

RAZVOJ I IMPLEMENTACIJA INTEGRISANOG VEB SISTEMA PRIMENOM SAVREMENIH INTERNET TEHNOLOGIJA

DEVELOPMENT AND IMPLEMENTATION OF AN INTEGRATED WEB SYSTEM USING MODERN WEB TECHNOLOGIES

Kandidat

LUKA PATARČIĆ

Mentor

DR ZLATKO ČOVIĆ

Subotica, 2020. godine

Izvod

Ovaj diplomski rad predstavlja razvoj i implementaciju društvene mreže pod imenom „Allshack“. U pitanju je web sajt i mobilna aplikacija koja sadrži sve odlike savremenih društvenih mreža poput Instagram, Twitter i Facebook. Prvobitno je bilo potrebno naučiti sve tehnologije neophodne za proizvodnju projekta, savladati ih i napraviti program koji je moguće koristiti u realnom svetu. Nakon toga je bilo potrebno testirati da li sistem radi ono za šta je namenjen. Projekat je postavljen na internet kao gotov proizvod koji se može koristiti.

Sadržaj

Izvod	2
Sadržaj	3
Popis oznaka	6
Zahvale	7
Zadatak završnog rada	8
1. Uvod	9
1.1. Opis problema	9
1.2. Ciljevi rada	9
2. Teorijske osnove	10
2.1. HTML	10
2.2. CSS	10
2.3. JavaScript	10
2.4. PHP	10
2.5. MySQL	11
2.6. Symfony (API)	11
2.6.1. REST	11
2.6.2. Doctrine ORM	11
2.6.3. JWT Autentifikacija	12
2.6.4. Twig	12
2.6.5. Swift Mailer	12
2.6.6. PHPUnit Tests	12
2.6.7. Composer	12
2.6.8. Postman	12
2.6.9. Symfony Commands	12
2.6.10. FCM	12
2.7. React (Front-End)	13
2.7.1. Class Based Components	13
2.7.2. JSX	13
2.7.3. NPM	13
2.7.4. Fetch API	13

2.7.5.	Material Design Bootstrap	14
2.7.6.	Cookie	14
2.7.7.	Prop-Types.....	14
2.7.8.	Adobe Illustrator	14
2.7.9.	Font Awesome.....	14
2.7.10.	Google Fonts.....	14
2.7.11.	React Best Practices i Optimizacija.....	14
2.7.12.	SEO optimizacija	15
2.8.	React Native	15
2.8.1.	React Navigation 5.....	15
2.8.2.	React Native AsyncStorage.....	15
2.8.3.	React Native paper	15
2.8.4.	React Native Image Picker.....	15
2.9.	Node.js.....	16
2.9.1.	Socket.io	16
2.10.	Git/GitHub	16
3.	Rešavanje problema	17
3.1.	Web sajt.....	17
3.1.1.	Početna stranica	17
3.1.2.	Login/Registracija	18
3.1.3.	Objave.....	21
3.1.4.	Pretraga	28
3.1.5.	Profil	28
3.2.	Mobilna Aplikacija	30
3.2.1.	„Splash Screen“	30
3.2.2.	Login/Registracija	31
3.2.3.	Početna stranica	33
3.2.4.	Pretraga	34
3.2.5.	Notifikacije.....	34
3.2.6.	Profil	35
3.2.7.	Čet	36
3.2.8.	„Push“ notifikacije	37
3.3.	Šema sistema.....	38

3.4.	Baza podataka	38
3.5.	Testiranje u Realnom okruženju.....	40
3.5.1.	API produkcija.....	40
3.5.2.	Web sajt produkcija.....	40
3.5.3.	Mobilna produkcija.....	40
3.5.4.	Node.js produkcija.....	40
4.	Zaključak.....	41
	Literatura	42
5.	Prilozi	43
5.1.	Elektronski materijali.....	43

