadržaj unutar resursnih datoteka zapakiranih s aplikacijom ili može preko mreže preuzeti (download) sadržaj sa weba ili aplikacijskog servera.

Cordova pruža paket Javascript API-ja koje omogućuju programeru da pokrene web aplikaciju unutar Cordova kontejnera kako bi programer mogao pristupiti mogućnostima uređaja izvan web konteksta. API-i se implementiraju u dva dijela: Javascript knjižica (library) koja prikazuju nativne mogućnosti web aplikacije i korespondirajućeg nativnog koda koji se pokreće u kontejneru koji implementira nativni dio API-a (slika 2.2).



Slika 2.2 Arhitektura nativne aplikacije

2.2. Dodaci i podržane mobilne platforme

Cordova dodaci (API) su:

- Accelerometer
- Camera
- Capture
- Compass
- Contacts
- Device
- Events
- File
- Geolocation
- Globalization
- InAppBrowser
- MediaNotification
- Splashscreen
- Storage

Kad programer implementira značajku u aplikaciji koja koristi jednu od Cordova dodataka, aplikacija poziva dodatak pomoću JavaScripta. Poseban sloj unutar aplikacije prevodi Cordova dodatak u pozive za odgovarajuću nativnu značajku. Način na koji se pristupa

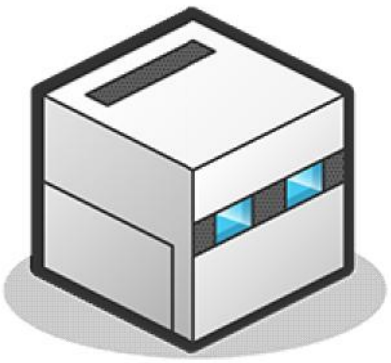
fotoaparatu na BlackBerryu je drugačiji od onoga kako se to radi na Androidu, tako da zajednički sloj omogućava programeru implementaciju jednog sučelje koji se prevodi iza scene (u spremniku aplikacija) u odgovarajući nativni dodatak za svaku podržanu mobilnu platformu.

Apache Cordova trenutno podržava sljedeće mobilne platforme:

- Android (Google)
- bada (Samsung)
- BlackBerry 10 (BlackBerry)
- iOS (Apple)
- Firefox OS
- Tizen (Linux)
- Windows Phone 7 i Windows Phone 8 (Microsoft)
- Windows 8 (Microsoft)

2.3. Programska zbirka PhoneGap

PhoneGap (http://phonegap.com/)(slika 2.3) je besplatna programska zbirka koja omogućava razvoj mobilnih aplikacija korištenjem standardiziranih sučelja za programiranje aplikacija. Omogućava softverskim inženjerima razvoj mobilnih aplikacija korištenjem JavaScript, HTML5 i CSS3 programskih jezika, umjesto specifičnih programskih jezika za određenu vrstu uređaja, kao objektni C. Rezultat takvog pristupa razvoju su hibridne mobilne aplikacije što znači da nisu niti nativne (jer se cijelo renderiranje sučelja odvija pomoću web prikaza umjesto nativnog radnog okvira određene tehnologije), a niti u potpunosti web orijentirane (zato jer nisu isključivo web aplikacije, već su pakirane kao aplikacije za distribuciju i imaju pristup nativnim sučeljima uređaja).[4]



Slika 2.3 PhoneGap

Podloga PhoneGap programske podrške je Apache Cordova programska zbirka. Da bi se shvatila veza između PhoneGap-a i Cordova. treba se vratiti u prošlost. PhoneGap je besplatan projekt stvoren od male kanadske kompanije Nitobi, kojeg 2011. kupuje Adobe, s naglaskom da Adobe nije kupio PhoneGap-ovu ”bazu kodova” (zbirku izvornih kodova koji se koriste za izgradnju određene aplikacije), nego ime i tim koji je radio na projektu. Projekt je doniran ASF-u (Apache Software Foundation), da bi se stvorio novi projekt pod nazivom Cordova (Cordova je ulica u Vancouver-u u kojoj je bila smještena kompanija Nitobi). PhoneGap je Adobe-eva binaran distribucija(distribucija softvera koji sadrži izvršne binarne datoteke, bez izvornog koda) Apache-eva projekta Cordova. Laički govereći, na Cordovu možemo gledati kao na ”engine” koji osnažuje PhoneGap, slično kao što WebKit osnažuje Chrome ili Safari.

PhoneGap distribucija sadrži dodatne alate koji je vežu s ostalim Adobe servisima a, koji nisu prikladni za Apache projekt. Upotrebom PhoneGap Build (https://build.phonegap.com/) i Adobe Edge Inspect (http://www.adobe.com/products/edge-inspect.html) razvoj aplikacija poprima potpuno novi strateški smisao. PhoneGap Build je ”cloud” baziran servis (platforma za smještaj, pokretanje i korištenje programske podrške) koji dopušta izgradnju mobilnih aplikacija u oblaku. Znači, programer pomoću navedenog servisa može izgraditi aplikaciju za različite mobilne platforme, bez prethodne lokalne instalacije razvojnih alata (SDK) za pojedine mobilne platforme. Edge Inspect je Adobe alat za pregled i testiranje aplikacija na različitim mobilnim uređajima.

Prednosti i nedostaci Cordova tehnologije

Prednosti:

1. Za razvoj aplikacija je dovoljno znanje HTML5, CSS3 i JavaScript programskih jezika. Kako je Cordova razvojni okvir koji u sebi sadrži sva sučelja za programiranje mobilnih aplikacija, nije potrebno znanje nekog od ”glomaznih” programskih jezika.
2. Mobilna aplikacija razvijena Cordova tehnologijom je podržana za sve vrste mobilnih uređaja, te je ovo svakako najveća prednost primjene Cordova razvojne tehnologije.
3. Smanjena cijena razvoja mobilne aplikacije. S obzirom na višestruku primjenjivost razvoj mobilnih aplikacija primjenom Cordova razvojne tehnologije je višestruko manja od razvoja nativnih aplikacija. Vrijeme razvoja je smanjeno, količina potrebnog

znanja je manja, a u konačnici i manji broj razvojnih inženjera je potrebno za razvoj mobilne aplikacije.

4. Podržan je pristup najvažnijim nativnim sučeljima za programiranje mobilnih aplikacija poput žiroskopa, kamere, kompasa, geolokacije, mreže, kamere, notifikacija, itd. Na taj način je smanjena razlika između korisničkog iskustva u korištenju mobilnih aplikacija razvijenih Cordova razvojnom tehnologijom u odnosu na nativne aplikacije.
5. Distribucija mobilnih aplikacija putem svih dostupnih programskih dućana (app stores).
6. Mobilne aplikacije razvijene Cordova tehnologijom imaju jednostavno i pregledno sučelje poput nativnih aplikacija, bez okvira internetskog preglednika oko njih.

Nedostatci:

1. Cordova mobilne aplikacije nemaju pristup svim ugrađenim funkcionalnostima mobilnih uređaja.
2. Zbog činjenica da Cordova aplikacije pokrivaju višestruke platforme, uvijek su jedan korak u zaostatku u primjeni novih funkcionalnosti koje pružaju mobilni uređaji s vremena na vrijeme.
3. Cordova mobilne aplikacije izgledaju identično na svim mobilnih tehnologijama. Sve dok izgledaju i funkcioniraju kao nativne aplikacije, sam izgled im je ipak pomalo generički – ne mora u svim slučajevima biti da u potpunosti Cordova aplikacija izgleda poput nativne iOS ili Android aplikacije. U prijevodu, kod nativnih aplikacija je ipak moguće sučelje aplikacije detaljnije dizajnirati i prilagođavati na mobilnoj platformi za koju se razvija u odnosu na Cordova aplikacije.

2.4. Programska zbirka jQuery Mobile

jQuery Mobile (http://jquerymobile.com/) je HTML5 baziran sustav, dizajniran za izradu responzivnih web mjesta i aplikacija koji su dostupni na smartphonima, tabletima i desktop uređajima. Zbirka je radni okvir koji je izgrađen je na široko korištenom jQuery-u te čistim HTML-om, koji osigurava kompatiblinost s web uređajima. Još jedna stavka koja ide u korist

jQuery Mobile-a je ta da on spaja nativne i mobilne web aplikacije. Isto tako, podržan je na većini mobilnih operativnih sustava (iOS, Android, Windows).

JQuery Mobile (slika 2.4) je stvoren od strane John Resiga, a zbirka se počela koristiti sredinom 2010. i ubrzo je postala jedna od najpopularnijih JavaScript programska zbirka. Danas se u web aplikacijama jQuery Mobile okvir koristi više nego bilo koji drugi radni okvir jQuery Mobile je open source projekt, smještene na Githubu s potpunom web stranicom, velikim brojem dokumentacije, primjera i referenci za stvaranje aplikacija. jQuery Mobile funkcionira na većini svih modernih desktop, smartphone, tablet i eReader platformi. Osim toga, stariji preglednici i telefoni također su podržani zbog progresivnog poboljšanog pristupa što je ujedno, vjerojatno jedna od najvažnijih karakteristika jQuery Mobile. [7]



Slika 2.4 jQuery Mobile

Kompatibilnost

Korisnici najnaprednijih mobilnih preglednika mogu uživati u potpunom poboljšanom doživljaju, s Ajax baziranim animiranim tranzicijama stranica koje su podržane u sljedećim OS/browser kombinacijama:

- iOS
- Android
- Windows
- Blackberry
- Palm WebOS
- Firebox Mobile
- Skyfire
- Opera Mobile
- Meego
- Samsung bada
- UC Browser
- Kindle and Kindle Fire

Na desktop platformama, jQuery Mobile je kompatibilan sa Windows, Linux i Mac OS X preglednicima:

- Safari
- Chrome
- Firefox
- Internet Explorer
- Opera

Jedna od najvećih koristi od navedenog popisa je ta da je jQuery Mobile je uistinu jedan od najkompatibilnijih mobilnih okvira koji su danas dostupni na tržištu. Njegova velika podrška desktop preglednicima omogućuje programerima da koriste različite platforme za izgradnju i testiranje aplikacija.

Najvažnija stvar koju bi trebalo znati o jQuery Mobile jest da je to knjižnica, a ne jQuery plug-in. Knjižnica koja će uzeti HTML oznake kao ulaz, te će ih formatirati pomoću unaprijed definiranih stilova i koja će ih prilagođavajući prema trenutnim mogućnostima preglednika. jQuery Mobile se oslanja na JavaScript i HTML5 značajke koje su podržane od strane hosting preglednika kako bi ponudio proširene funkcionalnosti. Ova prva karakteristika određuje veliku podršku jQuery Mobilea na mobilnim preglednicima, dok u isto vrijeme objašnjava zašto programeri moraju razvaijati vlastiti JavaScript kod za implementaciju kompleksnih ponašanja, (pohranjivanje podataka ili komunikacija s hardverskim značajkama koji su izloženi od strane host preglednika (Geolocation, kompas, itd)). Druga važna karakteristika jQuery Mobile je da ne postavlja bilo kakvu strukturu za JavaScript kod vašeg sustava. Općenito govoreći, programeri će upotrebljavati ponašanja pomoću standardnih jQuery sintaksi i idioma, baš kao što bi to bio slučaj sa normalnom web stranicom.

2.5. Izgradnja Cordova aplikacije

Cordova aplikacije su web aplikacija pokrenute unutar kllijentskog nativnog aplikacijskog spremnika. Dakle, web aplikacije pokrenute unutar Cordova aplikacija imaju veći utjecaj na HTML5 aplikacijsku strukturu od tradicionalnih serverskih web aplikacija.

HTML5, web aplikacije mogu koristiti nove mogućnosti koje dopuštaju aplikaciji da na mobilnom uređaju funkcioniraju učinkovitije (ili kod uređaja s ograničenom vezom) i mogu

koristiti klijentsku bazu podataka za pohranu aplikacijskih podataka. Ova funkcionalnost omogućava mobilnim uređajima lakši rad kako se one kreću unutar bežične pokrivenost. Osim toga, HTML5 podržava dodatak manifest datoteke koja navodi sve datoteke koji se nalaze unutar web aplikacije. Kada se učita index datoteka, preglednik čita manifest datoteku, dohvaća sve datoteke navedene u manifestu, te ih preuzima na klijentski uređaj. Ako mobilni uređaj izgubi mrežnu povezanost, web aplikacija će raditi odnosno pokretati će sve datoteke koje su učitane jer će svi podaci biti pohranjeni lokalno.

Web aplikacija mora biti napisana tako da može u potpunosti raditi unutar preglednika (ili, u slučaju Cordova aplikacija, u Cordova aplikacijskom spremniku).Index.html datoteka je obično jedina HTML datoteka u aplikaciji, a aplikacijski različiti prikazi (screenovi) su zapravo samo različiti spremnici koji aplikacija uključuje i isključuje po potrebi. HTML5 aplikacije će i dalje doprirati do server po podatke, koristeći XHR za asinkrono traženje podataka te će ih pohraniti lokalno.

Nakon što je web aplikacija završena, bilo da koristi jednu od Cordova APIa ili ne, mora biti pakirana u nativnu aplikaciju koja će se pokretati na uređaju. Svaka od mobilnih platformi koje podržava Cordova projekt ima svoje vlastite alate za pakiranje ili izgradnju nativne aplikacije. Prilikom izgradnje Cordova aplikaciju za svaku podržani mobilnu platformu, web sadržaj (HTML, CSS, JavaScript, i druge datoteke koje su sadržane u aplikaciji) mora biti dodan u aplikacijski projekt koji je odgovarajući za svaku mobilnu platformu, zatim izgrađen korištenjem alata koji pripadaju platformi. Ono što je zanimljivo i izazovnu u cijelom ovom postupku je da svaka mobilna platforma koristi u potpunosti različite alata te da aplikacijski projekti koriste različite konfiguracijske datoteke ovisno o platformi i da će najvjerojatnije projektna strukturu biti drugačija.

Neki od podržanih mobilnih alata (SDK) za razvoj radit će samo na pojedinim desktop operativnim sustavima:

- Android SDK (Linux, Microsoft Windows i Macintosh OS X)
- BlackBerry SDKs (Microsoft Windows i Macintosh OS X)
- iOS SDK (Macintosh OS X)
- Windows Phone SDK (Microsoft Windows)

2.5.1. Konfiguracija Cordove

Kao što je već prije istaknuto, za konfiguriranje razvojnih okruženja za različite podržane mobilne platforme, potrebno je instalirati platformin nativni softverski razvojni alat (SDK) ili integrirano razvojno okruženje (IDE). Za neke platforme, biti će potrebno instalirati dodatne alate kao što su Java, Ant. Gotovo svi alati koji su potrebni za lokalni razvoj Cordove su besplatni.

Mnogi aspekti ponašanje aplikacije se mogu kontrolirati sa globalnom configuracijskom datotekom config.xml. Ova XML datoteka je organizirana prema W3C Packaged Web Apps (Widgets) specifikaciji i produžuje osnovne specifične Cordova značajke, pluginove i specifične platformske postavke. Osim globalne konfiguracijske datoteke postoje i zasebne konfiguracijske datoteke koje se kreiraju ovisno o platformi za koju je namjenjena. Globalna konfiguracijska datoteka se nalazi u unutar glavnoga dokumenta tako da konstatno može biti vidljiva (app/config.xml) dok se pojedinačne konfiguracijske datoteke pasivno kopiraju u različite poddirektorije (app/platforms/android/res/xml/config.xml).

Konfiguracijski elementi

Sljedeći primjer prikazuje defaultnu config.xml datoteku:

```
<widget id="com.example.hello" version="0.0.1">
        <name>HelloWorld</name>
        <description>
            A sample Apache Cordova application that responds to
the deviceready event.
        </description>
        <author email="dev@callback.apache.org"
href="http://cordova.io">
            Apache Cordova Team
        </author>
        <content src="index.html" />
        <access origin="*" />
        <preference name="Fullscreen" value="true" />
 </widget>
```

Navedeni konfiguracijski elementi se nalaze u najvišoj config.xml datoteci i podržani u svim Cordova platformama.

<**widget**> elementov id atribut pruža aplikacijski identifikacijsku domenu i verziju sa punim brojem izraženim u većoj/manjoj notaciji. Ova oznaka isto može imati atribute koje specificiraju alternativne verzije, versionCode za Android i CFBundleVersion za iOS.

<name> element označava formalno ime aplikacije kako se ona pojavljuje na zaslonima uređaja i unutar app-store sučelja.

<description> i <author> elementi specificiraju meta podatke i kontakt informacije koji se mogu pojaviti unutar app-storea

<content> element definira početnu stranicu aplikacije u najvišem web direktoriju. Defaultna vrijednost je *index.html*.

<access> element definira set vanjskih domena s kojima aplikacija može komunicirati. Defaultna vrijednost koja se vidi u navedenom primjeru, dopušta aplikaciji pristup bilo kojem serveru.

<preference> element označava set različitih opcija kao parove imena(name) i vrijednosti(value).[5]

2.5.2. Web prikazi za različite platforme

Web pogled (web view) je nativna aplikacijska komponenta koja se koristi za renderiranje web sadržaja (obično HTML stranica) unutar nativnog apliakcijskog prozora ili zaslona. To je programatski dostupan omot (wrapper – podatak koji zauazima ili uokviruje glavni podatak ili program, koji postavlja drugi program kako bi on uspješno funkcionirao) oko ugrađenog web preglednika uključen u mobilnom uređaju.
Android – implementacijom *WebView* prikaza (korištenjem *android.webkit.Webview*)
iOS – implementacijom *UIWebView* prikaza (*System/Library/Framework/UIKit.framework*)
BlackBerry – Implementacijom *BrowserField* objekta (*net.rim.device.api.browser.field2*).

U osnovi web pogled osigurava korištenje dodataka (widgeta) koji ne koristi web preglednik, onemogućava JavaScript i pogreške web stranica se zanemaruju. Ukoliko se prikazuje samo HTML unutar određenog ekrana, WebView je dobro rješenje; korisnik ne mora koristiti elemente web preglednika da bi koristio sadržaj ekrana, a elementi web preglednika ne moraju komunicirati sa korisnikom da bi se sadržaj prikazao.

Web pogled kao tehnologija prikaza hibridnih mobilnih aplikacija sama po sebi je standardizirana i uvelike olakšava razvoj mobilnih aplikacija. Zapravo, mogao bi se zauzeti stav da WebView kao tehnologija može samo napredovati. Naravno, određena ograničenja postoje, jer ipak se korisničko sučelje ne može u potpunosti prilagoditi kao kod nativnih

aplikacija. No, sama činjenica da se razvoj određene mobilne aplikacije usmjerio na primjenu ove tehnologije opravdava sve njezine nedostatke.

2.5.3. Node.js i npm

Node.js je popularna Javascript platforma za server-side programiranje koja omogućava izgradnju i pokretanje web aplikacija. Node.js omogućava pisanje aplikacija u JavaScriptu na serveru. Izgrađen je na V8 JavaScript engineu te je napisan u C ++. Izvorno, namjena mu je bila da postane server okruženje za aplikacije, ali programeri su ga počeli koristiti za stvaranje alata kako bi im pomogli u lokalnoj automatizacije. Od tada se razvio cijeli novi sustav Node alata (kao što Grunt i Gulp) koji su transformirali front-end razvoj korisničkih sučelja. NPM ili Node Package Manager instalira Node alate (pakete) koje korisnik želi koristiti i pruža korisno sučelje za rad s njima. NPM su zapravo dvije stvari: prvo, to je on-line repozitoriji za objavljivanje open-source Node.js projekata; drugo, to je command-line alat za interakciju s navedenim repozitorijem koji pomaže u instalaciji paketa, upravljanju verzijama i upravljanju ovisnostima. Velik broj node.js knjižnica i aplikacija su objavljene na NPM i još mnogo se dodaju svaki dan.

NPM može instalirati pakete u lokalnom ili globalnom načinu rada. U lokalnom on instalira pakete u node_modules datoteci u matičnom radnom direktoriju. Ova lokacija je ujedno u vlasništvu trenutnog korisnika. Globalni paketi su instalirani u {prefiks}/lib/node_modules/ koja je u vlasništvu administratora (gdje je {prefiks} obično /usr/ ili /usr/local) što znači da će morati koristiti naredbu za globalnu instalaciju paketa. Ovakav način instalacije može dovesti do pogrešaka sa dozvolama prilikom rješavanja ovisnosti, kao i do sigurnosnih problema. [6]

2.5.4. Osnove rada s cordova naredbama

Nakon što su pomoću npm-a instalirane sve potrebne cordova komponente može se pristupit izradi cordova projekta. To se ostvaruje **cordova** naredbom iz naredbenog retka (engl. *Command line).* U daljenjem tekstu navest ćemo samo nekoliko naredbi koje služe za inicijalizaciju projekta i dodavanje pojedinačnih platformi.

Stvaranje Cordova projekta:

```
cordova create hello com.example.hello HelloWorld
```

Dodavanje platformi:

```
cordova platform add ios
cordova platform add amazon-fireos
cordova platform add android
cordova platform add blackberry10
cordova platform add firefoxos
```

Uklanjanje platformi:

```
cordova platform remove blackberry10
cordova platform rm amazon-fireos
```

Pokretanje projekta:

```
cordova build
```

Testiranje projekta na uređaju ili emulatoru:

```
cordova emulate android
```

Dodavanje cordova plugin-ova:

```
cordova plugin add cordova-plugin-device-motion
cordova plugin add cordova-plugin-device-orientation
cordova plugin add cordova-plugin-geolocation
```

Ukoliko se Cordova koristi iz NetBeans-a ili bilo koje druge razvojne okoline, naredbe nije potrebno "ručno" izvoditi već ih se pokreće pri stvaranju novog cordova projekta.

