Izrada mobilnih aplikacija korištenjem izomorfnog meteor.js radnog okvira

Prpić, Denis

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

IZRADA MOBILNIH APLIKACIJA KORIŠTENJEM IZOMORFNOG METEOR.JS RADNOG OKVIRA

Završni rad

Denis Prpić

Osijek, 2016.

SADRŽAJ

1. UVOD

Meteor ili Meteor.js je besplatni *open-source* JavaScript radni okvir napisan koristeći Node.js. Prvi put je predstavljen u prosincu 2011. godine pod imenom Skybreak razvijen od strane *Meteor Development Group* [1]. Node.js je *open-source cross-platform* (web, Android, iOS) JavaScript radno okruženje za razvoj raznolikih alata i aplikacija. Iako Node.js nije samo JavaScript okvir, većina njegovih modula je napisana u JavaScriptu, te se mogu razvijati novi. Meteor.js omogućuje brzu izradu i pruža *cross-platform* kod. Integrira s MongoDB bazom podataka, koristi DDP (*Distributed Data Protocol*) i *publish–subscribe* obrazac za automatsko propagiranje podataka za klijente bez potrebe da osoba koja razvija aplikaciju piše bilo kakav sinkronizacijski kod. Na klijentu, Meteor.js ovisi o jQuery-u, a može se koristiti s bilo kojom JavaScript UI *widget* bibliotekom. Osim koda jednostavnog za čitanje, Meteor.js pruža automatsku provjeru pogrešaka. Meteor.js koristi pametne pakete, odnosno male pakete koda koji se mogu izvoditi u pregledniku ili kao dio usluga u oblaku koji zajedno s ostalim bibliotekama olakšavaju razvoj aplikacije.

1.1. Zadatak rada

Zadatak rada je na primjeru opisati postupak koji će služiti za pomoć pri izradi mobilnih aplikacija korištenjem izomorfnog Meteor.js radnog okvira. Primjer sadržava sve osnovne i potrebne dijelove da bi aplikacija bila potpuna i funkcionalna. Uz sitne dorade se može prenamijeniti i za ostale svrhe, ali trenutno će služiti samo za javno dopisivanje. *Chat* je oblik komunikacije dvaju ili više korisnika putem računala i računalne mreže u realnom vremenu. Radi se o vrlo kratkim porukama koje korisnik vidi čim ih njegov sugovornik pošalje (obično pritiskujući tipku Enter). U nekim *chatovima* postoje tzv. sobe u kojima istovremeno priča i do nekoliko desetaka (pa i stotina) korisnika, u nekima je razgovor ograničen na nekoliko sudionika, dok neki mogu kombinirati te dvije vrste [2]. Za neke *chatove* ne treba ništa osim Web preglednika i Java Programskog jezika, dok su neki prerasli u zasebne računalne programe, od kojih su najpoznatiji:

- Windows Live Messenger (bivši MSN Messenger),
- Skype,
- Yahoo! Messenger,
- ICQ,
- Google Talk itd.

U *chatovskim* aplikacijama uobičajio se poseban jezik, zvan *leet*, još pisano 1337, u kojemu se zamjenjuju normalna slova brojevima i znakovima s tipkovnice, komuniciranja kraticama radi postizanja što veće brzine izmjene poruka

Tablica 1.0. Često korištene kratice i njihova značenja

BRB - Be Right Back	Vraćam se odmah
AFK - Away From Keyboard	Udaljen od tipkovnice (računala)
TY/THX - Thank You/Thanks	Hvala
BTW - By The Way	Usput rečeno
LOL - Laughing Out Loud	Glasno smijanje
ROFL - Rolling on floor laughing	Valjati se po podu od smijeha

Uz kratice se koriste i različiti smajlići (emotikoni). Oni su prvobitno bili stilizirani tipografski znaci koji se koriste kako bi prikazali emociju ili trenutno osjećanje. Porastom brzina komunikacija, umjesto tipografskih znakova koriste se male sličice. Nerado se gleda na pisanje velikim slovima (osim kratica), jer se to smatra vikanjem.

Tablica 1.1. Najčešće korišteni emotikoni

:-) ili **:)** ili **:o)**	Nasmijan
:-(ili **:(**	Tužan
;-) ili **;)**	Namigivanje

Potrebno je bilo napraviti mobilnu aplikaciju koja je povezana s bazom podataka kako bi bilo omogućeno pohranjivanje poruka i korisnika. Za korištenje aplikacije je potrebna registracija,

odnosno prijava kako bi dopisivanje s ostalim korisnicima bilo moguće. Bez prijave korisnik ima samo ograničen pogled. Za registraciju su potrebna polja:

- Korisničko ime
- Zaporka
- Potvrditi zaporku
- E-mail adresa

Nakon registracije, osobe se mogu prijaviti u aplikaciju. Osim registracije prijava je moguća i putem Facebook, Twitter, Google i ostalih servisa samo jednim klikom miša. Svaki korisnik se može dopisivati sa svim ostalim prijavljenim korisnicima u nekoliko soba koje u bilo kojem trenutku može promijeniti.

2. IZRADA MOBILNE APLIKACIJE I ODABIR ALATA

Mobilne aplikacije su aplikacijske programske podrške za pametne telefone, *tablet* računala i druge mobilne uređaje. Prvobitno su služile za brzu provjeru elektroničke pošte, ali je njihova velika potražnja dovela do proširenja i na druga područjima kao što su na primjer navigacijski uređaji, igrice za mobitele, gledanje video sadržaja ili pretraživanje interneta. Mobilne aplikacije mogu rabiti korisnici tzv. pametnih mobitela kao primjerice iPhone, BlackBerry uređaji i drugi android mobiteli. Mobilne aplikacije jedan su od najnovijih i ujedno najefikasnijih kanala komunikacije sa tržištem. U odnosu na klasične komunikacijske kanale, osim što su korisniku uvijek dostupni gdje god se nalazili, nude i najveću razinu interaktivnosti. Velikom broju korisnika mobilni uređaji predstavljaju izvor informacija, komunikacije i zabave, a oglašivačima novi, atraktivan i direktan komunikacijski kanal.

2.1. Korišteni alati i programski jezici

U ovom radu odabran je i korišten uređivač teksta zvan Atom, koristi se za pisanje programskog koda u većini programskih jezika. Datoteka se spremi u odabrani format koji je u skladu s korištenim programskim jezikom. Atom onda prepoznaje format te olakšava pisanje i organiziranje koda, po pravilima koja pripadaju korištenom programskom jeziku. Korišteni su HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*), JavaScript, Jquery i kombinirani su na razne načine. Uz to, dodano je još nekoliko funkcionalnosti. Također za izradu ovog rada potreban je instaliran Meteor.js koji se nakon preuzimanja instalira ili se instalira putem naredbenog retka ovisno o platformi na kojoj se radi. Naredbeni redak je potreban i za ostale funkcionalnosti poput kreiranja Meteor.js projekta, pokretanja projekta, dodavanje paketa i sl.