Popis oznaka

Oznaka	Značenje
JSON	JavaScript Object Notation (JavaScript Objektna Notacija)
OOP	Object Oriented Programming (Objektno Orijentisano Programiranje)
PHP	Hypertext Preprocessor (Hipertekst Preprocesor)
JSX	JavaScript XML (JavaSkript XML)
HTML	Hypertext Markup Language (Jezik za opis Hiper-Teksta)
CSS	Cascading Style Sheets (Kaskadne liste stilova)
JS	JavaScript (JavaSkript)
AJAX	Asynchronous JavaScript and XML (Asinhroni JavaSkript i XML)
ORM	Object Relational Mapper (Objektni Relacioni Mapper)
SMTP	Simple Mail Transfer Protocol (Jednostavni protokol za prenos e-pošte)
JWT	JSON Web Token (JSON serverski token)
NPM	Node Package Manager (Node menadžer paketa)
API	Application Programming Interface (Aplikativni Programski Interfejs)
SSL	Secure Socket Layer (Sigurnosni Protokol Komunikacija)
REST	Representational State Transfer (Reprezentativni Prenos Stanja)
URI	Uniform Resource Identifier (Uniformni Identifikator Resursa)
HTTP	Hyper Text Transfer Protocol (Hipe-Tekst Prenosni Protokol)
RSA	Rivest Shamir Adleman (Sistem kriptovanja javnim ključem)
FCM	Firebase Cloud Messaging (Firebase za Web Poruke)
iOS	iPhone Operating System (iPhone Operativni Sistem)
DOM	Document Object Model (Model Objekta Dokumenta)
UI	User Interface (Korisnički Interfejs)
SSR	Server Side Rendering (Renderiranje na serverskoj strani)
XSS	Cross Site Scripting (Skriptovanje sa drugog sajta)
CSRF	Cross-Site Request Forgery (Krivotvorenje zahteva sa drugog sajta)
Less	Linear Style Sheets (Linearna Lista Stilova)
CTA	Call To Action (Poziv na akciju)
DTD	Document Type Definition (Definicija Tipa Dokumenta)
CMS	Content Management System (Sistem za upravljanje sadržajem)
CRON	Scheduling one-time tasks (Zakazivanje jednokratnih zadataka)
TTL	Time To Live (Vreme za život)

Zahvale

Ovim putem bih se zahvalio svojoj porodici na konstantnoj podršci i motivaciji kao i kolegama bez kojih školovanje ne bi bilo toliko lepo. Pogotovo bih se zahvalio profesoru Zlatku Čoviću i profesorki Sanji Maravić Čisar jer sam zahvaljujući njima zavoleo programiranje i pronašao zanimanje kojim ću se baviti u životu.

Zadatak završnog rada

- Proučiti, naučiti i savladati Symfony, React, React Native i Node.js
- Realizovati integrisani web sistem primenom savremenih internet tehnologija
- U realizaciji sistema koristiti: Symfony, React, React Native, Node.js
- Realizovani sistem postaviti na web server
- Testirati integrisani sistem u realnom okruženju

1. Uvod

Završni rad predstavlja razvoj i implementaciju integrisanog web sistema, u vidu društvene mreže koja se sastoji od baze podataka, API, web sajta i mobilne aplikacije.

1.1. Opis problema

Za početak je bilo neophodno odabrati tip projekta na koji će se primeniti sledeće tehnologije. Odabrana je društvena mreža jer poseduje kompleksne funkcionalnosti koje se mogu odraditi na visokom nivou pomoću baš ovih tehnologija. Specifično su odabrane ove tehnologije pošto su veoma tražene u današnjem svetu programiranja. Nakon proučavanja i savladanja tehnologija započet je projekat koji je imao cilj napraviti što sličniju društvenu mrežu kao ostale aktuelne mreže u svetu. Ovo obuhvata čet sistem u realnom vremenu (engleski: real-time), potpuni asinhroni sajt sa što manje osvežavanja i dinamički učitanim sadržajem. Nakon dovođenja aplikacije do zadovoljavajućeg nivoa sve je postavljeno na internet u vidu produkcije.

1.2. Ciljevi rada

Cilj ovog rada je bilo proučiti, naučiti i implementirati moderne web tehnologije za jedan uniformni sistem koji je spreman za produkciju i korišćenje.

2. Teorijske osnove

U izradi ovog projekta primenjene su sledeće tehnologije:

2.1. HTML

HTML (engleski: Hypertext Markup Language) je opisni jezik koji se koristi za osnovu svih web sajtova.

Sastoji se iz više različitih delova koji se mogu kategorisati u tri grupe:

- Verzija datoteke (engleski: DTD)
- Deklarativno zaglavlje dokumenata
- Telo dokumenta

HTML fajl koji opisuje jednu stranicu, sastavljen je od teksta i tagova. Tag predstavlja osnovu svake web stranice koja se piše na sledeći način „<“ ime tag-a „>“ (<html>). Ekstenzija za HTML fajlove je .html [7].