2.5.5. Instalacija programske zbirke Cordova

Cordova se mora instalira na lokalni sistem (Local Disk (C)) a, pokreće se na Node.js platformi, stoga se kao prvi korak instalacije, mora instalirati Node.js.

1. Korak. Preuzimanje i instalacija Node.js-a (https://nodejs.org)
2. Korak. Preko npm-a (node package manager), instalirati Cordovu

```
npm install –g cordova
```

3. Korak. Preuzimanje i instalacija Git-a (https://git-scm.com/)

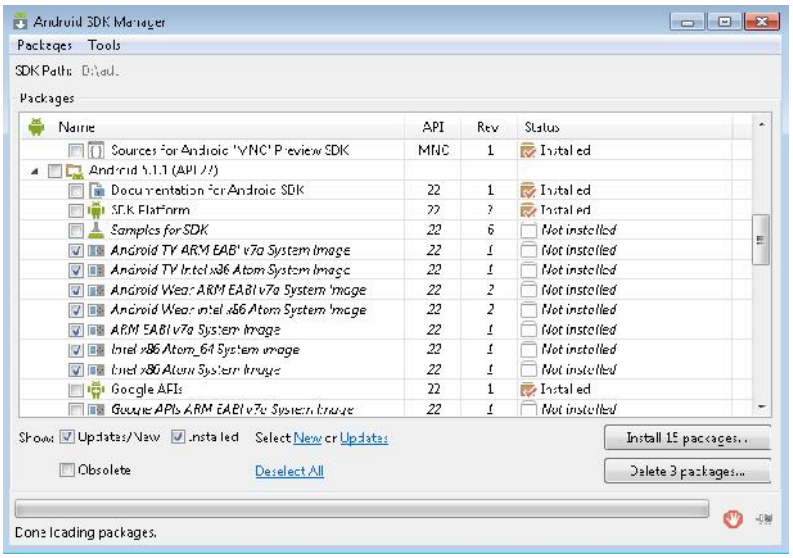
4. Korak. Dodavanje Git-a i Cordove na putanju (Path environment) *System properties > Advanced system settings > Environment Variables > Path Variable-value =* dodaju se putanje prema Gitu i Cordovi s lokalnog diska C

   ;C:\Program Files (x86)\Git\bin;C:\ProgramFiles (x86)\Git\cmd;
   ;C:\Users\User\AppData\Roaming\npm

5. Korak. Stvaranje Cordova aplikacije unutar Netbeansa (IDE) i dodavanje jquery, jquerymobile programskih zbirki.

   File > New Project > Categories: HTML5 > Cordova Application

6. Korak. Preko SDK managera (slika 2.5), instalirati najnoviju verziju Androida (API 22), alate za pokretanje Android mobilne platforme i emulator za testiranje.



Slika 2.5 Android SDK Manager

2.6. Zaključak Cordove

Apache Cordova je set sučelja za programiranje koji u kombinaciji sa razvojnim okvirima korisničkog sučelja poput jQuery Mobile, omogućava da aplikacije za mobilne uređaje budu razvijene isključivo korištenjem programskih jezike HTML5, CSS i JavaScript. Kada se koristi Apache Cordova aplikacije za mobilne uređaje mogu se razvijati bez i jedne linije koda nativnih programskih jezika (Java, Objektni C, itd.). Umjesto toga, koriste se web tehnologije, koje se same poslužuju lokalno na mobilnom uređaju unutar aplikacije (generalno nisu poslužene na nekom udaljenom http serveru). Zbog tih JavaScript sučelja za programiranje

mobilne aplikacije su konzistentne na više platformi i izrađene prema web standardima, te bi aplikacije trebale biti prenosive na sve podržane mobilne platforme (Apple iOS, Google Android, Microsoft Windows Phone, BlackBerry, LG webOS) s minimalnim ili bez promjena u pisanju koda.

Cordova aplikacija je napisana na način da se može pokretati unutar preglednika ili kako je u slučaju kod Cordove, unutar Cordova spremnika. Nakon što je aplikacija završena, pakira se u nativnu aplikaciju koja će se pokretati na mobilnom uređaju. Svaka od platformi koje su podržane unutar Cordova projekta ima svoje vlastite alate za pakiranje i izgradnju. Prilikom izgradnje za svaku dodanu platformu, sadržaj aplikacije je dodan u aplikacijski projekt koji je odgovarajući za pojedinu platformu te izgrađen korištenjem alata koji pripadaju određenoj platformi. Ono što je zanimljivo i izazovno u cijelom ovom postupku je da svaka mobilna platforma koristi u potpunosti različite alate te da aplikacijski projekti koriste različite konfiguracijske datoteke ovisno o platformi i da će najvjerojatnije projektna struktura biti drugačija. Aspekti ponašanje palikacije se mogu kontrolirati sa globalnom konfiguracijskom datotekom config.xml. Ova datoteka je organizirana prema W3C Packaged Web Apps (Widgets) specifikaciji i produžuje osnovne specifične Cordova API značajke, pluginove i specifične platformske postavke. Osim globalne konfiguracijske datoteke postoje i zasebne konfiguracijske datoteke koje se kreiraju ovisno o platformi za koju je namjenjena. Web pogled je osnovna aplikacijska komponenta koja se koristi za prikaz web sadržaja (obično HTML stranica) unutar nativnog aplikacijskog spremnika. Web pogled kao tehnologija prikaza hibridnih mobilnih aplikacija sama po sebi je standardizirana i uvelike olakšava razvoj mobilnih aplikacija.

3. Alternative Cordovi

Hibridni aplikacijski pristup koji koristi Cordova nije jedinstven na tržištu. Cordova projekt je možda započeo trend, ali na tržištu postoje i mnogi drugi proizvodi koji koriste sličan pristup. U nastavku slijedi popis nekih od proizvoda koje koriste hibridni aplikacijski pristup. Neki su poput Cordove, dok su drugi izgrađeni iznutra sa Cordovom.

- Programski okvir Ionic
- Programski okvir Kendo UI
- Programski okvir Sencha Touch
- Programski okvir Appcelerator Titanium

3.1. Programski okvir Ionic

Ionic (http://ionic.io/) programski okvir omogućuje razvoj hibridnih nativnih mobilnih aplikacija (trenutno su podržani samo za Android i iOS uređaje), pomoću HTMLa, JavaScripta, CSSa i, u osnovi je kao i bilo koji drugi mobilni okvir. Ono što ga čini jedinstvenim je AngularJS i podrška za SASS (Syntactically Awesome Style Sheets) CSS ekstenziju. AngularJS, obično se naziva Angular, je open-source web radni okvir održavan od Googlea i zajednice individualnih developera i korporacija za rješavanje mnogih izazova s kojima se susreću u razvoju aplikacija. Ionicov cilj je pojednostaviti razvoj i testiranje aplikacija pružajući okvir za klijentsku stranu Model-View-Controller (MVC) arhitekturu, zajedno s komponentama koje se obično koriste u raskošnim internet aplikacijama. Isto tako, koristi jednostavnu sintaksu (barem u odnosu na Backbone ili Knockout) i kao jQuery, ima veliki broj pluginova i ekstenzija. Ionic pruža set AngularJS pokretanih grafičkih elemenata, koji nisu web komponente i ne mogu se koristiti izvan Ionic/Angular. Kao dodatak grafičkim elementima, Ionic će omogućiti svaku drugu potrebnu mobilnu funkcionalnostu, kao što su touch, animacije i asinkronu komunikaciju i nativno pakiranje. Za razliku od jQuery Mobile, Ionic mobilni okvir se koristi samo za hibridne mobilne aplikacije (HTML5 aplikacije zapakirane kao nativne mobilne aplikacije) i ne može se koristiti za klasične desktop web aplikacije (za takvu svrhu se koristi Angular).

Glavne značajke:

- Open-source i besplatan
- Izgrađen oko Angulara
- iOS i Android podrška
- SASS podrška
- Podržava Cordovu i Phonegap

3.2. Programski okvir Kendo UI

Kendo UI (http://www.telerik.com/kendo-ui) je radni okvir za moderna HTML sučelje, stvorena od strane Telerika. Upotrebom najnovijih HTML5, CSS3 i JavaScript standarda, Kendo UI pruža sve što je potrebno za klijentsku stranu, jQuery pokreće razvoj u jednom integriranom, kompaktnom pakiranju, zajedno sa integracijom AngularJS i Bootstrap podrškom. Budući da je ovo komercijalni proizvod dolazi u dva odvojena paketa. Prvi se zove Kendo UI Core i on je open source, dok se drugi zove Kendo UI Professional i za njega se mora platiti. Core izdanje sadrži 40 widgeta, sve mobilne widgete, kao i okvira za mobilne aplikacije, jezgru funkcionalnosti Kendo UI radnog okvira, uključujući i Datasource komponentu, predloške i tematske grafičke elemente s integriranim animacijama te AngularJS direktivom. Kendo UI je izgrađen na jQueryu tako da, ako je programer upoznat sa jQuery sintaksom, neće imati problema prilikom njegovog korištenja. Kendo UI je uspješna mješavina nekoliko različitih razvojnih filozofija. S jedne strane, može se koristiti na sličan način kao i jQuery Mobile, kroz HTML5 oznake. S druge strane, može se koristiti kao Ionic okvir, koristeći Angular. Oba pristupa imaju svoje prednosti i mane. Prvi će se svidjeti programerima početnicima, dok će drugi privući one iskusnije. Kendo UI je komercijalni proizvod i nema istu potporu open source zajednice stoga takvo korištenje povećava troškove cjelokupnog razvoja.

Glavne značajke:

- Licenciran pod dozvoljenom Apache v2 licencom
- Jednostavan za korištenje
- Izgrađen oko AngularJSa
- Ugrađena ThemeRoller podrška

- Podržava nativni pakirajući sustav
- Koristi se za mobilni i klasični web development
- 40 widgeta uključeno unutar Core izdanja

3.3. Programski okvir Sencha Touch

Korisničko sučelje ili radni okvir (https://www.sencha.com/), posebno izgrađen za mobilni web. Mogu ga koristiti web programeri za razvoj korisničkih sučelja za mobilne web aplikacije koje izgledaju kao nativne aplikacije na podržanim mobilnim uređajima (Android, iOS, Windows, Tizen i BlackBerry uređaji). U potpunosti se temelji na web standardima kao što su HTML5, CSS3 i OOP JavaScript. To je ujedno i jedan od najstarijih mobilnih okvira, ako ne i najstariji. U to vrijeme, prilikom razvoja mobilnih okvira obično se koristio jQuery (u to vrijeme AngularJS nije postojao i BackboneJS je još uvijek bio u fazi razvoja), Sencha Touch je otišao na drugu stranu korištenjem ExtJs kao JavaScript temelj i jQTouch kao UI okvir. Sencha Touch obuhvaća široki raspon grafičkih elemenata (ili widgeta, više od 50) za uporabu u mobilnim web aplikacijama. Sve te komponente su optimizirane za touch input. Komponente su: tipke sa određenim temama i efektima, elementi forme poput tekstualnih polja za e-mail, datum i adresa, slideri, selektori, liste, minimalni set ikona, alatne trake i izbornici, itd. Sencha Touch arhitekturu slijedi uobičajeni MVC model. Iako postoje mnoge MVC arhitekture (klasični MVC, moderni MVVM), od kojih se većina neznatno razlikuje jedan od drugog, Sencha Touch MVC slijedi ova pravila: Models, Stores, Views, and Controllers. Ako ne volite objektno orijentirani razvoj (OOP), onda Sencha Touch nije za vas. Zahvaljujući ovom značajkom ima višu krivulju učenja od drugih mobilnih okvira.

Glavne značajke:

- Komercijalni proizvod
- Ugrađena Theme Builder podrška
- 50+ grafičkih elemenata
- Podržava nativni pakirajući sustav kao i Cordova
- Koristi se za mobilni i klasični web development

3.4. Programski okvir Appcelerator Titanium

Appcelerator Titanium (http://www.appcelerator.com/) je razvojna okolina namijenjena za izradu aplikacija za više uređaja bazirana na Eclipse razvojnom okruženju. Titanium je nastao krajem 2008 godine. Kada je izašao bio je bez potpore za Android i iPhone uređaje koji su uvedeni u 6. mjesecu 2009. Uz titanium je moguće izrađivati aplikacije za Android mobilne telefone za iPhone za tablete, pa i računala i web. Ono što ga čini posebnim je da baza aplikacije tj. kod uglavnom ostaje jednak i nisu potrebne velike promjene da bi aplikacija radila na više različitih uređaja.

Titanium se zasniva na javascript jeziku, ali podržava i većinu popularnih jezika poput PHPa, Pythona, CSS-a i HTML-a. Omogućava izradu programskog koda, editiranje gotovog koda te raznovrsno debagiranje i testiranje aplikacije na emulatorima za ciljani uređaj poput Adroida ili iPhone-a. Emulatori omogućuju precizan prikaz aplikacije u radnom okruženju što olakšava i ubrzava izradu samih aplikacija. Moguća je i instalacija aplikacije na postojeći uređaj te postoji jednostavno automatizirano rješenje da se gotova aplikacija proslijedi i objavi na Android i apple store. Titanium ima implementiranu pomoć za nadopunu koda kod izrade aplikacija i podržava HTML, CSS, PHP, Javascript, Python i Ruby.

Titanium za razliku od većine drugih opcija kreira izvorne aplikacije, a ne samo simulaciju izvorne aplikacije. To čini tako da koristi izvorne UI komponente pa aplikacija nakon procesa prevođenja (Compiling) funkcionira poput izvorne u svakom pogledu. Titanium u teoriji zapravo radi izvorni kod za svaku platformu, koristeći samo JavaScript, njegov napredni prevoditelj spaja JavaScript kod sa Titanium API-om koji se zatim provjerava kod sa interpreterom koji se nalazi na ciljanom Operativnom sistemu i zatim se ubacuje kao međulinijski objekt ili poveznica između izvornih komponenti i to uparen sa izvornim objektima u omjeru 1:1, što ga zapravo čini premosnicom koja omogućuje korištenje izvornih svojstava uređaja u aplikacijama.

Najvažnija stvar koju bi trebalo znati o jQuery Mobile jest da je to knjižnica, a ne jQuery plug-in. Knjižnica koja će uzeti HTML oznake kao ulaz, te će ih formatirati pomoću unaprijed definiranih stilova i koja će ih prilagođavajući prema trenutnim mogućnostima preglednika. jQuery Mobile se oslanja na JavaScript i HTML5 značajke koje su podržane od strane hosting preglednika kako bi ponudio proširene funkcionalnosti. Ova prva karakteristika određuje veliku podršku jQuery Mobilea na mobilnim preglednicima, dok u isto vrijeme objašnjava zašto programeri moraju razvaijati vlastiti JavaScript kod za implementaciju kompleksnih ponašanja, (pohranjivanje podataka ili komunikacija s hardverskim značajkama koji su

izloženi od strane host preglednika (Geolocation, kompas, itd)). Druga važna karakteristika jQuery Mobile je da ne postavlja bilo kakvu strukturu za JavaScript kod vašeg sustava. Općenito govoreći, programeri će upotrebljavati ponašanja pomoću standardnih jQuery sintaksi i idioma, baš kao što bi to bio slučaj sa normalnom web stranicom.

3.5. Zaključak alternativa Cordove

Za razliku od ostalih sličnih okvira, Cordova pruža podršku za gotovo sve mobilne platforme. Za razliku od Ionica koji je po nekim istraživanjima najpopulrniji hibridni framework i koji se bazira isključivo na razvoju mobilnih aplikacija, Cordova aplikacije se mogu koristiti kako za mobilne tako i za desktop aplikacije. Isto tako, za razliku od Sencha Toucha za čije korištenje se mora izdvojiti znatan iznos novca, Cordova je besplatna tako da ga mogu koristiti svi koji to žele.

4. Rad s jQuery Mobileom

4.1. Osnove programske zbirke jQuery

jQuery radni okvir je okosnica jQuery Mobile radnog okvira, stoga je bitno poznavati neke osnove jQuery. Iako nije potrebno, razumjevanje jQuerya će učiniti rad sa jQuery Mobile-om još lakšim. jQuery je robusna Javascript knjižica (library) koja pojednostavljuje Javascipt kodiranje, povećava mogućnosti CSS-a (CascadingStyleSheets), eliminira kompatibilne probleme s cross-browserima i u suradnji je sa CSS3-om. Što znači, brže skriptiranje, manje testiranja i manje kodiranja za različite preglednik.

Uključivanje jQuerya u dokument:

1. direktna referenca prema CDN – hostu

```
<script type=”text/javascript
src=”https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.
js”></script>
```

2. lokalno, uključivanjem individualnih paketa

```
<script type=”text/javascript
src=”js/libs/jquery/jquery.js”></script>
```

Uključivanje Javascript datoteka unutar <*head*> elementa je tradicionalan pristup. No ipak, 80 % vremena krajnjeg korisnika se odvija na front-endu. Većina tog vremena se troši na preuzimanje sredstava kao što su različiti stilovi, slike, skripte itd. Stoga je očigledno važno smanjivanje broja sredstava ali, je i postalo uobičajeno uključivanje Javascripta na kraju HTML dokumenta. Ovaj se pristup koristi zato jer skripte blokiraju paralelno preuzimanje, što znači da se druga sredstva neće preuzeti sve dok se svaka skripta ne preuzme pojedinačno. Kako bi se osigurala svo preuzimanje, skripte se uključuju i prije završne *</html>* oznake.

Isto tako, bolje je koristiti min (minified) verziju u razvijajućim okruženjima, jer je manja od izvorne verzije. Osim toga, iako je pakirana verzija manja od minified verzije, ona zahtijeva procesno vrijeme za dekompresiju datoteka na klijent strani i nije dostupna u najnovijim verzijama. Prema Yahoo-u, prilikom pregled deset najboljih američkih web stranica, minimiziranjem se postiže smanjenje veličine za 21%.[8]

JQuery (slika 4.1) je JavaScript zbirka. Osnove su iste kao i kod pisanja JavaScripta, samo što će se pisati na način koji se koristi u jQuery okviru. Drugim riječima, dok se dodaju poboljšanja određenim osnovama, jezgra JavaScript - varijable, funkcije, uvjetne izjave, i tako dalje - se nije promijenila. Dakle, i dalje će se koristiti var ključna riječ, if i switch izjave i funkcije, ali će se sigurno primijetiti puno dodatnih povećanje i različitih načina pisanja stvari, prilikom pristupa HTML elemenata.

jQuery nudi mnoge poboljšanja JavaScript jeziku i, kao što je već spomenuto, najbolji dio je taj da je cross-browser kompatibilan,[9] tako da se programer ne mora brinuti o pisanje većeg broja verzija iste skripte za obradu različitih preglednika, što prilično olakšava rad. Osobito je korisno kada radite s događajima (events), Ajaxom i drugim funkcionalnostima koje tradicionalno zahtjevaju neke uvjetne izjave kako bi se utvrdilo kako će preglednik interpretirati kod.



Slika 4.1 jQuery

4.1.1. Pristup DOM elementima

Umjesto kontinuiranog korištenja ”document.getElementById()” za pristup HTML elementima unutar web stranice, jednostavno se može koristiti jQuery selektor jQuery () ili $ () funkcija koja je skraćena verzija. Koristeći jQuery selektore ne samo da se daje manje znakova za unos, nego omogućuje mnogo više od pristupa elemenata pomoću Id-a. JQuery selektori, omogućavaju pristup cijelom nizu HTML elemenata ili pristup elementu i objektu po imenu. Selektor jQuery pakira element ili skup elemenata u jQuery objekt, omogućavajući primijenu jQuery metoda za sam objekt. Isto tako elementu se može pristupiti po imenu klase ili korištenjem CSS selektora, kao što su: :first-child, :nth-child.

Primjeri jQuery selektora:

- Za pristup elementu po IDu pomoću jQuery selektora, koristi se #ID selektor, baš kao što bi koristio sa CSS-om kod dodavanja klase za element po ID:

```
$ ('# Foo');
```

- Za pristup element po imenu klase, koristi se selektor .class, baš kao što bi se koristio pri izradi CSS klase. Ako postoji više elemenata sa istim nazivom klase, svi će oni biti izabrani:

```
$ ('. Foo');
```

- Prilikom ciljanja određenog elementa sa imenom klase, dodaje se naziv elementa prije imena klase:

```
$ ('Div.foo');
```

- Može se čak i ciljati određene elemente ili skup elemenata pomoću selektora elementa:

```
$ ('Div');
```

4.1.2. Upravljanje događajima i funkcijama

Događaji se koriste u jQuery prilikom interakcije korisnika s web stranicom, kao što su klikovi mišem. Događaji su vrlo laki za upravljanje u jQuery-u, pouzdani su i funkcionalni kroz sve glavne preglednike, što je velika stvar, zato što to nije uvijek slučaj s tradicionalnim JavaScript-om. Uobičajeno, događaji se koriste za obavljanje neke vrste funkcija.

U sljedećem primjeru, klik događaj vezan je uz div element s "foo" imenom klase:

```
$ ('Div.foo "). click (function (event) {
// kod
});
```

Vezivanjem klik događaja za div, funkcija je povezana s klikom miš na određeni div. Dakle, kada se bilo kada klikne div.foo, izvršit će se kod koji se nalazi unutar funkcije.