2.1.1. HTML

HTML opisuje strukturu web-stranice ili aplikacije, što ga čini prezentacijskim jezikom, a ne programskim jezikom. Vrlo je jednostavan i jako se brzo uči, zbog čega je i postao vrlo popularan [3]. Valja istaknuti i to da je besplatan. HTML su obične tekstualne datoteke s ekstenzijama .html ili .htm, a osnovni građevni element stranica su znakovi (tags) poput <html>. Preglednici ne prikazuju znakove i skripte već ih koriste kao opisnike kako će se nešto prikazati u pregledniku. HTML može ubacivati skripte poput onih napisanih u JavaScriptu koje utječu na ponašanje HTML web-stranica, ali se isto tako može koristiti CSS kako bi bio definiran izgled teksta i ostalih materijala [4]. HTML5 je službeno završen i objavljen 28. listopada. 2014.

godine. HTML5 ima puno zanimljivih mogućnosti, poput reproduciranja videa na stranicama bez korištenja *Adobe Flash Playera* ili *Microsoft Silverlighta*, upravljanje pomoću tipkovnice i mnogi drugi elementi. Korišten je HTML5 u izradi koji služi za normalan prikaz stranice, odnosno okvira i za prikaz svih njezinih elemenata. Bez HTML-a ne bi bio moguć prikaz stranice, pa je zato on njezin ključan dio. Osim što služi za prikaz stranice i njezinih elemenata, služi i za dodavanje eksternih datoteka poput stilskih datoteka, a koje omogućuju mijenjanje izgleda stranice po vlastitim željama. Moguće je dodati koliko god treba stilskih datoteka, da okvir bude pravilno oblikovan i prikazan. Isto tako HTML služi i za povezivanje skripti koje dodaju razne mogućnosti i obavljaju različite funkcije na okviru. Skripte se pišu pretežito u JavaScriptu i onda se pozivaju u HTML-u kako bi se izvodile pri nekom specifičnom događaju i sl.

Sl. 2.1. Službeni logo HTML5 standarda

Na slici 2.1. je prikazan logo trenutne HTML inačice, koja se zove HTML5.

2.1.2. CSS

CSS je stilski jezik koji služi za opis izgleda i formata dokumenta napisanog pomoću HTML jezika. Zajedno s HTML-om i JavaScript-om, CSS je temeljna tehnologija korištena na većini web-stranica i aplikacija kako bi se kreirale da budu vizualno privlačne. CSS je prvenstveno dizajniran kako bi se omogućilo odvajanje sadržaja dokumenta od prezentacije dokumenta, uključujući elemente poput rasporeda, boja i fontova [4]. Ova razdioba može poboljšati dostupnost sadržaja, omogućiti veću fleksibilnost i kontrolu u specifikacijama prezentacijskih karakteristika, omogućiti da više HTML stranica dijeli formatiranje i izgled stranice povezivanjem na određeni CSS u odvojenoj .css datoteci te smanjiti kompleksnost i ponavljanje u strukturi sadržaja. CSS omogućuje odvajanje prezentacijske instrukcije u zasebnu datoteku ili u stilski odjeljak HTML datoteke. Svakom odgovarajućem HTML elementu CSS dodjeljuje listu

instrukcija za oblikovanje [5]. Na primjer može se odrediti da svi odlomci <p> budu podebljani, pri čemu HTML samo traži gdje se nalaze svi <p> i oblikovanje odrađuje CSS. CSS ima jednostavnu sintaksu koja koristi veliki broj engleskih riječi za određivanje imena raznih svojstava stilova. Datoteka se sastoji od liste pravila, a svako pravilo se sastoji od jednog ili više selektora i deklaracijskog bloka u koji se unose svojstva poput boje, fonta i ostalog. Selektori se koriste za deklariranje toga na koji dio oznake se određeni stil odnosi, a to može biti na sve elemente određenog tipa poput <h2>. Dakle, zaglavlja veličine dva će imati određena svojstva. Selektori mogu biti korišteni i za atribute poput jedinstvenog identifikatora ili klase koja grupira više elemenata u dokumentu. Trenutna inačica je CSS3 koja je potpuno kompatibilna sa starijim inačicama, a u njoj se pojavljuju novi elementi poput 2D i 3D transformacija, animacija i mnogih drugih zanimljivih opcija [5]. CSS je korišten u aplikaciji za izgled okvira, a korišteno je dosta stilskih datoteka, pri čemu je svaka navlastito oblikovana.

Sl. 2.2. Logo novog CSS3

Na slici 2.2. je prikazan logo najnovije inačice CSS-a, koja se zove CSS3.

2.1.3. JavaScript

JavaScript je dinamični, tipizirani i interpretacijski programski jezik. Standardiziran je u specifikacijama ECMAScript jezika. Uz HTML i CSS jedna je od 3 bitne tehnologije za kreiranje sadržaja WWW-a (*World Wide Web*). Koristi ga većina web-stranica i aplikacija, te je podržan na svim modernim web-preglednicima bez dodataka. JavaScript podržava objektno orijentirane, proceduralne i funkcionalne stilove programiranja. JavaScript je tumač, što znači da ne treba prevoditi cijeli program i kreirati izvršnu datoteku, već se skripta odma izvodi naredbu po naredbu [6]. Korišten je JavaScript u kombinaciji s Jquery-iem za dinamički izgled aplikacije i za dinamičke funkcije. JavaScript je temelj za izradu mobilne aplikacije i svih funkcionalnosti u

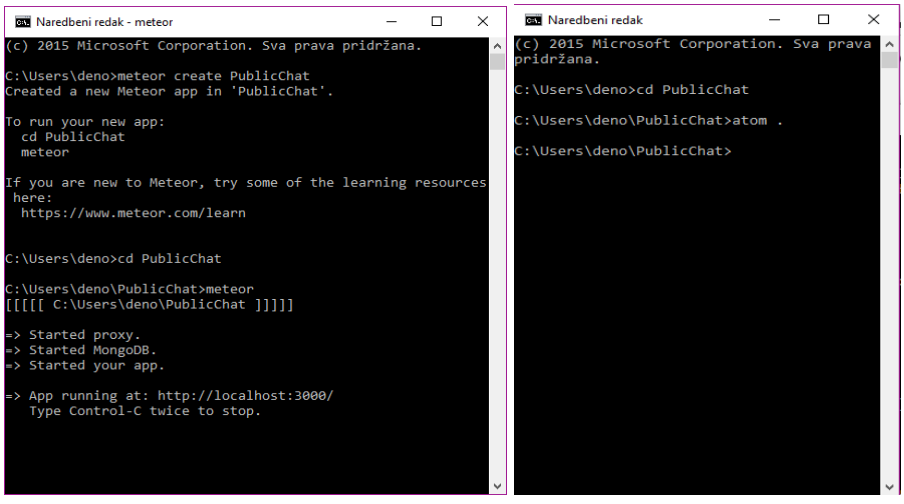
samoj aplikaciji. Funkcionira u kombinaciji sa ostalim korištenim datotekama, jer su međusobno povezani. Za izradu kompletne aplikacije dovoljno je koristit samo jedan jezik – JavaScript.

Sl. 2.3. Logo JavaScripta

Na slici 2.3. je prikazan logo najnovije inačice JavaScript.

2.2. Meteor.js

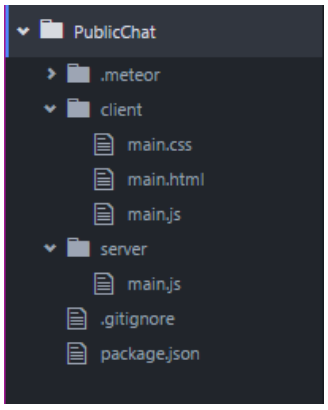
Prvi korak izrade mobilne aplikacije korištenjem Meteor.js radnog okvira je instalacija istoga. Instalacija se vrši kroz naredbeni redak ili preuzimanjem i pokretanjem instalacije ovisno o platformi na kojoj se radi. Za instalaciju na OS X i Linux platformi u terminal se upisuje *curl https://install.meteor.com/ | sh.* Nakon instalacije potrebno je stvoriti aplikaciju i pokrenuti je kroz naredbeni redak. Instalacija nam daje mogućnost alata naredbenog retka zvanog Meteor alat, koji nam daje Meteor.js naredbe. Za pregled značajki Meteor alata dovoljno je pokrenuti jednostavnu naredbu *meteor --help*.



