Sveučilište u Rijeci – Odjel za informatiku

Jednopredmetna informatika

Dominik Barjaktarić

Web aplikacije za mobilne uređaje - Cordova

Završni rad

Mentor: dr. sc. Igor Jugo

Rijeka, rujan 2017.

SADRŽAJ

1. Uvod 1

2. Web tehnologije 2

2.1. HyperText Markup Language (HTML) 2

2.2. Cascading Style Sheets (CSS) 6

2.3. JavaScript 10

2.3.1. jQuery 11

3. Poslužiteljske web tehnologije 12

3.1. PHP 12

4. Hibridne mobilne aplikacije 16

5. Apache Cordova 19

5.1. Upotreba 20

5.2. Arhitektura hibridne mobilne aplikacije 21

5. Ogledna hibridna mobilna aplikacija 24

5.1. Struktura aplikacije 24

5.2. Opis funkcionalnosti aplikacije 25

5.2.1. Registracija novog korisnika 25

5.2.2. Prijava korisnika 27

5.2.3. Pregled profila 29

5.2.4. Objavljivanje i brisanje poruka 31

5.2.5. Pretraga profila 33

5.2.6. Pregled traženog profila 34

5.2.7. Praćenje traženog profila 35

5.2.8. Prikaz liste praćenih profila 35

5.2.9. Ažuriranje profilne slike pomoću kamere ili odabirom iz galerije slika 36

5.2.10. Odjava korisnika 38

6. Zaključak 40

7. Popis literature 42

8. Popis slika 42

9. Popis tablica 43

1. Uvod

U današnje vrijeme sve je veći zahtjev za proces brzog razvijanja aplikacija. Programeri moraju učiti i koristiti sve više programskih jezika i alata, kako bi bili konkurentni na tržištu. Pojavom prvih pametnih telefona na tržištu, pojavio se veliki interes za razvoj mobilnih aplikacija. Tri su glavne mobilne platforme na tržištu: Android, iOS i Windows phone. Svaka od tih platformi koristi drugi programski jezik pomoću kojeg se razvijaju mobilne aplikacije. Do poteškoća i problema dolazi kada želimo mobilnu aplikaciju razvijati za sve tri mobilne platforme. Potrebno je više programera za razvoj aplikacije, što i ujedno povećava troškove projekta. Razvija se ista aplikacijska logika koristeći različite programske jezike. Takav razvoj nije idealan i teško je održavati. Kako bi se olakšao razvoj mobilnih aplikacija za različite platforme, razvijena je tehnologija koja koristi jedan programski jezik – JavaScript. Pomoću JavaScript-a dovoljno je aplikacijsku logiku implementirati samo jednom kako bi aplikacija radila na više mobilnih platformi.

U ovom radu opisuje se tehnologija za razvoj hibridnih mobilnih aplikacija. Kroz praktični primjer prikazat će se na koji način se razvijaju hibridne mobilne aplikacije te koje tehnologije je poželjno prethodno poznavati kako bi proces razvoja bio što kraći i efikasniji. U prvom poglavlju objasnit ćemo osnovne web tehnologije kao što su HTML, CSS i JavaScript. Drugo poglavlje opisuje rad poslužiteljskih tehnologija, konkretno PHP i MySQL bazu podataka. U trećem poglavlju objašnjavaju se vrste mobilnih aplikacija te uspoređuju prednosti i nedostatke. Radni okvir Apache Cordova te popis njenih API-ja prikazani su u četvrtom poglavlju. Peto poglavlje prikazuje ogledni praktični primjer izrađene hibridne mobilne aplikacije. Prikazuju se slike izgleda aplikacije, te se objašnjava aplikacijska logika.

Ovaj rad koristan je programerima koji žele naučiti razvijati mobilne aplikacije za više platformi koristeći jedan programski jezik. Web programeri koji imaju predznanje u HTML, CSS i JavaScript, proces učenja tehnologije bit će kraći od programera koji se prvi put susreće s web tehnologijama.

2. Web tehnologije

2.1. HyperText Markup Language (HTML)

HyperText Markup Language (skraćeno HTML) se koristi za strukturiranje i prikazivanje web stranice i njenog sadržaja. HTML nije programski jezik, on je označni (engl. markup) jezik i upotrebljava se kako bi web pregledniku rekao kako prikazati web stranice.

(https://mzl.la/2dqvAVa, 2017.)

HTML se sastoji od skupa elemenata, koji se dalje sastoje od oznaka i sadržaja. Oznake se koriste kako bi označile elemente i prikazale ih na određeni način. Oznake mogu riječ, paragraf ili sliku prikazati kao hipervezu na neko drugo mjesto, mogu istaknuti riječi, povećati ili smanjiti font itd. Na primjer:

Ovo je paragraf.

Ako želimo da rečenica bude prikazana kao paragraf u HTML dokumentu, moramo dodati oznake <p> i </p> oko nje:

<p>Ovo je paragraf.</p>

Dijelovi elementa su:

* Oznaka za otvaranje – označava gdje element počinje djelovati. Zapisuje se određena oznaka između znakova manje (<) i više (>). U navedenom primjeru, otvarajuća oznaka je <p> te označava početak paragrafa.
* Oznaka za zatvaranje – analogno oznaci za otvaranje, označava gdje element prestaje djelovati. Sastoji se od oznake omotan između znakova manje (<), kose crte (/) i više (>). U navedenom primjeru, zatvarajuća oznaka je </p> te označava kraj paragrafa. Neki elementi zahtijevaju samo oznaku za otvaranje. Radi se o praznim elementima (engl. *void elements*) koji nemaju sadržaja (npr. elementi img i br)
* Sadržaj – sadržaj elementa u ovom primjeru je rečenica „Ovo je paragraf“.
* Element – kompletan element koji se sastoji od oznake za otvaranje, sadržaja i oznake za zatvaranje: „<p>Ovo je paragraf.</p>“.

HTML dopušta umetanje elementa unutar elemenata. To su ugniježđeni elementi (engl. *nested elements*). Primjerice želimo li istaknuti jednu ili više riječi u paragrafu, radimo to ovako:

<p>Ovo je <strong>novi</strong> paragraf.</p>

Izlaz:

Ovo je **novi** paragraf.

Kod ugniježđenih elemenata treba paziti na pravilno ugnježđivanje, dakle pogrešno bi bilo sljedeće:

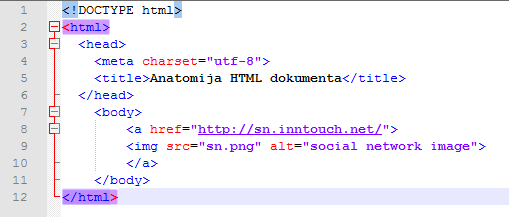
<p> Ovo je <strong>novi paragraf.</p></strong>

Izlaz:

Ovo je **novi paragraf.**

zato što nismo zatvorili element koji je prethodno bio otvoren unutar drugog elementa. Web preglednik u tim slučajevima „pogađa“ što je programer htio reći.

Što sve HTML datoteka treba sadržavati kako bi bila potpuna nalazi se na sljedećoj slici.

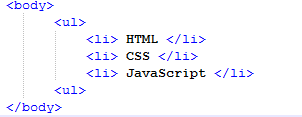


Slika 1: Anatomija HTML dokumenta

* <!DOCTYPE html> - DOCTYPE-ovi su skup pravila koja HTML datoteka treba slijediti.
* <html></html> - *html* element obuhvaća cijeli sadržaj stranice; još se naziva i korijenski element (engl. *root element*).
* <head></head> - *head* element ponaša se kao kontejner za sve elemente koje želimo uključiti u HTML dokument čiji sadržaj nije vidljiv korisniku. Tu spadaju elementi kao što su ključne riječi, opis stranice, naslov stranice, eksterni CSS dokument, deklaracija skupa znakova i slično. Prateći sliku primjećujemo kako je u *head* elementu naveden naslov web stranice i *meta* elementkoji govori da je sadržaj kodiran pomoću *utf-8* skupa znakova.
* <body></body> - *body* element sadrži cijeli sadržaj kojeg korisnik vidi u trenutku učitavanja stranice; tekst, slike, video, audio zapisi i drugi.

Svaki HTML dokument mora sadržavati minimalno gore navedene elemente. Dalje slijede elementi koji su opcionalni:

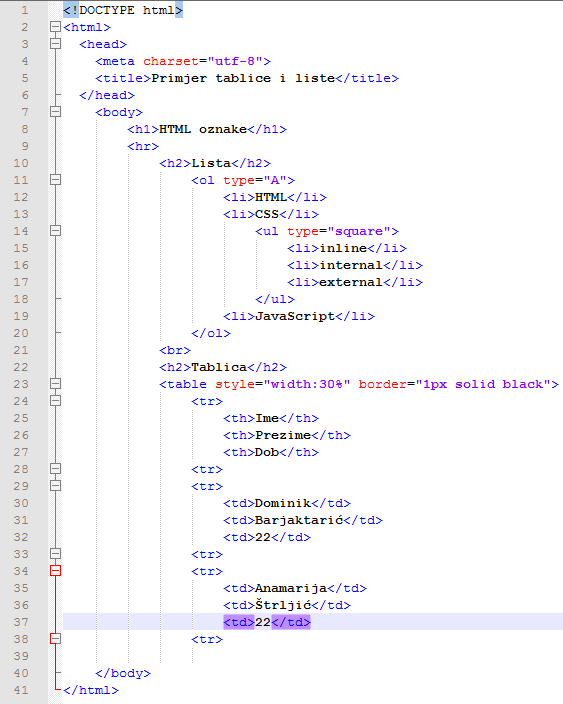
* <a></a> - *a* element definira hipervezu koja povezuje dva HTML dokumenta. Dakle veza postavljena na jednoj stranici koja vodi na drugu stranicu. Atribut *href* oznake *a* sadrži putanju poveznice.
* <img> - *img* element jedan je od praznih elemenata. Navedeni atributi pridodani elementu *img* su *src* i *alt* (engl. *source* i *alternative*). *Src* sadrži putanju lokacije slike, a *alt* je sadržaj koji će se prikazati ako se slika neuspješno učita.
* <h1></h1> - označava naslov najviše razine (engl. *heading 1*), analogno postoje *h2, h3, h4, h5, h6.*
* <br> - Prilikom pisanja HTML kôda, tipka za novi red je zanemarena. Oznaka *br* jednaka je pritiskom tipke za novi red. Ovaj element je prazan element.
* <ol></ol> i <ul></ul> - *ol* oznaka (engl. *ordered* *list*) označava listu s rednim brojem. Dodatno postoji atribut gdje redne brojeve možemo zamijeniti rimskim brojevima, slovima i slično. Ugniježđeni elementi potrebni su nam kod izrade lista. Oznaka <li></li> označava element liste (engl. *list item*). Analogno postoji *ul* oznaka (engl. *unordered* *list*) koja elemente liste nabraja pomoću simbola (krug, kvadrat, slika).