- Jquery selektor se koristi za selektiranje "div.foo", koji tada postaje jQuery objekt.
- Div.foo objekt tada koristi klik metodu za pokretanje funkcije kada se "div.foo" klikne.
- Funkcija se koristi za izvršavanje koda. Isto tako, funkcija ima pristup "event" objekt, koji se prenosi kao argument. u ovom slučaju, to je event argument u funkciji.
-

- **Document.ready() funkcija**

Uobičajena praksa među programerima je čekanja da se web stranica učita prije izvršavanja bilo kakvog JavaScript-a. Razlog tome je taj da se osigura da su elementi web-stranice dostupni prije nego što se pokuša njima pristupiti ili manipulirati. Pokušaj pristupa nedostupnim elementima dokumenta može dovesti do neočekivanog ponašanja i potencijalnog prekida naknadnog JavaScript-a. Uz tradicionalni JavaScript, najčešći način čekanja da se stranica učita je korištenje window.onload događaja. Međutim, ovaj pristup se odvija nakon što se dokument učita, zato jer čeka sve slike i bannere da se učitaju unutar web stranice, što može odgoditi pokretanje skripti. Srećom, jQuery pruža ready događaj koji omogućuje kod da odmah odgovori kada dokument postane dostupan.

Primjer $(document).ready funkcije

```
$ (document) .ready (function () {
// kod
});
```

Prvo je potrebno odabrati samome objekt dokumentu, a zatim primijeniti ready događaj. Kada se ready događaj pokrene, izvršiti će se funkcija koja će sadržavati kod. Potrebno je dodati kod ready događaj kako bi se osiguralo da je dokument spreman prije nego što se pokuša primijeniti klik događaj:

Primjer $(document).ready funkcije sa klik događajem

```
$(document).ready(function() {
$('div.foo').click(function() {
    alert($(this).html());
 });
});
```

Skraćena verzija koja eliminira potrebu za pristupom dokumentu (document) i postavlja ready događaj.

```
$(function() {
 $('div.foo').click(function() {
     alert($(this).html());
 });
});
```

- **Primjena specijalnih efekata**

JQuery je dobro poznat po specijalnim efektima koji omogućuju stvaranje efekata bez posebnih pluginova, kao što je Flash. Knjižnica pruža mnoge tehnike za uključivanje animacije u web stranicu. Animacije mogu stvoriti vizualno privlačanu web stranicu ili može poslužiti i u druge, praktične svrhe, kao što je pružanje vizualne povratne informacije korisniku. Mnoge metode animacije su uključene u jQuery okviru, kao što su: fadeIn, fadeOut, fadeTo, show, toggle, slideUp i slideDown, itd.

Primjer jQuery koda za animaciju

```
$(document).ready(function() {
    $('div.foo').click(function() {
          $(this).animate(function(){
           height: '+=50'
          }, 1000, function() {
                                  // kod kad se animacija završi
          });
 });
});
```

U ovom primjeru, kada se div.foo klikne, njegova visina se povećana za 50 piksela tijekom razdoblja od 1 sekunde (1000 milisekundi), a kada animacije završi, povratna funkcija će izvršiti kod. Povratna funkcija koristi se za odgodu izvršenja koda dok se nešto drugo ne dogodi.

4.1.3. AJAX

Ajax (slika 4.2) je kratica za asinkroni JavaScript i XML. Asinkroni znači da se može napraviti zahtjev prema serveru putem Hypertext Transfer Protocol (HTTP) i dalje obrađivati druge podatke, dok se čeka na odgovor. Na primjer, mogu se napraviti pozivi na server strani skripti za preuzimanje podataka iz baze podataka kao XML, slanje podataka na prema server strani skripti koji će biti pohranjeni u bazi podataka, ili jednostavno učitati XML datoteku za



Slika 4.2 Ajax

popunjavanje stranica na web stranici, bez osvježavanja web stranice. Funkcionalnost dostupna preko jQuery radnog okvira čini razvoj Ajaxa lakšim od tradicionalnog JavaScripta, zahtijevajući pritom manje kodiranja i nudeći dodatne metode i događaje. Količina jQuery koda potrebna za rukovanjem Ajaxa je minimalna u usporedbi s tradicionalnim JavaScriptom, čak i kada se razvoja kompleksna funkcionalnost, što u konačnici čini razvoj znatno bržim. [10]

Primjer Ajax requesta

```
jQuery.ajax({
 url: 'url for request',
 success: function(xml) {
          // obrada odgovara
 }
});
```

Ovaj kod koristi jQuery ajax metodu koja uključuje URL za zahtjev, te obrađuje povratnu funkciju uspjeha s anonimnom funkcijom. Mnoga svojstva se mogu koristiti u postupku jQuery Ajax metode, kao što su vrsta zahtjeva, POST ili GET (zadani), korisničko ime i lozinku, i crossDomain. Ovaj primjer pokazuje Ajax metodu u najosnovnijem obliku za ilustraciju jednostavnosti jQuery Ajax poziva. Ajax se koristi u jQuery Mobile radnom okviru za obradu promjena na stranica i učitavanje stranice.

JQuery je glavna okosnica jQuery Mobile radnog okvira, tako da je korisno znati neke osnove jQuery prije nego se počne sa razvojom jQuery Mobile projekata. Iako nije potrebno, razumijevanje jQuery će omogućiti lakše razumjevanje jQuery Mobilea, pogotovo ako postoji interes za pisanje bilo kakvih prilagođenih funkcionalnosti. jQuery je učinio poboljšanja korisničkog sučelje daleko lakšim pojednostavljujući JavaScript i kombinirajući ga sa sintaksom CSSa. Radni okvir pruža način koji omogućava manje potrebnog vremena za pisanje koda, trošenje manje vremena na testiranje i postizanje kompleksnijih rezultata u kraćem vremenu. jQuery omogućuje poboljšanja unutar aplikacije dodavanjem prilagođenih interakcije, kao i efekata za pružanje vizualne povratne informacije korisniku, koja će u konačnici stvoriti bolje korisničko iskustvo.

4.2. Osnove HTML 5

Polazišna točka za sva razvoj jQuery Mobile je HTML5 koji ima ključnu ulogu u jQuery Mobile okviru. HTML5 (slika 4.3) pruža vezu za sve, od definiranja kako se web stranice renderira u mobilnim, tabletima ili računalnim preglednicima do atributa koji definiraju grafičke elemente,teme itd. Dostupnost je glavni fokus i prioritet jQuery Mobile okvira. To je jedan od razloga zašto se okvir temelji na semantičkom HTML i zašto je dostupan na najširem mogućem raspon uređaja. Tehnike koje se koristi za okvirnu podršku reda preglednika i omogućuju pristup za korisnike s čitača zaslona takvim kao što su VoiceOver za Apple iPhone. jQuery Mobile koristi tri razine razred za potporu okvira: A, B, i C. A je puna; B je puna, minus Ajax; i C je osnovna podrška. Standardni, semantički HTML vam daje sigurnost pri pružanja mobilnog pregleda web koji je dostupan najvećem krugu korisnika bez obavljanja dodatnog testiranje.



Slika 4.3 HTML5

4.2.1. Viewport meta oznaka

Viewport meta oznaku je uveo mobile Safaria internetski preglednik dopuštajući tako web programerima da kontroliraju veličinu i omjer prozora. Mnogi drugi glavni mobilni preglednici sada podržavaju tu oznaku. Oznaka se koristi za postavljanje raspored prozor preglednika za poboljšanu reprezentaciju web stranica. Mobilni web preglednici očito imaju puno manju veličinu zaslona od desktop preglednika i zbog toga oni imaju različit izgled prikaza. Trenutno standardna opcija za ispravan prikaz web stranice je korištenje viewport meta oznake. Ova oznaka govori pregledniku da optimizira web stranice na temelju najbolje širine za gledanje u tom pregledniku. Korištenje viewport meta oznake za postavljanje prikaza, potrebno je postavite naziv meta tag u ViewPort, zatim koristiti atribut sadržaja za postavljanje svojstva i vrijednosti koje zadovoljavaju potrebe. U atributu sadržaja, mogu se definirati svojstva i vrijednosti koje želite postaviti za prikaz. Za dodavanje više svojstava viewport meta tag, potrebno je stvoriti zarezom razgraničeni popis svojstava i vrijednosti.

Najčešća postavka za viewport meta tag u mobilnom developmenta:

```
<html>
 <head>
 <meta name=”viewport” content=”width=device-width, initial-
scale=1”>
             <title>Page Title</title>
 </head>
<body>
</body>
</html>
```

4.2.2. Data-attributes

HTML5 data atributi omogućuju pohranu podataka koji nisu viđeni od strane korisnika. Drugim riječima, podaci nisu renderirani ili se čak niti ne koriste u pregledniku. U prethodnim verzijama HTMLa, prilagođeni podaci se često pohranjuju u naslovu (title), rel, klasi (class) ili ID atributima, ili pak u skrivenim HTML elementima. Podaci pohranjeni u tim atributima su dostupni JavaScriptu. Postoje mnogi razlozi zašto je pristup tim podacima koristan putem JavaScripta; mogu se koristi iza kulisa za stvaranje prilagođenih funkcionalnosti ili za prikazivanje korisniku kada se događa nekakva specifična interakcija. Dobar primjer korištenja prilagođenih podataka na web-stranici su galerije slika. Naslov i opis se mogu dodati slici te se zatim koristi JavaScript za prikaz informacija o slici koju korisnik gleda. Poanta je ta da je to pogrešna upotreba HTML atributa i sve do sada, to je bio jedini način za obradu određenih funkcionalnosti koje inače nisu podržana. Srećom, programeri mogu početi koristiti HTML atribute na odgovarajući način uz pomoć data-atributa.

HTML5 specifikacija podržava bilo koji atributi koji počinje s data kao područje za spremanje podataka. Svako ime koje se želi dodati na data-prefiks će biti podržano. Na primjer, ako se dodaje data atribut slici koja pripada određenoj galeriji slika ili se mogu dodavati prilagođeni data-title i data-description atributi.

```
<img src=”image-path/img.jpg” data-title=”My image title” data-
description=”My image description”>
```

Definiranje jQuery Mobile headera:

```
<body>
   <div data-role=”header”>
   <h1>Header</h1>
   </div>
```

```
</body>
```

Obeshrabrujuće je misliti da je potrebno razumjeti HTML5 i jQuery kako bi se mogli koristiti sa jQuery Mobile okvirom. Srećom, okvir je izgrađen na način koji podržava korisnike sa različitim razinama vještina. Nije potrebno nužno znati HTML5 i jQuery, jer okvir pruža predložak za HTML5 koji se može preuzeti direktno sa službene stranice http://jquerymobile.com/ i jQuery je dostupna one programere koji ga žele koristiti. Jedina stvar koja je apsolutno neophodno je razumjevanje korištenja atributa (data- attributes), jer je to način na koji se grafički elementi dodaju unutar aplikacije bez korištenja jQuerya.

4.3. Izrada osnovne aplikacije korištenjem jQM razvojnog okvira

Postoji mnogo načina za izgradnju jQuery Mobile web projekta. Međutim, u svom najjednostavnijem obliku, okvir funkcionira pomoću HTML5 i jQuery knjižnice za transformaciju ”data-attributes” elemenata u komponente, koje je vrlo lako uklopiti u već postojeći razvoj. Najosnovniji dijelovi okvira su stranice i alatne trake no prije svega, potrebno je uključiti jQuery Mobile okvir u HTML5 web projekt. [11]

Postoje dva načina za dodavanje jQuery Mobile okvir u web projekt. Prvi način je preuzmanje pojedinačnih paketa, koji uključuju punu i minified verzije JavaScript knjižnice i CSS datoteke. Također je potrebno preuzeti jQuery knjižnicu. Drugi način izravno pozivanje projekta sa CDN datotekama.

Uključivanje jQuerya Mobilea u dokument:

1. direktna referenca prema CDN – hostu

```
<link rel=”stylesheet” href=”http://code.jquery.com/mobile/1.0/
jquery.mobile-1.0.min.css” />
<script src=”http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js”></script>
<script src=”http://code.jquery.com/mobile/1.0/jquery. mobile-
1.0.min.js”></script>
```

2. lokalno, uključivanjem individualnih paketa

```
<script type=”text/javascript
src=”js/libs/jquery/jquery.js”></script>
<script type=”text/javascript src=”js/libs/jquery/jquery-
mobile/jquery.mobile.js”></script>
```

4.3.1. Stranice i alatne trake

U jQuery Mobile, stranica je doslovno definirana pomoću HTML elementa s ”data-role” atributom koji ima vrijednost stranica (page) kao što se vidi iz navedenog primjera:

Primjer jQuery Mobile stranice

```
<div data-role=”page”></div>
```

jQuery Mobile će pretvoriti ovaj element (najčešće korišten) u komponentu stranice. Sve što se treba učiniti je koristiti ovu oznaku i ugraditi okvir u glavu (<head>) u web projekt. Unutar data-role = "page" elementa, mogu se koristiti bilo koje HTML oznake. Najčešći elementi koji se mogu vidjeti unutar data-role = "page" elementa su elementi s data-role atributima koji ima vrijednost header, content i footer. U jQuery Mobileu header i footer data-role atributi se smatraju kao alatne trake, dok se data-role=”content” koristi za definiranje područja sadržaja web projekta u koju se mogu dodati bilo koje HTML oznake (slika 4.4).



Slika 4.4 jQM stranica

Sljedeći primjer pokazuje tipičnu HTML strukturu za osnovnu jQuery Mobile web stranicu pomoću data-role atributa.

```
<body>
 <div data-role=”page”>
<div data-role=”header”>
            <h1>Page title</h1>
            </div>
            <div data-role=”content”>
            Body copy
            </div>
            <div data-role=”footer”>
            Copyright
            </div>
      </div>
```

4.3.2. Struktura mobilnih stranica

Postoje dva načina za strukturiranje web stranice u jQuery Mobileu. Prvi način je da se uključe sve stranice unutar iste datoteke dok je drugi način stvaranje zasebnih datoteka, kao

što je slučaj kod tipičnih web stranica. Da bi se stvorilo više stranica unutar jedne HTML datoteke, višestruke stranice s ”data-role” atributima treba razgraničiti prilikom čega se omogućuje da svaka stranica sadržii vlastito zaglavlje, sadržaj i podnožje.

Sljedeći primjer daje osnovnu ideju o tome kako u jQuery Mobile izgleda višestruka stranica unutar jedne HTML datoteke.

```
<div data-role=”page” id=”page-one”>
     <div data-role=”header”>
          <h1>Page 1</h1>
     </div>
     <div data-role=”content”>Body copy for page 1</div>
      <div data-role=”footer”>Copyright</div></div>
<div data-role=”page” id=”page-two”>
      <div data-role=”header”>
           <h1>Page 2</h1>
     </div>
      <div data-role=”content”>Body copy for page 2</div>
      <div data-role=”footer”>Copyright</div> </div>
```

Izrada stranica i uključivanje jQuery Mobile okvira je jednostavna jer otkrivanjem i spoznajom načina kako radni okvir funkcionira, otkrivaju se pojedinosti koje čine ovaj okvir tako snažnim i velikim. ”Data” atributi pružaju način za pohranjivanje prilagođenih podataka koje korisnik ne vidi no koji ostaju dostupni JavaScriptu pa tako u konačnici i jQueryu. Upravo je to razlog zašto data atributi igraju veliku ulogu u jQuery Mobileu. Radni okvir koristi ove atribute za transformaciju osnovnih HTML elementi u stilizirane grafičke elemente. [12]

4.3.3. Vrste poveznica kod jQM-a

JQuery Mobile podržava standardne vrste HTML poveznica, kao i cijeli niz prilagođenih vrsta poveznica koje se odnose na mobilno iskustvo. Sljedeći primjeri prikazuju popis podržanih vrsta poveznica koje su dostupne u jQuery Mobile radnom okviru. Svaki primjer prikazuje mogućnosti kategorizirane na temelju njihovih krajnji rezultat i podrške Ajaxu.

```
<a href=”http://www.jquerymobile.tv”>Hyperlink within same
domain</a>
```

Standardni HTML link koji je transformiran od JQM okvira za korištenje Ajaxa, uključuje tranzicije stranica i podržana je u povijest stranice.

```
<a href=”http://www.jquerymobile.tv” data-rel=”dialog”> Open a dialog</a>
```

Opcija koja se koristi za dijaloge koja nije zapisana u povijesti.

```
<a href=”http://www.jquerymobile.tv” data-rel=”back”>Back button</a>
```

Ova opcija se koristi za navigaciju u povijest stranice i služi kao opcija za pružanje back buttona sa stranice ili dijaloga.

```
<a href=”http://www.jquery.com”>External hyperlink</a>
```

Poveznica prema stranici na vanjskoj domeni koja automatski onemogućava Ajax.

```
<a href=”http://www.jquery.com” rel=”external”>External hyperlink</a>
```

Po defaultu ovaj atribut definira hiperlink kao vanjski, koji ne samo da onemogućava Ajax nego ga i miče iz povijesti stranice te osvježava stranicu.

```
<a href=”tel:15556667777”>Phone Number</a>
```

Poveznica inicijalizira telefonski poziv.

```
<a href=”mailto:jdoe@jquerymobile.tv”>Email link</a>
```

Poveznica inicijalizira email.

4.3.4. Tranzicije stranica

U jQuery Mobile okviru se može koristiti veliki broj tranzicija stranica. Sve se tranzicije baziraju na CSS efektima. Kada se koristi Ajax navigacija, tranzicije funkcioniraju između poveznica ili obrazaca.

Metode: **slide, slideup, slidedown, pop, fade, flip.**

Svaka se od navedenih tranzicija mogu lako postaviti na globalnoj razini kao zadana opcija pomoću svojstva ”defaultPageTransition”. Za ispravno korištenje, potrebno je to svojstvo vezati za mobileinit događaj, koji je dostupan putem API-ja preko jQuery Mobile objekta.

Primjer koda defaultnih tranzicija

```
$(document).bind(“mobileinit”, function() {
    $.mobile.defaultPageTransition = ‘fade’;
});
```

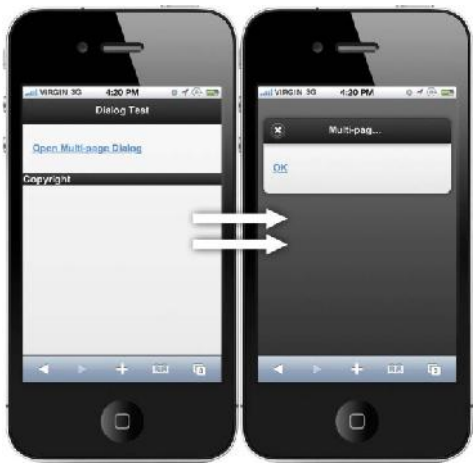
Isto tako, tranzicije mogu biti postavljena na pojedinim poveznicama kako bi se premostile defaultne tranzicije stranica. Ova opcija je korisna u brojnim situacijama poput stvaranja pop-up prozora koje koristite pop tranzicije.

Primjer koda pop-up prozora sa pop tranzicijom

```
<a href="popup.html" data-transition="pop">Pop-up</a>
```

4.3.5. Dijaloški prozor

Poput i ostalih stvari u jQuery Mobile okviru, stvaranje dijaloških prozora je jako jednostavno. Za stvaranje dijaloškog prozora potrebno je koristiti data-rel atribut i postaviti njegovu vrijednost na dialog: data-rel="dialog" (slika 4.5). Dijaloški prozori mogu biti uključeni u stranicu ili stranice u jednoj HTML datoteci baš kao i višestranični predložak ili mogu biti vanjske web stranice poput jednostraničnog predloška. Da bi se stvorio višestranični predložak koji uključuje dijaloški prozor, dodaje se još jedna jQuery Mobile stranica. Razlika je samo u tome kako se stranice povezuje sa ostalim stranicama. Drugim riječima, svaki stranica može biti dijaloški prozor a, ono što ga čini dijaloškim prozorom je način na koji je otvoren. [13]



Slika 4.5 Dijaloški prozor

Primjer prikazuje poveznicu sa data-rel atributom koji otvara stranicu u dijaloškom prozoru:

```
<div data-role="page">
     <div data-role="header"><h1>Dialog Test</h1></div>
     <div data-role="content">
       <p><a href="#multipage-dialog" data-rel="dialog">Open Multi-
page Dialog</a></p> </div>
     <div data-role="footer">Copyright</div>
     </div>
     <div data-role="page" id="multipage-dialog">
          <div data-role="header">
            <h1>Multi-page dialog window</h1></div>
          <div data-role="content">
   <p><a href="dialog.html" data-rel="back">OK</a></p> </div></div>
```

JQuery Mobile dijalog se također može koristiti kao pop-up prozor. Umjesto učitavanja dijaloga koji pruža ili traži povratnu informacij, dijalog može biti još jedan web stranica koja uključuje kopiranje, medije, i tako dalje. Po defaultu, dijalog (ili pop-up) prozor uključuje gumb za zatvaranjei, tako da se ne trebaju dodavati svi ostali linkovi ili funkcionalnost. Dijaloški prozori također uključuju tranzicije. Zadana tranzicija je pop, ali se može promijeniti na flip ili slidedown. Za promjenu tranzicije, dodaje se data-transition atribut na poveznicu koja uključuje data-rel = "dialog" postavku.

Primjer koda za dodavanje tranzicija dialogu

```
<a href=”popup-window.html” data-rel=”dialog” data-transition=
”slidedown”>Play now &gt;</a>
```

4.3.6. Gumbi

JQuery Mobile pruža način za stvaranje gumbića s HTML i ulaznim elementima. Gumbići pružaju korisne alternative standardnim poveznicama i inputima stvaranjem većeg klikajućeg područje. Za pretvaranje poveznice u gumb, postavlja se data-role atribut s vrijednosti ”button”.