2.2. CSS

CSS (engleski: Cascading Style Sheets) je tehnologija za formatiranje web sajtova [7]. Osnovni ciljevi uvođenja ove tehnologije su bili:

- Razdvojiti HTML dokument i dokumente za stilizovanje (CSS)
- Unaprediti izgled web stranica

Stilove je moguće primeniti na 3 načina: eksterni fajl, interni stil i umetnuti stil. Pravilo stila predstavlja osnovu CSS tehnologije i sastoji se iz dva glavna dela:

- **Selektor** – koji određuje na koji HTML element se primenjuje stilizovanje.
- **Deklarativni blok** – koji se sastoji od stilova koji će biti primenjeni na selektor

2.3. JavaScript

JavaScript je objektno bazirani skriptni programski jezik. Primeri upotrebe na web sajtovima:

- Dodavanje dinamičnosti stranicama
- Kreiranje „kolačića“ (engleski: cookies)
- Programiranje igrica putem HTML5 Canvas
- Asinhrono prikupljanje podataka od servera

JavaScript je moguće primeniti na 3 načina: eksterni fajl, interna skripta ili umetnuta skripta [7].

2.4. PHP

PHP (engleski: Hypertext Preprocessor) je skriptni serverski programski jezik koji omogućuje razvoj dinamičkih web sajtova. Podržava više web servera kao i operativnih sistema. Danas najveći broj sajtova su napravljeni pomoću PHP-a. Ima veoma dobru integraciju sa bazama podataka. Ekstenzija za php fajlove je .php. Prilikom pisanja PHP koda mora da počinje sa „<?php“ i završava sa „?>“ [7].

2.5. MySQL

MySQL je sistem za upravljanje relacionim bazama podataka otvorenog koda. Poznat je po brzini, pouzdanosti i lakoći upotrebe. Radi sa svim operativnim sistemima i programskim jezicima kao što su PHP, C, C++, JAVA itd.

2.6. Symfony (API)

Symfony je PHP „web framework“. Objavljen je 18. oktobra 2005 godine kao besplatan software. Inspirisan je od strane Spring „framework-a“.

Symfony funkcioniše na principu modularnih komponenti. Može da bude jako lagan „framework“ koji se koristi samo za neki API ili da bude drastično kompleksan i jak „framework“. Sve je to moguće zbog modularnih komponenti koje mogu da se dodaju ili oduzimaju. Danas je Symfony „open source“ projekat koji podržava SensioLabs [10].

Zašto baš Symfony?

- **Podržan je od strane kompanije**

Mnogi „framework-ovi“ su napravljeni od strane ljudi iz „open source“ zajednice, Symfony iako je „open source“ je podržan od strane kompanije SensioLabs. Samim tim možete biti sigurni da radite sa pouzdanim i testiranim kodom na vašem projektu.

- **Jako fleksibilan**

Symfony u svojoj srži sadrži dve najznačajnije funkcionalnosti. Pravljenje „bundles“ i „components“ što omogućava skalijabilne web aplikacije i modularnost.

- **Dokazan kao najbolji**

Za razliku od drugih PHP „framework-ova“ Symfony je dokazan u praksi. Ima bezbroj sajtova napravljenih pomoću njega kao što su Spotify, Dailymotion, National Geographic i mnogi drugi. Mnogobrojne kompanije kao i Infostud koriste ovaj „framework“. Takođe većina CMS sistema su napravljeni u Symfony-ju kao što su Drupal, Joomla!, Magento i drugi [1][2].

2.6.1. REST

REST je stil (softverska arhitektura) pisanja web servisa, ako prate ove konvencije nazivaju se i „RESTful Web Services“. REST omogućava da sistem koji zahteva podatke pristupa i manipuliše tekstualnim reprezentacijama web servisa korišćenjem uniformnih i predefinisanih setova operacija. Zahtevi koji su upućeni ka jednom URI će prouzrokovati odziv (engleski: response) u vidu nekog formata, obično JSON. Najčešći vid prenosa podataka je preko HTTP a metode koje su dostupne su GET, POST, PUT, PATCH, DELETE itd. Cilj REST aplikacija je brzina, pouzdanost i održivost.

2.6.2. Doctrine ORM

Doctrine je ORM(engleski: Object Relational Mapper) za PHP koji pruža „persistance“ za PHP objekte. Koristi „Data Mapper“ patern u srži, ciljajući da odvoji biznis logiku od „persistance“ u relacionim bazama podataka. Prednost koju pruža Doctrine je ta da programer može u potpunosti da se oslanja na OOP programiranje. Entiteti su PHP objekti koji mogu biti identifikovani pomoću jedinstvenog identifikatora poput primarnog ključa. Ovim klasama nije potrebno nasleđivanje nikakve apstraktne klase ili interfejsa. Entiteti sadrže promenljive i metode koje su direktno povezane sa kolonama u bazi pomoću Doctrine mapirajućih sposobnosti.