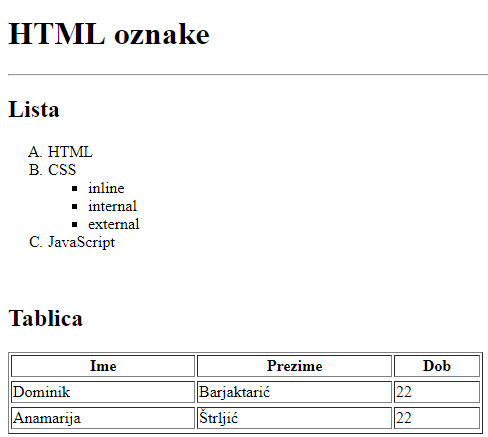
Slika 2: Primjer neuređene liste

* <table></table> - *table* oznaka potrebna nam je za izradu tablice. Sastoji se od više ugnježđenih elemenata gdje su <table></table> krajnje vanjski.

Primjer ugniježđenih listi te tablice dan je u jednostavnom primjeru na sljedećoj slici.



Slika 3: HTML oznake kod



Slika 4:Izlaz HTML koda

2.2. Cascading Style Sheets (CSS)

*Cascading Style Sheets* (skraćeno CSS) je kôd koji upotrebljavamo za opisivanje stila HTML dokumenta.

CSS također nije programski jezik. CSS je jezik stila (engl. *style sheet language*). Kôdom CSS-a odlučujemo kako će se HTML kôd prikazivati.

Na primjer, želimo li obojiti cijeli sadržaj koji se nalazi u oznakama za paragraf crvenom bojom, napravit ćemo to ovako:

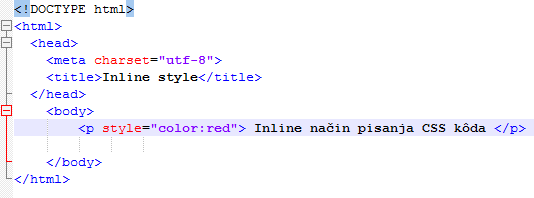
p {color: red;}

CSS kôd se može nalaziti i koristiti na tri različita načina:

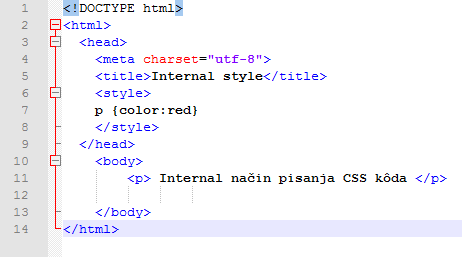
* CSS u redu – CSS kôd direktno ubacujemo u HTML elemente
* Interni CSS – CSS kôd pišemo u HTML dokumentu pod elementom

<style></style>

* Eksterni CSS – CSS kôd se piše u zasebnoj datoteci koja ima .css nastavak te se taj dokument poziva u *head* elementu HTML dokumenta.



Slika 5: CSS kôd u redu.



Slika 6: Interni CSS kôd

Za trivijalne web stranice može se koristiti interni stil i stil u redu, ali kod većih stranica preporučeno je koristiti eksterni stil.

Da bi se koristio eksterni stil, ubacuje se linija kôda u *head* elementu HTML-a koji izgleda ovako:

<link href=“stil.css“ rel=“stylesheet“ type=“text/css“>

Href atribut označava putanju CSS datoteke, atribut rel definira vezu, u većini slučajeva će to biti stylesheet. Na kraju je atribut type koji označava vrstu internetskog medija povezanog dokumenta.

CSS datoteka sastoji su od jednog ili više skupa pravila (engl. *rule set*).

p {color: red;}

Skup pravila sastoji se od selektora i deklaracije. Selektor je u ovom primjeru p, koji znači da će svaki paragraf u HTML dokumentu biti u crvenoj boji. Uzmimo u obzir da skup pravila može sadržavati više selektora:

p,h1,h2 {deklaracija}

Deklaracija se sastoji od svojstva i vrijednosti svojstva (engl. *Properties* i *Property value*).

U navedenom primjeru, svojstvo je color. Svojstvo daje stil HTML elementu[[1]](#footnote-1). Red je vrijednost svojstva color. Svojstvo color ima mnoštvo predeklariranih boja odnosno vrijednosti. Nadalje, vrijednost svojstva color može se specificati i na drugi način:

color: #ff0000;

color: rgb(255,0,0)

Tri su osnovne pravila sintakse za pisanje CSS stilova:

* Svaki skup pravila mora biti zamotan u vitičaste zagrade te se selektor nalazi prije lijeve vitičaste: h1{deklaracija}
* U svakoj deklaraciji potrebno je odvojiti svojstvo i vrijednost svojstva dvotočkom
* U svakoj deklaraciji poslije svake vrijednosti svojstva koristimo točku-zarez koja odvaja jedno svojstvo od drugoga

Na više se načina može pristupiti pojedinom elementu HTML dokumenta:

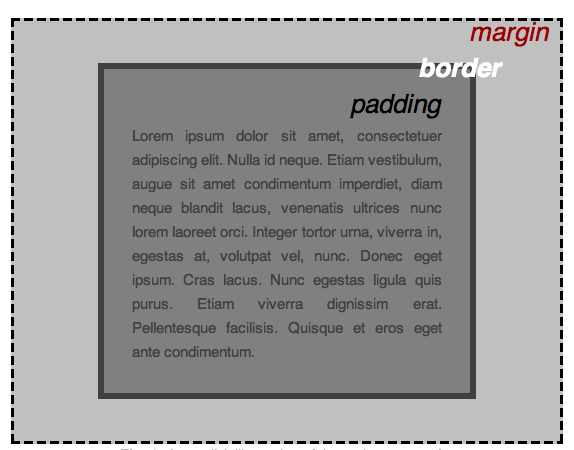
Tablica 1: Vrste CSS selektora.

|  |  |  |
| --- | --- | --- |
| **Ime selektora** | **Što obuhvaća** | **Primjer** |
| Selektor elementa | Sve HTML elemente specificirane oznake | p  selektira sve paragrafe HTML dokumenta |
| ID selektor | Element u HTML dokumentu kojem je dodijeljen jedinstveni ID atribut | #my-id  selektira element čiji je atribut *id=“my-id“* |
| Klasni selektor | Identično ID selektoru, ali može imati više instanci na istoj stranici | .my-class  selektira elemente čiji je atribut *class=“my-class“* |
| Atributni selektor | Element na stranici sa specificiranim atributom | img[src] selektira  *<img src=“myimage.png“>* |
| Pseudo-klasni selektor | Element sa specificiranim stanjem (npr. kada se kursorom pređe preko elementa) | a:hover selektira <a> ali samo kad se kursor nalazi na elementu |

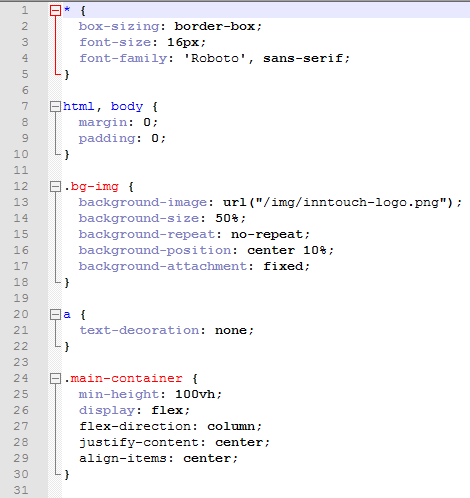
Postoji još mnogo varijacija selektora.[[2]](#footnote-2)

CSS je baziran na kontejner modelu (engl. *box model*). Svaki kontejner uzima mjesto na HTML stranici i ima sljedeća svojstva:

* *padding* - prostor oko sadržaja
* *border* – okvir oko sadržaja
* *margin* – margina odnosno prostor izvan elementa



Slika 7: Prikaz kontejner modela



Slika 8: Primjer CSS kôda

2.3. JavaScript

JavaScript je programski jezik koja na web stranici omogućuje interaktivnost s korisnikom (na primjer: igre, pritisci gumbova, podaci uneseni u obrasce i slično). JavaScript (dalje JS) je dinamički programski jezik koji kada se primjeni u HTML dokumentu može pružiti dinamičnu interaktivnost na web stranicama. (https://mzl.la/2bKKtll, 2017.)

Veliki je izbor alata zasnovan na jezgri JS-a koji omogućuju dodatne funkcije uz minimalni napor. To uključuje:

* Aplikacijsko programsko sučelje (skraćeno API) web preglednika – API-ji ugrađeni u web preglednike koje pružaju funkcionalnosti poput dinamičkog kreiranja HTML-a i postavljanja CSS stilova, stvaranje 3D grafike i audio uzoraka.
* API-ji treće strane koji omogućavaju programerima da uključe funkcionalnosti na svojim web stranicama od drugih davatelja sadržaja kao što su Twitter ili Facebook
* Radni okviri i biblioteke trećih strana mogu se koristiti kako bi brzo izgradile web stranice i aplikacije. (https://mzl.la/2bKKtll, 2017.)