Sl. 2.4. Naredbeni redak za stvaranje i pokretanje aplikacije

Na slici 2.4. je prikazan naredbeni redak s naredbama za stvaranje i pokretanje aplikacije, te pokretanje u programu Atom.

Nakon izvršenja naredbi Meteor.js stvara sve potrebne datoteke za daljnje razvijanje aplikacije. Osim .meteor mape vidljive su klijent i poslužitelj mape s potrebnim datotekama za rad aplikacije. Vidljive datoteke već sadrže demonstracijski kod, te je aplikaciju već moguće pregledavati na adresi *localhost:3000*. Sve datoteke unutar klijent mape su isključivo na strani klijenta i neće biti učitavane od strane poslužitelja. Sve datoteke unutar poslužitelj mape su isključivo na strani poslužitelja i neće se slati klijentu. Izmjenjivanjem koda se aplikacija u pregledniku automatski ažurira što uvelike ubrzava proces pregledavanja, provjeravanja i izrade projekta. Na adresi za pregled projekta vidljivo je kako se stranica osvježava te je vidljivo i u terminalu gdje je upisana naredba za pokretanje Meteor.jsa. Dok se aplikacija izvodi u razvojnom modu, Meteor.js će otkriti sve promjene koje su napravljene i automatski ponovno izgraditi aplikaciju. Nakon što završi s izradom aplikacije koristi *Hot Code Push*, te gura sve promjene svim klijentima koji su spojeni na aplikaciju, bez potrebe da sam klijent osvježava. Poslužitelj može slati poruke klijentu u bilo koje vrijeme. Ako se napravi promjena samo na poslužitelju koja ne utječe na klijenta, ona se ne gura. Informacije o trenutnoj verziji Meteor.js izdanja su pohranjene u *.meteor/release*. Moguće je pokrenuti naredbu za ažuriranje Meteor.jsa koja provjerava je li trenutna verzija najnovija, ako ne, ažurira se. Dobra je praksa uvijek koristi zadnju verziju, pogotovo kada su u pitanju sigurnosne ispravke.



Sl. 2.5. Prikaz nakon stvaranja aplikacije u Atom-u

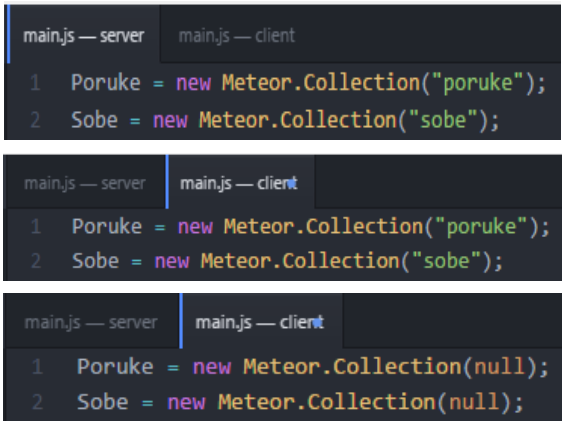
Na slici 2.5. je prikazano početno stanje datoteka u programu Atom nakon što je aplikacija stvorena.

Sl. 2.6. Službeni logo Meteor.js.js-a

Na slici 2.6. je prikazan logo Meteor.js radnog okvira.

2.2.1. Zbirke i sheme

U Meteor.js-u se podatkovni sloj obično pohranjuje u MongoDB bazu podataka. Skup povezanih podataka u MongoDB-u predstavlja zbirku podataka. MongoDB bazi podataka u Meteor.js-u se pristupa kroz zbirke, čineći ih primarnim mehanizmom pohrane za aplikacijske podatke. Međutim, zbirke su mnogo više nego način pohrane i dohvaćanja podataka, one sudjeluju u pružanju interaktivnog, povezanog korisničkog iskustva. Kad se stvara zbirka na poslužitelju, stvara se zbirka u MongoDB bazi podataka i sučelje na tu kolekciju koja će se koristiti na poslužitelju. To je sloj na vrhu temeljnog Node MongoDB upravljačkog programa, ali sa sinkronim API-em. Na strani klijenta se piše ista naredba, ali radi drugu stvar. Na strani klijenta ne postoji izravna veza s MongoDB bazom podataka i nije moguć sinkroni API. Umjesto toga, na strani klijenta, zbirka predstavlja pred memoriju baze podataka zahvaljujući MiniMongo biblioteci. Način na koji se podaci s zbirke klijenta prijenose na zbirku poslužitelja je pretplata na publikaciju koja gura podatke s poslužitelja na klijenta. Za pisanje podataka natrag na poslužitelja se koriste metode. Osim zbirki na strani klijenta i poslužitelja, postoji i treći način korištenja zbirki u Meteor.js-u, a to je lokalni način. Stvaraju se zbirke na strani servera ili klijenta na način da se umjesto imena zbirke upisuje *null*. To je zbirka Minimongo koja nema vezu s bazom podataka (inače bi imenovana zbirka bila ili izravno povezana s bazom podataka na serveru ili preko pretplate na klijentu). Lokalna zbirka je pogodan način za iskorištavanje pune snage biblioteke Minimongo za pohranjivanje memorije. Primjerice, može se koristiti umjesto jednostavnog niza ako je potrebno izvršiti složene upite na podacima. Može se iskoristiti i njezina reaktivnost na klijentu da bi se pokrenuo neko korisničko sučelje na način koji se čini prirodnim u Meteor.js-u. U Meteor.js-u zbirke su ekvivalent SQL tablicama. Ovisno o korištenoj verziji Meteor.js radnog okvira umjesto riječi Meteor se može upisivati riječ Mongo.



Sl. 2.7. Primjeri stvaranja zbirki

Na slici 2.7. su prikazani kodovi za stvaranje zbirki na sva 3 načina.

Premda je MongoDB baza podataka bez sheme, što omogućuje maksimalnu fleksibilnost u strukturiranju podataka, općenito je dobro upotrijebiti shemu da bi se ograničili sadržaji zbirke, a da bi se prilagodilo poznatom formatu. Kako to često biva, podatci se čitaju više nego što se pišu, pa je uglavnom lakše i događa se manje grešaka ako se upotrijebi shema pri pisanju. U Meteor.js-u, istaknuta shema paketa je *aldeed:simple-schema*. To je izražajna shema utemeljena na MongoDB-u koja se koristi za ubacivanje i ažuriranje dokumenata.

2.2.2. Atmosphere i npm paketi

Atmosphere paketi su napisani posebno za Meteor.js te imaju nekoliko prednosti nad npm-om kada se koriste s Meteor.js-om. Točnije, Atmosphere paketi mogu:

- ovisiti o jezgrenim paketima Meteor.js, kao što su *Distributed Data Protocol* (DDP) i Blaze
- izričito uključivati datoteke koje nisu JavaScript, uključujući CSS, Less, Sass, Stylus i statička sredstva
- iskoristiti Meteor.js-ov sustav izgradnje da budu automatski prevedeni iz jezika poput CoffeeScript-a
- imati dobro određen način za prenošenje drugačijeg koda za klijenta i poslužitelja, time omogućujući različito ponašanje u svakom danom kontekstu
- dobiti pristup Meteor.js-ovu prostoru imena paketa i globalnom izvozu paketa bez izričite upotrebe ES2015 uvoza
- primijeniti točnu inačicu ovisnosti između paketa koji koriste Meteor.js-ov rješavač ograničenja
- uključivati ugrađene priključke za Meteor.js-ov sustav izgradnje
- uključivati unaprijed izgrađen binarni kod za različite ustroje poslužitelja, kao što su Linux ili Windows

Npm je spremište osnovnih JavaScript paketa. Ti su paketi izvorno namijenjeni jedino za Node.js okruženje radnji isključivo od strane poslužitelja, ali kako je ekosustav JavaScript-a sazrijevao, pojavila su se rješenja koja bi omogućila upotrebu npm paketa u drugim okruženjima poput preglednika. Danas se npm koristi za sve vrste JavaScript paketa. Dodavanje glavnog npm paketa kroz naredbeni redak vrši se naredbom *meteor npm install.*Za pregled verzija svih paketa potrebno je otvoriti datoteku *.meteor/versions*.