Primjer koda za gumb poveznicu

```
<a href=”#” data-role=”button”>Hyperlink button</a>
```

Po defaultu, gumbići imaju punu širinu sadržavajućeg elementa ali, se mogu postaviti da budu u istoj liniji pomoću data-inline atributa. Kada se ovom atributu postavi vrijednost na istina (”true”), gumbići se smješteni u istoj liniji čija je širina definirana s tekstom naziva poveznice.

Primjer koda za gumb poveznicu smještenu u istoj liniji

```
<a href=”#” data-role=”button” data-inline=”true”>Hyperlink
button</a>
```

Tabela 4.1 Tablica sa atributima ”button ” komponente

data- ATTRIBUTES	VALUES
data-corners	true | false (true is the default value)
data-icon	home | delete | plus | arrow-u | arrow-d | check | gear | grid | star | custom | arrow-r | arrow-l | minus | refresh | forward | back | alert | info | search
data-iconpos	left | right | top | bottom | notext (left is the default value)
data-iconshadow	true | false (true is the default value)
data-inline	true | false (false is the default value)
data-shadow	true | false (true is the default value)
data-theme	swatch letter (a–z)

Isto tako, gumbi mogu uključivati i ikone pomoću ”data-icon” atributa. Broj defaultnih ikona je poprilično impresivan a, ikone se kao i sve ostalo u okviru, vrlo lako koriste.

Tabela 4.2 Tablica ikona kod jQM-a

NAME	ATTRIBUTE	ICON
Left arrow	data-icon="arrow-l"	
Right arrow	data-icon="arrow-r"	
Up arrow	data-icon="arrow-u"	
Down arrow	data-icon="arrow-d"	
Delete	data-icon="delete"	
Plus	data-icon="plus"	
Minus	data-icon="minus"	
Check	data-icon="check"	
Gear	data-icon="gear"	
Refresh	data-icon="refresh"	
Forward	data-icon="forward"	
Back	data-icon="back"	
Grid	data-icon="grid"	
Star	data-icon="star"	
Alert	data-icon="alert"	
Info	data-icon="info"	
Home	data-icon="home"	
Search	data-icon="search"	

Tablica 4.2 Tablica ikona kod jQM-a

4.3.7. Liste

Liste se u jQuery Mobile radnom okviru često koristi za prikazivanje podataka.Okvir nudi mnoštvo mogućnosti oblikovanja listi s kojima se može postići bilo kakav uzorak dizajna koji se može zamisliti, poput osnovnih lista, numerirane lista, nenumeriranih lista, ugniježđenih lista itd.



Slika 4.6Osnovna lista

Jedan od najčešćih uzoraka dizajn koje se mogu vidjeti s jQuery Mobile listama je osnovna lista (slika 4.6). Osnovna lista je jednostavna nenumerirana lista koje sadrže stavke s poveznicama i često se koristi za prikazivanje navigacije. jQuery Mobile poboljšava ovu liste kada se koristi data-role atribut s vrijednošću ”listview” koji je pričvršćen na otvarajuću oznaku od nenumerirane lista.

Sljedeći primjer prikazuje kako stvoriti osnovnu listu koristeći listview data-role atribut:

```
<ul data-role=”listview”>
 <li><a href=”#”>Home</a></li>
 <li><a href=”#”>Articles</a></li>
 <li><a href=”#”>Videos</a></li>
 <li><a href=”#”>Books</a></li>
 <li><a href=”#”>More</a></li>
</ul>
```

Listview oznaka je poboljšana od strane okvira tako da proizvodi listu poveznica koje uključuju ikone strjelica, pozadinske sjene itd...

4.3.8. Teme jQuery Mobilea

jQuery Mobile uključuje kompletan tematski okvir koji omogućava prilagođavanje uzoraka boja i setova ikona za stvaranje prilagođenih tematskih stranica, alatnih traka, sadržaja, obrazaca, lista, gumbića itd. Okvir koristi CSS3 za mnoga svoja poboljšanja, kao što su sjene, gradijenti i zaobljeni kutovi, tako da se niti jedna slika ne koristi za ove vizualne aspekte, što temu više laganom i brzom za učitavanje. jQuery Mobile ima tematski sustav koji

sadrži pet uzoraka definiranih slovima od "a" do "e" (slika 4.7). Uzoraka se mogu kombinirati i podudarati unutar web projekta kako bi se omogućila potpunu modificiranje. Defaultni uzorci sadrže boje i teksture, padding, i predefinirane dimenzije koje ih odvajaju jedni od drugih.

Tabela 4.3 Tablica defaultnih uzoraka boje

SWATCH	COLOR	PRIORITY
a	Black	Highest
b	Blue	Secondary
c	Gray	Baseline
d	White	Alternate secondary
e	Yellow	Accent



Slika 4.7 Teme jQM-a

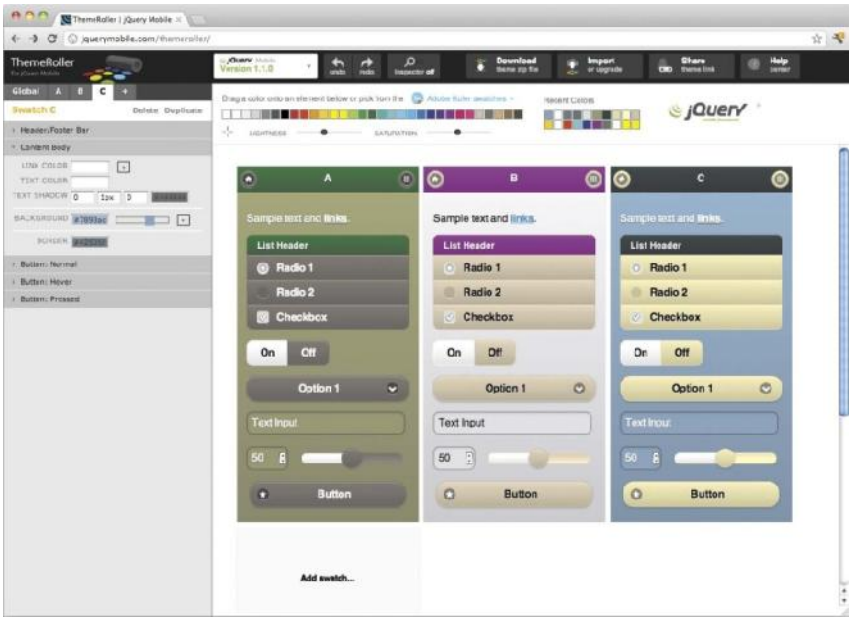
Korištenje defaultnih uzoraka boja se ostvaruje pomoću data-theme atributa koji se može primjeniti na bilo koji HTML element sa jQM okvirom. Bilo da se radi o setu gumbića, listama ili samoj strukturi stranice, okvir pruža mogućnost definiranja različitih tema za svaki postojeći element.

Sljedeći primjer koda prikazuje implementaciju tema

```
<a href="#" data-role="button" data-theme="a">Swatch A</a>
<a href="#" data-role="button" data-theme="b">Swatch B</a>
<a href="#" data-role="button" data-theme="c">Swatch C</a>
<a href="#" data-role="button" data-theme="d">Swatch D</a>
<a href="#" data-role="button" data-theme="e">Swatch E</a>
```

4.3.9. Themeroller

ThemeRoller (https://themeroller.jquerymobile.com/) (slika 4.8) je "drag and drop" sučelje koje omogućuje da povučete boje u blokovima radi oblikovanja uzoraka boja. Alat počinje rad sa tri vrste uzoraka koji su zastupljeni s nekoliko komponenti. Na vrhu se nalaze, boje blokova koje se mogu povući i ispustiti na zaglavlja, gumbiće, tekst, pozadine, poveznice itd. Isto tako, korisnik u svojoj kreaciji može biti detaljniji tako što će lijevoj bočnoj traci prilagoditi svoj izbor, prilikom čega može definirati fontove, sjene, zaobljene kutove, ikone itd. [14]



Slika 4.8 ThemeRoller sučelje

4.3.10. Zaključak jQM-a

jQuery Mobile je Javascript knjižica koja uz korištenje HTML5 data atributa omogućuje brže, efikasnije i estetski ljepše grafičko uređivanje elemenata koji se nalaze unutar aplikacije. Isto tako, okvir omogućuje stvaranje bilo kakve funkcionalnosti koju programer želi ostvariti. Okvir podržava mnoge načine za dodavanje prilagođenih funkcionalnosti koje programeru omogućuju stvaranje moćnih i interaktivnih mobilnih web aplikacija i stranica. Uključivanjem jQuery Mobile okvira u web aplikaciju, okvir pretvara kod iz semantičkog HTML u bogato, interaktivno i pristupačno mobilno iskustvo koje se odvija pomoću jQuerya i CSSa. jQuery Mobile pristup čini mobilni razvoj web aplikacija nevjerojatno jednostavnim, brzim i učinkovitim. jQuery Mobile nije stvoren isključivo za programere, dizajneri imaju pristup jQuery UIu, koji pruža potpunu kontrolu nad dizajnom mobilnih web aplikacija. Ugrađeni grafički elementi, kao što liste, dijalozi, alatne trake, search, i različiti elementi su prilagodljivi putem tematskog okvira.

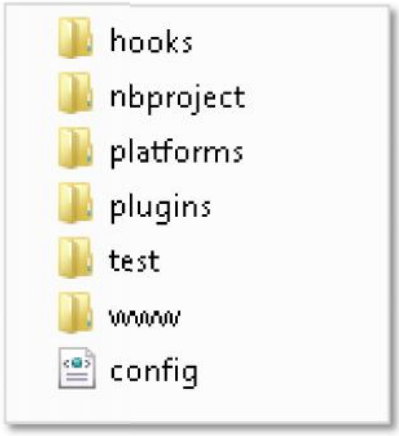
5. Izrada aplikacije ”Sjeveroljubac”

U ovom poglavlju će biti opisan postupak izrade aplikacije ”Sjeveroljubac”. Kada sam razmišljao o samoj aplikaciji odnosno o temi i svrsi koju bi aplikacija trebala pokrivati, shvatio sam da aplikacija mora biti vezanu uz Sveučilište Sjever, odnosno uz student koji pohađaju navedeno Sveučilište. Glavna poanta aplikacije je ta da student što brže pristupi ili dođe do informacija vezanim uz Sveučilište.

Ponekad je studentima glavni razlog ili problem doći do željenih informacija a, zbog tog gubitka vremena, studenti često ne ispunjavaju svoje obaveze pa sukladno s tim gubi potrebne bodove ili čak ne odlaze na ispite. Stoga je po mom mišljenju sama aplikacija koncipirana tako da se sastoji od poveznica pomoću kojih će studentu omogućiti da sa jednog mjesta, pristupi ili dođe do željenih informacija. Poveznice od kojih se aplikacija sastoji su službena stranica Sveučilišta, službeni mail Sveučilišta, moodle i studomat. Upravo su te poveznice koje studenti najlakše koriste kroz svoje studentsko razdoblje. Isto tako, budući da je Sveučilište Sjever okosnica odnosno temelj same aplikacije, unutar aplikacije su stavljene i poveznice pomoću kojih se student može povezati sa Sveučilištem na društvenim mrežama. Kako smo vidjeli iz teorijskog dijela, Cordova podržava gotovo sve mobilne platforme i dodavanje platformi nije problem u radu. Budući da se za izgradnju apliakcije za različite mobilne platforme mora imati odgovarajući SDK (Software Development Kit), bilo je potrebno koristiti se u radu sa Mac računalom. Stoga je aplikacija napravljena za Android i iOS mobilne platforme budući jer su upravo te dvije platforme najčešće korištene.

5.1. Struktura projekta

U prvom dijelu opisana je kreacija Cordova projekt. Sama struktura projekta odnosno direktoriji od kojih se projekt sastoji su koncipirani tako da omogućavaju što lakše snalaženje i što bolju organizaciju (slika 5.1). Direktoriji koji je navažniji za sam rad je ”www” direktorij unutar kojeg se nalazi glavna index.html stranica unutar koje se odvija svo programiranje i sve izmjene koje se rade, rade se u ovom direktoriju. Isto tako, svi dodaci poput jQuery knjižica i CSS stilova se dodaju unutar navedenog direktorija te se svi oni



Slika 5.1 Struktura projekta

nakon izgradnje (build) projekta kopiraju u direktorij platforme unutar kojeg se nalaze mobilne platforme za koje će projekt biti namjenjen. U ovom slučaju to su Androiod i iOS platforme. Struktura ”platforms” direktorija je gotovo ista kao i struktura www direktorija uz dodatak različitih pluginova i Cordova knjižica pomoću koje projekt ostvaruje svoju nativnu namjenu. Unutar glavnog direktorija se nalazi i globalna konfiguracijska dataoteka pomoću koje projekt funkcionira unutar natvinog okvira.

WWW direktorij je koncipiran kao bilo kakva tradicionalna web stranica i kao što sam već rekao, organizirana je tako da omogućava što lakše snalaženje unutar samog dokument. Prilikom dodavanja jQuery i jQuery Mobile knjižica, bilo je potrebno sukladno sa organizacijom, kreirati direktorij jquery unutar kojeg se nalaze navedene knjižice. JQuery Mobile CSS stilovi su nakon preuzimanja spremljeni css direktorij. Osim klasičnog jQuery Mobile stila, unutra se nalazi stil koji povećava opseg ikona koje se mogu koristit u radu. Unutar js direktoriji se nalazi index.js skripta koja obrađuje pokretanje aplikacije, preusmjeravanje i druge funkcije te zahtjeva druge module za dodavanje funkcionalnosti.

5.2. Struktura aplikacije

Budući da je aplikacija namjenjena za sve studente Sveučilišta Sjever i one koji će to i postati bilo je potrebno kreirati dvije poveznice koje će se odnositi na status korisnika kojeg on ima u odnosu na Sveučilište. Korisnik će za početak morati odabrati da li već studira na Sjeveru ili je početnik. Stoga je uz ***index.html*** bilo potrebno kreirati stranicu za početnike, ***postani_sjeveroljubac.html.*** Isto tako, zbog lakše navigacije i funkcionalnosti aplikacije, za kategoriju odnosno status Student, dodatno su kreirane stranice ***odjeli.html, profesori.html, i rss.html***. Stranica odjela će se sastojati od liste koje će voditi na stranice za smjerova odnosno odjela koji se nalazi u sklopu Sveučilišta. Kako se u sklopu Sveučilišta Sjever nalazi čak devet smjerova smještenih u Varaždinu i Koprivnici te je zbog navedene količine bilo potrebno kreirati stranicu koja će sadržavati listu u kojoj će korisnik odabrati odjel. Isto tako, za potrebe same aplikacije bilo je potrebno kreirati nekakvu uvodnu stranicu na kojoj će se nalaziti informacije vezane uz samu aplikaciju. Prilikom kreiranja, odlučio sam se za izradu višestruke jQuery Mobile stranice unutar istog HTML dokumenta. Kako smo u teorijskom dijelu rada ustanovili, jQM stranice su definirane dodavanjem data-role atributa koji ima vrijednost page i sve što se nalazi unutar stranice postaje relativno za tu stranicu.

5.3. Višestranični predložak

Sad kada je stvoren koncept odnosno razrada plana kako će aplikacija izgledati i od kojih stranica će biti sačinjena, došlo je vrijeme da krenemo sa programiranjem. Prije pisanja bilo kakvog koda, trebalo je uključiti jQuery i jQuery Mobile knjižice u projekt. Navedene knjižice preuzete su sa službene jQuery stranice te su stvorene poveznicu unutar glave aplikacije. Isto tako, uz javascript jquery knjižice, u projekt je uključen i jQuery Mobile CSS stil te stil koji sadrži dodatni paket ikona koje će se koristiti u radu.

Primjer koda "glave" dokumenta

```
<head>
<meta charset="UTF-8">
<meta name="format-detection" content="telephone=no">
<meta name="viewport" content="user-scalable=no, initial-scale=1,
maximum-scale=1, minimum-scale=1, width=device-width">
<script type="text/javascript"
src="jquery/jquery/jquery.js"></script>
<script type="text/javascript"
src="jquery/jquery_mobile/jquery.mobile-1.4.5.js"></script>
<link rel="stylesheet" type="text/css" href="css/index.css">
<link rel="stylesheet" type="text/css" href="css/jqm-icon-pack-
fa.css">
<link rel="stylesheet" type="text/css" href="css/jquery.mobile-
1.4.5.min.css">
        <title>Sjeveroljubac</title>
</head>
```

<meta name="format-detection" /> // određuje da li će se ili ne telefonski brojevi u HTML sadržaju pojaviti kao poveznice. Korisnik može kliknuti poveznicu s telefonskim brojem za pokretanje telefonskog poziva na navedeni broj telefona.

<meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width"> // skaliranje širine zaslona prema širini određenog uređaja sa max i min vrijednostima

5.4. Stranica index.html

Nakon uključivanja jQuery Mobile radnog okvira, moglo se započeti sa programiranjem. Za početak je bilo potrebno stvoriti homepage stranica. Homepage stranica se kreira kao i bilo koja druga stranica u jQM okviru jedina razlika je u to me što joj je zadan id="homepage". Struktura svake jQM stranice je podijeljena na header, content i footer. Kao što je u teorijskom dijelu već rečeno, najčešće korištena HTML oznaka unutar okvira je . Stoga je cijela struktura stranice razdvojena sa div elementima koji imaju različite vrijednosti

odnosno header, content i footer. Budući da u teorijskom dijelu nisu opisavani svi atributi koji se mogu koristit u jQM okviru, kroz praktični dio će biti opisati atribute koji su korišteni.

Primjer koda zaglavlja (headera) dokumenta

```
<div data-role="page" id="homepage"><!-- početak homepage -->
   <div data-role="header"  data-position="fixed">
       <h3>Sjeveroljubac</h3>
       <a href="#dialog" data-iconpos="notext" data-icon="info"
data-transition="flip" data-direction="reverse" class="ui-btn-
right"> </a>
</div>
```

<**data-position**="**fixed**"> // odnosi se na fiksiranje elementa. Pomoću ovog atributa, header i footer su fiksirani na vrhu i dnu prikazanog polja dok se sadržaj stranice slobodno pomiče između njih čineći header i footer uvijek vidljivim korisniku.

<**data-iconpos**="**notext**"> // ikona bez teksta

<**data-direction**="**reverse**"> // upotreba istog efekta tranzicije stranice prilikom povratka unazad

<**class**="**ui-btn-right**"> // klasa pomoću koje se određuje pozicija linka (desno).

<**a href**=**'’#dialog’’**> // poveznica koja vodio na dijaloški prozor

Primjer koda sadržaja (content) dokumenta

```
<div data-role="content">
    <center> <p>Dobrodošli na mobilnu aplikaciju Sjeveroljubac,
namijenjenu za sve studente Sveučilišta Sjever.</p><br><img
src="img/UNIN_1.png"/><center>
    <strong>Odaberite svoj status na Sjeveru</strong>
    <a href="postani_sjeveroljubac.html" data-ajax="false" data-
role="button" data-inline="true" data-theme=""
class="blink">Početnik</a>
    <a href="#student" data-role="button" data-inline="true" data-
theme="" data-transition="flip"  class="blink">Student </a>
</center>
    <p id="pbol">Povežite se sa Sveučilištem Sjever na društvenim
mrežama</p>
     <div style=" margin-left: 10px;">
    <a href="https://m.facebook.com/pages/Sveu%C4%8Dili%C5%A1te-
Sjever-University-North/781535971918819" data-role="button" data-
inline="true" rel="external" data-icon="facebook" data-
iconpos="notext"> </a>
     <a href="https://plus.google.com/+UninHr/about" data-
role="button" data-inline="true" data-icon="google-plus" data-
iconpos="notext" rel="external"> </a>
     <a href="https://linkedin.com/edu/school?id=162166&trk=edu-cp-
title" data-role="button" data-inline="true" data-icon="linkedin"
data-iconpos="notext" rel="external"> </a>
                <a href="https://m.twitter.com/UniNorthHR" data-
role="button" data-inline="true" data-icon="twitter" data-
iconpos="notext" rel="external"> </a>
                </div>
                </center></div>
```

** ** // HTML oznaka koji postavlja odnosno centrira sadržaj elementa

<a href="postani_sjeveroljubac.html" data-ajax=”false” data-role="button" data-theme="" data-transition="slideup" data.inline=”true” class="blink">
<a href="#student" data-role="button" data-inline=”true” data-theme="" data-transition="flip" class="blink">

// gumb poveznice koja vode na stranice za studente i početnike, koji ima defaultnu a temu sa tranzicijom fllip, i klasom blink koja ima beskonačni fade efekt
.blink {animation:fade 3000ms infinite; -webkit-animation:fade 3000ms infinite;}

<a href=”m.facebook, plus.google, linkedin, m.twitter” data-role="button" data-inline="true" data-icon="facebook, google-plus, linkedin, twitter" data-iconpos="notext" rel="external"> // gumb poveznice koje vode na profil Sveučilišta Sjever na društvenim mrežama, postavljene u istoj liniji i bez teksta sa atributom vanjskog linka. Malo slovo m označava da se radi o mobilnim verzijama poveznica

Primjer koda podnožja dokumenta

```
<div data-role="footer" data-theme="b" data-position="fixed">
                <h2>Sjever rocks!</h2></div>
</div><!-- kraj homepage-->
```

** ** // footer sa fiksiranom pozicijom koji ima temu b (crna tema)

5.4.1. Izrada dijaloškog prozor

Dijaloški prozor je kreiran s namjerom da da neke osnovne informacije o aplikaciji. U sklopu dijaloga se nalazi i gumb poveznica koja vodi na stranicu sa odjelima te back button odnosno gumb koji se vraća unazad. Poveznica na dijaloški prozor se jedino nalazi na početnoj stranici odnosno homepage budući da je upravo na tom mjesto bitno dati neke osnovne informacija.