JS zahtjeva više vremena za učenje, za razliku od HTML-a i CSS-a.

* JS kôd može se nalaziti u HTML dokumentu unutar <script></script> elementa ili u vanjskoj datoteci s ekstenzijom .js.
* Vanjski JS dokument povezujemo s HTML dokumentom tako da se ubacuje sljedeća linija koda u head elementu:

<script src=“scripts/main.js“></script>

Značajke JS-a poput varijable, komentari, operatori, if grananja, funkcije i drugih, nisu veoma različite drugim popularnim programski jezicima kao što su C++, Java i Python. Kao i kod svakog programskog jezika, sintaksa je drugačija.

2.3.1. jQuery

JQuery je JavaScript biblioteka. Lagan je za korištenje te čini funkcije poput manipulacija HTML dokumenata, rukovanje događajima, animacije i Ajax pozivi mnogo jednostavnijim. S kombinacijom svestranosti i proširivosti, jQuery je promijenio način na koji milijuni ljudi pišu JavaScript. (https://jquery.com/, 2017.)

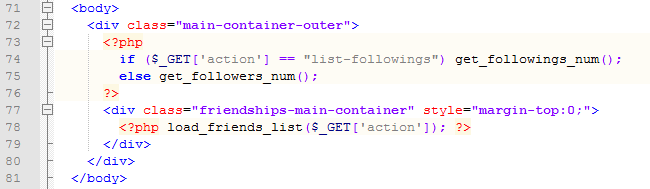
Popis značajki jQuery-a:

* DOM manipulacija – olakšan je način selekcije elemenata DOM-a.
* Upravljanje događajima – nudi elegantan način za dohvaćanje širokog raspona događaja.
* AJAX podrška – jQuery je znatno olakšao funkcije AJAX tehnologije.
* Animacije – dolazi s mnogo izgrađenih animacijskih efekata koje možete koristiti na svojim web stranicama.
* Laganost – jQuery je vrlo lagana biblioteka, teži svega 19KB.
* Podrška za više preglednika – jQuery ima podršku i dobro funkcionira u programima IE 6.0+, FF 2.0+, Safari 3.0+, Chrome i Opera 9.0+.
* Najnovija tehnologija – jQuery podržava CSS3 selektore i osnovnu XPath sintaksu.

3. Poslužiteljske web tehnologije

3.1. PHP

PHP (*PHP Hypertext preprocessor*)[[3]](#footnote-3) je skriptni jezik koji može biti ugrađen (engl. *embedded*) u HTML. Sintaksa je uzeta iz C-a, Jave i Perla. Cilj PHP-a je mogućnost brzog i jednostavnog pisanja dinamički generiranih stranica. U HTML dokumentu moguće je otvoriti PHP skriptu koja počinje s <?php i završava s ?>. Na slici se nalazi primjer:



Slika 9: PHP unutar HTML-a

PHP je poseban po tome što se kôd izvršava na poslužitelju (engl. *server*), generira se HTML koji se zatim šalje klijentu. Značajke poslužiteljske skripte su:

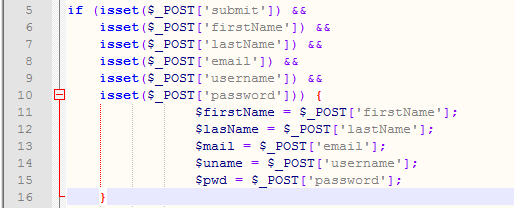
* Pokreće se na poslužitelju – kôd je ugrađen u HTML dokumentu
* Napravljen je za interakciju s kontinuiranom pohranom podataka – nalik je izravnoj komunikaciji korisnika i baze podataka
* Olakšava prijenos podataka s poslužitelja u preglednik
* Radi na zahtjev (engl. *on demand*)
* Posebne funkcije u dinamičkim web aplikacijama – validacija korisnika, spremanje i dohvaćanje podataka, kretanje među ostalim stranicama
* Ima veliku ulogu u tome kako se gradi baza podataka te kako se kasnije manipulira njome (http://bit.ly/1q2804m, 2017.)

PHP ima nekoliko unaprijed definiranih varijabli zvanih superglobalna polja (engl*. superglobals*). Ona su uvijek dostupna bez obzira na opseg i možemo im pristupiti iz bilo koje funkcije, klase ili datoteke. Superglobalna polja su:

* $GLOBALS
* $\_SERVER
* $\_GET
* $\_POST
* $\_FILES
* $\_COOKIE
* $\_SESSION
* $\_REQUEST
* $\_ENV

U oglednoj aplikaciji korištena su superglobalna polja $\_GET, $\_POST i $\_REQUEST. Kada korisnik ispuni neki obrazac, na primjer obrazac za prijavu ili registraciju, poziva se PHP skripta koja se nalazi na poslužitelju gdje se podaci ispunjenog obrasca koriste za daljnju obradu i/ili spremanje u bazu, ili za neki drugi rad. Obje metode slanja podataka obrasca, GET i POST, izrađuju asocijativno polje ključeva i vrijednosti gdje su ključevi imena pojedinačnih polja obrasca. Razlika GET-a i POST-a je što GET prosljeđuje vrijednosti kroz parametre URL-a, a POST preko HTTP POST metode. Podaci poslani GET metodom nakon slanja vidljivi su u URL adresi. Zaključujemo da obrasce s osjetljivim podacima poput lozinke nećemo slati GET metodom. Nadalje, GET metoda ima ograničenje na količinu poslanih znakova (2048), dok POST metoda nema ograničenja.

$\_REQUEST je superglobalno polje koje sadrži $\_GET, $\_POST i $\_COOKIE polja.



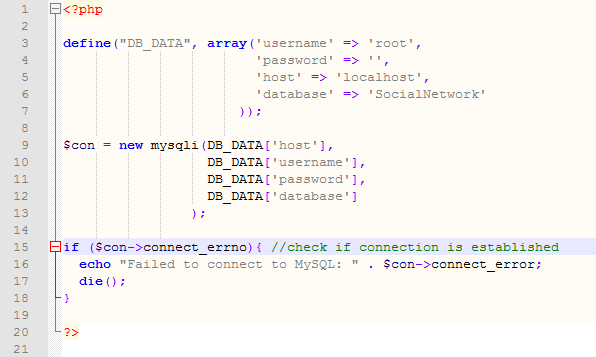
Slika 10: Primjer POST metode PHP skripte

Za primjer ćemo uzeti registraciju korisnika. Podatke ćemo slati POST metodom, jer putem obrasca korisnik šalje lozinku. Na slici je pojednostavljeni kôd POST metode. U PHP skripti, prvo se provjerava da li su postavljeni pojedini ključevi unutar $\_POST superglobalnog polja. Ako jesu, vrijednosti tih ključeva se deklariraju novim varijablama koje će se koristiti za daljnji rad kao što je izrada upita za bazu.

3.2. MySQL

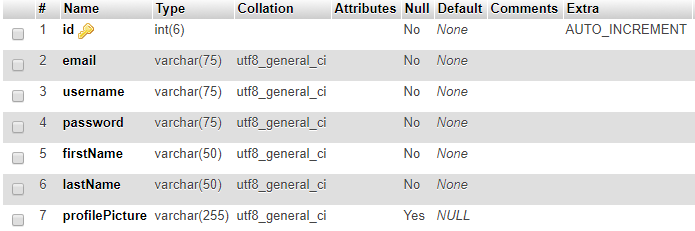
Baza podataka zasebna je poslužiteljska aplikacija za pohranu podataka. U današnje vrijeme koriste se sustavi za upravljanje relacijskim bazama podataka (engl. *relational database management system,* skraćeno RDBMS) za pohranu i upravljanje podacima. Baza podataka naziva se relacijska jer se može sastojati od više tablica koje mogu biti međusobno povezane. Dakle, podaci mogu biti pohranjeni u različitim tablicama, a relacije tablica se uspostavljaju putem primarnog ključa (engl. *primary key*) i jednog ili više vanjskih ključeva (engl. *foreign keys*). (http://bit.ly/2wmuogV, 2017.)

MySQL je RDBMS otvorenog koda koja se temelji na strukturiranom upitnom jeziku (engl. *structured query language,* skraćeno SQL). Radi na svim glavnim platformama; Linux, UNIX, Windows i drugi. Često se koristi za razvoj web aplikacija.



Slika 11: Primjer izrade objekta u PHP skripti za povezivanje s MySQL bazom

U PHP skripti ogledne aplikacije, stvara se objekt klase mysqli i prosljeđujemo parametre za pristupanje bazi. Taj objekt koristi se u ostalim dijelovima PHP skripte kako bi se izvršili upiti na bazu. Baza se sastoji od tri tablice: *Users*, *Posts* i *Friendships*. Tablica *Users* sadrži podatke o korisniku, te ima primarni ključ *id* čija se vrijednost automatski inkrementira i dodjeljuje prilikom spremanja nove n-torke.



Slika 12: Relacija Users

Tablica *Friendships* ima dva vanjska ključa *User1* i *User2,* te primarni ključ *id* koji identificira prijateljstvo. U usporedbi s Facebook stranicom, ogledna aplikacija nema način obostranog prijateljstva. Kada korisnik pronađe drugog korisnika te želi s njime stvoriti vezu, može to učiniti pritiskom na gumb *Follow* na korisničkom profilu. Stvara se nova n-torka u tablici *Friendship*:



Slika 13: Ubacivanje n-torke kada korisnik pritisne na gumb Follow u oglednoj aplikaciji

Korisnik. 4500 tada prati korisnika 4502, no ne i obrnuto.

Tek kada korisnik s ID-om 4502 pritisne gumb *Follow* na profilu korisnika s ID-om 4500, tada su obostrano povezani.