2.6.3. JWT Autentifikacija

JWT (engleski: JSON Web Token) je otvoreni standard (RFC 7519) koji definiše kompaktan i samoizolovan način da se bezbedno prenose podaci između klijenta i servera kao JSON objekat. Ovi podaci mogu biti verifikovani i poverljivi pošto se digitalno potpisuju. JWT može da se potpiše pomoću javnog i privatnog para ključeva korišćenjem RSA.

2.6.4. Twig

Twig je najpopularnija platforma za interpretaciju (engleski: templating engine) PHP jezika. Pomoću Twig-a mogu se praviti stranice (engleski: templates) koji mogu da se iskoriste na više mesta. Ovo omogućava pisanje manje koda, lakše održavanje i bržu proizvodnju. Twig sadrži interni sistem za keširanje (engleski: internal caching system) koji ga čini veoma brzim, što poboljšava UX.

2.6.5. Swift Mailer

SwiftMailer je biblioteka koja se integriše u bilo koji web sistem koji koristi PHP. Omogućava slanje mejlova SMTP protokolom na OOP način.

2.6.6. PHPUnit Tests

PHPUnit je unit test „framework“ za PHP programski jezik. Razvio ga je Sebastian Bergmann. PHPUnit je zasnovan na ideji da bi programeri trebali da pronađu greške na novo dodatom kodu i da provere da nema novonastalih grešaka u starom kodu. PHPUnit kao i ostali unit test „framework-ovi“ koristi potvrđivanje (engleski: assertion) za proveru da li se kod pravilno ponaša sa određenim podacima. Cilj unit testova je da se izoluje svaki deo programa i da prikaže da li su ti delovi pravilno napisani.

2.6.7. Composer

Composer je upravljač paketa (engleski: dependency manager) za PHP programski jezik. Pruža standardni format rukovođenja paketa PHP softvera i biblioteka. Composer radi iz komandne linije i instalira PHP biblioteke za PHP aplikaciju. Omogućava takođe automatsko učitavanje (engleski: autoload) funkcionalnosti za biblioteke koje su implementirale ovu mogućnost. Ovo znači da su sve klase unutar programa učitane pokretanjem programa i pozivaju se korišćenjem „use“ sintakse.

2.6.8. Postman

Postman je softver koji omogućava HTTP zahteve ka stranicama i olakšava proces pravljenja API programa.

2.6.9. Symfony Commands

Symfony omogućava pravljenje komandi koje se mogu pokretati iz terminala. Ovo omogućava pisanje koda za CRON i slične programe za automatizaciju.

2.6.10. FCM

FCM (engleski: Firebase Cloud Messaging) je multi-platformsko rešenje za poruke (notifikacije) koja omogućava slanje poruka na pouzdan i lak način.

FCM implementacija uključuje dva glavna dela slanja i primanja:

- Pouzdano okruženje kao što je Firebase konzola ili aplikacija koja šalje zahtev na Firebase da se pošalje poruka.
- Mobilni uređaj koji radi na Android-u ili iOS koji prima poruku i prikazuje je na ekranu.

2.7. React (Front-End)

React je JavaScript biblioteka za pravljenje UI. Nije „framework“ jer ne pruža rešenja za sve probleme već se pored React moraju instalirati dodatne biblioteke kao što su React Router za rutiranje unutar aplikacije Next.js za SSR, Redux za „state management“. Napravljen je i održavan od strane Facebook što predstavlja veliku prednost zbog kvaliteta koda, popravljivanja grešaka itd. React kod se sadrži od entiteta nazvanim komponente. Ove komponente je moguće prikazati na web čitaču pomoću React DOM biblioteke. Dve najvažnije stvari kod React biblioteke su stanja (engleski: state) i svojstva (engleski: props) koji predstavljaju čuvanje podataka i manipulisanje tim podacima kako bi web stranica bila dinamična [3].

2.7.1. Class Based Components

Postoje dve vrste komponenti u obliku klasa i funkcija. Klasne komponente rade na principu OOP programiranja. Svaka klasa nadovezuje React klasu „Component“ koja ima jedan obavezan metod „render“ koji prikazuje React Komponente na web čitaču. U Klasnim komponentama postoje takozvane „lifecycle methods“ koje se pokreću u određenim trenucima prikazivanja sadržaja na čitaču [11]. Primeri su:

- **shouldComponentUpdate** - koji dozvoljava programeru da zaustavi nepotrebna ponovna pokretanja „render“ metode ukoliko su „props“ i „state“ isti kao i pre.
- **componentDidMount** - se poziva kada se komponenta napravi na korisničkom delu. Tu se obično stavljaju API pozivi ka serveru.
- **componentWillUnmount** - se poziva kada se komponenta uništava. Ovde se obično stavljaju stvari koje želimo da obrišemo kada se i sama komponenta obriše na primer „event listeners“.