Sl. 2.8. Službeni logo npm-a

Na slici 2.8. je prikazan logo npm upravitelja paketa.

2.2.3. PhoneGap

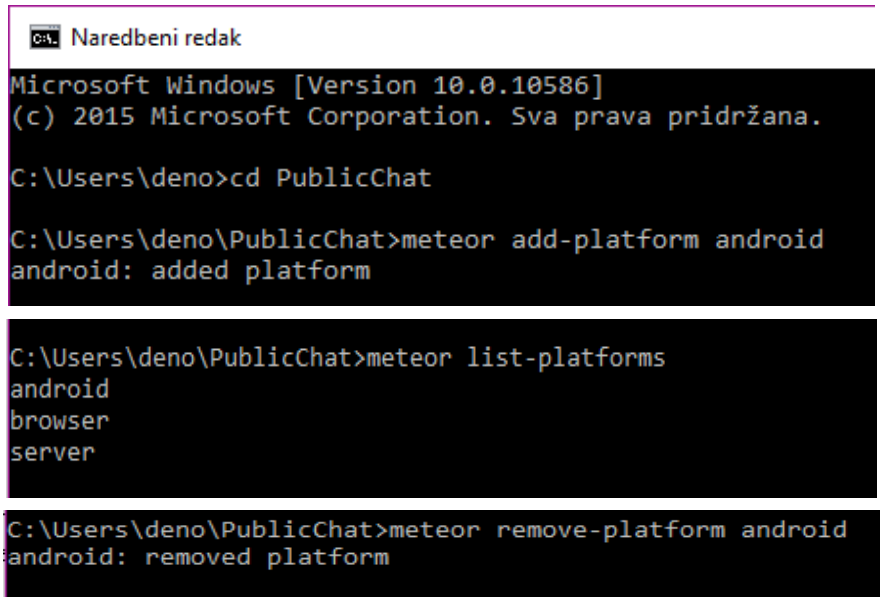
Apache Cordova (bivši PhoneGap) je popularan okvir za razvoj aplikacija za mobilne uređaje koji je izvorno izradio Nitobi. Adobe Systems je kupio Nitobi u 2011., nazvao ga PhoneGap te kasnije izdao inačicu softvera s otvorenim kodom zvanu Apache Cordova [7]. Apache Cordova omogućuje programerima softvera izgradnju aplikacija za mobilne uređaje koristeći CSS3, HTML5 i JavaScript umjesto oslanjanja na API-e specifične za pojedina sučelja, kao što su oni u Androidu, iOS-u ili uređaju Windows. Ona omogućuje sažimanje CSS, HTML i JavaScript koda ovisno o sučelju uređaja, proširuje značajke HTML-a i JavaScripta da bi mogli raditi s uređajem. Nastale aplikacije su hibridne, što znači da nisu ni zaista izvorne aplikacije za mobilne uređaje (jer cijelo prikazivanje izgleda vrši se preko interneta umjesto preko izvornog okvira korisničkog sučelja), niti su potpuno utemeljene na internetu (jer one nisu samo internetske aplikacije, nego su spakirane kao aplikacije za raspodjelu te imaju pristup API-ima izvornog uređaja). S Meteor.js-om nema potrebe za instalacijom Cordove ili za korištenjem cordova naredbe izravno.

Sl. 2.9. Logo PhoneGap/Apache Cordova

Na slici 2.9. su prikazani stari i novi logo PhoneGapa, odnosno Apache Cordove.

Izrađivanje projekta Cordova događa se kao dio naredbi za pokretanje i izgradnju Meteor.js-a i sam se projekt smatra tvorevinom (spremljen u *.meteor/local/cordova-build* u direktoriju aplikacije) koja može bilo kada biti izbrisana i ponovno izrađena. Umjesto

izmjenjivanja Cordovine *config.xml* datoteke, Meteor.js čita *mobile-config.js* datoteku u *root* direktoriju aplikacije i koristi postavke tamo opisane da bi uobličio proizvedeni projekt. Osim instaliranog PhoneGapa za izradu mobilne aplikacije potrebni su instalirani *Android SDK* i *Java Development Kit* (JDK) koje je prethodno moguće preuzeti s njihove službene stranice. Nakon instaliranih svih potrebnih servisa, potrebno je dodati *mobile-config.js* unutar projekta.



Sl. 2.10. Naredbeni redak s naredbama za dodavanje, uklanjanje i prikaz platformi.

Na slici 2.10. je prikazan naredbeni redak koji sadrži naredbe za dodavanje, uklanjanje i prikaz trenutnih platformi.

Na isti način prikazan na slici 2.10. se mogu dodati i ukloniti i druge platforme poput iOS. Za pokretanje emulacije na android platformi potrebno je imati spojen android uređaj na računalo i istu mrežu kao i računalo, te omogućenu komunikaciju između računala i mobilnog uređaja. Emulacija se pokreće naredbom *meteor run android* ili za emulaciju na računalu *meteor run android-device* za koju je potrebno prethodno podesiti *Android Virtual Devices* (AVD). Da bi aplikacija mogla biti istovremeno pokrenuta i na uređaju i web pregledniku potrebno je podesiti portove.

2.2.4. Implementacija i nadzor

Kada je Meteor.js aplikacija izrađena i provjerena, treba ju učitati na mrežu kako bi je drugi vidjeli. Implementacija Meteor.js aplikacije je slična implementaciji bilo koje druge Node.js aplikacije utemeljene na *websocketu*, ali je različita u određenim nijansama. Implementacija je web aplikacije temeljno različita od izdavanja većine drugih vrsta softvera zbog toga što se može implementirati koliko god često se želi. Ne mora se čekati da korisnici naprave nešto da bi dobili

novu inačicu softvera jer će im je poslužitelj dobaviti. Međutim, i dalje je važno temeljito provjeriti promjene dobrim procesom osiguranja kvalitete. Premda je lagano izbaciti ažuriranja za kvarove, ti kvarovi i dalje mogu prouzročiti velike probleme korisnicima te čak potencijalno kvar na podacima. Najlakši način za pouzdano upravljanje aplikacijom je korištenje Galaxya, usluge koju je razvila *Meteor Developmet Group* isključivo da bi pokrenula Meteor.js aplikacije. Galaxy je raspodijeljen sustav kojeg pokreće Amazon AWS. Danas većinu velikih Meteor.js aplikacija pokreće Galaxy te su mnoge od njih zamijenile prilagođena rješenja koja su koristila prije nego je Galaxy započeo s radom. Da bi se implementirao Galaxy, potrebno je izraditi korisnički račun i odvojiti MongoDB bazu podataka. Nakon toga, implementirati Galaxy je lako. Potrebno je samo dodati varijable okruženja datoteci za postavke da bi ga se usmjerilo prema MongoDB-u. Za implementaciju se koristi naredba *DEPLOY_HOSTNAME=us-east-1.galaxy-deploy.meteor.com meteor deploy naziv-aplikacije.com --settings production-settings.json* unutar naredbenog redka pozicioniranog u direktorij aplikacije. Da bi Galaxy ispravno radi s vlastitom domenom potrebno je usmjeriti DNS na Galaxy. Kada se pokreće aplikacija u proizvodnji, vrlo je važno pratiti učinak aplikacije i osigurati da glatko radi. Osim Galaxy-a, postoje i druga rješenja poput Kadire.