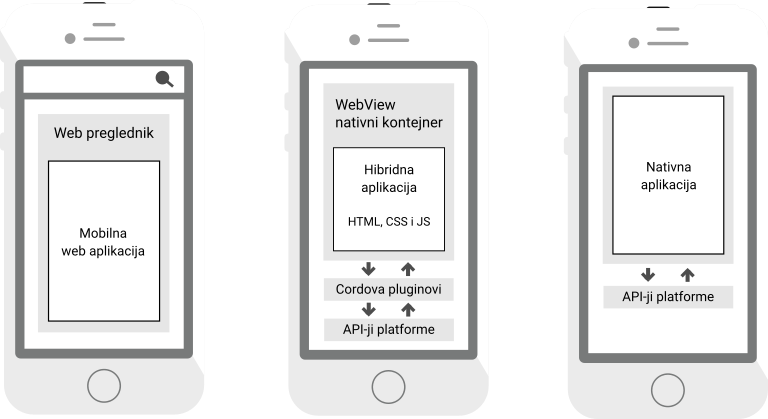


Slika 14: Primjer n-torke prilikom praćenja drugog profila

Tablica *Posts* sadrži podatke o porukama koje svaki korisnik objavljuje na svom profilu*.* Ima primarni ključ *id* koji jedinstveno identificira svaku n-torku koja se automatski inkrementira prilikom svake nove n-torke, kao što je to slučaj kod tablica *Users* i *Friendships*. Vanjski ključ je *FK\_Users\_id.*

4. Hibridne mobilne aplikacije

Postoje tri vrste mobilnih aplikacija; mobilne web aplikacije, hibridne mobilne aplikacije i nativne mobilne aplikacije. Da bi razumijeli razliku između hibridnih mobilnih aplikacija i nativnih mobilnih aplikacija, te mobilnih web aplikacija, moramo jasno definirati njihove uloge, tehnologiju za razvoj, prednosti i nedostatke.



Slika 15: Razlika mobilne web, hibridne i nativne aplikacije

Mobilna web aplikacija je takva vrsta mobilne aplikacije kojoj se pristupa putem web preglednika na mobilnom uređaju. Takvim aplikacijama se može pristupiti i preko web preglednika na računalu. Mobilna web aplikacija je dizajnirana da bude responzivna; dizajn se prilagođava prema različitim rezolucijama ekrana mobilnih i tablet uređaja kako bi se aplikacija pravilno prikazala. Da bi se mogla koristiti, mobilni uređaj mora imati vezu na internet. Koriste se standardne web tehnologije kao što su HTML, CSS i JavaScript. Mogu imati isti izgled i korisničko iskustvo kao nativna mobilna aplikacija. Mobilna web aplikacija se ne može uploadati na app store mobilne platforme, te je pristup i korištenje nativnih funkcija mobilnog uređaja ograničeno.

Hibridne i nativne mobilne aplikacije imaju isti fizički format, ali su razvijeni korištenjem različitih tehnologija. Hibridne mobilne aplikacije se razvijaju pomoću standardnih web tehnologija (HTML, CSS i JavaScript), dok se nativne mobilne aplikacije razvijaju nativnim programskim jezicima za određenu mobilnu platformu (Java za Android platformu, Objertive-C/Swift za iOS platformu, te .NET programski jezici za Windows Phone i Windows platformu).

Kod razvoja nativne mobilne aplikacije za različite mobilne platforme, svaka aplikacija se mora razvijati zasebno za određenu platformu. Što znači da kada se razvija nativna aplikacija za jednu određenu platformu, mora se reimplementirati njena ista programska logika za druge mobilne platforme. Re-implementiranje iste aplikacijske logike za različite platforme je teško i nije idealno rješenje iz razloga jer se moraju koristiti nativni programski jezici ciljane mobilne platforme, te rukovati različitim problemima specifični za svaku platformu.

Hibridna mobilna aplikacija ima prednost naspram nativne mobilne aplikacije jer se koristi isti odnosno jedan *code base (*HTML, CSS i JavaScript) za razvoj aplikacije za različite mobilne platforme. Drugim riječima, kôd aplikacije piše se jedno, te se isti koristi za druge mobilne platforme uz minimalne promjene koje su specifične za svaku platformu. Sva aplikacijska logika implementirana je koristeći jedan programski jezik – JavaScript. Hibridne mobilne aplikacije mogu implementirati i koristiti sve nativne funkcije mobilne platforme kao što je to kod nativnih mobilnih aplikacija.

Vrijeme potrebno za razvoj hibridne mobilne aplikacije za različite mobilne platforme je daleko kraće od vremena potrebno za razvoj nativne mobilne aplikacije za svaku mobilnu platformu. Nadalje, potrebni resursi za implementiranje hibridne mobilne aplikacije su minimizirani u usporedbi s razvojem nativne mobilne aplikacije. Razlog je taj što hibridne mobilne aplikacije koriste ujedinjeni programski jezik, dok nativne mobilne aplikacije koriste više ne ujedinjenih programskih jezika, što za razvoj zahtijevaju više programera s različitim znanjem i vještinama.

Nativna mobilna aplikacija radi malo brže od hibridne mobilne aplikacije, iz razloga jer je aplikacijski kôd pisan u nativnom programskom jeziku određene platforme koji je optimiziran. Korištenjem najboljih praksi prilikom razvoja hibridne mobilne aplikacije, mogu se povećati njene performanse kako bi bile blizu, pa čak i iste performansama nativne mobilne aplikacije.

Dakle, najvažnija razlika između nativne mobilne aplikacije i mobilne web aplikacije je da nativna mobilna aplikacija ima pristup i mogućnost korištenja nativnih funkcija mobilnog uređaja, dok su funkcije dostupne za mobilne web aplikacije ograničene.

Za hibridnu mobilnu aplikaciju može se reći da je ona mobilna web aplikacija koja se prikazuje pomoću nativnog kontejnera, tzv. WebView, o kojem će se detaljnije reći u idućem poglavlju. Ponaša se kao nativna mobilna aplikacija i ima kompletan pristup nativnim funkcijama uređaja.

Iz perspektive web programera, hibridne mobilne aplikacije odlično su rješenje jer se koriste web tehnologije za razvoj. Vještine koje web programer ima uključujući radni okvir Apache Cordova biti će dovoljne za razvoj hibridne mobilne aplikacije za različite platforme u kratkom vremenskom razdoblju.

Iz perspektive programera sa znanjem nativnog programskog jezika mobilne platforme, morat će provesti neko vrijeme za učenje standardnih web tehnologija. Na kraju će moći razvijati mobilne aplikacije za više platformi u kraćem vremenskom razdoblju i uz manje napora nego za razvoj iste aplikacije za svaku platformu zasebno koristeći njen nativni programski jezik.

Tablica 2: Ključne razlile između mobilnih aplikacija.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Mobilna web aplikacija | Hibridna mobilna aplikacija | Nativna mobilna aplikacija |
| Mogućnost uploada na app store platforme | ne | da | da |
| Tehnologije za razvoj | Standardne web tehnologije:  HTML, CSS i JavaScript | | Nativni programski jezik mobilne platforme:  Java, Swift/Objective-C, .NET programski jezici |
| Složenost | Normalno | Normalno | Visoko |
| Podrška za više mobilnih platformi | Da | Da | Ne |
| Korištenje nativnih funkcija mobilnog uređaja | Ograničeno | U potpunost (pomoću hibridnih aplikacijskih okvira kao na primjer Apache Cordova) | U potpunosti |
| Performanse | Vrlo dobro | | Odlično |

Tablica iznad ujedinjuje i prikazuje ključne razlike između vrsta mobilnih aplikacija koje smo definirali i naveli u poglavlju.

5. Apache Cordova

Projekt Apache Cordova[[4]](#footnote-4) je projekt otvorenog koda koji ima svrhu izrade hibridnih mobilnih aplikacija korištenjem standardne web tehnologije HTML, CSS i JavaScript. Apache Cordova nudi skup JavaScript API-ja, čijom se implementacijom omogućuje pristup nativnim funkcijama mobilnog uređaja. Te nativne funkcije mogu biti sljedeće:

* Pristup i obrada kontakt liste
* Pristup i obrada datoteke
* Fotografiranje pomoću kamere
* Dohvat fotografije iz galerije
* Snimanje zvuka pomoću mikrofona
* Dohvat smjera uređaja pomoću kompasa
* Lokacija uređaja pomoću GPS-a
* Dohvat podataka o pokretu uređaja
* Dohvat podataka o povezanosti uređaja s mobilnom mrežom ili internetom

Apache Cordova podržava širok spektar različitih mobilnih platformi:

* Android
* iOS
* Windows platforma
* Windows Phone 8
* Windows 8
* BlackBerry
* Tizen
* Web OS
* Firefox OS
* Ubuntu

5.1. Upotreba

Da bi razumjeli važnost upotrebe Apache Cordova, moraju se definirati trenutni izazovi prilikom razvoja mobilnih aplikacija:

* Svaka mobilna platforma ima svoju programsku filozofiju.
* Svaka mobilna platforma ima svoj skup jedinstvenih problema.
* Razvoj, testiranje i održavanje nativnih mobilnih aplikacija za različite mobilne platforme je težak i skup proces..

Jedan od najvećih izazova mobilnog razvoja je ta da svaka mobilna platforma ima svoju programsku filozofiju; nekoliko je programskih jezika i alata potrebno kako bi se razvijala mobilna aplikacija za različite mobilne platforme.

Rezultat navedenog izazova, te razvoj, testiranje i održavanje jedne takve aplikacije s više implementacija za više platformi je teško i skupo. Ovime bi postojalo više *code base*-a koji su nekonzistentni, jer svaki *code base* bit će pisan u drugom programskom jeziku od drugog developera. Vrlo je teško naći developera koji ima znanje i vještine svih nativnih programskih jezika i alata.

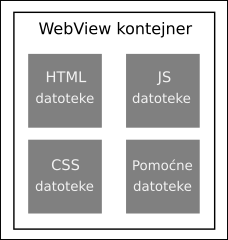
Daljnji izazov je rukovanje s nekompatibilnošću mobilnih platformi. Primjer problema s kojim se može susresti prilikom razvoja nativne mobilne aplikacije za Android i iOS je ta da se kod Android platforme, SMS poruke mogu slati direktno koristeći API platforme, dok se kod iOS platforme mora pokrenuti njena platformska SMS aplikacija kako bi se mogle slati poruke. To znači da je teret ne samo prilikom reimplementacije aplikacijske logike za različite platforme, nego je i teret za održavanje konzistentnosti ponašanja aplikacije za različite platforme, jer se moraju pronaći različita rješenja i načina za implementaciju funkcije.