2.7.2. JSX

JSX (engleski: JavaScript XML) je ekstenzija na JavaScript sintaksu. Veoma je sličan HTML i ovo ga čini veoma lakim za savladanje. React komponente su pisane korišćenjem JSX sintakse. Sličan primer JSX je Facebook-ov XHP za PHP jezik.

2.7.3. NPM

NPM (engleski: Node Package Manager) je menadžer paketa za JavaScript programski jezik. Sastoji se iz dva dela:

- konzola gde korisnik navodi koje pakete (biblioteke) želi da preuzme za svoj projekat
- online baza podataka koja sadrži sve moguće pakete koji mogu da se preuzmu

2.7.4. Fetch API

Fetch API pruža interfejs za preuzimanje resursa preko interneta. Radi na veoma sličan način kao „XMLHttpRequest“ ali pruža mnogo bolji i jači set funkcionalnosti.

Fetch API koristi fetch() metod koji uzima jedan obavezan argument što je putanja do resursa koji želimo da preuzmemo. Kao rezultat vraća obećanje (engleski: promise) koji se pretvara u odziv (engleski: response) od tog zahteva bez obzira da li je bio uspešan ili ne. Obećanje (engleski: promise) je zastupnik (engleski: proxy) za vrednost koja nije prvobitno poznata kada se poziva ovaj metod. Omogućava povezivanje rukovatelja sa asinhronom metodom koja će na kraju prouzrokovati uspešnu ili neuspešnu vrednost. Kao drugi opcioni argument postavlja se jedan objekat gde možemo navesti dodatne informacije kao Header, Method, Body.

2.7.5. Material Design Bootstrap

Material Design je dizajnerski jezik koji je Google napravio u 2014 godini. To je set pravila i načina pravljenja UI komponenti.

Bootstrap je CSS biblioteka koja je usmerena ka responzivnom „mobile first“ razvoju web sajtova. Sastoji se od predefinisanih CSS klasa za stilizovanje i mnoštvo JS gotovih funkcionalnosti.

Material Design Bootstrap prati Google-ov dizajn patern za Bootstrap biblioteku. Pored obične verzije postoje verzije za JavaScript „framework-ove“ kao sto su Angular, Vue i naravno React.

2.7.6. Cookie

Kolačići (engleski: cookies) su mali setovi podataka koji se čuvaju na uređaju korisnika. Kolačići su napravljeni sa svrhom da budu pouzdan način da se zapamti informacija u čitaču (kao npr. proizvodi u korpi pri kupovini ili temu veb sajta). Autentifikacioni kolačići su najčešći metod korišćen da se zna koji je korisnik ulogovan na web stranici. Kolačići predstavljaju problem jer je moguće da haker dođe do njihovih podataka korišćenjem metoda poput XSS i CSRF. Načini da se zaštitite su da se kolačići za autentifikaciju koriste samo preko HTTPS metoda gde se kolačić ne može pristupiti preko JS i radi preko sigurne konekcije.

2.7.7. Prop-Types

Prop-Types biblioteka je namenjena za proveru „props“ koje neka komponenta prima. Ovo je veoma korisno jer kako aplikacija raste postoji sve više komponenti i greške postaju češće. Na ovaj način se te greške mogu smanjiti proverom tipa „props“ koji se prosleđuju i da li su uopste prosleđeni.

2.7.8. Adobe Illustrator

Adobe Illustrator je editor za vektorsku grafiku i dizajn. Orginalno napravljen za „Apple Macintosh“, razvoj je započet 1985 godine. Illustrator se smatra jednim od najboljih editora za vektorsku grafiku.

2.7.9. Font Awesome

Font Awesome je biblioteka za fontove i ikonice zasnovana na CSS i Less. Napravio je Dave Gandy kako bi se mogla koristiti uporedo sa Bootstrap.

2.7.10. Google Fonts

Google Fonts je biblioteka besplatnih fontova. Veoma dobro se integriše sa CSS i Android-om gde se samo prosleđuje link bez ikakve obaveze skidanja font fajlova.

2.7.11. React Best Practices i Optimizacija

Kako bi React radio najbrže i najbolje prvobitno je neophodno pre postavljanja na web sajt da se napravi produkciona verzija. Ovde React u pozadini napravi optimizovanu verziju aplikacije koja je spremna za produkciju.

Ukoliko želite korak više da napravite potrebno je skinuti dodatne biblioteke sa NPM koje će još više optimizovati aplikaciju.