Sl. 2.11. Službeni logo Kadire

Na slici 2.8. je prikazan logo Kadira sustava za nadzor izvođenja na Meteor.js-u.

3. STRUKTURA MOBLINE APLIKACIJE

U ovom dijelu rada bit će prikazani kodovi koji su napisani i korišteni za izradu mobilne aplikacije. Također bit će prikazani dijelovi koda i njihovo pojašnjenje. Mobilna aplikacija se sastoji od nekoliko datoteka od kojih svaka ima svoju posebnu funkciju. Meteor.js je *full-stack* okvir, što znači da uključuje kod i na strani klijenta i na strani servera. Sastoji se od nekoliko posebnih direktorija koji nam pomažu razdvojiti kod:

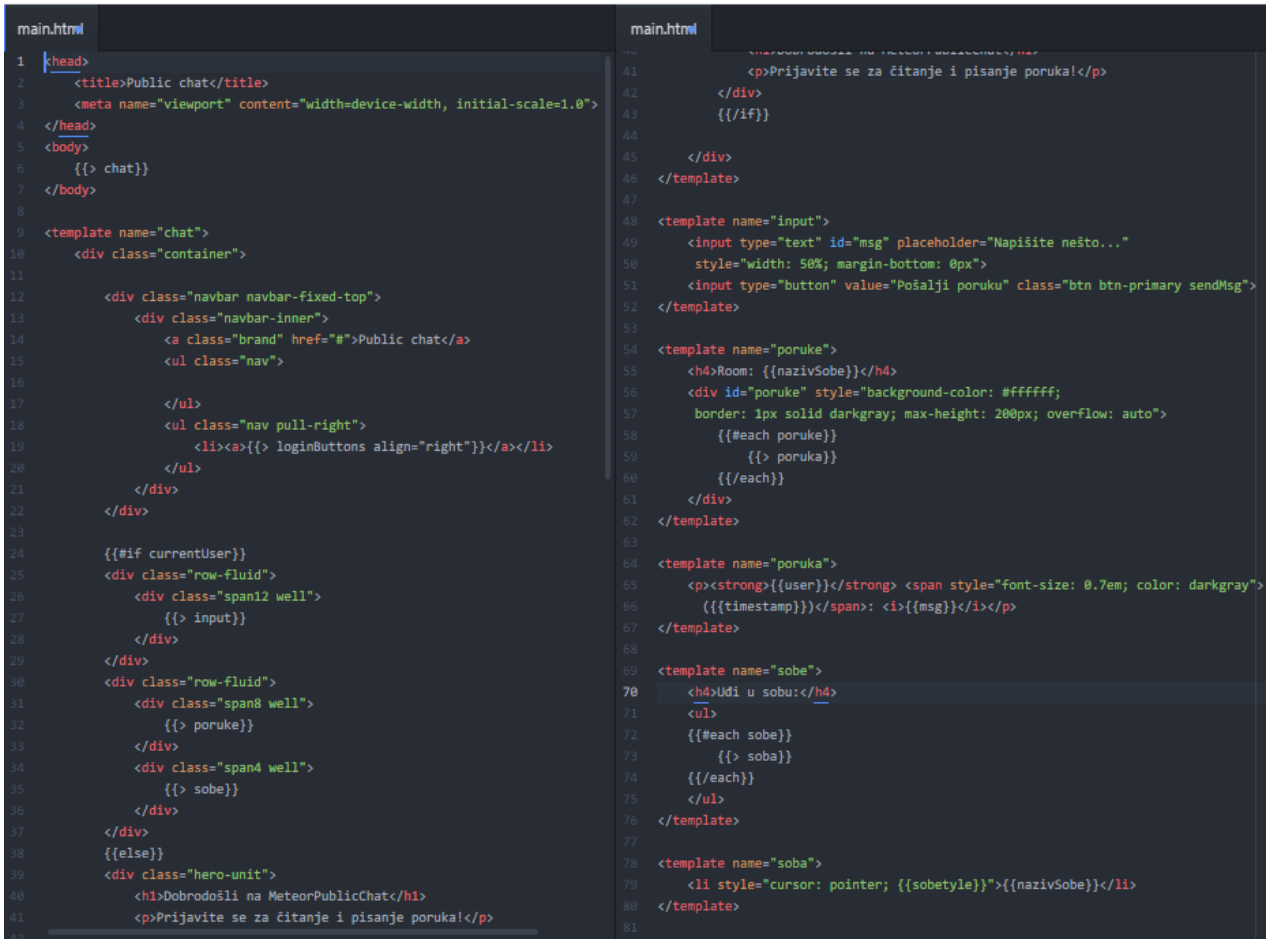
- client/ - klijentska strana
- server/ - poslužiteljska strana
- public/ - sredstva koja trebaju poslužiti klijentskoj strani
- private/ - sredstva koja trebaju poslužiti poslužiteljskoj strani

3.1. Početni okvir

Korišten je HTML5 i dodane su razne CSS datoteke kojima se oblikuje izgled okvira. HTML5 je uveo novi <template> element, koju Meteor.js koristi kako bi se odvojile oznake u više komada. Jednostavno se uzme neka od oznaka i stavi unutar <template> elementa, nakon toga se može dinamički umetnuti ili ukloniti taj predložak. Element <template> treba imati atribut ime kao i zaglavlje, kako bi se moglo na njega odnositi kada se koristi predložak. Meteor.js koristi svoju verziju Handlebar zvanu Spacebar. Postoji 4 glavne vrste Spacebarova:

- {{ varijabla }} – sve unutar duplih zagrada je tiskano kao običan tekst
- {{> poruka }} – umeće predložak po imenu
- {{#each}} – definira logičku operaciju
- {{{}}} – sve unutar trostrukih zagrada se umeće kao HTML

Spacebarovi omogućuju puno lakše i preglednije rukovanje oznakama. Moguće je koristiti drugi naziv početnog HTML dokumenta umjesto *index.html* bez usmjeravanja iz razloga što Meteor.js prolazi kroz sve HTML datoteke koje nisu unutar poslužitelja u potrazi za <head>, <body> i <template> elementima. Sadržaj tih elemenata se zbraja i prevodi u JavaScript jezik koji se šalje klijentu.



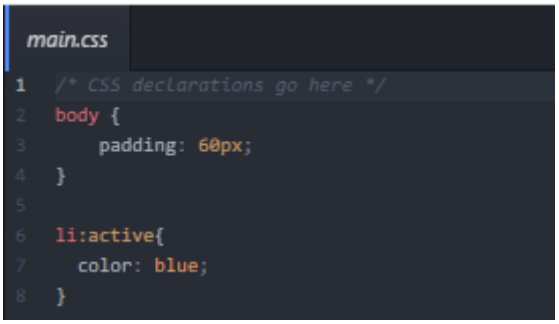
Sl. 3.1. Početni okvir – main.html

Na slici 3.1. je prikazan cijeli HTML kod početnog okvira aplikacije.

3.1.1. Kod za CSS stilsku datoteku

U datoteci *main.css* korišten je CSS da se oblikuje izgled elemenata u HTML dokumentu. Kao što se vidi na slici 3.2, ovo je samo jedna od stilskih datoteka. Korišteno je oko sedam stilskih datoteka koje sadrže mnogo linija koda, pri čemu su neke od tih već bile gotove i potrebno ih je samo prepravljat po želji. Korišteni su jedinstveni identifikatori, klase, cijeli elementi HTML-a i razne kombinacije. Identifikatori se koriste tako da se izabere neki HTML element koji treba promijeniti i dodaje se id = „(identifikator)“, dok se za klase koristi *class* oznaka. Ako se navede HTML element u CSS-u, to znači kako će svaki puta kada se pojavi taj element u HTML-u biti primijenjena svojstva koja smo mu zadali u CSS-u. Zbog toga su korištene klase i identifikatori, naime ako se nešto ponavlja više puta, ali ne treba za sve elemente, već za samo neke odabrane. Meteor.js nudi mogućnost dodavanja gotovog dizajna kroz naredbeni redak korištenjem naredbe *meteor add Naziv:style*. Na slici 3.2. vidljivo je da

datoteka sadrži samo dvije stilske promjene, razlog tome je što su ostale promjene unutar stilskih datoteka u .meteor direktoriju, te su neke promjene pisane unutar same HTML datoteke.