Apache Cordova smanjuje tu složenost navedenih izazova. Daje mogućnost korištenja jednog programskog jezika JavaScript za pisanje aplikacijske logike za različite mobilne platforme. Drugim riječima, ne mora se koristiti skup nativnih programskih jezika za razvoj. Apache Cordova omogućava postojanje samo jednog *code base*-a za sve implementacije mobilne aplikacije za različite platforme. Time se drastično smanjuje složenost razvoja, testiranja i održavanja mobilnih aplikacija.

Kada postoji samo jedan *code base*, omogućava se fleksibilnost za rukovanje jedinstvenih problema svake platforme. Sve je centralizirano na jednom mjestu u aplikacijskom kodu, što rezultira lakšim čitanjem i održavanjem.

5.2. Arhitektura hibridne mobilne aplikacije

Glavne komponente jedne hibridne mobilne aplikacije su HTML, CSS i JavaScript datoteke. Također može sadržati i dodatne pomoćne datoteke kao na primjer JSON datoteke ili slike. U HTML datotekama se pozivaju JavaScript i CSS datoteke. Za korištenje nativnih funkcija mobilnog uređaja, u JavaScript datoteci se pozivaju funkcije Apache Cordova API-ja.



Slika 16: Apache Cordova arhitektura

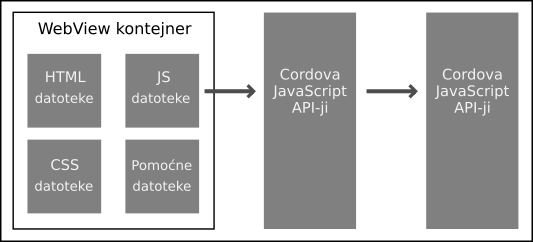
Apache Cordova stvara WebView kontejner koji koristi dostupni prostor ekrana mobilnog uređaja. U WebView kontejneru se učitavaju aplikacijske HTML, CSS i JavaScript datoteke.

Važno je istaknuti da je WebView kontejner nativna komponenta platforme koja se koristi za prikaz web stranice ili sadržaja HTML datoteke u aplikacijskom prozoru. Može se reći da je to ugrađeni mobilni web preglednik unutar hibridne aplikacije koja omogućuje prikaz web sadržaja odnosno datoteka.

Kada se hibridna aplikacija na mobilnom uređaju pokrene, učitava se zadana početna datoteka index.html u aplikacijskom WebView kontejneru, te zatim dodjeljuje kontrola, dopuštajući tako interakciju korisnika s aplikacijom. Interakcija se vrši putem unosa podataka, dodirom na akcijski gumb i pregledom rezultata u aplikacijskom WebView kontejneru.

Zahvaljujući toj tehnici i zato što je WebView nativna komponenta platforme koja omogućuje iscrtavanje web sadržaja, korisnik ima osjećaj da vrši interakciju s nativnom mobilnom aplikacijom. WebView kontejner je implementiran na svim glavnim mobilnim platformama.

Da bi imali pristup nativnim funkcijama mobilnog uređaja kao što su snimanje zvuka ili fotografiranje, Apache Cordova pruža skup JavaScript API-ja koje se mogu koristiti pozivima iz JavaScript koda, kao što je prikazano na slici.

Slika 17: Prikaz rada Apache Cordova API-ja

Apache Cordova podržava sljedeće nativne funkcije mobilnog uređaja:

* *Battery status* – Dohvat podataka o stanju baterije.
* *Camera* – Korištenja zadane aplikacije za slikanje
* *Console* – Korištenje console.log() funkcije za određene platforme.
* *Contacts* – Pristup bazi kontakt liste.
* *Device* – Dohvat informacija o hardveru i softveru uređaja; na primjer dohvat modela uređaja, platforme i verzije, te ime uređaja
* *Device* *Motion* – Prikupljanje podataka o pokretu uređaja.
* *Device* *Orientation* - Dobivanje smjera prema kojem pokazuje uređaj, tzv. kompas
* *Dialogs* - Prikaz nativnih dijaloga korisniku s popratnim zvukom ili vibracijom.
* *File* – Omogućava obradu podataka; čitanje, pisanje i navigaciju kroz *filesystem.*
* *File* *Transfer* – Omogućuje upload i download datoteka.
* *Geolocation* – Dohvat lokacije uređaja pomoću GPS-a.
* *Globalization* – Dohvat korisničkih regionalnih postavki i izvršavanje funkcija.
* *Inappbrowser* – Prikaz korisnih članaka, video zapisa ili web resursa unutar hibridne aplikacije.
* *Media* – Omogućuje snimanje i reprodukciju audio zapisa.
* *Media* *Capture* – Snimanje audio zapisa, slikanje i snimanje videa zapisa pomoću zadane aplikacije mobilne platforme
* *Network* *information* – Dohvat podataka o povezanosti uređaja s mobilnom mrežom ili Wi-Fi-jem.
* *Splashscreen* – Prikaz početnog zaslona prilikom otvaranje aplikacije.
* *Vibration* – Omogućuje vibriranje uređaja.
* *Statusbar* – Omogućuje prilagodbu statusne trake za iOS i Android platforme.
* *Whitelist* – Implementira listu pravila za WebView kontejner.

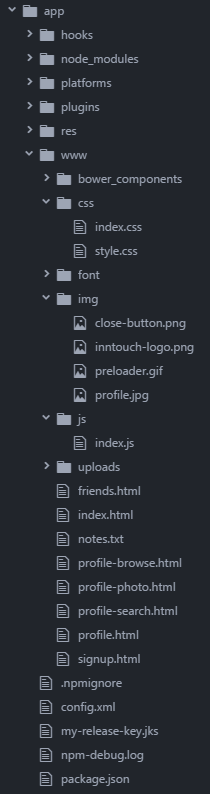
Pozivi prema Apache Cordova API-ju se prevode u API pozive mobilne platforme. U Apache Cordova radnom okviru, nativnim API-jima platforme se pristupa uporabom *plugin*-ova, gdje su implementirani.

Ovakav pristup omogućava korištenje jedinstvenog API sučelja kako bi se izvršila određena nativna funkcija na različitim mobilnim platformama.

5. Ogledna hibridna mobilna aplikacija

Za praktičan primjer ovoga rada razvijena je hibridna mobilna aplikacija pomoću Apache Cordova. Radi se o maloj društvenoj mreži gdje se korisnici mogu registrirati, objavljivati poruke na profilu, pretraživati druge profile te mogućnost praćenja istih.

5.1. Struktura aplikacije

U www mapi nalaze se sve datoteke aplikacijskog front-enda i programske logike.

U Mapi bower\_components nalazi se najnovija verzija jQuery-a kojeg pozivamo u svakom od navedenih HTML datoteka.

Mapa css sadrži stilove; index.css datoteka je predefinirana od strane Apache Cordova, datoteka style.css je glavni dokument sa stilovima kojeg pozivamo u svakom od navedenih HTML datoteka.

Mapa font sadrži font datoteke koje se također pozivaju u svakoj HTML datoteci.

Mapa img sadrži slike koje se koriste u aplikaciji, kao što su gumb za brisanje poruke, logo, *preloader* stranice i zadana profilna slika.

U mapi js nalazi se datoteka index.js koja sadrži svu aplikacijsku logiku, te se poziva u svakom od navedenih HTML dokumenata.

Index.html – početna točka prilikom pokretanja aplikacije; stranica za prijavu korisnika

signup.html – stranica za registraciju novog korisnika

profile.html – pregled vlastitog profila

profile-search.html – stranica za pretragu profila

Slika 18: Struktura aplikacije

profile-browse.html – pregled drugih profila

friends.html – stranica za pregled liste praćenih profila i profila koji prate korisnički profil

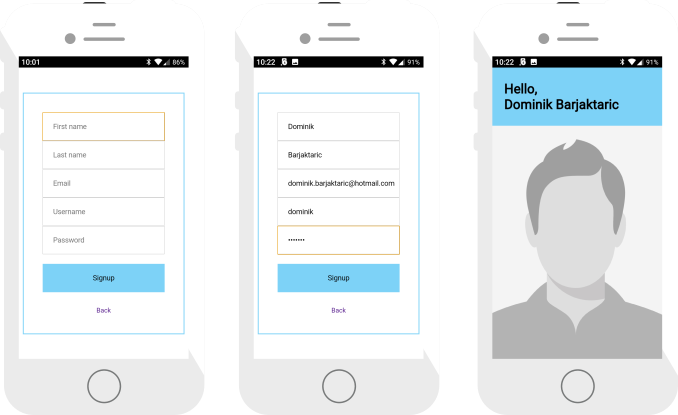
profile-photo.html – stranica za upload profilne slike

config.xml – sadrži općenite podatke o aplikaciji kao što su ime, opis i autor.

5.2. Opis funkcionalnosti aplikacije

5.2.1. Registracija novog korisnika

Prozor registracije novog korisnika sadrži obrazac (lijevi prozor). Kada korisnik ispuni obrazac i pritisne gumb „Signup“ (prozor u sredini), podaci se šalju poslužitelju i ako je registracija bila uspješna, korisnika se preusmjerava i prikazuje stranica profila (desni prozor).

Slika 19: Izgled prozora prilikom procesa registracije korisnika

U signup.html dokumentu, povezuje se eksterni CSS dokument u *head* elementu (8), te JS dokumenti kao što su jQuery (24), cordova.js (25) i index.js (26). U *body* elementu nalazi se kôd obrasca (11).