Primjer koda za dialog prozor

```
<div data-role="dialog" id="dialog"><!--početak dialog-->
    <div data-role="header" data-add-back-btn="true" data-back-btn-
text="Nazad">
       <h3>Sjeveroljubac</h3>
    </div>
    <div data-role="content"><p>Aplikacija je stvorena s namjerom
kako bi studentima Sveučilišta Sjever olakšala pristup do
informacija  odnosno kako bi stdudent mogao s jednog mjesta
pristupiti željenim informacijama. Aplikacija se sastoji od
poveznica kao što su službena stranica sveučilišta, službeni
sveučilišni mail, Studomata i Moodlea.</p>
                <center><strong>Nastavite sa odabirom svog statusa
na Sjeveru</strong><br>
     <a href="postani_sjeveroljubac.html" data-ajax="false" data-
role="button" data-inline="true" data-theme="b">Početnik</a>
```

```
    <a href="#student" data-role="button" data-ajax="false" data-
inline="true" data-theme="b">Student </a>
                </center>            </div>
        </div><!--kraj dialog-->
```

** ** // dodavanje ”back” gumba. Okvir automatski generira back gumb pomoću navedenog atributa. Isto tako, uz gumb je dodana i tekst.

Kako zbog same mobilne aplikacije i estetike, nisam želio koristiti tekst prilikom dodavanja ”back” gumba odlučio sam koristiti jQuery Mobile funkciju kako bi maknuo tekst iz back gumba. Okvir prilikom dodavanja back gumba sam generira tekst i ikonu no ja sam želio koristiti samo ikonu. U nastavku slijedi funkcija pomoću koje sam to ostvario.

Primjer funkcije za uklanjanje teksta iz ikona

```
$(document).on('pagebeforeshow', function () {
    $(this).find('a[data-rel=back]').buttonMarkup({
    iconpos: 'notext'
    });
});
```

$(document) // objekt sam objasnio u teorijskom dijelu. Znači, on se izvršava svaki put kada se dokument pokrene

.**on()** // metoda pridružuje jedan ili više događaja za selektirane elemente

('pagebeforeshow') // događaj koji se pokreće prije nego što započne animacija

$(this) // odnosi se na element koji je povezan sa kontekstom. On dopušta pristup elementu koji se izvršava u nekom događaju.

.find() // pronalazi određeni element na koji se referira

.buttonMarkup // dodaje stilove za gumb određenom elementu

Laički govoreći, ono što ova funkcija radi je da se prilikom pokretanja dokument, pronalaze svi ”back” gumbi koji se nalaze u dokumentu te se stiliziraju na način da im pridružuje samo ikonu bez teksta.

5.5. Stranica student.html

Kreirana je stranica sa id-om=”student”. Ova stranica je namijenjena studentima Sveučilišta Sjever. Unutar sadržaja se nalazi lista sa poveznicama, ***odjeli.html, profesori.html i rss.html.***

Stranica sa id-om student

```
<div data-role="page" id="student">
    <div data-role="header" data-theme="a" data-position="fixed">
```

```
    <a href="#" data-rel="back" data-icon="arrow-l" data-
theme="d"></a><h3>Sjeveroljubac</h3>
    </div>
    <div data-role="content"  data-theme="a">
        <h2>Student Sjeveroljubac</h2>
        <img src="img/plakat.jpg" style="width: 100%;"/>
        <ul data-role="listview" style="margin-top: 20px;">
                    <li id="page1"><a href="odjeli.html" data-
ajax="false">Odjeli</a></li>
                    <li id="page2"><a href="profesori.html" data-
ajax="false">Profesori</a></li>
                    <li id="page3"><a href="rss.html" data-
ajax="false">RSS feed</a></li>
        </ul>
    </div>
    <div data-role="footer" data-theme="b" data-position="fixed">
                <h2>Sjever rocks!</h2>
    </div>
</div>
```

5.6. Izrada stranice popisa odjela

Stranica odjeli je kreiran zbog broja odjela Sveučilišta. Budući da u sklopu Sveučilišta postoji 7 preddiplomskih stručnih studija i 2 sveučilišna studija, bilo je potrebno stvoriti stranicu na kojoj će nalazi osnovna lista koja će sadržavati poveznice na određene odjele kako bi student mogao odabrati odjel koji pohađa u sklopu Sveučilišta. U tu svrhu sam stvorio dvije liste i podijelio ih na preddiplomske i sveučilišne studije. Svaka stavka unutar liste nema definiranu poveznicu jer ćemo pomoću funkcije upotrijebiti swipe (klizanje po zaslonu) efekt koja će nam koristiti za promjenu stranice.

Primjer stranice odjeli.html

```
<div data-role="page" id="odjeli"><!--početak odjeli-->
            <div data-role="header" data-add-back-btn="true" data-
back-btn-text="Nazad" data-position="fixed">
                <h3>Sjeveroljubac</h3>
            </div>
            <div data-role="content">
                <p>Preddiplomski stručni studij:</p>
                <ul data-role="listview" data-inset="true"
id="swipelist">
                    <li id="page1"><a
href="">Elektrotehnika</a></li>
                    <li id="page2"><a href="">Multimedija,
oblikovanje i primjena</a></li>
                    <li id="page3"><a href="">Proizvodno
strojarstvo</a></li>
```

```
                    <li id="page4"><a
href="">Graditeljstvo</a></li>
                    <li id="page5"><a href="">Logistika</a></li>
                    <li id="page6"><a href="">Sestrinstvo</a></li>
                    <li id="page7"><a href="">Novinarstvo</a></li>
                    <li id="page8"><a href="">Medijski
dizajn</a></li>
                    <li id="page9"><a href="">Poslovanje i
menadžment u medijima</a></li>
                </ul><br>
                <p>Sveučilišni studij:</p>
                <ul data-role="listview" data-inset="true"
id="swipelist">
                    <li id="page10"><a href="">Poslovna
ekonomija</a></li>
                    <li id="page11"><a href="">Odnosi s
javnostima</a></li>
                </ul>
            </div>
```

<**div data-role="header" data-add-back-btn="true" data-back-btn-text="Nazad" data-position="fixed**"> // header unutar kojeg se nalazi gumb za nazad i koji je fiksiran za gornji dio vidljivog područja

<**ul data-role="listview" data-inset="true" id="swipelist**"> // lista sa vrijednosti ”listview” koja generira osnvnu listu unutar okvira. Pomoću atributa data-inset=”inset” lista se stilizira sa zaobljenim kutovima i marginama.

5.6.1. Klik i each funkcije

Budući jer je aplikacija Sjeveroljubac mobilna aplikacija u projekt je trebalo uključiti i neke mobilne događaje a, stranica odjeli je odlično mjesto za korištenje takve vrste događaja. Kako se na navedenoj stranici nalaze dvije osnovne liste od kojih jedna ima čak 7 poveznica, korisniku će klikom na pojedinu poveznicu otvoriti iduću stranicu pomoću ”flip” tranzicije. Isto tako, po mom mišljenju, listu su savršeni elementi na koje se može primjeniti ovaj efekt. U nastavku slijedi funkcija pomoću koje su korištene liste:

Primjer koda za click funkciju

```
$(document).on('pageinit' , function(event){
       $("li").each(function(index){
         var elementId = $(this).attr("id");
           elementId = '#' + elementId;
            $(function(){
               $(elementId) .click (function (event){
                  $.mobile.changePage( elementId, { transition:
"flip"});
                              });  });  });  });
```

$(document) // izvršava se svaki put kada se dokument pokrene
.on() // metoda pridružuje jedan ili više događaja za selektirane elemente
(’pageinit’) // aktivira se kada je stranica inicijalizirana i nakon što je jQuery Mobile završio unaprijedio izgled sadržaja stranice
.each() // generička funkcija iteracije koja se može koristiti za iteraciju kroz objekte i nizove
.attr() // uzima vrijednost atributa (u ovom slučaju ”id”)
$(this) // odnosi se na element koji je povezan sa kontekstom. On dopušta pristup elementu koji se izvršava u nekom događaju.
$.mobile.changePage // promjena stranice

Ono što je najzanimljivije u ovoj jQuery funkciji je upotreba each funkcije. Each funkcija možemo usporediti sa for petljom jer to je upravo ono što ona radi. Znači, prilikom učitavanja dokumenta each funkcija prolazi kroz sve stavke liste na stranici odjeli te uzima njihov id kao atribut kojeg sprema u varijablu. Unutar varijable svakom atributu se dodaje hashkey koji u tom slučaju postaje poput poveznica. Nakon kreiranja poveznica, na varijablu se primjenjuje click efekt koji prilikom klikanja stavke pokreće funkciju mobile.changePage pri čemu se događa ”flip” tranzicija stranica odnosno stavki unutar liste. Ovo je zaista jedan od odličnih primjera kako jQuery olakšava rad svakom programeru.

5.6.2. Pojedinačne Stranice odjela

Kako sam je već prije istaknuto, u sklopu Sveučilišta Sjever se nalazi 9 odjela koji su podijeljeni na preddiplomske i sveučilišne studije. Budući da je svaka stranica odjela koncipirana na isti način, uz razlike u slikama i poveznicama koje pripadaju određenom odjelu, u ovom ulomku ću opisati samo jednu stranicu. Stranice odjela se sastoje od slike i četiri poveznice. Prva i najvažnija poveznica vodi na službenu stranicu Sveučilišta Sjever odnosno na obavijesti koje se nalaze unutar pojedinog odjela. Druga poveznica vodi na službeni mail kojeg student ima u sklopu Sveučilišta. Treće poveznica vodi moodle odnosno na stranicu za prijavu na moodle. Četvrta poveznica vodi na Studomat tako da student može odmah direktno prijaviti ispit ili pogledati određene informacije na Studomatu. Studomat kao poveznicu sam dodao iz razloga što on pruža mobilnu verziju prikaza za razliku od moodlea i maila. Sve ikone koje su korištene na stranicama, nalaze se u sklopu jQM paketa koji dodatno obogaćuje već ionako veliku zbirku ikona u okviru. Isto tako, sve slike koje se nalaze unutar navedenih stranica sam preuzeo sa službene stranice Sveučilišta.

Dodatna interaktivna stavka koja sam uključio za korisnika, je mogućnost dodavanja vlastitih gumb poveznica. Ako korisnik misli da bi u njegovom sučelju trebalo biti još dodatnih poveznica, korisnik će moći kreirati vlastitu poveznicu te će korištenjem localStorage biti u mogućnosti pohraniti podatke tako da oni uvijek budu vidljivi.

Primjer koda za pojedinu stranicu odjela

```
<div data-role="page" id="page1"><!--početak page1-->
    <div data-role="header" data-add-back-btn="true" data-
position="fixed">
       <h3>Sjeveroljubac</h3> </div>
     <div data-role="content">
       <h2>Odjel za elektrotehniku</h2>
         <img src="img/ethnic.jpg"/>
         <a href="http://www.unin.hr/category/et/" data-
role="button" data-theme="b" data-iconpos="right" data-
icon="info">Obavijesti</a>
        <a
href="https://login.aaiedu.hr/sso/module.php/core/loginuserpass.php
?AuthState=_677462ef18525c8e8cd10c901aa888e40fd7c1c6ff%3Ahttps%3A%2
F%2Flogin.aaiedu.hr%2Fsso%2Fsaml2%2Fidp%2FSSOService.php%3Fspentity
id%3Dgoogle.com%252Fa%252Funin.hr%26cookieTime%3D1438178255%26Relay
State%3Dhttps%253A%252F%252Fwww.google.com%252Fa%252Funin.hr%252FSe
rviceLogin%253Fservice%253Dmail%2526passive%253Dtrue%2526rm%253Dfal
se%2526continue%253Dhttps%25253A%25252F%25252Fmail.google.com%25252
Fa%25252Funin.hr%25252F%2526ss%253D1%2526ltmpl%253Ddefault%2526ltmp
lcache%253D2%2526emr%253D1" data-role="button" data-theme="b" data-
iconpos="right" data-icon="mail">Mail</a>
        <a href="https://moodle.vz.unin.hr/moodle/login/index.php"
data-role="button" data-theme="b" data-iconpos="right" data-
icon="book">Moodle</a>
        <a
href="https://www.isvu.hr/studomat/prijava?site_preference=mobile"
data-role="button" data-theme="b" data-iconpos="right" data-
icon="pencil">Studomat</a>
        <ul id="todos"></ul>
           <center><strong>Želite li kreirati vlastite gumb
poveznice?<br>Kliknite na gumb</strong></center>
           <a href="#dialog1" data-role="button" data
theme="b">Stvori</a>
</div>
<div data-role="footer" data-theme="a" data-position="fixed">
   <div data-role="navbar">
     <ul>
       <li id="page1"><a href="odjeli.html" data-ajax="false"
class="ui-btn-active ui-state-persist">Odjeli</a></li>
       <li id="page2"><a href="profesori.html" data-
ajax="false">Profesori</a></li>
       <li id="page3"><a href="rss.html" data-ajax="false">RSS
feed</a></li>
     </ul>
</div>
</div></div><!--kraj page1-->
```

** ** // header koji ima fiksiranu poziciju te gumb za nazad

Obavijesti // gumb poveznica koja je prikazana kao tema ”b” i koja ima ”info” ikonu smještenu na desnom kraju gumba

Moodle // gumb poveznica koja je prikazana kao tema ”b” i koja ima ”book” ikonu smještenu na desnom kraju gumba

Studomat // gumb poveznica koja je prikazana kao tema ”b” i koja ima ”book” ikonu smještenu na desnom kraju gumba

**<ul id=’’todos’’>
** // lista unutar koje će se ispisivati podatci iz localStoragea

Stvori // gumb poveznica koja vodi na dijaloški prozor za kreiranje gumb poveznica

5.6.3. Dijaloški prozor za kreiranje poveznica

Dijaloški prozor se sastoji od forme, unutar koje će korisnik unositi ili odabrati određene podatke. Od korisnika se traži da unese vrijednosti za poveznicu te da ju imenuje i naknadno odabere ikonu koju će koristit, poziciju ikone i odabir boje same gumb poveznice. Nakon što korisnik klikne na gumb Dodaj, podaci se obrađuju i spremaju lokalno. Prilikom osvježavanja aplikacije, podaci iz lokalne pohrane se ispisuju unutar stranica pojedinih odjela.

Primjer dialoga za kreiranje poveznica

```
<div data-role="dialog" id="dialog1" data-theme="a" ><!--početak
dialog-->
  <div data-role="header" data-add-back-btn="true">
      <h3>Sjeveroljubac</h3>
  </div>
  <div data-role="content" id="bbb" data-theme="a">
     <form id="form" action="#" method="POST">
     <label for="select-choice-1" class="select">Zalijepite ili
upišite poveznicu</label>
     <input id="description" name="description" type="text" />
     <label for="select-choice-1" class="select">Upišite
naziv</label>
     <input id="naziv" name="naziv" type="text" />
  <div data-role="fieldcontain">
    <label for="select-choice-1" class="select">Odaberite boju
gumba</label>
    <select name="select-choice-1" id="select-choice-1">
       <option id="a" value="a">Bijela</option>
       <option id="b" value="b">Crna</option>
    </select>
```

```
</div>
  <div data-role="fieldcontain">
   <label for="select-choice-1" class="select">Odaberite
ikonu</label>
   <select name="select-choice-1" id="select-choice-1-1">
      <option value="facebook">Facebook</option>
      <option value="book">Knjiga</option>
      <option value="info">Info</option>
      <option value="archive">Arhiva</option>
      <option value="calendar">Kalendar</option>
   </select>
</div>
  <div data-role="fieldcontain">
   <label for="select-choice-1" class="select">Odaberite poziciju
ikone</label>
   <select name="select-choice-1" id="select-choice-1-1-1">
      <option value="left">Lijevo</option>
      <option value="right">Desno</option>
      <option value="top">Gore</option>
      <option value="bottom">Dolje</option>
   </select>
</div>
  <input id="add" type="submit" value="Dodaj" />
  <button id="clear">Obriši</button>
  </form>
</div>
```

<input id="description" name="description" type="text" /> // polje za unos poveznice

<input id="naziv" name="naziv" type="text" /> // polje za unos naziva poveznice

** <select name="select-choice-1" id="select-choice-1"> <option id="a" value="a">Bijela</option> <option id="b" value="b">Crna</option></select>** // jqm atribut fieldcontain unutar kojeg se nalaze opcije sa vrijednostima za boju. Isti takav tip atributa je korišten i za vrijednosti pozicija ikona te za ikone.

5.6.4. Lokalna pohrana podataka

Lokalna pohrana podataka (LocalStorage) je metoda i protokol za pohranu podataka u web pregledniku koja pruža trajnu pohranu podataka (poput cookiea). Za razliku od cookiea, kojima pristup ima i server i klijent, localStorage se odvija isključivo na klijent strani i takvi podaci se nikada ne prenose na server. Local Storage pruža rješenje poput baze podataka. Vrijednosti se pohranjuju po domeni tako da niti jedna druga domena ne može doći do podataka. Podaci se pohranjuju kao parovi ”ključ-vrijednost” (key-value). Isto tako,

localStorage pruža 5MB prostora za pohranu. U nstavku slijedi skripta za spremanje i prikazivanje podataka.

Nakon što korisnik u formu unese sve potrebne podatke, klikom na gumb ”Dodaj” skripta uzima sve vrijednosti iz forme, te ih ispisuje na stranici pojedinog odjela. Nakon ispisa, forma se resetira, dok se lista sa id-om ”todos” u obliku html-a pohranjuje unutar localStorage-a. Nakon osvježavanja aplikacije, prikazuju se podaci iz localStorage

Primjer koda za upis i ispis podataka iz LocalStoragea

```
 $('#add').click( function() {
   var Description = $('#description').val();
   var naziv = $('#naziv').val();
   var tema = $('#select-choice-1').val();
   var icon = $('#select-choice-1-1').val();
   var poz = $('#select-choice-1-1-1').val();
   $('#todos').prepend('<a href="'+ Description +' " data-
role="button" style="margin-left: -40px;"data-theme="'+ tema +'"
data-iconpos="'+ poz +'" data-icon="'+icon+'">' + naziv + '<a>');
   $('#form')[0].reset();
   var todos = $('#todos').html();
   localStorage.setItem('todos', todos);
   location.reload();
   return false;
});

if(localStorage.getItem('todos')) {
$('#todos').html(localStorage.getItem('todos'));
}

$('#clear').click( function() {
window.localStorage.clear();
location.reload();
return false;
});
```

$('#add').click(function() // prilikom klika pokreće se funkcija

var Description = $('#description').val(); // u varijablu se spremaju vrijednosti iz inputa

$('#todos').prepend('<a href="'+ Description +' " data-role="button" style="margin-left: -40px;"data-theme="'+ tema +'" data-iconpos="'+ poz +'" data-icon="'+icon+'">' + naziv + '<a>'); // ispis dobivenih vrijednosti iz inputa

var todos = $('#todos').html(); // u varijablu se sprema čiatava lista s id-om todos

localStorage.setItem('todos', todos); // pohranjivanje podataka u localStorage pomoću naredbe .setItem(key, value).

if(localStorage.getItem('todos')) {$('#todos').html(localStorage.getItem('todos')); } // uvjet s kojim se provjerava da li u pohrani postoji stavka todos i ako postoji, ispiši ju unutar liste s id-om todos.

$('#clear').click(function() { window.localStorage.clear(); location.reload(); }); // klikom na gumb Obriši, brišu se svi podaci koji se nalaze unutar localStoragea.

5.7. Stranica s popisom nastavnika

Za potrebe aplikacije bilo je potrebno kreirati stranicu unutar koje će biti prikazani kontakt podaci od svih profesora koji predaju na Sveučilištu Sjever. Budući da se na Sveučilištu trenutno nalazi 318 profesora, bilo je potrebno pronaći način za dohvaćanje i prikazivanje podataka o profesorima. Znači, aplikacija se trebala spojiti s drugom domenom(u ovom slučaju službena stranica Sveučilišta) te s te domene dohvatiti određene podatke i prikazati ih unutar aplikacije. Za ostvarenje takve funkcionalnosti unutar jQM okvira koristio sam AJAX. AJAX se u jQM okiru koristi za komunikaciju odnosno slanje HTTP zahtjeva prema drugim stranicama ili serverima te za dohvaćanje podataka. No postoji jedna bitna stavka za komunikaciju sa različitim domenama unutar AJAX-a a, to je Cross Origin problematika. Cross Origin ima negativne konotacije i ne upotrebljava se toliko često u praksi, jedino u slučajevima kada korisnik ima omogućen pristup severu na kojoj se nalazi aplikacija ili kada se obadvije aplikacija nalazi na istoj domeni.

Da bi se uopće pristupilo podacima o profesorima, koristio sam još jedan dodatan alat, YQL. YQL (Yahoo Query Language) je upitni jezik sličan SQL-u koji se koristi za dohvaćanje i manipulaciju podacima iz drugih izvora.

Primjer YQL upita:

```
Select * from html where url =“http://www.unin.hr/o-sveucilistu/nastavnici/“;
```

JSON (JavaScript Object Notation), je otvoreni standardni format koji koristi ljudski-čitljiv tekst za prijenos objekt podataka koji se sastoje od parova atribut-vrijednost. Json je primarni format podataka koji se koristi za asinkronu komunikaciju preglednik/server. Iako izvorno potječe iz skriptnog jezika JavaScript, JSON je format podataka koji je neovisan o jeziku.