- Brunch - koji vrši minifikaciju (stavi ceo kod u jednu liniju) kako bi se smanjio fajl i ubrzalo učitavanje
- Webpack – webpack se koristi uz React i on automatski optimizuje kod ali se mogu dodati dodatni

parametri da se poboljša taj proces.

Radi optimizacije React biblioteke i ukoliko se koriste klasne komponente potrebno je koristiti „lifecycle metod“ kao „shouldComponentUpdate“ kako bi se izbeglo nepotrebno ponovno učitavanje koda.

2.7.12. SEO optimizacija

SEO optimizacija funkcioniše malo drugačije kada se koriste JavaScript „framework-ovi“ za pravljenje web stranica pošto se sve učitava dinamički, ne postoji statičan HTML, moramo pomoći Google da nam pronađe sve stranice i da ih uspešno indeksira. Jedna veoma korisna biblioteka za ovo je „react-router-sitemap“ koji koristi „React-Router“ biblioteku da izgeneriše „sitemap.xml“ dokument koji će poslužiti da Google pronađe svaku našu stranicu. Pored ovoga naravno potrebno je da sajt sadrži sve meta tagove potrebne i „robots.txt“ fajl. Najbolji način za rešavanje problema JavaScript „framework-ova“ je metod „Pre-Rendering“ gde se JavaScript kod pretvori u statičan HTML format i na taj način ovi sajtovi funkcionišu kao i svi ostali. SSR je veoma dobar za razliku od CSR pošto Google robot i klijent dobiju HTML kao rezultat što znatno olakšava indeksiranje stranica.

2.8. React Native

React Native je mobilni „framework“ napravljen od strane Facebook. Namenjen je za pravljenje Android i iOS aplikacija korišćenjem React uz native mogućnosti svakog operativnog sistema. React i React Native su u suštini potpuno identični sem što React Native ne manipuliše DOM, već interpretira JavaScript koji je programer napisao, u kod koji nativni program može da razume [4][12].

2.8.1. React Navigation 5

React Navigation je JavaScript biblioteka koja je namenjena za React Native „framework“. Služi za navigaciju između stranica unutar mobilnih aplikacija. Posедуje sve moguće varijacije Rutiranja od „Bottom Tab Navigation“, „App Drawer“ i mnogi drugi sa animacijama koje programer može da izmeni. Veoma bitna biblioteka za pravljenje aplikacija.

2.8.2. React Native AsyncStorage

React Native AsyncStorage je biblioteka namenjena za čuvanje podataka na uređaju. Funkcioniše na principu kolačića za web čitače. Jedna od mana je to što ne poseduje TTL što znači da će ostati u memoriji sve dok korisnik manuelno ne obriše.

2.8.3. React Native paper

React Native Paper je JavaScript biblioteka zasnovana na Google Material Design paternu. Poseduje gotove komponente za sve UI elemente.

2.8.4. React Native Image Picker

React Native Image Picker je moćna biblioteka namenjena za korišćenje nativnih UI elemenata kako bi se mogla odabrati slika iz galerije, koristila kamera za slikanje ili video.

2.9. Node.js

Node.js je JavaScript radno okruženje (engleski: runtime environment) koji pokreće JavaScript kod van web čitača. Node.js omogućava programeru da piše JavaScript kao backend jezik [5].

2.9.1. Socket.io

Socket.io je JavaScript biblioteka koja omogućava komunikaciju u realnom vremenu (engleski: real-time) između dva uređaja bilo to web čitača ili mobilnih telefona. Sastoji se od:

- Node.js servera
- JavaScript klijenta

Ovo funkcionira na taj način da klijent pokušava da ostvari konekciju sa „WebSocket“ serverom, ukoliko to ne uspe druga opcija mu postaje „HTTP polling“. „WebSocket“ je komunikacioni protokol koji pruža dupleks i nisko kašnjenje između klijenta i servera [9].

2.10. Git/GitHub

Git je sistem za verzionisanje (engleski: version control) tj. praćenje promena u kodu tokom razvoja nekog softvera. Namenjen je za koordinisani rad između programera, ali se može i koristiti za bilo šta drugo. Cilj je da ubrza proces razvoja softvera, ostvari integritet podataka i pruži distribuirani ne-linearni tok rada. Git je napravio Linus Torvalds u 2005 godini za razvoj Linux operativnog sistema [8].

GitHub je korporacija koja pruža hosting za razvoj softvera i „Version Control“ korišćenjem Git. Pruža sve mogućnosti Git sa svojim dodatnim funkcionalnostima.