01: <!DOCTYPE html>

02: <html>

03: <head>

04: <meta charset="utf-8">

05: <meta name="format-detection" content="telephone=no">

06: <meta name="msapplication-tap-highlight" content="no">

07: <meta name="viewport" content="user-scalable=no,

initial-scale=1, maximum-scale=1, minimum-scale=1,

width=device-width">

08: <link rel="stylesheet" type="text/css" href="css/style.css">

09: <title>Signup | SocialNetwork</title>

10: </head>

11: <body class="" data-page="signup">

12: <div class="main-container">

13: <form class="" id="signup-form" action="" method="">

14: <input type="text" name="firstName" value=""

placeholder="First name" required>

15: <input type="text" name="lastName" value=""

placeholder="Last name" required>

16: <input type="email" name="email" value=""

placeholder="Email" required>

17: <input type="text" name="username" value=""

placeholder="Username" required>

18: <input type="password" name="password" value=""

placeholder="Password" required>

19: <input id="signup-btn" type="submit" name="submit"

value="Signup">

20: <br>

21: <a href="index.html">Back</a>

22: </form>

23: </div> <!-- end main container -->

24: <script type="text/javascript"

src="bower\_components/jquery/dist/jquery.js"></script>

25: <script type="text/javascript" src="cordova.js"></script>

26: <script type="text/javascript" src="js/index.js"></script>

27: </body>

28: </html>

U dokumentu index.js napisana je aplikacijska logika. Izdvojit ćemo kôd koji se odnosi na ovaj proces:

097: $("#signup-form").on("submit", function(event) {

098: event.preventDefault();

100: var data = $(this).serialize() + "&ajax=1" + "&action=signup";

101: $.ajax({

102: type: "POST",

103: url: ajax\_PHP\_API,

104: data: data,

105: dataType: "json",

106: cache: true,

107: })

108: .done(function(data) {

109: if (data['status'] == "success") {

110: set\_credentials(data);

111: } else alert("Failed to sign up. Please try again.");

112: })

113: .fail(function() {

114: alert("Could not connect to server. Please try again

later");

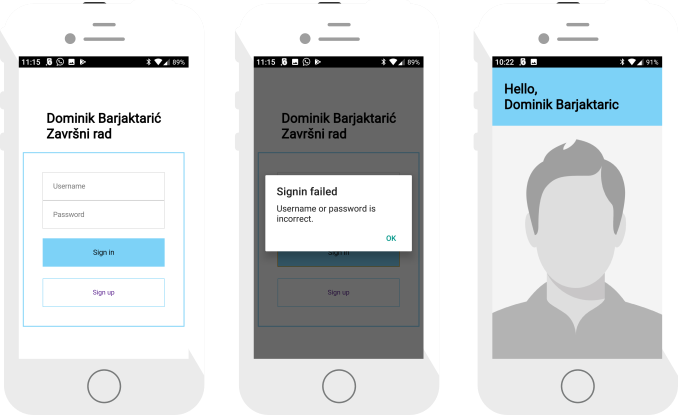
115: });

116: });

Kada korisnik pritisne gumb „Signup” pokreće se event (97). Prikupljaju se svi podaci forme i spremaju u varijablu data (100). Zatim se radi AJAX poziv poslužitelju (101-107) te se očekuje odgovor u JSON obliku (105). Ako je AJAX poziv bio uspješan (108-112), provjerava se vrijednost ključa „status”. Ako je vrijednost „success”, poziva se funkcija „set\_credentials” te prosljeđuje JSON. U slučaju neke druge vrijednosti, prikazat će se obavijest o neuspješnoj registraciji (111). Ako AJAX poziv nije uspio, također će se pojaviti obavijest (113-115).

5.2.2. Prijava korisnika

Prilikom otvaranja aplikacije učitava se i prikazuje dokument index.html. Na slici je prikazan izgled procesa prijave korisnika. Lijevi prozor prikazuje obrazac za prijavu. Ako je prijava bila neuspješna, prikazuje se obavijest korisniku (prozor u sredini). Ako je prijava bila uspješna, korisnika se preusmjerava i prikazuje stranica njegovog profila.

Slika 20: Izgled prozora za prijavu korisnika  


Isjećak HTML koda iz dokumenta index.html u kojemu je definiran obrazac za prijavu:

14: <form class="" id="login-form" action="" method="">

15: <input type="text" name="username" value=""

placeholder="Username" id="username" required>

16: <input type="password" name="password" value=""

placeholder="Password" id="password" required>

17: <input type="submit" name="submit" value="Sign in" id="signin-

btn">

18: <a href="signup.html" class="signup-btn">Sign up</a></span>

19: </form>

Isjećak koda iz dokumenta index.js odgovoran za aplikacijsku logiku prilikom procesa prijave:

62: $("#login-form").on("submit", function(event) {

63: event.preventDefault();

64:

65: var data = $(this).serialize() +

66: "&ajax=1" +

67: "&action=signin";

68:

69: $.ajax({

70: type: "POST",

71: url: ajax\_PHP\_API,

72: data: data,

73: dataType: "json",

74: cache: false,

75: })

76: .done(function(data) {

77: if (data['status'] == "success") {

78: set\_credentials(data);

79: }

80: else if (data['status'] == "fail") {

81: navigator.notification.alert(

82: 'Username or password is incorrect.', // message

83: function(){}, // callback

84: 'Signin failed', // title

85: 'OK' // buttonName

86: );

87: }

88: })

89: .fail(function() {

90: alert("Could not connect to server. Please try again later");

91: });

92: });

Kada korisnik pritisne gumb „Sign in”, pokreće se event (62) u kojoj se priprema varijabla „data” (65-67) koja sadrži podatke iz obrasca za AJAX poziv (69-75). Odgovor se očekuje u JSON formatu (73). Ako je AJAX poziv bio uspješan (76-88), te ako je vrijednost ključa „status” jednaka „success” (77), tada se poziva funkcija „set\_credentials” i prosljeđuje dobiveni JSON (78). Ako je vrijednost ključa „status“ jednaka „fail“ poziva se funkcija „navigator.notification.alert” (81-86) koja služi za prikaz obavijesti da je korisničko ime ili lozinka netočna. Ta funkcija je implementirana u Apache Cordova API-ju, te prikazuje obavijesti koje imaju nativni izgled mobilne platforme.

44: function set\_credentials(data) {

45: myStorage.setItem('id', data['id']);

46: myStorage.setItem('username', data['username']);

47: myStorage.setItem('firstName', data['firstName']);

48: myStorage.setItem('lastName', data['lastName']);

49: myStorage.setItem('email', data['email']);

50: myStorage.setItem('signedin', 1);

52: if (data['profilePicture'] == null) {

myStorage.setItem('profilePicture',

"img/profile.jpg");}

53: else myStorage.setItem('profilePicture', domainBase +

data['profilePicture']);

54:

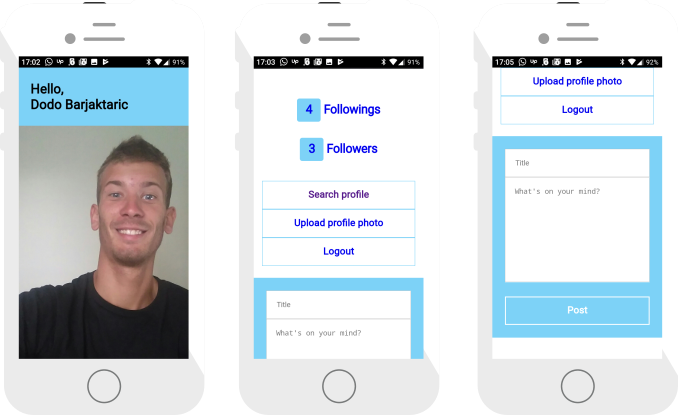
55: window.location.replace("profile.html");

56: }

Funkcija „set\_credentials” sprema vrijednosti iz proslijeđenog JSON-a u localStorage (45-50). Provjerava se da li profil ima uploadanu sliku (52). Ako nema dodjeljuje se zadana profilna slika (52). Ako ima tada se dodjeljuje URI adresa slike koja se nalazi na poslužitelju (53). Zatim se korisnika preusmjerava na stranicu profila (55).

5.2.3. Pregled profila

Korisnik na svom profilu vidi sljedeće:

* Ime, prezime i profilnu sliku (lijevi prozor).
* Broj profila koje prati, broj profila koje prate korisnika, gumb za pretragu profila, gumb za ažuriranje profilne slike i gumb za odjavu (prozor u sredini).
* Obrazac za objavu poruke i lista objavljenih poruka (desni prozor).

Slika 21: Izgled prozora za pregled profila

Isjećak HTML kôda iz dokument profile.html odgovoran za prikaz profila korisnika:

16: <div class="main-container-outer">

17: <div class="main-container-inner">

18: <div class="profile-info">

19: <h2 data-name="" class="profile-name">Hello,<br></h2>

20: <div class="profile-photo-wrapper" data-profilePicture="">

21: </div>

22: <br>

23: <a href="#" data-followings-num="" class="profile-

followings"></a>

24: <a href="#" data-followers-num="" class="profile-

followers"></a>

25: <a href="profile-search.html" class="profile-search-btn">

26: <h2>Search profile</h2></a>

28: <a href="profile-photo.html" class="profile-settings-btn">

29: <h2>Upload profile photo</h2></a>

31: <a href="#" class="logout-btn"><h2>Logout</h2></a>

34: </div>

35: <div class="profile-posts">

36: <div class="form-wrapper post">

37: <form class="" id="post-form" action="" method="">

38: <input type="text" name="title" placeholder="Title">

39: <textarea name="postText" rows="10" cols="80"

placeholder="What's on your mind?"></textarea>

40: <a href="#" class="post-submit-btn"><h2>Post</h2></a>

41: </form>

42: </div>

43: <div class="posts-wrapper">

44: </div>

46: </div>

47: </div>

48: </div>

Sadržaj koji se dinamički ubacuje u HTML kôd prilikom učitavanja je: Ime i prezime (19), profilna slika (20), broj profila koje korisnik prati (23), broj profila koji prate korisnika (24) i lista objavljenih poruka (43).