Prilikom učitavanja stranice, pokreće se funkcija čija je glavna stavka url odnosno poveznica na službenu stranicu Sveučilišta na kojoj se nalaze podaci o profesorima. Zatim se pomoću Ajax-a uzimaju podaci sa navedene poveznice. Ajax zahtjev se satoji od GET metoda, url unutra kojeg je smješteno izgenerirano YQL pitanje s kojim se dohvaćaju svi HTML podaci koji se nalaze na navedenoj poveznici. Podaci koje se dohvaćaju su u JSON

formatu. Na Sveučilištu Sjever predaje 318 profesora, stoga je prvi zadatak bio pronaći dužinu polja (array) kako bi pomoću for petlje mogao iterirati kroz sve elemente odnosno profesore. Zatim je trebalo pronaći sve podatke o profesorima(kontakt podaci, slike, itd.) koji se nalaze u dodatnim poljima unutar elementa za svakog pojedinog profesora (još jedna for petlja). Nakon što sam pronašao sve podatke, bilo je potrebno napraviti ispis.

Primjer funkcije za dohvaćanje i manipulaciju podataka sa domene unin.hr

```
$(document).on('pageinit', '#homepage', function(){
     url = 'http://www.unin.hr/o-sveucilistu/nastavnici/';
        $.ajax({
        type: "GET",
        url:
'https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20ht
ml%20where%20url%3D%22http%3A%2F%2Fwww.unin.hr%2Fo-
sveucilistu%2Fnastavnici%2F%22%20and%20xpath%3D%22%2Fhtml%2Fbody%2F
div%5B5%5D%2Fdiv%5B6%5D%2Fdiv%5B1%5D%2Fdiv%22%20&format=json&diagno
stics=true&callback=',
        dataType: 'json',
        error: function(){
            },
        success: function(data) {
        var a = data.query.results.div.div.div.length;
        var b = data.query.results.div.div.div[0].div.length;
        var c =
data.query.results.div.div.div[0].div[1].div.strong.length;
        var output = '<br><div data-role="listview" data-
filter="true" data-inset="true" data-theme="" >

        for ( var i = 0; i < a; i++){
          var dd =
data.query.results.div.div.div[i].div[0].a.content;
          var ddd =
data.query.results.div.div.div[i].div[1].div.content;
          var imgalt =
data.query.results.div.div.div[i].div[1].div.img.alt;
          var imgsrc =
data.query.results.div.div.div[i].div[1].div.img.src;
          var imgsty =
data.query.results.div.div.div[i].div[1].div.img.style;
          var dddd =
data.query.results.div.div.div[i].div[1].div.strong;
          var arrr = ddd.split("\n");

          output+='<div data-role="collapsible" data-inset="true"
data-theme="a" data-content-theme="a">';
              output+='<h3>'+ dd +'</h3>';
              output+='<img src="'+imgsrc+'" alt="'+imgalt+'"
style="height: 100px; width: 90px; float: left; margin-right:
10px;"/>';
              for (var k=0; k < c; k++){
              output+='<strong>'+dddd[k]+'</strong>';
              output+='<strong style="font-weight:
lighter;">'+arrr[k]+'</strong>';
```

```
            if(dddd[k] !== arrr[k]){
                output+='<br>';    } }
            output+='</div>'; }
    output+='</div>';
    $("#bbb").html(output);
    $("#bbb div[data-role=listview]").listview();
    $("#bbb div[data-role=collapsible]").collapsible();
    } }); });
```

var a = data.query.results.div.div.div.length; // dužina niza u kojem su smješteni profesori

var c = data.query.results.div.div.div[0].div[1].div.strong.length; // dužina niza u kojem su smješteni podaci o profesorima

**var output = '
 ** // kriranje ispisa

var dd = data.query.results.div.div.div[i].div[0].a.content; // u for petlji, iteracija kroz sve elemente u kojima se nalaze imena profesora

var ddd = data.query.results.div.div.div[i].div[1].div.content; // u for petlji, iteracija kroz sve elemente u kojima se nalaze podaci o profesora

var imgsrc = data.query.results.div.div.div[i].div[1].div.img.src; // u for petlji, iteracija kroz sve elemente u kojima se nalaze slike profesora

var dddd = data.query.results.div.div.div[i].div[1].div.strong; // u for petlji, iteracija kroz sve elemente u kojima se podaci o profesora

var arrr = ddd.split("\n"); // pretvaranje podataka iz stringa u array radi lakšeg ispisa

Nakon toga je bilo potrebno kreirati još jednu petlju koja će prolaziti kroz sve podatke o profesorima. Budući da su se podaci ispisivali jedni pokraj drugih, bilo je potrebno napraviti uvjet pomoću kojeg ćemo svaku novu liniju podataka ispisati u novom redu.

if(dddd[k] !== arrr[k]){ output+='<br>'; } } // poravnanje ispisa

$("#bbb").html(output); // ispis podatak unutar bbb selektora

5.8. Stranice vijesti i obavijesti

U dogovoru s mentorm i administratorom službene stranice Sveučilišta, na službenoj stranicu su aktivirani rss feedovi (obavijesti) koji će se prikazivati unutar aplikacije. U tu svrhu, administrator službene stranice je kreirao pet kategorija obavijesti. Kategorije su podijeljene prema sljedećem rasporedu:

- Sve vijesti sortirane kronološki: http://www.unin.hr/feed/
- Aktualno: http://www.unin.hr/category/aktualno/feed/
- Izdvojene novosti: http://www.unin.hr/tag/primary/feed/
- Novosti, objave i informacije: http://www.unin.hr/tag/secondary/feed/
- Ostale vijesti: http://www.unin.hr/tag/other/feed/

Prvo je bilo potrebno kreirati stranicu u koju će se sastojati od liste sa navedenim kategorijama. Zatim je trebalo kreirati stranicu za svaku pojedinu kategoriju (5 stranica) tako da se RSS podaci mogu ispisivati unutar te stranice. Budući da je bilo potrebno poslati zahtjev (request) putem AJAX-a za svaku pojedinu stranicu kategorija, trebalo je formirati pet zasebnih AJAX poziva sa navedenih url-ovima. Stoga ću u ovom opisu opisati samo jedan postupak.

Primjer stranice rss.html

```
 <div data-role="page" id="homepage">
    <div data-role="header" data-theme="b" data-position="fixed">
    <a href="#" data-rel="back" data-icon="arrow-l" data-theme="b"
></a><h3>Sjeveroljubac</h3>
    <a href="index.html" data-iconpos="notext" data-icon="home"
data-transition="flip" data-direction="reverse" class="ui-btn-
right"> </a></div>
    <div data-role="content"  data-theme="c" id ="postlist">
    <ul data-role="listview" id="swipelist">
       <li id="page1"><a href="#page1">Sve vijesti
(kronološki)</a></li>
       <li id="page2"><a href="#page2">Aktualno</a></li>
       <li id="page3"><a href="#page3">Izdvojene novosti</a></li>
       <li id="page4"><a href="#page4">Novosti, objave i
informacije</a></li>
       <li id="page5"><a href="#page5">Ostale vijesti</a></li>
    </ul></div>
    <div data-role="footer" data-theme="b" data-position="fixed">
      <div data-role="navbar">
       <ul>
          <li id="page1"><a href="odjeli.html" data-
ajax="false">Odjeli</a></li>
          <li id="page2"><a href="profesori.html" data-
ajax="false">Profesori</a></li>
          <li id="page3"><a href="rss.html" data-ajax="false"
class="ui-btn-active ui-state-persist">RSS feed</a></li>
</ul></div></div></div>

<div data-role="page" id="page1" data-theme="">
     <div data-role="header" data-theme="b" data-position="fixed">
     <a href="#" data-rel="back" data-icon="arrow-l" data-theme="b"
></a><h3>Sjeveroljubac</h3> </div>
    <div data-role="content"  data-theme="" id ="postlist1"></div>
    <div data-role="footer" data-theme="b" data-position="fixed">
        <div data-role="navbar">
         <ul>
             <li id="page1"><a href="odjeli.html" data-
ajax="false">Odjeli</a></li>
             <li id="page2"><a href="profesori.html" data-
ajax="false">Profesori</a></li>
             <li id="page3"><a href="rss.html" data-ajax="false"
class="ui-btn-active ui-state-persist">RSS feed</a></li>
</ul></div></div></div>
```

Prilikom učitavanja stranice, pokreće se funkcija čija je glavna stavka url odnosno poveznica na stranicu na kojoj se nalaze RSS obavijesti. Zatim se pomoću Ajax-a uzimaju podaci sa navedene poveznice. Ajax zahtjev se satoji od GET metoda, enkodiranog url za pristup feedovima i tipa podataka koji su u JSON formatu.

Primjer funkcije za dohvaćanje RSS feedova

```
$(document).on('pageinit', '#page1', function(){
  url = 'http://www.unin.hr/feed/';
    $.ajax({
        type: "GET",
        url: document.location.protocol +
'//ajax.googleapis.com/ajax/services/feed/load?v=1.0&num=1000&callb
ack=?&q=' + encodeURIComponent(url),
        dataType: 'json',
        error: function(){
            alert('Unable to load feed, Incorrect path or invalid
feed');
        },
        success: function(xml){
            var postlist = xml.responseData.feed.entries;
            var html = '<div data-role="collapsible-set" data-
inset="true">';
            $.each(postlist, function(index, data) {
                html += '<div data-role="collapsible" data-
inset="true" data-theme="b" data-content-theme="b">';
                html += '<a href="'+ data.link +'" data-
role="button" data-inline="true" data-mini="true" data-
iconpos="top" data-icon="arrow-r" style="float: right; margin-top:
-1px;">više</a>';
                html += '<h3>' + data.title + '</h3>';
                html += '<strong>' + data.contentSnippet +
'</strong><br>';
                html += '<strong style="font-weight: lighter;">' +
data.author + '</strong><br>';
                html += '<strong style="font-weight: lighter;">' +
data.publishedDate + '</strong><br>';
                html += '<strong style="font-weight: lighter;">' +
data.content + '</strong><br>';
                html += '</div>';
            });
            html += '</div>';
            $("#postlist1").append(html);
            $("#postlist1 div[data-role=collapsible-
set]").collapsibleset();
           $("#postlist1 div[data-
role=collapsible]").collapsible();
           $("#postlist1 a[data-role=button]").button();
```

var postlist = xml.responseData.feed.entries; // varijabla u koju spremaju odgovori (feedovi)

$.each(postlist, function(index, data) {} // iteracija kroz sve dobivene rss podatke

html += ’<h3>’ + data.title + ’</h3>’; // ispis naslova obavijesti
**html += ’’ + data.author + ’
’;** // ispis autora obavijesti
**html += ’’ + data.publishedDate + ’
’;** // ispis datuma objave
**html += ’’ + data.content + ’
’**; // ispis sadržaja obavijesti
$("#postlist1").append(html); // dodavanje rss feedova unutar postlist1 selektora

5.9. Stranica postani_sjeveroljubac.html

Ova stranica je napravljena za sve one korisnike koji žele postati studenti na Sveučilištu Sjever. Stranica je koncipirana tako da daje neke osnovne informacije o Sveučilištu kao što su O Sveučilištu, Programi, Vodič za studente, Lokacija i informacije o upisima. Sve podaci koji se nalaze unutar ove stranice su preuzeti sa službene stranice Sveučilišta Sjever. U poveznici ”O Sveučilištu” se nalazi kratka informacija o tome što je zapravo unin te misija i vizija Sveučilišta Sjever. U poveznici ”Programi” se nalaze sudijski programi koji su u sklopu rada Sveučilišta Sjever. ”Vodič za student” je pdf dokument. U poveznici lokacija se nalazi adrese Sveučilišta Sjever sa poveznicama na googlemaps, dok se pod poveznicom ”Upisi” nalaze poveznice na službenu stranicu Sveučilišta vezane za upis studenata. Unutar poveznice ”Upisi” korišteni su AJAX zahtjev i YQL konzola kako bi se dohvatile i prikazale informacije o upisima.

Za početak je bilo potrebno kreirati stranicu koja će sadržavati listu sa navedenim poveznicama. Nakon toga, trebalo je kreirati pet zasebnih stranica unutar kojih će biti smješteni navedeni podaci.

Primjer homepage stranice sa listom

```
<div data-role="page" id="homepage">
    <div data-role="header" data-theme="b" data-position="fixed">
        <a href="#" data-ajax="false" data-rel="back" data-
icon="arrow-l" data-theme="b"></a><h3>Sjeveroljubac</h3>
        <a href="index.html" data-ajax="false" data-
iconpos="notext" data-icon="home" data-transition="flip" class="ui-
btn-right"> </a>
    </div>
    <div data-role="content"  data-theme="c" id ="postlist">
        <center><h2>Postani Sjeveroljubac</h2></center>
        <img src="img/plakat.jpg" style="width: 100%;"/>
        <ul data-role="listview" id="swipelist">
           <li id="page1"><a href="#page1">O Sveučilištu </a></li>
           <li id="page2"><a href="#page2">Programi</a></li>
           <li id="page3"><a href="#page3">Vodič za
studente</a></li>
```

```
          <li id="page4"><a href="#page4">Upisi</a></li>  </ul>
</div>
<div data-role="footer" data-theme="b" data-position="fixed">
                <h2>Sjever rocks!</h2></div></div>
```

Stranice ”O sveučilištu”

```
<div data-role="page" id="page1" data-theme="">
    <div data-role="header" data-theme="b" data-position="fixed">
        <a href="#" data-rel="back" data-icon="arrow-l" data-
theme="b" ></a><h3>Sjeveroljubac</h3></div>
    <div data-role="content"  data-theme="" id ="postlist1">
        <h2>Što je Unin?</h2><p>Sveučilište Sjever je ustanova čiji
su trenutni osnivači gradovi Koprivnica i Varaždin, te je
organizirano kroz dva ravnopravna sveučilišna centra koji djeluju u
navedenim gradovima. Dana 12. rujna 2012. godine potpisan je ugovor
o partnerstvu gradova Varaždina i Koprivnice, kojim su postavljeni
temelji i utvrđene smjernice u razvoju modernog europskog
sveučilišta,  orijentiranog prema potrebama lokalne zajednice.</p>
        <h2>Misija</h2><p> Misija Sveučilišta Sjever je izobrazba
kompetentnog stručnog kadra za potrebe realnog gospodarstva  i
zdravstvenog sustava u regiji sjeverozapadne Hrvatske kroz
kvalitetno izvođenje stručnih i diplomskih studija prema zahtjevima
Bolonjske deklaracije. U provedbi ovog cilja Sveučilište Sjever  se
ustrojava kao dinamična organizacija koja stalno prati, primjenjuje
i ugrađuje znanstvene i stručne  spoznaje u osuvremenjivanje
postojećih i razvoj novih studijskih programa, promovira koncept
cjeloživotnog obrazovanja te produbljuje i održava veze s
gospodarstvom i suradnju sa srodnim visokoškolskim ustanovama  u
zemlji i inozemstvu. </p>
       <h2>Vizija</h2> <p>Vizija Sveučilišta Sjever je biti vodeća
obrazovna, znanstvena, stručna i društveno odgovorna visokoškolska
ustanova za obrazovanje kadrova iz područja tehničkih, ekonomskih,
biomedicinskih i zdravstvenih, biotehničkih  i interdisciplinarnih
znanosti te umjetničkog područja u sjeverozapadnoj
Hrvatskoj.Završeni studenti Sveučilišta Sjever jesu i ostat će
poželjni i zapošljivi stručnjaci zbog visoke razine i širine
usvojenih znanja i kompetencijspremni na samostalan i kreativan rad
u struci. U svom radu Sveučilište Sjever njeguje načela kvalitete u
visokom obrazovanju, načela etičnosti, kreativnosti,
transparentnosti, suradnje s drugim visokoškolskim  ustanovama te
nadasve dobre međuljudske odnose.</div> </div>
<div data-role="footer" data-theme="b" data-position="fixed">
<h2>Sjever rocks!</h2></div>
```

Stranica ”Programi”

```
<div data-role="page" id="page2" data-theme="">
            <div data-role="header" data-theme="b" data-
position="fixed">
                <a href="#" data-rel="back" data-icon="arrow-l"
data-theme="b"></a><h3>Sjeveroljubac</h3></div>
    <div data-role="content"  data-theme="" id ="postlist2">
```

```
        <h2>Studijski programi</h2>
        <p> Sveučilište Sjever izvodi nastavu na 11 studijskih
programa iz tehničkog, biomedicinskog,  društvenog i umjetničkog
područja, od čega je 9 studija na preddiplomskoj razini:<br><br>
            Elektrotehnika, s dva usmjerenja:<br>
            Automatizacija,<br>
            Biomedicinska elektronika,<br>
            Proizvodno strojarstvo,<br>
            Multimedija, oblikovanje i primjena,<br>
            Tehnička i gospodarska logistika,<br>
            Graditeljstvo,<br>
            Sestrinstvo,<br>
            Novinarstvo,<br>
            Medijski dizajn i<br>
            Poslovanje i menadžment u medijima.<br><br>
            Dva studija na diplomskoj razini:<br>
            Odnosi s javnošću<br>
            Poslovna ekonomija, <br><br>s dva usmjerenja:<br>
            Međunarodna trgovina,<br>
            Turizam.<br><br>
U suradnji sa Sveučilištem u Rijeci izvodi se i sveučilišni
poslijediplomski doktorski studij "Mediji i izdavaštvo".

U planu je uvođenje novih studija te u srednjoročnom razdoblju i
podizanje postojećih stručnih studija na sveučilišnu
razinu.</p></div>
    <div data-role="footer" data-theme="b" data-
position="fixed"><h2>Sjever rocks!</h2></div> </div>
```

Stranica "Vodič za studente"

```
<div data-role="page" id="page3" data-theme="">
            <div data-role="header" data-theme="b" data-
position="fixed">
                <a href="#" data-rel="back" data-icon="arrow-l"
data-theme="b"></a><h3>Sjeveroljubac</h3> </div>
<div data-role="content"  data-theme="" id ="postlist3">
<center><h2>Vodič za studente</h2></center>
        <object data="img/Brošura-Studiranje-na-Sjeveru.pdf"
type="application/pdf" style="width: 100%; height: 500px;"> <a
href="img/Brošura-Studiranje-na-Sjeveru.pdf"></a></object> </div>
    <div data-role="footer" data-theme="b" data-
position="fixed"><h2>Sjever rocks!</h2></div></div>
```

Stranica "Upisi"

```
<div data-role="page" id="page4" data-theme="">
            <div data-role="header" data-theme="b" data-
position="fixed">
                <a href="#" data-rel="back" data-icon="arrow-l"
data-theme="b"></a><h3>Sjeveroljubac</h3> </div>
    <div data-role="content"  data-theme="" id ="postlist4"></div>
```

```
    <div data-role="footer" data-theme="b" data-
position="fixed"><h2>Sjever rocks!</h2></div></div>
```

Primjer funkcije za dohvaćanje podataka

```
$(document).on('pageinit', '#page4', function(){
    url = 'http://www.unin.hr/category/upisi_preddiplomski/';
        $.ajax({
        type: "GET",
        url:
"https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20ht
ml%20where%20url%3D%22http%3A%2F%2Fwww.unin.hr%2Fcategory%2Fupisi_p
reddiplomski%2F%22%20and%20xpath%3D%22%2F%2Fdiv%5Bcontains(%40class
%2C%20'article')%5D%22&format=json&callback=",
        dataType: 'json',
        async: true,
        error: function(){
            alert('Unable to load, Incorrect path or invalid
data');},
        success: function(data) {
        var ako = data.query.results.div.length;
        var output = '<ul data-role="listview">';
        for (var i=0; i<ako; i++){
            var aa =
data.query.results.div[i].div[0].div.h1.a.content;
            var ss =
data.query.results.div[i].div[0].div.h1.a.href;
            var cc = data.query.results.div[i].div[0].div.p;
            output+='<li>';
            output+='<a href="' +ss+ '">';
            output+='<h3>'+ aa +'</h3>';
            if ( typeof cc !== 'undefined'){
                output += '<p>'+cc+'</p>';
            }
            output += '</a>';
            output += '</li>';
        }
        output+='</ul>;
    $("#postlist4").html(output);
    $("#postlist4 ul[data-role=listview]").listview();
    $("#postlist4 div[data-role=collapsible]").collapsible();
    } }); });
```

typeof // koristi se za provjeru vrste podataka

if (typeof cc !== 'undefined'){} // uvjet koji se koristi ako je vrsta podataka unutar varijable cc različit od undefined

5.9.1. Implementacija Google mapsa

Prilikom implementacije mapa (googlemaps), koristiti će se jedna od nativnih mogućnosti uređaja. U tu svrhu, bilo je potrebno koristiti Cordova plugin geolocation kako bi se odredila

trenutna pozicija korisnika te da bi se izračunala udaljenost i prikazao put od korisnikove trenutne pozicije do Sveučilišnog centara u Varaždinu i Koprivnici. Isto tako, potrebno je napomenuti da je prilikom implementacije bilo potrebno izmjeniti configuracijsku datoteku, kako bi android platformi dozvolili pristup određenim skriptama.