Sl. 3.2. CSS kod početnog okvira

Na slici 3.2. je prikazana *main.css* datoteka, odnosno CSS kod početnog okvira.

3.1.2. Spajanje na bazu podataka

Kod korištenja Meteor.js-a nema potrebe za posebnim spajanjem na bazu podataka poput uključivanja datoteke koja sadrži kod za povezivanje na bazu podataka. Samom instalacijom Meteor.js-a dobiva se pristup MongoDB bazi podataka, te se na principu *publish-subscribe* može upravljati shemama i zbirkama koje predstavljaju tablice u bazi podataka. Na slikama 3.3 i 3.4 mogu se vidjeti JavaScript kodovi za stvaranje zbirki, pretplate i objave, te za upravljanje zbirkama. Osim zbirki poruke i sobe, postoji i zbirka korisnici koja dolazi uz *accounts-ui* paket.

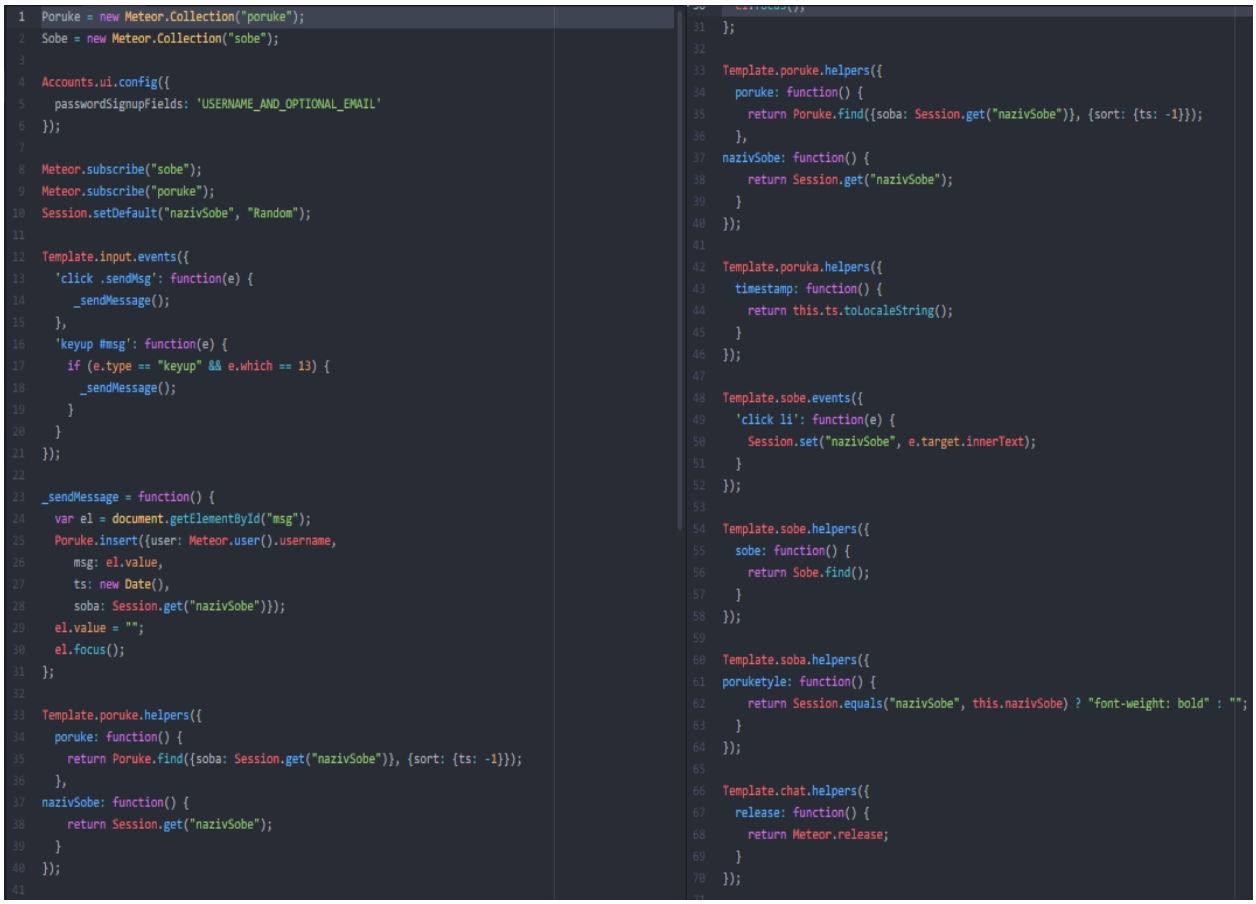
3.1.3. Registracija i prijava

Za korištenje registracije i prijave, nema potrebe za pisanjem koda koji bi to omogućio. Meteor.js pruža mogućnosti dodavanja cijelog sučelja samo jednom naredbom. Nakon što je paket dodan, može ga se uređivati unutar JavaScript datoteka, te se lako dodaju nove mogućnosti poput prijave putem Facebooka i slično. Osim jednostavnog dodavanja novih servisa za prijava, moguće je oblikovanje postojećeg poput odabira je li za prijavu potreban email ili korisničko ime. Glavne funkcije Meteor.js računa:

- Poništavanje lozinke – kada korisnik klikne na link u svom emailu, preusmjerava se na stranicu gdje može unijeti novu lozinku
- Upis korisnika – ukoliko je korisnik kreiran od strane administratora, a nije postavljena lozinka, koristi se metoda vrlo slična poništavanju lozinke
- Potvrda putem emaila – kada korisnik klikne na link u svom emailu, program bilježi da taj email doista pripada tom korisniku

3.2. Korisničke mogućnosti

Osim registracije i prijave, svaki prijavljeni korisnik ima mogućnosti pregleda i pisanja poruka, te mijenjanja soba. Korisnici koji nisu prijavljeni imaju ograničeni pogled. Svaka poruka sadrži korisnika, sadržaj poruke te vrijeme slanja poruke. Za promjenu sobe, potrebno je u izborniku odabrati željenu sobu nakon čega se izbacuje upozorenje, te se nakon potvrde soba promijeni. Na slici 3.3. vidljivo je kako je većina koda napisana na strani klijenta. *Template events* određuje sve rukovatelje događajima u predlošku, dok *Template helpers* možemo gledati kao varijable koje možemo koristiti u predlošku.



```
Poruke = new Meteor.Collection("poruke");
Sobe = new Meteor.Collection("sobe");

Accounts.ui.config({
  passwordSignupFields: 'USERNAME_AND_OPTIONAL_EMAIL'
});

Meteor.subscribe("sobe");
Meteor.subscribe("poruke");
Session.setDefault("nazivSobe", "Random");

Template.input.events({
  'click .sendMsg': function(e) {
    _sendMessage();
  },
  'keyup #msg': function(e) {
    if (e.type == "keyup" && e.which == 13) {
      _sendMessage();
    }
  }
});

_sendMessage = function() {
  var el = document.getElementById("msg");
  Poruke.insert({user: Meteor.user().username,
    msg: el.value,
    ts: new Date(),
    soba: Session.get("nazivSobe")});
  el.value = "";
  el.focus();
};

Template.poruke.helpers({
  poruke: function() {
    return Poruke.find({soba: Session.get("nazivSobe")}, {sort: {ts: -1}});
  },
nazivSobe: function() {
    return Session.get("nazivSobe");
  }
});

};

Template.poruke.helpers({
  poruke: function() {
    return Poruke.find({soba: Session.get("nazivSobe")}, {sort: {ts: -1}});
  },
nazivSobe: function() {
    return Session.get("nazivSobe");
  }
});

Template.poruka.helpers({
  timestamp: function() {
    return this.ts.toLocaleString();
  }
});

Template.sobe.events({
  'click li': function(e) {
    Session.set("nazivSobe", e.target.innerText);
  }
});

Template.sobe.helpers({
  sobe: function() {
    return Sobe.find();
  }
});

Template.soba.helpers({
poruketyle: function() {
    return Session.equals("nazivSobe", this.nazivSobe) ? "font-weight: bold" : "";
  }
});

Template.chat.helpers({
  release: function() {
    return Meteor.release;
  }
});
```