129: function bind\_profile\_data(page) {

130: var firstName = myStorage.getItem('firstName');

131: var lastName = myStorage.getItem('lastName');

132:

133: if (myStorage.getItem('profilePicture') == "reload")

reload\_profile\_photo();

134:

135: $('[data-name]').append(firstName + ' ' + lastName);

136: $('[data-profilePicture]').css('background-image', 'url(' +

myStorage.getItem('profilePicture') + ')');

137:

138: load\_followings\_num(page);

139: load\_followers\_num(page);

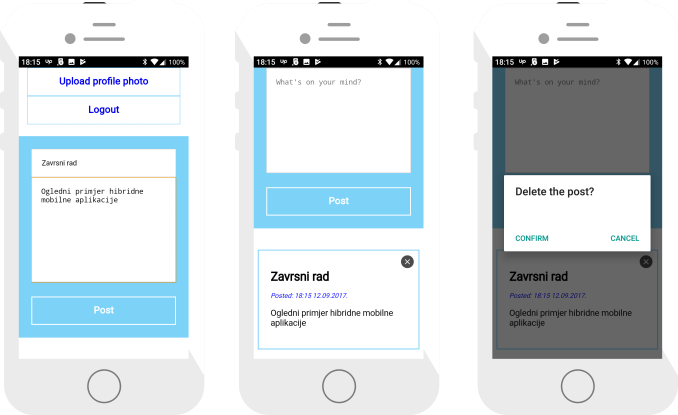
140: load\_posts();

141: }

Prilikom učitavanja stranice profila, pokreće se funkcija „bind\_profile\_data” (129). Ime i prezime profila (135), i profilna slika (136) dinamički se ubacuju u HTML. Ako je vrijednost ključa „profilePicture” jednaka „reload”, poziva se funkcija „reload\_profile\_photo” (133). Ovaj poziv funkciji je kada korisnik uploada novu profilnu sliku, te se deklarira nova vrijednost varijable profilne slike. Pozivaju se još funkcije „load\_followings\_num” (138) i „load\_followers\_num” (139) koje AJAX pozivom poslužitelju dohvaćaju podatke o broju profila. Na kraju se poziva funkcija „load\_posts” (13) koja također AJAX pozivom dohvaća sve objavljene poruke profile (140) koji se također dinamički ubacuju u HTML kôd.

5.2.4. Objavljivanje i brisanje poruka

Korisnik na svom profilu ima obrazac koji služi za objavu nove poruke. U obrascu se unosi naslov poruke i sadržaj (lijevi prozor). Kada korisnik pritisne gumb „Post“, poruka se sprema te se prikazuje na profilu, ispod obrasca (prozor u sredini). Korisnik ima mogućnost brisanja poruke pritiskom gumba „X“ koji se nalazi gornjem desnom kutu svake objavljene poruke. Prikazuje se obavijest o brisanju poruke koju korisnik mora potvrditi (desni prozor).

Slika 22: Izgled prozora za objavljivanje i brisanje poruka

Dvije su funkcije odgovorne za ovaj proces, „post\_form\_listener“ za objavu i „delet\_post\_listener“ za brisanje poruke. Isječak aplikacijske logike iz dokumenta index.js:

242: function post\_form\_listener() {

243: $(".post-submit-btn").on("click", function(event) {

244: event.preventDefault();

245:

246: var userid = myStorage.getItem('id');

247: var data = $('#post-form').serialize() +

248: "&id=" +

249: userid +

250: "&ajax=1" +

251: "&action=new-post";

252:

253: $.ajax({

254: type: "POST",

255: url: ajax\_PHP\_API,

256: data: data,

257: cache: false,

258: })

259: .done(function(data) {

260: $('#post-form input, #post-form textarea').val('');

261: load\_posts();

262: })

263: .fail(function() {

264: alert("Could not connect to server. Please try again later");

265: });

266: });

267: }

Kada korisnik pritisne gumb „Post“ pokreće se event (243). Varijabli „data“ deklariraju se sve vrijednosti iz obrasca, a koja će se koristiti za AJAX poziv poslužitelju (253-258). Ako je poziv bio uspješan (259), polja obrasca se resetiraju (260) i poziva se funkcija (261) koja učitava sve objavljene poruke i prikazuje ih na profilu.

270: function delete\_post\_listener() {

271: $(".posts-wrapper").on("click", ".delete-post-button",

function(event) {

272: event.preventDefault();

273: var postid = $(this).data('post-id');

274:

275: navigator.notification.confirm(

276: '',

277: onConfirm,

278: 'Delete the post?',

279: ['Cancel','Confirm']

280: );

281:

282: function onConfirm(buttonIndex) {

283: if (buttonIndex == 1) {} // do nothing when cancelled

284: else if (buttonIndex == 2) {

285: var data = "id=" +

286: postid +

287: "&ajax=1" +

288: "&action=delete-post";

289:

290: $.ajax({

291: type: "POST",

292: url: ajax\_PHP\_API,

293: data: data,

294: cache: false,

295: })

296: .done(function(data) {

297: load\_posts();

298: })

299: .fail(function() {

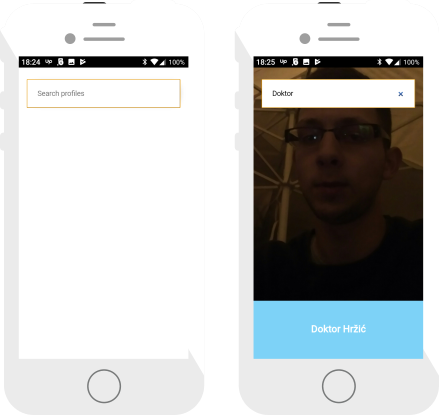
300: alert("Could not connect to server. Please try again

later");

301: });}}});}

Prilikom pritiska gumba za brisanje pokreće se event (271). Prvo se poziva Cordova API funkcija „navigator.notification.confirm“ (275-280) koja prikazuje obavijest u nativnom izgledu mobilne platforme. Ako je korisnik pritisnuo gumb za potvrdu (284), pokreće se AJAX poziv (290). Ako je poziv vio uspješan poziva se funkcija (297) koja učitava sve objavljene poruke i prikazuje ih na profilu.

5.2.5. Pretraga profila

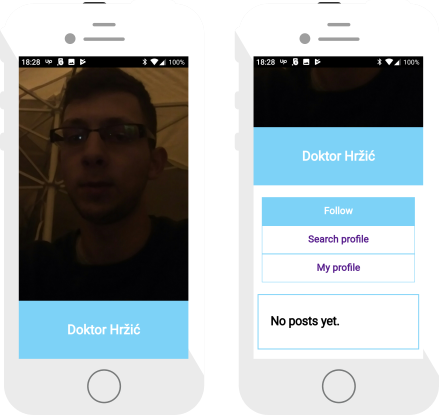
****Korisnik ima mogućnost pretrage profila drugih korisnika. Pritiskom na gumb „Search profile“ otvara se stranica profile-search.html.

Slika 23: Izgled prozora za pretragu profila

Na stranici je prikazan obrazac za unos imena i/ili prezimena profila kojeg korisnik želi pronaći (lijevi prozor). Kada je korisnik potvrdio pretragu, prikazuju se traženi profili (desni prozor). Korisnik ima mogućnost pregleda traženog profila pritiskom na sliku ili imena.

5.2.6. Pregled traženog profila

Stranica profile-browse.html učitava se kada korisnik odabere profil prilikom pretrage. Izgled traženog profila sličan je izgledu pregleda vlastitog profila. Korisnik može vidjeti profilnu sliku i ime traženog profile (lijevi prozor).

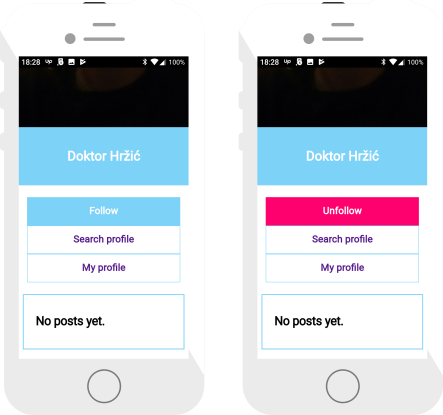
****

Slika 24: Izgled prozora za pregled traženog profila

Nadalje, prikazan je gumb „Follow“ koji služi za praćenje traženog profila. Gumb „My profile“ preusmjeruje korisnika na vlastiti profil. Ispod skupina gumba prikazana je lista objavljenih poruka (desni prozor).

5.2.7. Praćenje traženog profila

Na traženom profilu prikazan je gumb „Follow“ ili „Unfollow“. Ako korisnik ne prati traženi profil, prikazat će se gumb „Follow“ (lijevi prozor).

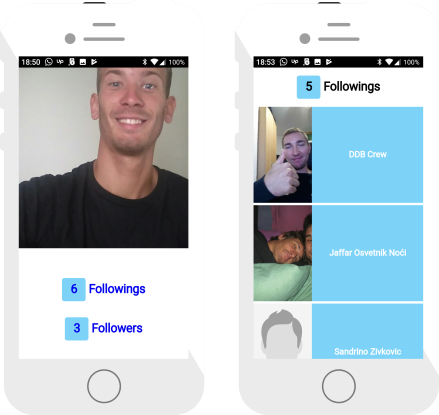
****

Slika 24: Izgled prozora za praćenje profila drugih korisnika

Pritiskom na taj gumb, traženi profil ubacit će se u listu profila koje korisnik prati, te će se umjesto gumba „follow“ prikazati gumb „unfollow“ (desni prozor). Ako korisnik pritisne na taj gumb, prestat će pratiti traženi profil te će se ponovno prikazati gumb „follow“ (lijevi prozor).