Primjer skripte googlemapsa

```
<script type="text/javascript">
            var map,
                currentPosition,
                directionsDisplay,
                directionsService,
                destinationLatitude = 46.299919,
                destinationLongitude = 16.326297;
            function initializeMapAndCalculateRoute(lat, lon)
            {
                directionsDisplay = new
google.maps.DirectionsRenderer();
                directionsService = new
google.maps.DirectionsService();
                currentPosition = new google.maps.LatLng(lat, lon);
                map = new
google.maps.Map(document.getElementById('map_canvas'), {
                   zoom: 15,
                   center: currentPosition,
                   mapTypeId: google.maps.MapTypeId.ROADMAP });
                directionsDisplay.setMap(map);
                 var currentPositionMarker = new
google.maps.Marker({
                    position: currentPosition,
                    map: map,
                    title: "Current position" });
                calculateRoute();}
            function locError(error) {
                 alert('the current position could not be
located');}
            function locSuccess(position) {

initializeMapAndCalculateRoute(position.coords.latitude,
position.coords.longitude);}
            function calculateRoute() {
                var targetDestination =  new
google.maps.LatLng(destinationLatitude, destinationLongitude);
                if (currentPosition != '' && targetDestination !=
'') {
                    var request = {
                        origin: currentPosition,
                        destination: targetDestination,
                        travelMode:
google.maps.DirectionsTravelMode["DRIVING"]};
                    directionsService.route(request,
function(response, status) {
                        if (status ==
google.maps.DirectionsStatus.OK) {
```

```
directionsDisplay.setPanel(document.getElementById("directions"));

directionsDisplay.setDirections(response);
                            $("#results").show();}
                        else {
                            $("#results").hide(); });}
                else {
                    $("#results").hide();}}
            $(document).on("pagebeforeshow", "#map_page",
function() {
                // find current position and on success initialize
map and calculate the route

navigator.geolocation.getCurrentPosition(locSuccess, locError,
{timeout: 10000, enableHighAccuracy: true});
            });
        </script>
```

5.10. Autorizacija kroz LDAP

LDAP (od engl. Lightweight Directory Access Protocol je standardni, na Internetu dostupan protokol za pristup imeničkim servisima. LDAP je imenični servis koji služi kao centrano mjesto za pohranu informacija (korisnički računi, kontakti, postavke za progame, itd.). Podaci se pohranjuju u LDAP stablo, posebno prilagođenu bazu podataka. Prednost je lako dodavanje, modificiranje i uklanjanje podataka sa štednjom resursa. Još jedna prednost je dodavanje unosa za korisnika u odnosu na kasičnu SQL bazu. LDAP koristi takozvane "sheme" za određivanje tipa podataka koji će biti pohranjeni u stablu, npr "core.shema" i "inetorgperson.shema". Različite "sheme" daju na korištenje različite "objectClasse". objectClass-a je predložak za podatke koji će se koristiti kod unosa - on definira set atributa koji će biti prisutni u unosu. Atributi se mogu poistovjetiti sa varijablama i tipovima podataka kod programskih jezika, služe za pohranu vrijednosti. Za razliku od varijabli, jedan atribut može pohraniti više vrijednosti.

Lightweight? Korijeni LDAPa su iznimno povezani s X.500 imeničkim servisom; LDAP je originalno dizajniran kao „lakši“ desktop protokol za usmjeravanje zahtjeva prema X.500 poslužiteljima. X.500 je skup standarda i smatra se ”težim” protokolom jer zahtjeva komunikaciju klijent – poslužitelj putem OSI modela. Ovaj sedam – slojni model je bio dobar predložak za kreiranje mrežnih protokola, ali je u usporedbi s TCP/IP poprilično teži i

kompliciraniji. LDAP je lakši jer koristi male podatke u prijenosu informacija koje su direktno preslikani na TCP sloju unutar TCP/IP mrežnog modela.

Directory? LDAP je samo protokol a, pojednostavljeno znači skup poruka za pristup određenim vrstama podataka. Protokol kao takav ne spominje ništa o mjestu pohrane podataka. Stoga, kada se govori o LDAPu kako ne podržava transakcije i druge funkcije baza podataka, misli se na LDAP kao protokol koji nema komunikacijske mogućnosti za izvršavanjem navedenih upita i koji ne zahtjeva od pozadinskog skladišta podataka istu mogućnost.

Access Protocol? LDAP predstavlja podatke u stablastom prikazu, i koristi se kao klijent – poslužitelj protokol (prema definiciji iz RFC 2251), koji omogućava asinkroni pristup. On omogućava da klijent izvrši nekoliko upita te da odgovori mogu stići u drukčijem poretku nego što su poslani.

Svaki unos u LDAP stablu mora imati "objectClass" atribut - njime se određuje o kakvom se tipu podataka radi. Za dodavanje podataka u stablo koristi se LDIF unos. LDIF (LDAP Interchange Format) je tekstualna datoteka koja sadrži kolekciju unosa, te atributa i njihovih vrijednosti koji će biti pohranjeni u imeniku. Cijeli sadržaj imenika moguće je izvesti (export) u LDIF datoteku (backup).

Sljedeći primjer LDIF datoteke predstavlja unos za administratorski račun imenika:

```
dn:cn = admin, dc = tvrtka, dc = hr
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword: {crypt}b2Ri7FFx2nx
```

LDAP ima devet osnovnih operacija koje se mogu svrstati u tri skupine:

- Ispitivajuće operacije: search, compare. Ove operacije omogućavaju izvršavanje upita nad imenikom.
- Ažurirajuće operacije: add, delete, modify, modify DN(rename). Ove operacije dozvoljavaju ažuriranje informacija u imeniku.
- Autentikacija i kontrolne operacije: bind, unbind, abandon. Bind operacija omogućava klijentima indetifikaciju nad imenikom; unbind omogućava zatvaranje veze s imenikom; abandon operacija omogućava obavijest kako klijent nije više zainteresiran za rezultate operacije koju je prethodno tražio.

ldapjs je JavaScript okvir za implementaciju LDAP klijenata i servera u Node.js. Namijenjen je za developere koji se služe interakcijom s HTTP servisima. ldapjs implementira većinu zajedničkih operacija u LDAP v3 RFC-u, kako za klijenta tako i za poslužitelja. U skladu je sa LDAP protokolom, te je interoperabilan s OpenLDAP i bilo kojom drugom LDAPv3 implementacijom. Njegova namjera izgradnja LDAP sa svim onim što želite, a ne samo sa tradicionalnim bazama podataka.

Express.js je Node.js web aplikacijski server okvir, namijenjen za izgradnju single-page, multi-page, i hibridnih web aplikacija. On je de facto standardni server okvir za node.js. Autor, TJ. Holowaychuck, opisao ga je kao nadahnutim poslužiteljem, što znači da je relativno minimalan s mnogim značajkama koje su dostupne kao dodatci(plugins). Express je backend dio MEAN slog, zajedno sa MongoDB bazama podataka i AngularJS frontend okvirom. Express je minimalna i fleksibilna Node.js web aplikacijski okvir koji pruža robustan skup značajki za razvoj web i mobilnih aplikacija. Omogućava brzi razvoj Node baziranih web aplikacija.

Slijedeće značajke su neke od osnovnih značajki Express okvira:

- Omogućuje postavljanje međuprograma za odgovor na HTTP zahtjeve.
- Definira tablica usmjeravanja koje se koristi za obavljanje različitih akcije na temelju HTTP metode i URLa.
- Omogućuje dinamički prikazuje HTML stranice na temelju prenošenja argumenta za predloške.

Primjer express.js Hello World

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
   res.send('Hello World');
})
var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

Express Aplikacija koristi povratnu funkciju čiji su parametri request (zahtjev) i response (odgovor) objekti.

Request objekt - predstavlja HTTP zahtjeva i ima svojstva za zahtjev upita, parametre, HTTP zaglavlja, itd.

Response objekt - predstavlja HTTP odgovor koji Express aplikacija šalje kada dobiva HTTP.

5.10.1. Izrada stranice LDAP_login.html

Unutar aplikacije je bilo potrebno napraviti formu za logiranje. Kako bi se cijeli postupak olakšao, korisnici će se logirati sa svojim službenim računima. Prijava će se odvijati pomoću postojećeg LDAP protokola. Kako se prijava na određeni servis ili web stranicu ne može ostvariti preko client strane (frontend), za izradu ovoga zadatka bilo je potrebno koristiti node.js. Node.js je javascript koji se odvija na serveru, pa u skladu s tim, pomoću node će aplikacija Sjeveroljubac komunicirati sa serverom te će preko LDAP protokola provjeravati da li korisnik uopće postoji u imeniku. U tu svrhu, kreiran je Node.js projekt koji će poslije biti uploadan na heroku server (https://heroku.com/). Zašto heroku server? Navedni node projekt mora konstanto raditi, stoga je bilo potrebno pokretati projekt na serveru. Isto tako, da bi se node.js projekt uopće mogao pokretati na serveru, na tom istom serveru mora biti instaliran node.js. Iz tog raloga odlučeno je da će se u radu koristiti heroku. Prilikom rada u node.js koristiti će se određeni moduli točnije ldap.js i express.je.

Unutar apliakacije kreirana je stranica LDAP_login.html koja će komunicirati sa node projektom kako bi se ostavrila uspješna prijava. Komunikacija će se odvijati pomoću Ajaxa, koji će uzimati vrijednosti iz forme i slati na provjeru node projektu koji se nalazi na domeni http://ldap-node-js.herokuapp.com/. Isto tako, korisniku je ponuđeno da spremi svoje podatke unutar localStoragea, kako prilikom svakog logiranja ne bi morao upisivati podatke.

Primjer skripte za pohranu podataka u localStorage i slanje ajax zahtjeva

```
 <script>
            function zapamti(){
                if (localStorage.chkbx && localStorage.chkbx !== 
’’) {

$(’#remember_me’).attr(’checked’, ’checked’);

$(’#username’).val(localStorage.usrname);

$(’#password’).val(localStorage.pass);
                                    }
                else {
```

```
$('#remember_me').removeAttr('checked');
        $('#username').val('');
        $('#password').val('');                          }
$('#remember_me').click(function() {
          if ($('#remember_me').is(':checked')) {
                   // save username and password
          localStorage.usrname = $('#username').val();
          localStorage.pass = $('#password').val();
          localStorage.chkbx = $('#remember_me').val();
          } else {
          localStorage.usrname = '';
          localStorage.pass = '';
          localStorage.chkbx = '';
        }                                  });                          }
         function zovi(){
                var username = $('#username').val();
                var password = $('#password').val();
if($("#username").val() === '' && $("#password").val() === '') {
                    alert("Niste unijeli sve podatke!");
                    return false;
                        }
else{
      jQuery.ajax({
                type: "GET",
                url: 'https://ldap-node-
js.herokuapp.com/login?username=' + username + '&password=' +
password,
               dataType: 'html',
               accept: 'html',
                error: function(jqXHR, textStatus, errorThrown){
                        alert("server is down" + textStatus +
errorThrown);
       },
                success: function(data) {
                        if (data === 'Success') {
                            window.location.href="#dialog_uspjeh";
                        }
                        if (data === 'Fail'){
                            alert("Podaci koje ste unijeli nisu
točni!");                 }        }   });    }}  </script>
```

Primjer koda node.js projekta (server.js)

```
var http = require('http');
var express = require('express');
var app = express();
var ldap = require('ldapjs');
var cors = require('cors');
app.use(cors());

app.get('/login', function(req, res) {
    res.setHeader("Access-Control-Allow-Origin","*");
    res.status(200);
```

```
  var authenticated = login(req.query.username, req.query.password,
res);
  res.type('html');

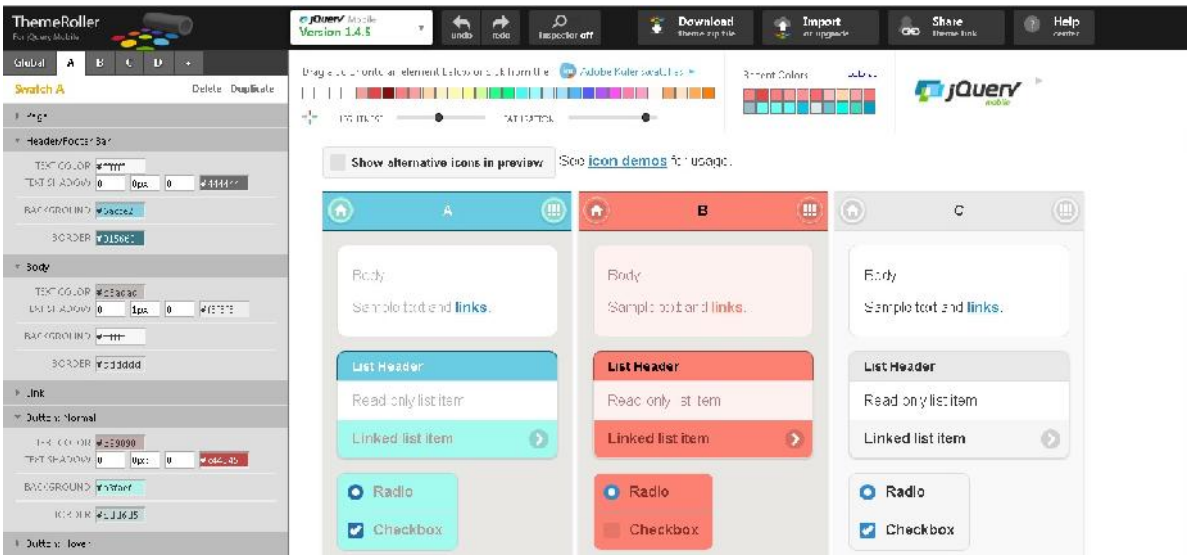
}); var port = Number(process.env.PORT || 9080);
app.listen (port);
var login = function (username, password, res) {
      ldap.Attribute.settings.guid_format = ldap.GUID_FORMAT_B;
    var client = ldap.createClient({
          url: 'ldap://ldap.unin.hr:389/ ',
          timeout: 5000,
          connectTimeout: 10000
    });
    console.log('--- going to try to connect user ---');
    try {
        client.bind('UID=' + username + ',DC=unin,DC=hr', password,
function (error) {
            if(error){
                res.send("Fail");
                console.log('not bind', error.message);
                client.unbind(function(error)
{if(error){console.log(error.message);} else{console.log('client
disconnected');}});
            } else {
                console.log('connected');
                res.send("Success");
            }
        });
    } catch(error){
        console.log(error);
        client.unbind(function(error)
{if(error){console.log(error.message);} else{console.log('client
disconnected');}});
    }
};
    var opts = {
      filter:
'(&(objectclass=student)(samaccountname='+'judizdar'+'))',
      scope: 'sub',
      attributes: ['objectGUID']
    };
```

5.11. Izrada tema u ThemeRolleru

Izrada tema pomoću ThemeRollera je izrazito lagano i omogućava brzu kreaciju tako da programer ne mora gubiti puno vremena za izradu izgleda aplikacije. Programer u principu odabire boju iz paleta boja koje mu se nude i "baca" ju na pojedini element koji želi obojati. U padajućem izborniku s lijeve strane nalaze se elementi od kojih se sastoji aplikacija, kao što su header, footer, button, body itd. unutar kojih se mogu mijenjati boje teksta, sjene, pozadine

i obruba. Nakon što se završilo s kreiranjem tema, teme se preuzimaju na lokalni disk te se uključuju u dokument unutar <head> oznake. Isto tako, da bi tema u potpunosti funkcionirala, sa službene jQuery Mobile stranice (http://jquerymobile.com/) je potrebno preuzeti i strukturnu datoteku odnosno css stil koji se koristi prilikom izrade custom tema.

Za potrebe aplikacije Sjeveroljubac, unutar navedenog sučelja kreirane su tri teme. Prva tema A, bazira se na plavoj boji i kombinaciji različitih tonova plave boje i svijetlo sive pozadine. Druga tema B, bazira se na crvenoj boji i kombinaciji tonova crvene boje i svijetlo sive pozadine. Treće tema C, je defaultna tema u sučelju a, bazira se na sivoj i bijeloj boji (slika 5.2).



Slika 5.2 Teme aplikacije

5.12. Izrada vizualnog identiteta aplikacije

Kako aplikacija ne bi koristila defaultnu Cordova ikonu, kreirana je ikona koja će zamijeniti navedenu. Ikona (slika 5.3)je izrađena u Photoshopu te je pomoću online alata za generiranje ikona (http://makeappicon.com/) kreirana kako bi odgovarala svim rezolucijama ikona za ios i android. Da bi se ikona uspješno implementirala u aplikaciju, potrebno je definirati pravila unutar globalne konfiguracijske datoteke. Isto procedura se radi i prilikom mijenjanja splashscreena, odnosno screena koji se prikazuje kada se aplikacija učitava. Screenovi su izrađeni u Photoshopu a, izrađuju se zasebno za svaku rezoluciju.



Slika 5.3 Ikona

Primjer koda za implementaciju custom ikona i splashscreena za android platformu

```
<platform name=’’android’’>
        <icon src="www/img/icon_36.png" density="ldpi" />
        <icon src="www/img/icon_48.png" density="mdpi" />
        <icon src="www/img/icon_72.png" density="hdpi" />
        <icon src="www/img/icon_96.png" density="xhdpi" />
        <splash src="www/img/screen-land-hdpi.png" density="land-
hdpi"/>
        <splash src="www/img/screen-land-ldpi.png" density="land-
ldpi"/>
        <splash src="www/img/screen-land-mdpi.png" density="land-
mdpi"/>
        <splash src="www/img/screen-land-xhdpi.png" density="land-
xhdpi"/>
        <splash src="www/img/screen-port-hdpi.png" density="port-
hdpi"/>
        <splash src="www/img/screen-port-ldpi.png" density="port-
ldpi"/>
        <splash src="www/img/screen-port-mdpi.png" density="port-
mdpi"/>
        <splash src="www/img/screen-port-xhdpi.png" density="port-
xhdpi"/>
</platform>
```

5.13. Prilagođeni stilovi

JQuery Mobile radni okvir posjeduje neke svoje zadane postavka. Kako u svim okvirima pa tako i u ovom, potrebno je napraviti neke sitne izmjene unutar projekta kako bi se određeni elementi prikazivali onako kako programer to želi.

Prvi problem na koji sam naišao je bio taj da je okvir skraćivao naslov Sjeveroljubac u headeru na stranicama tako da je naslov izgledao **Sjeverolj....** . Problem je riješen dodavanjem stila za header i njegov naslov.

```
.ui-header .ui-title { overflow: visible !important;  white-space:
normal !important;}
```

overflow: visible !important; // označava što se događa sa sadržajem kada on prekoračuje okvire elementa. U ovom slučaju je vidljiva sa important oznakom što znači da je header naslov uvijek vidljiv u pregledniku

white-space: normal !important; // označava kako upravlja sa praznim prostorom unutar elementa

Drugi problem je bio taj što je okvir po defaultnim postavkama sam transformirao tekst tako da je sav tekst unutar stranice bio ispisan velikim slovima. Problem je riješen dodavanjem stila za tekst koji sam primjenio za cijelu <body> oznaku.

```
body{ text-transform: none; }
```

Slike nisu zauzimel čitav prostor content elementa. Riješeno tako da što sam postavio širinu slike na 100%.

```
img { width: 100%; }
```

Uređivanje naslova h2(postavljanje u centralnu poziciju) i h3 (transformacija u velika slova)

```
h3{ text-transform: uppercase; }
h2{ text-align: center; }
```

5.14. Konfiguracijska datoteka config.xml

Config.xml datoteka je XML datoteka koja se koristi za definiranje svojstava aplikacije. Ona pruža generalne informacije o aplikaciji kao što su ime aplikacije, opis aplikacije, početna stranica i specificira značajke i funkcionalnosti kojima aplikacija može pristupiti. [15] Navedeni primjer config.xml datoteke je dafultni primjer koji se generira prilikom stvaranja projekta. Isto tako, u samom projektu postoji više config.xml datoteka. Osim config datoteke koja se nalazi u najvišoj razini projekta, svaka platforma koja se dodaje projektu ima svoju config.xml datoteku. Sve promjene koje se vrše unutar config.xml datoteke odvijaju se u globalnoj config datoteci te se nakon izgradnje projekta za pojedinu platformu kopiraju u platforms direktorij.