3. Rešavanje problema

Integrisani web sistem je napravljen tako da korisnik ima što bolje korisničko iskustvo. To znači da se sve učitava asinhrono i da nema nepotrebnih ponovnih učitavanja stranica. To je sve moguće jer se koriste moderne web tehnologije i metode.

3.1. Web sajt

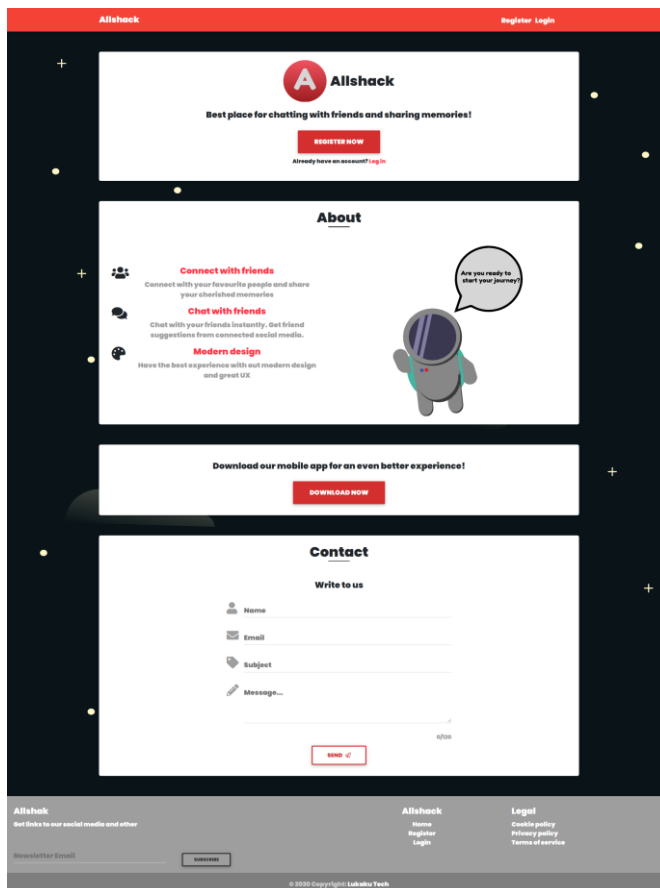
Web sajt je napravljen pomoću React biblioteke i dodatnih JavaScript biblioteka koje su ubrzale proces pravljenja aplikacije. Postoji 5 glavnih stranica. Glavna stranica koja u zavisnosti od toga da li je korisnik ulogovan ili ne prikazuje početnu stranicu ili stranicu sa listom objava. Registracija je stranica gde korisnik može da se prijavi ukoliko ne poseduje nalog, a „Login“ je stranica ukoliko poseduje gde sa svojim informacijama može da se prijavi na aplikaciju. Profil stranica gde se može gledati sopstveni profil ili profil bilo koje druge osobe na mreži. Sve ostale funkcionalnosti se prikazuju putem modala.

3.1.1. Početna stranica

Početna stranica (Slika 1) na početku ima jedan CTA koji vodi ka registraciji. Nakon toga ima kratak opis glavnih funkcionalnosti web sajta. Posle je prisutno dugme koje skida .apk fajl mobilne aplikacije. Na kraju je kontakt forma koja šalje email administratoru web sajta o bilo kakvim pitanjima u vezi aplikacije.

Primer dinamičnosti početne stranice u zavisnosti od toga da li je korisnik ulogovan ili ne.

```
render() {  
  const {authenticated} = this.context;  
  document.title = authenticated ? 'Allshack' : 'Welcome to Allshack';  
  
  return (  
    <MDBContainer className={authenticated ? null : 'text-center pt-5'}>  
      {authenticated ?  
        <PostContainer />  
        :  
        <>  
          <Banner/>  
          <About/>  
          <MobileApp/>  
          <Contact/>  
        </>  
      }  
    </MDBContainer>  
  );  
}
```



Slika 1 Početna stranica

3.1.2. Login/Registracija

Login i Registracija stranice (Slika 2 i 3) poseduju validaciju za svako polje i ukoliko validacija prođe korisnik dobija indikator u obliku „spinner“. Ako je registracija uspešna, korisnik dobija potvrdu da mu je poslat verifikacioni email na njegovu adresu (Slika 4).

Allshack Register Login

Register

First Name Last Name

@ Email Username

Your password Confirm password

REGISTER ✓

Already have an account? Log in!

Allshack Get links to our social media and other

Newsletter Email SUBSCRIBE

Allshack Home Register Login

Legal Cookie policy Privacy policy Terms of service

© 2020 Copyright: Lukaku Tech

Slika 2 Registracija

Allshack Register Login

Login

@ Email

Your password

Forgot Password?