5.2.8. Prikaz liste praćenih profila

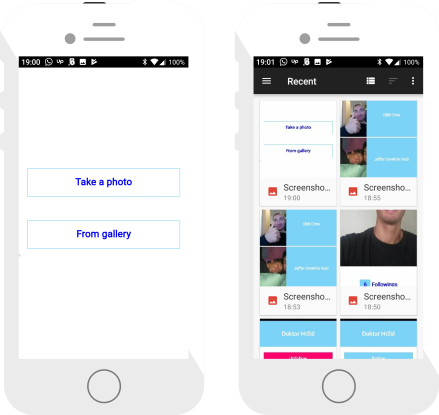
Korisniku se na svom profilu prikazuje broj profile koje prati i broj profila koje prate korisnika (lijevi prozor). Pritiskom na jedan od tih brojeva, preusmjerava se korisnika na stranicu friends.html.

Slika 25: Izgled prozora za prikaz liste praćenih profila

Na stranici friends.html prikazuje se ukupan broj praćenih profila i lista profila s profilnom slikom i imenom (desni prozor). Korisnik može pregledati pojedini profil pritiskom na sliku ili ime.

5.2.9. Ažuriranje profilne slike pomoću kamere ili odabirom iz galerije slika

Korisnik na svom profilu ima gumb „Upload profile photo“ koji prilikom pritiska preusmjeruje korisnika na stranicu profile-photo.html. Na toj stranici nalaze se dva gumba „Take a photo“ i „From Gallery“ (lijevi prozor).

Slika 26: Izgled prozora za ažuriranje profilne slike

Ako korisnik pritisne gumb „Take a photo“, pokrenuti će se aplikacija kamere mobilnog uređaja. Ako korisnik pritisne gumb „From Gallery“, otvara se galerija. Kada korisnik odabere sliku ili fotografira novu, profilna slika će se ažurirati i biti će preusmjeren na stranicu vlastitog profila.

Isječak aplikacijske logike za ovaj proces iz dokumenta index.js:

535: function profile\_photo\_listener() {

536: $('#open-camera-btn, #open-gallery-btn').on("click", function(event{

537: event.preventDefault();

538: var sourceType;

539:

540: if ($(this).attr('id') == "open-camera-btn") sourceType = 1;

541: else if ($(this).attr('id') == "open-gallery-btn") sourceType = 0;

542:

543: var options = {

544: quality: 100,

545: sourceType: sourceType, //0 album, 1 camera

546: mediaType: 0, //0 pictures only

547: destinationType: 1, //1 FILE\_URI

548: correctOrientation: true,

549: saveToPhotoAlbum: true

550: };

551:

552: navigator.camera.getPicture(onSuccess, onFail, options);

553:

554: function onSuccess(imageURI) {

555: var win = function (r) {

556: // alert("Profile photo updated successfully");

557: myStorage.setItem('profilePicture', "reload");

558: window.location.replace('profile.html');

559: }

560:

561: var fail = function (error) {

562: alert("An error has occurred: Code = " + error.code);

563: }

564:

565: var ft = new FileTransfer();

566: var serverURL = ajax\_PHP\_API +

567: "?action=upload-profile-photo&ajax=1&id=" +

568: myStorage.getItem('id');

569:

570: ft.upload(imageURI,

571: encodeURI(serverURL),

572: win,

573: fail

574: );

575: }

576:

577: function onFail(message) {

578: if (message != "Selection cancelled.") {

579: alert('Failed because: ' + message);

580: }

581: }

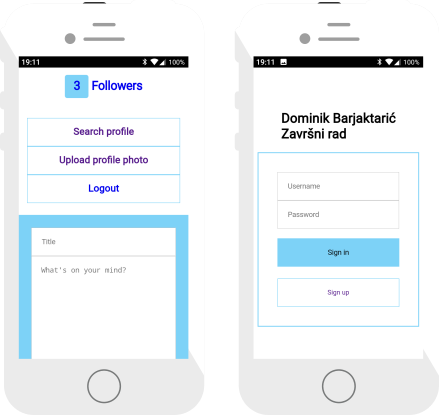
582: });

583: }

Prilikom pritiska na gumb kamere ili galerije pokreće se event (536). Provjerava se koji gumb je pritisnut (540-541) te se dodjeljuje određena vrijednost varijabli „sourceType“. Objekt „options“ (543-550) sadrži postavke kao što su kvaliteta slike, izvor (kamera ili galerija) i orijentacija slike. Poziva se funkcija „navigator.camera.getPicture“ koja je implementirana u Cordova API-ju (552). Prosljeđuju se tri parametra, prva je funkcija koja se poziva kada je slika uspješno odabrana, druga je također funkcija koja se poziva ako slika nije uspješno odabrana i treći parametar je objekt „options“ kojega smo prethodno deklarirali. Ako je slika uspješno odabrana, stvara se novi objekt klase „FileTransfer“ (565) čija se implementacija također nalazi u Cordova API-ju. Služi za prijenos datoteke na poslužitelj pozivom funkcije „upload“ (570).

5.2.10. Odjava korisnika

Na profilu korisnika nalazi se gumb „Logout“ (lijevi prozor). Kada korisnik pritisne gumb, aplikacija ga odjavljuje i preusmjerava ga se na stranicu index.html (desni prozor). Korisnik se tada može ponovno prijaviti ili registrirati novi profil.

****

Slika 27: Izgled prozora prilikom odjave korisnike

Isjećak aplikacijske logike iz dokumenta index.js:

120: function signout\_listener() {

121: $('.logout-btn').on("click", function(event) {

122: event.preventDefault();

123: myStorage.clear();

124: window.location.replace("index.html");

125: });

126: }

Pritiskom na gump „Logout”, pokreće se event (121) gdje se poziva funkcija koja čisti objekt „localStorage“ od deklariranih vrijednosti (123),, te se korisnika preusmjerava na stranicu za prijavu (124).

6. Zaključak

U ovom radu prikazano je na koji način se mobilne aplikacije mogu razvijati i pojasnila se je tehnologija koja koristi samo jedan programski jezik za razvoj mobilne aplikacije za više platformi. Ovakav pristup razvoja omogućio je projekt Apache Cordova, koji je otvorenog koda. Cordova obuhvaća skup API-ja koje se implementiraju JavaScript programskim jezikom u aplikacijskoj logici, kako bi se pozivale nativne funkcije mobilnog uređaja kao što su kamera i GPS lokacija. Omogućen je brži razvoj aplikacije i smanjenje troškova projekta. Nije više potrebno reimplementirati istu aplikacijsku logiku kako bi aplikacija bila razvijena za više platformi. Performanse ovakve aplikacije blizu su ili jednake performansama nativne mobilne aplikacije. Objašnjene su vrste mobilnih aplikacija i po čemu se razlikuju, te smo definirali njihove prednosti i nedostatke. Naučili smo se tehnologiju za razvoj hibridne mobilne aplikacije, te se detaljno upoznali s API-jem Cordova radnog okvira.

7. Popis literature

MDN Web Docs – HTML Basics - https://mzl.la/2dqvAVa

MDN Web Docs – CSS Basics - https://mzl.la/2dc1T8h

W3schools – CSS - https://www.w3schools.com/css/

MDN Web Docs – JavaScript - https://mzl.la/2bKKtll

W3schools – Javascript - https://www.w3schools.com/Js/

jQuery official site - https://jquery.com/

jQuery API documentation - https://api.jquery.com/

PHP official site – http://php.net/

PHP manual - http://php.net/manual/en/index.php

W3schools – PHP - https://www.w3schools.com/php/

NuSphere – PHP superglobals - http://bit.ly/2xJukZ9

UpWork – Server-side scripting - http://bit.ly/1q2804m

Tutorialspoint – MySQL - http://bit.ly/2wmuogV

W3schools – PHP MySQL database - http://bit.ly/2uN94f0

Techtarget – MySQL - http://bit.ly/2xr23pf

H. Saleh, E. Holmes, T. Bray, S. Yusuf – Mobile application Development: Javascript Frameworks

8. Popis sliKa

Slika 1: Anatomija HTML dokumenta 5

Slika 2: Primjer neuređene liste 6

Slika 3: HTML oznake kod 7

Slika 4:Izlaz HTML koda 7

Slika 5: CSS kôd u redu. 8

Slika 6: Interni CSS kôd 8

Slika 7: Prikaz kontejner modela 11

Slika 8: Primjer CSS kôda 11

Slika 9: PHP unutar HTML-a 14

Slika 10: Primjer POST metode PHP skripte 15

Slika 11: Primjer izrade objekta u PHP skripti za povezivanje sa MySQL bazom 16

Slika 12: Relacija Users 17

Slika 13: Ubacivanje n-torke kada korisnik pritisne na gumb Follow u oglednoj aplikaciji 17

Slika 14: Primjer n-torke prilikom praćenja drugog profila 17

Slika 15: Razlika mobilne web, hibridne i nativne aplikacije 18

Slika 16: Apache Cordova arhitektura 23

Slika 17: Prikaz rada Apache Cordova API-ja 24

Slika 18: Struktura aplikacije 26

Slika 19: Izgled prozora prilikom procesa registrancije korisnika 27

Slika 20: Izgled prozora za prijavu korisnika 29

Slika 21: Izgled prozora za pregled profila 31

Slika 22: Izgled prozora za objavljivanje i brisanje poruka 33

Slika 23: Izgled prozora za pretragu profila 35

Slika 24: Izgled prozora za pregled traženog profila 36

Slika 24: Izgled prozora za praćenje profila drugih korisnika 37

Slika 25: Izgled prozora za prikaz liste praćenih profila 38

Slika 26: Izgled prozora za ažuriranje profilne slike 39

Slika 27: Izgled prozora prilikom odjave korisnike 41

9. Popis tablicA

Tablica 1: Vrste CSS selektora. 10

Tablica 2: Ključne razlile između mobilnih aplikacija. 20

1. Popis svih svojstava može se pronaći na: https://www.w3schools.com/cssref/ [↑](#footnote-ref-1)
2. Detaljnija lista selektora može se naći na adresi: https://mzl.la/2xr20dc [↑](#footnote-ref-2)
3. PHP je rekurzivni akronim, prva inicijala „P“ znači „PHP“, dakle PHP = „PHP Hypertext Preprocessor“ [↑](#footnote-ref-3)
4. Službena dokumentacija nalazi se na adresi http://docs.cordova.io/ [↑](#footnote-ref-4)