Primjer koda konfiguracijske datoteke

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<widget xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0"
id="com.coolappz.Sjeveroljubac" version="1.0.0">
    <name>Sjeveroljubac</name>
    <description>
        A sample Apache Cordova application that responds to the
deviceready event.
    </description>
    <author email="dev@cordova.apache.org"
href="http://cordova.io">
        Apache Cordova Team
    </author>
    <content src="index.html"/>
    <plugin name="cordova-plugin-whitelist" version="1"/>
    <access origin="*"/>
    <allow-intent href="http://*/*"/>
    <allow-intent href="https://*/*"/>
```

```
    <allow-intent href="tel:*"/>
    <allow-intent href="sms:*"/>
    <allow-intent href="mailto:*"/>
    <allow-intent href="geo:*"/>
    <platform name="android">
        <access origin="*"/>
        <allow-intent href="market:*"/>
        <preference name="Fullscreen" value="true" />
        <preference name="permissions" value="INTERNET" />
        <preference name="SplashScreen" value="screen" />
        <icon src="www/img/icon_36.png" density="ldpi" />
        <icon src="www/img/icon_48.png" density="mdpi" />
        <icon src="www/img/icon_72.png" density="hdpi" />
        <icon src="www/img/icon_96.png" density="xhdpi" />
        <icon src="www/img/icon_144.png" density="xxhdpi" />
        <icon src="www/img/icon_192.png" density="xxxhdpi" />
        <splash src="www/img/screen-land-hdpi.png" density="land-
hdpi"/>
        <splash src="www/img/screen-land-ldpi.png" density="land-
ldpi"/>
        <splash src="www/img/screen-land-mdpi.png" density="land-
mdpi"/>
        <splash src="www/img/screen-land-xhdpi.png" density="land-
xhdpi"/>
        <splash src="www/img/screen-port-hdpi.png" density="port-
hdpi"/>
        <splash src="www/img/screen-port-ldpi.png" density="port-
ldpi"/>
        <splash src="www/img/screen-port-mdpi.png" density="port-
mdpi"/>
        <splash src="www/img/screen-port-xhdpi.png" density="port-
xhdpi"/>
    </platform>
    <platform name="ios">
        <access origin="*"/>
        <preference name="Fullscreen" value="true" />
        <preference name="permissions" value="INTERNET" />
        <allow-intent href="itms:*"/>
        <allow-intent href="itms-apps:*"/>
        <icon src="www/img/Icon-60@3x.png" width="180" height="180"
/>
        <icon src="www/img/Icon-60.png" width="60" height="60" />
        <icon src="www/img/Icon-40@3x.png" width="120" height="120"
/>
        <icon src="www/img/Icon-76.png" width="76" height="76" />
        <icon src="www/img/Icon-76@2x.png" width="152" height="152"
/>
        <icon src="www/img/Icon-40.png" width="40" height="40" />
        <icon src="www/img/Icon-40@2x.png" width="80" height="80"
/>
        <icon src="www/img/Icon.png" width="57" height="57" />
        <icon src="www/img/Icon@2x.png" width="114" height="114" />
        <icon src="www/img/Icon-72.png" width="72" height="72" />
        <icon src="www/img/Icon-72@2x.png" width="144" height="144"
/>
```

```
        <icon src="www/img/Icon-Small.png" width="29" height="29"
/>
        <icon src="www/img/Icon-Small@2x.png" width="58"
height="58" />
        <icon src="www/img/Icon-Small-50.png" width="50"
height="50" />
        <icon src="www/img/Icon-Small-50@2x.png" width="100"
height="100" />
        <splash src="www/img/screen_320_480.png" width="320"
height="480"/>
        <splash src="www/img/screen_640_960.png" width="640"
height="960"/>
        <splash src="www/img/screen_768_1024.png" width="768"
height="1024"/>
        <splash src="www/img/screen_1536_2048.png" width="1536"
height="2048"/>
        <splash src="www/img/screen_1024_768.png" width="1024"
height="768"/>
        <splash src="www/img/screen_2048_1536.png" width="2048"
height="1536"/>
        <splash src="www/img/screen_640_1136.png" width="640"
height="1136"/>
        <splash src="www/img/screen_750_1334.png" width="750"
height="1334"/>
        <splash src="www/img/screen_1242_2208.png" width="1242"
height="2208"/>
        <splash src="www/img/screen_2208_1242.png" width="2208"
height="1242"/>
    </platform></widget>
```

<**allow intent href**> // element koji daje dozvole pojedinim akcijama kao što je pristup emailu, sms, telefonskom broju itd.

<**platform name**> // element određuje konfiguraciju koja će se pojaviti u pojedinoj platormi

<**plugin name**> element koji uključuje određeni plugin

5.15. Testiranje aplikacije

Aplikacija je testirana na Bluestacks android simulatoru i u Chrome browser. Testiranje na android simulatoru je dugotrajno i pomalo besmisleno jer se za svaku promjenu unutar koda treba pokretati čitav niz Cordova knjižica i pluginova a, prosječno vrijeme za izgradnju i pokretanje aplikacije na simulatoru traje u prosjeku oko 5 minuta što u velikoj mjeri otežava rad. Stoga je aplikacija Sjeveroljubac uglavnom testirana u Chrome browseru budući da je vrijeme izgradnje gotovo minimalno. Isto tako, da bi se aplikacija testirala na android simulatoru potrebno je da računalo ima HAXM Accelerator. HAXM (Hardware Accelerated Execution Manager) je hardverski virtualizacijski engine koji koristi virtualizacijsku tehnologiju za prikazivanje android aplikacija na simulatoru.

Aplikacija se nalazi na domeni http://arwen.unin.hr/~judizdar/Sjeveroljubac/.

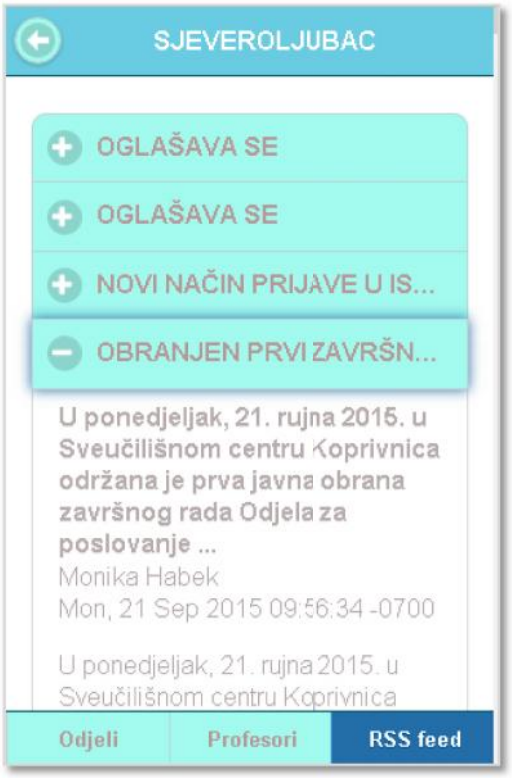
index.html



profesori.html



rss.html



odjeli.html



Stranica pojedinačnog odjela



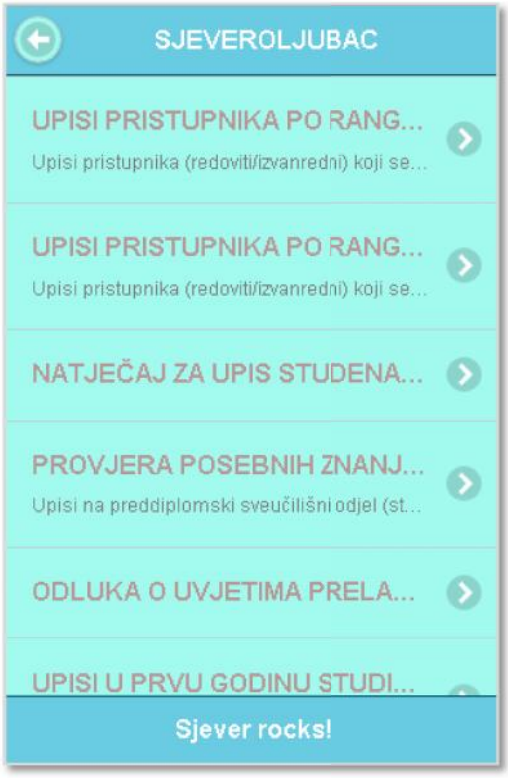
Dijalog za kreiranje poveznica



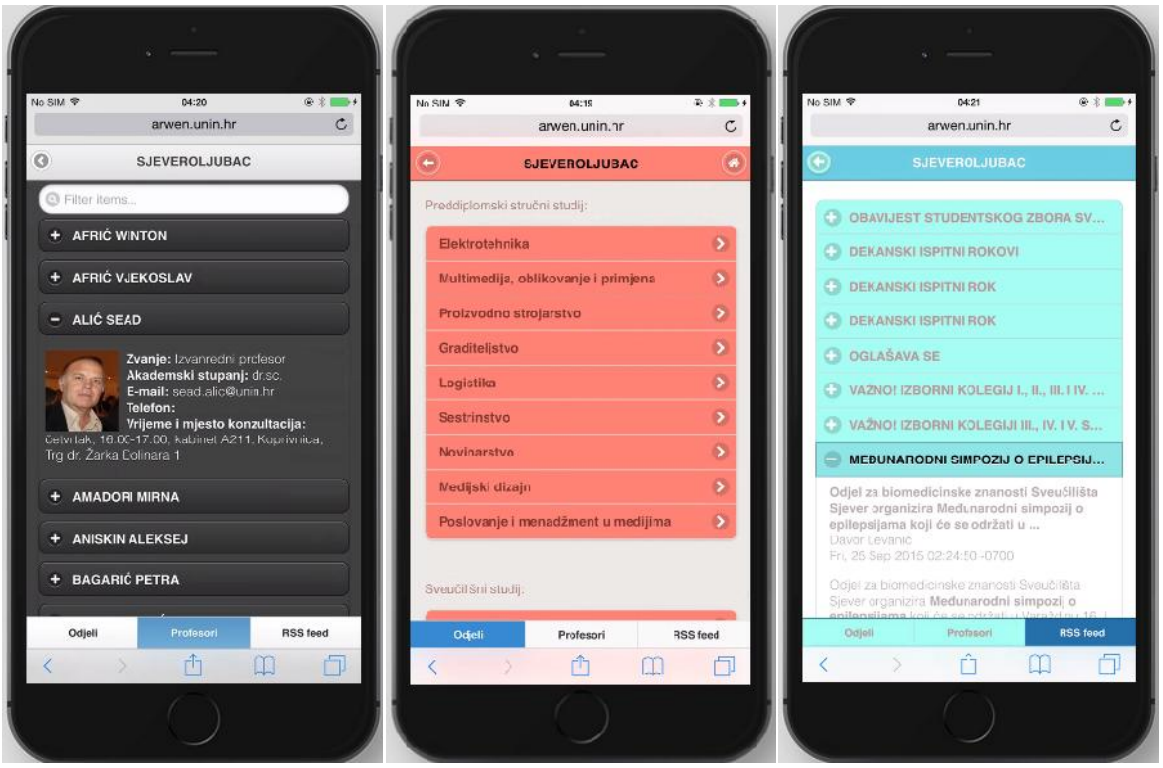
postani_sjeveroljubac.html



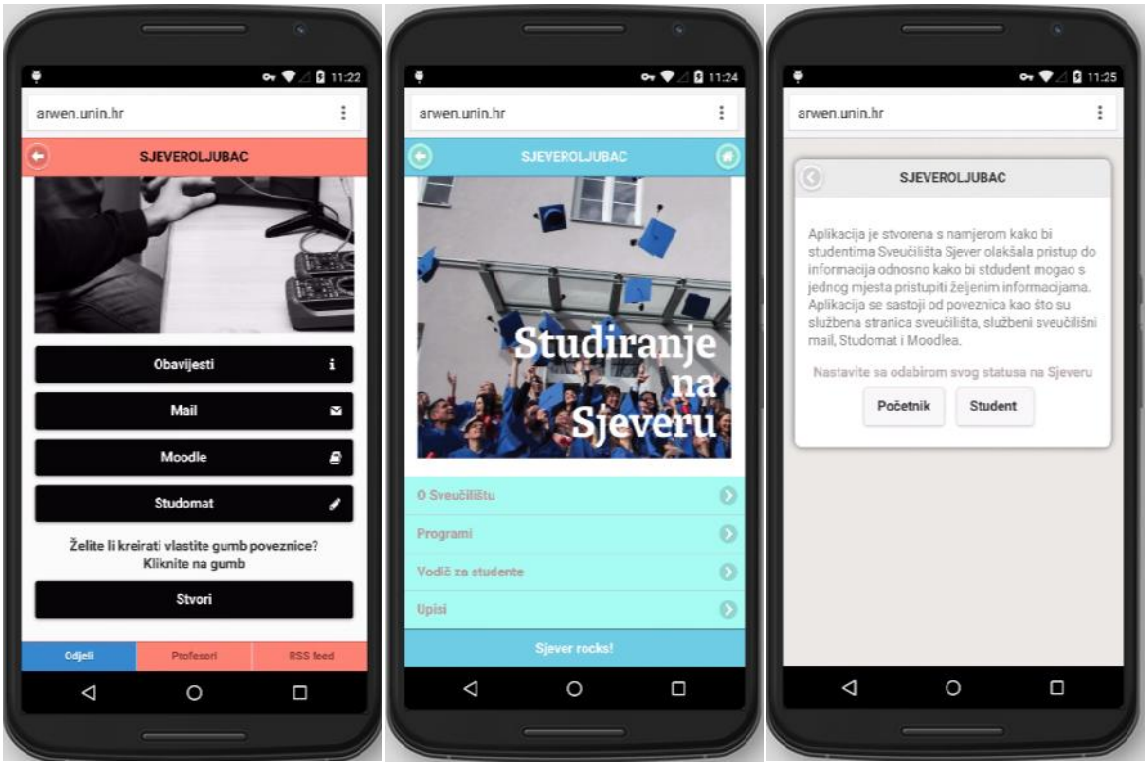
Stranica upisa



iOS simulator iPhone 5



Android Simulator Nexus 6



6. Zaključak

Cordova i jQuery Mobile su radni okviri koji u velikoj mjeri olakšavaju rad svakom programeru koji se korištenjem ovih radnih okvira može više usredotočiti na funkcionalnost same aplikacije. Korištenjem ovih radnih okvira skraćuje se vrijeme koje je potrebno za izgradnju aplikacije i vrijeme koje se troši na testiranje u različitim preglednicima i mobilnim platformama. Kad govorimo o Cordovi, uz svoje prednosti i nedostatke, Cordova programeru omogućuje da uz web tehnologije pristupi nativnim mobilnim mogućnostima uređajima. Možda Cordova nije najbolje rješenje za ovakvu vrstu razvoja pogotovo ako aplikacija mora koristiti sve mogućnosti mobilnih uređaja no kako vrijeme ide tako će se i smanjivati razlike između nativnih i hibridnih mobilnih aplikacija. Za razvoj nativnih mobilnih aplikacija potreban je veliki tim koji će stvoriti aplikaciju za svaki mobilnu platformu što uvelike povećava trošak i vrijeme za izgradnje takvog projekta. Isto tako, pri većem broju članova tima javljaju se problemi oko upravljanje projektom i svim zadacima koje je potrebno odraditi da bi se projekt uspješno završio. Cordova ne zahtjeva veliki tim i veliku cijenu troška rada te se ne javljaju problemi oko odabira mobilne platforme jer je za rad s Cordovom dovoljan samo jedan programer koji sam može odlučiti koje će mobilne platforme biti podržane u aplikaciji. I ako već ima mogućnost stvaranja projekta za sve mobilne platforme, zašto ih ne bi iskoristio.

Nakon rada sa jQueryom, tradicionalni Javascript postaje besmislen. Iako je jQuery Javascript knjižica, on u mnogočemu olakšava funkcionalnost i produktivnost same aplikacije. Tradicionalni Javascript je pomalo dosadan i ponekad zahtjeva pretjeranu redundanciju što uvelike otežava rad na samom projektu. Tu na scenu stupa jQuery. Kada se prvi put vidi sintaksa jQuery, svaki programer ostane malo zatečen. No kad se jednom svlada i nauči tek tada se vide sve prednosti koje jQuery pruža. jQuery Mobile je Javascript knjižica koja uz korištenje HTML5 data atributa omogućuje brže, efikasnije i estetski ljepše grafičko uređivanje elemenata koji se nalaze unutar aplikacije. Isto tako, radni okvir omogućuje stvaranje bilo kakve funkcionalnosti koju programer želi ostvariti. Radni okvir podržava mnoge načine za dodavanje prilagođenih funkcionalnosti koje programeru omogućuju stvaranje moćnih i interaktivnih mobilnih web aplikacija i stranica.

Što se praktičnog dijela rada tiče, u ovom radu je razvijena aplikacija ”Sjeveroljubac” koja služi kao sučelje za brži pristup informacijama. Aplikacija ima zadatak da studentu omogući bržu navigaciju do željene stranice. Stoga je osnova aplikacije bila kreirati listu koja će se sastoji od poveznica: službena stranica sveučilišta, službeni mail, moodle i studomat. Isto

tako, kreirane je lista sa odjelima koji se nalazi u sklopu sveučilišta. Tehnologije koje su korištene u radu su HTML, CSS, Javascript, jQuery, jQueryMobile, Ajax, XML, YQL, Node.js, JSON. Sama aplikacija je koncipirana tako da se sastoji od devet pojedinačnih html stranice unutar kojih se nalaze višestranične jQM stranice. Razlog tome je lakša navigacija kroz projekt i smanjivanje rizika od koalizije između skripti i ajax zahtjeva. Za sam izgled aplikacije je najzaslužniji jQuery Mobile koji je formirao aplikaciju kako bi ona u potpunosti zadovoljila sve zahtjeva mobilnog iskustva. Uz jQM u aplikaciju su implementirani razni grafički elementi kao što su ikone, dijaloški prozori, sklopive(collapsible) liste itd. Što se same funkcionalnosti tiče, aplikacija odrađuje sve zadatke koji se odvijaju na klijentskoj strani. Na stranici **profesori.html** uspješno su prikazani podaci sa web stranice (u ovom slučaju podaci o profesorima sa službene stranice sveučilišta), koji se dohvaćaju pomoću Ajax poziva i YQLa (Yahoo Query Language). Isto tako, na stranici **RSS.html** uspješno su dohvaćene i prikazane RSS obavijesti koje su u XML formatu. Da bi se u radu prikazala jedna od nativnih mogućnosti mobilnih uređaja, na stranici **postani_sjeveroljubac.html** se koristio Cordova dodatak ”geolocation” kako bi se dobila trenutna pozicija korisnika te uz implementaciju ”googlemapsa” prikazao put koji korisnik mora prijeći kako bi došao do Sveučilišta. Kako bi aplikacija imala interaktivni karakter, na stranici **odjeli.html** u zasebnim stranicama pojedinog odjela stvoren je dijaloški prozor unutar kojeg korisnik može sam dodavati svoje poveznice. Poveznice se pohranjuju u lokalnu pohranu i prikazuju u listi na zasebnim stranicama odjela. Iako se aplikacija isključivo bazira na front-end odnosno na klijentskoj strani, u aplikaciju je uspješno implementiran Node.js projekt (server strana) s kojom je ostvarana LDAP autorizacija korisnika, naravno, korisničko ime i lozinka se pohranjuju (ako to korisnik želi) u lokalnu pohranu. Za kraj je kreirana ikona aplikacije s kojom je stvoren vizualni identitet.

Po mom mišljenje aplikacija je iskoristila sve mogućnosti koje se pružaju pomoću programske zbirke Cordova. Naravno, uvijek se nešto može poboljšati i popraviti no to opet sve ovisi o zahtjevim korisnika i mogućnostima navedene zbirke. Možda je jedini nedostatak, nemogućnost implementacije SSO (Single-Sign-On) jer bi inače u tom slučaju apliakcija u potpunosti zadovoljila sve korisnikove želje.

Jurica Dizdar

7. Literaturne reference

[1] Zainul Setyo Pamungkas: PhoneGap Build Starter, Painless Mobile Apps Development, USA, 2014.

[2] John M. Wargo. Addison-Wesley: Apache Cordova 3 Programming, USA, 2014.

[3] Raymond K. Camden. Manning Publishing: Apache Cordova in Action, USA, 2014.

[4] Gustavo De La Vega Alvarez, Packt Publishing: Instant PhoneGap, USA, 2013.

[5] Apache Cordova.
URL: https://cordova.apache.org/docs/en/4.0.0/config_ref_index.md.html (01.09.2015.)

[6] Guillermo Rauch. Wiley Publishing: Smashing Node.js Javascript Everywhere, USA, 2012.

[7] Adrian Kosmaczewski, O'Reilly Media: Mobile Javascript Application Development, USA, 2014.

[8] Yahoo. URL: https://www.yahoo.com/ (01.09.2015.)

[9] jQuery. URL: https://jquery.com/ (02.09.2015.)

[10] Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012.

[11] Shane Gliser. Packt Publishing: Creating Mobile Apps with jQuery Mobile, USA, 2013.

[12] jQuery Mobile. URL: http://jquerymobile.com/ (03.09.2015.)

[13] Matt Doyle. Elated: Master Mobile Web Apps with jQuery Mobile (Fourth Edition), USA, 2014.

[14] Adam Boduch. Packt Publishing: jQuery UI Themes Beginner's Guide, USA, 2011.

[15] URL: https://developer.blackberry.com/html5/documentation/v2_2/modifying_your_config_file.html (04.09.2015.)

Popis slika

Slika 1.1 JQM i Cordova Izvor: URL: http://miamicoder.com/ 2
Slika 2.1 Cordova komponente Izvor: John M. Wargo. Addison-Wesley: Apache Cordova 3 Programming, USA, 2014. 4
Slika 2.2 Arhitektura nativne aplikacije Izvor: Raymond K. Camden. Manning Publishing: Apache Cordova in Action, USA, 2014. 5
Slika 2.3 PhoneGap Izvor: URL: http://phonegap.com/ 6
Slika 2.4 jQuery Mobile Izvor:URL: http://jquerymobile.com/ 9
Slika 2.5 Android SDK Manager Izvor: John M. Wargo. Addison-Wesley: Apache Cordova 3 Programming, USA, 2014. 16
Slika 4.1 jQuery Izvor: URL: https://jquery.com/ 24
Slika 4.2 Ajax Izvor: URL: https://jquery.com/ 27
Slika 4.3 HTML5 Izvor: Shane Gliser. Packt Publishing: Creating Mobile Apps with jQuery Mobile, USA, 2013. 29
Slika 4.4 jQM stranica Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012. 32
Slika 4.5 Dijaloški prozor Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012. 35
Slika 4.6 Osnovna lista Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012. 38
Slika 4.7 Teme jQM-a Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012. 39
Slika 4.8 ThemeRoller sučelje Izvor: Adam Boduch. Packt Publishing: jQuery UI Themes Beginner's Guide, USA, 2011. 40
Slika 5.1 Struktura projekta 41
Slika 5.2 Teme aplikacije 70
Slika 5.3 Ikona 70

Popis tabela

Tabela 4.1 Tablica sa atributima ”button ” komponente Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012. 37

Tabela 4.2 Tablica ikona kod jQM-a Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012. 37

Tabela 4.3 Tablica defaultnih uzoraka boje Izvor: Kris Hadlock. Peachpit Press: jQuery Mobile: Develop and Design, USA, 2012..... 39