Slika 3.3. JavaScript kod na strani klijenta

Na slici 3.3 je prikazana main.js datoteka na strani klijenta s cijelim JavaScript kodom.

```
Poruke = new Meteor.Collection("poruke");
Sobe = new Meteor.Collection("sobe");

Meteor.startup(function () {
  Poruke.remove({});
  Sobe.remove({});
  if (Sobe.find().count() === 0) {
    ["Random", "Soba1", "Soba2", "Soba3"].forEach(function(r) {
      Sobe.insert({nazivSobe: r});
    });
  }
});

Sobe.deny({
  insert: function (userId, doc) {    return true;  },
  update: function (userId, doc, fieldNames, modifier) {    return true;  },
  remove: function (userId, doc) {    return true;  }
});
Poruke.deny({
  insert: function (userId, doc) {    return (userId === null);  },
  update: function (userId, doc, fieldNames, modifier) {    return true;  },
  remove: function (userId, doc) {    return true;  }
});
Poruke.allow({
  insert: function (userId, doc) {    return (userId !== null);  }
});

Meteor.publish("sobe", function () {  return Sobe.find();});
Meteor.publish("poruke", function () {  return Poruke.find({}, {sort: {ts: -1}});
});

```

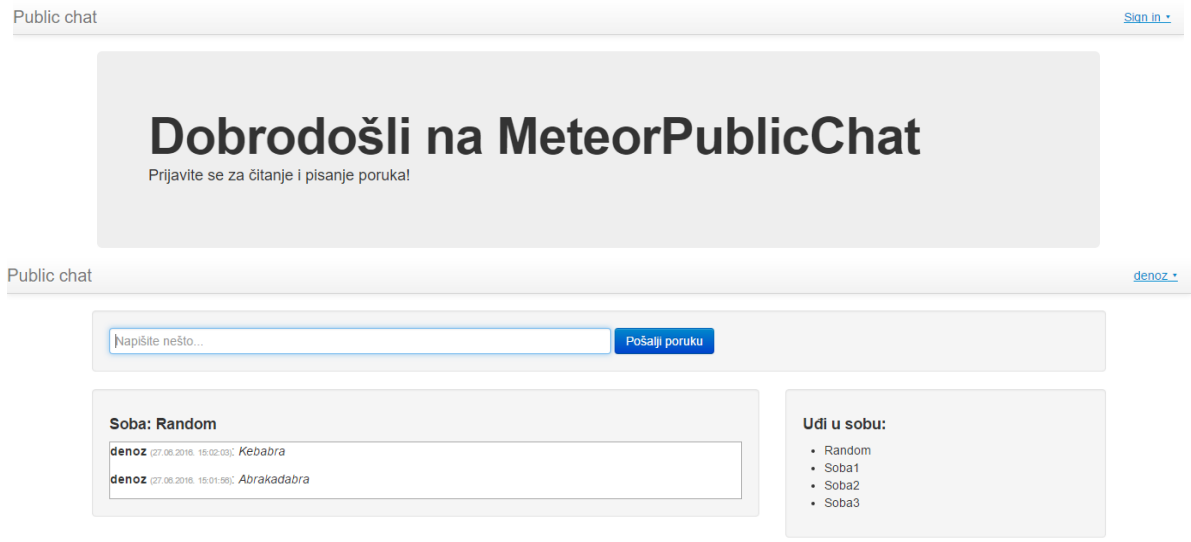
Slika 3.4. JavaScript kod na strani poslužitelja

Na slici 3.4 je prikazana main.js datoteka na strani poslužitelja s cijelim JavaScript kodom.

4. PRIKAZ MOBILNE APLIKACIJE

Nakon što se napiše HTML kod i spremi sa ekstenzijom .html ili .htm, otvaranjem te datoteke mobilni uređaj pretvara zapisani kod u odgovarajući prikaz na zaslonu. Na tom se mjestu mogu vidjeti vizualni prikazi koda.

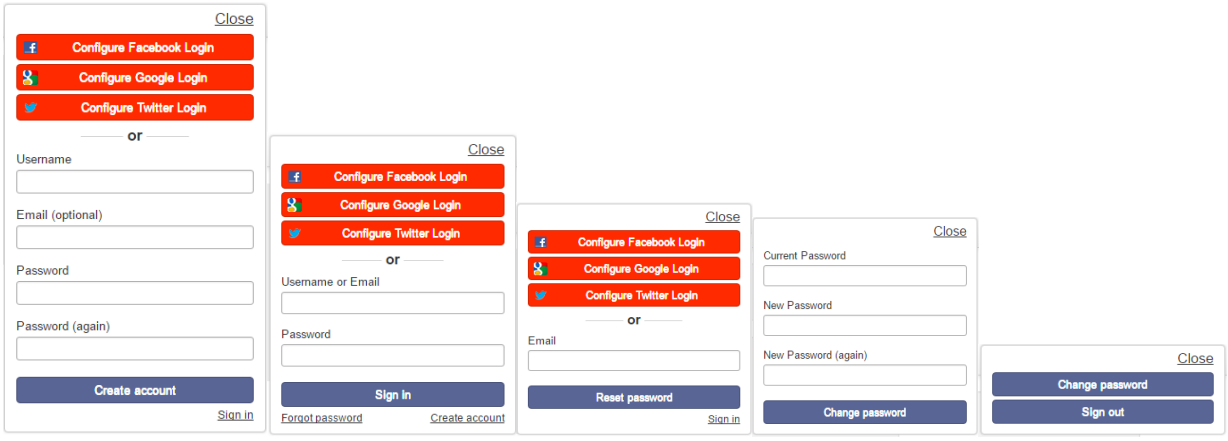
4.1. Početni okvir, registracija i prijava



Slika 4.1. Prikaz početnog okvira prijavljenog i ne prijavljenog korisnika

Slika 4.1. Prikazuje početni okvir u slučaju prijavljenog i ne prijavljenog korisnika.

Kod ne prijavljenog korisnika vidljivi su naslov, pozdravna poruka te mogučnosti prijave/registracije. Nakon prijave vidljivo je polje za unos poruke, popis soba te sve poruke koje su napisane u sobi u kojoj se korisnik trenutno nalazi.



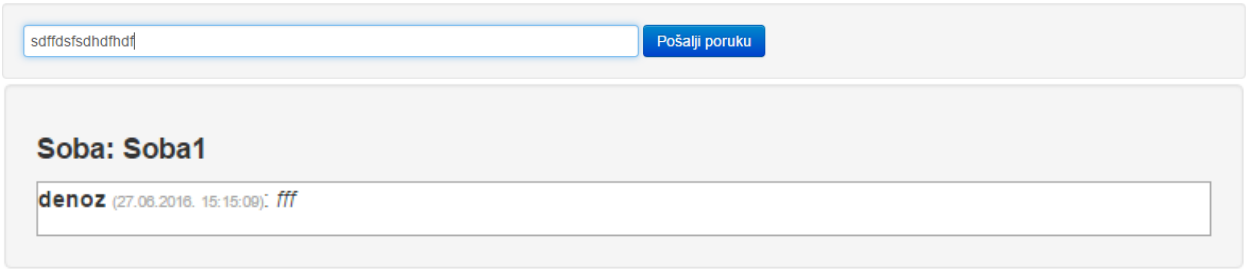
Slika 4.2. Prikaz korisničkog sučelja za prijavu

Slika 4.2. prikazuje izgled korisničke forme za vrijeme registracije, prijave, resetiranja lozinka, promjene lozinke, te odjave.

Osim uobičajenih polja na korisničkoj formi vidljivi su gumbi za postavljanje prijave putem drugih servisa. Nakon pristika na gumb *Configure NazivServisa Login* otvara se prozor s uputama za podešavanje prijave putem pojedinog servisa, nakon što se prijava podesi gumb *configure* se automatski zamijeni s gumbom *Sign in with NazivServisa*.

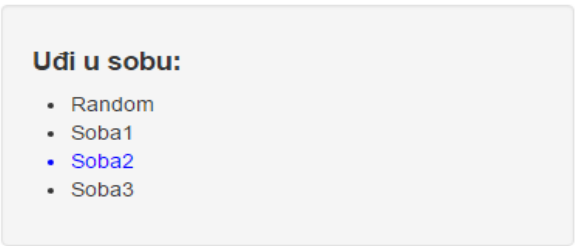
4.2. Upravljanje događajima

Za pisanje poruka potrebna je korisnička prijava, nakon čega se u odabranoj sobi u prostoru za poruke upisuje žaljeni sadržaj poruke. Poruka se šalje pristikom na gumb pošalji poruku ili pritiskom entera na tipkovnici računala ili mobilnog uređaja. Nakon pritiska gumba za slanje poruka, poruka se odmah prikazuje u prostoru za prikaz poruke bez potrebe za osvježavanjem okvira, odnosno stranice. Također potrebe za osvježavanjem nema ni kod promijene sobe, te se odmah može pisati i čitati poruke u novo odabranoj sobi.



Slika 4.3. Prostor za slanje poruka i prostor za sadržaj

Slika 4.3. prikazuje porostore za upis sadržaja poruke i gumb za slanje poruke, te prostor gdje se nalaze sve poruke unutar određene sobe.



Slika 4.4. Prostor za odabir soba

Na slici 4.4. je prikazan prostor koji sadrži listu soba. Promjena se vrši pritiskom miša na naziv sobe.

5. ZAKLJUČAK

Zadatak ovog rada bio je napraviti vodič za izradu mobilnih aplikacija korištenjem izomorfnog Meteor.js radnog okvira koji bi služio drugima kao pomoć pri izradi aplikacija u Meteor.js-u. Za primjer je napravljena aplikacija za javno čavrljanje. Opisani su glavni programski jezici i alati koji su korišteni u izradi rada. JavaScript je korišten za sve elemente i temelj je aplikacije. CSS oblikuje elemente u HTML-u i tako upravlja izgledom aplikacije. HTML povezuje sve korištene programske jezike i datoteke, što omogućuje prikaz funkcionalne mobilne aplikacije. Prikazani su i dijelovi koda korišteni u izradi, a nakon koda je prikazan izgled i opisane su mogućnosti aplikacije. Mobilna aplikacija ima sve funkcije koje su potrebne za ispravan i kontinuiran rad. Postoji mogućnost poboljšanja mobilne aplikacije da se koristi i za privatno čavrljanje, što bi zahtijevalo značajne promjene u kodu i izgledu aplikacije. Velike prednosti Meteor.jsa su što je cijelu aplikaciju moguće napraviti pomoću samo jednog jezika – Javascript, pametni paketi štede vrijeme, aplikacija je u stvarnom vremenu, optimiziran je, ima zajednicu koja je prijateljska i podržava programere, te je u skladu s tehnologijom.

LITERATURA

[1] Vanian, Jonathan, Meteor wants to be the warp drive for building real-time apps, Gigaom, 2014.

[2] Informatički enciklopedijski rječnik. Zagreb: Sv. 1. 2005.

[3] Elizabeth Robson, Eric Freeman, Head First HTML and CSS (Second Edition), O'Reilly Media,Inc.,2012.

[4] Jeniffer Niederst Robbins, Learning Web Design: A Begginer's Guide to HTML, CSS, JavaScript, and Web Graphics (Fourth Edition), Littlechair,Inc. 2012.

[5] Jon Duckett, HTML and CSS: Design and build websites (First Edition), John Wiley & Sons,Inc. 2011.

[6] Dane Cameron, A Software Engineer Learns HTML5, JavaScript and jQuery, Cisdal Publishing, 2014.

[7] Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap, Adobe, 2012.

SAŽETAK

Temelj je rada mobilna aplikacija zamišljena kao javno čavrljanje putem mobilnog uređaja. Čavrljanje je jako popularno diljem cijelog svijeta, te postoje mnoge aplikacije i web stranice koje pružaju mogućnosti javnog i privatnog čavrljanja. Za izradu aplikacije korišten je Meteor.js radni okvir koji ubrzava i olakšava izradu mobilne aplikacije. Osim JavaScripta korišteni su CSS i HTML. Aplikacija omogućava čavrljanje s ostalim korisnicima nakon prijave, za koju je potrebna registracija. Meteor.js nudi razne pakete poput kompletnog sučelja za registraciju i prijavu, dodavanja prijave putem Facebooka, Twittera i ostalih servisa ukucavanjem samo jedne naredbe u naredbeni redak. Osim paketa Meteor.js sadrži integriranu MongoDB bazu podataka. Dovoljan je samo jedan jezik, JavaScript, za izradu potpune aplikacije. Također, objava projekta je jednostavna i vrši se jednom naredbom kao i dodavanje platformi poput Anroid i iOS. Mobilna aplikacija omogućava svim prijavljenim korisnicima međusobno čavrljanje kroz nekoliko soba, također svaki korisnik ima uvid u listu korisnika u trenutnoj sobi.

Ključne riječi: Čavrljanje, mobilna aplikacija, kod, Meteor.js.

ABSTRACT

Development of mobile applications using isomorphic Meteor.js framework

The basis of the work is the mobile application designed as a public chat on a mobile device. Chat is very popular all over the world, and there are many applications and websites that provide public and private chat. To create applications used Meteor.js framework that accelerates and facilitates the development of mobile applications. In addition to the JavaScript used also CSS and HTML. The application allows you to chat with other users after registration, for which registration is required. Meteor.js offers various packages such as a complete interface for registration and login, adding the logon with Facebook, Twitter and other services by typing just one command at a command prompt. In addition the package includes an integrated Meteor.js MongoDB database. It takes only one language, JavaScript, for making complete application. Also, the deployment of the project is simple and is done with one command as well as adding platforms like Anroid and iOS. The mobile application allows all registered users to chat with each other through several rooms, each user also has access to a list of users in the current room.

Key words: Chat, mobile application, code, Meteor.js.

ŽIVOTOPIS

Denis Prpić, rođen u Požegi 15.Veljače 1994. Osnovnu školu završio u Požegi u razdoblju od 2000. do 2008. godine. 2008. godine upisuje srednju tehničku školu u Požegi, smjer elektrotehničar koju završava 2012. godine. Nakon toga upisuje Elektrotehnički fakultet u Osijeku, smjer informatika kojega trenutno pohađa.