LOGIN

Don't have an account? Sign up!

Allshack Get links to our social media and other

Newsletter Email SUBSCRIBE

Allshack Home Register Login

Legal Cookie policy Privacy policy Terms of service

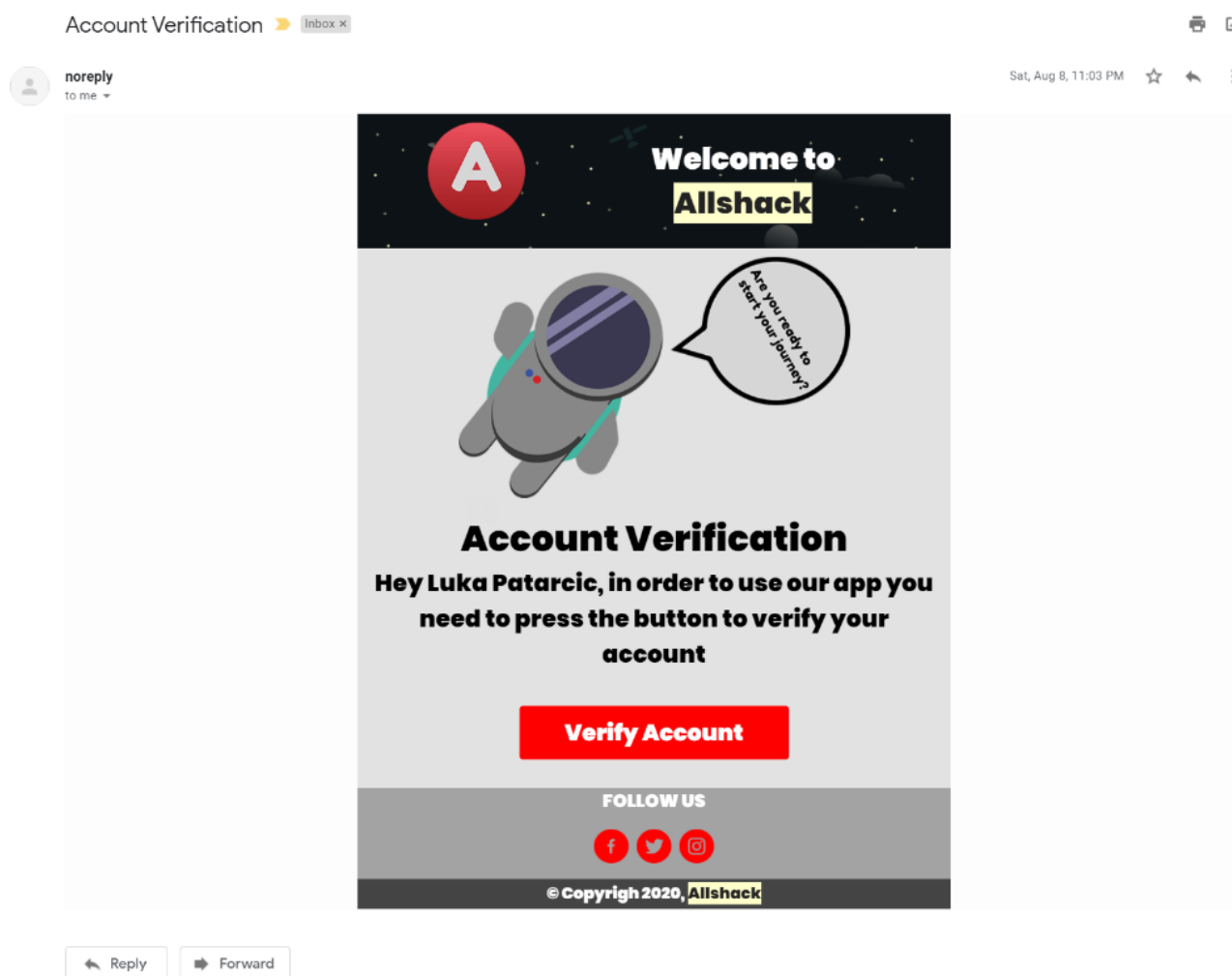
© 2020 Copyright: Lukaku Tech

Slika 3 Login

A verification link has been sent to your email account

Slika 4 Uspešna registracija

Nakon uspešne registracije korisniku je poslata email adresa (Slika 5). Stilizovana je pomoću HTML i CSS i klikom na dugme „Verify Account“ korisnik je preusmeren na „Login“ stranicu web sajta ili ako je mobilna aplikacija instalirana preusmeren je tamo pomoću „Deep Linking“.

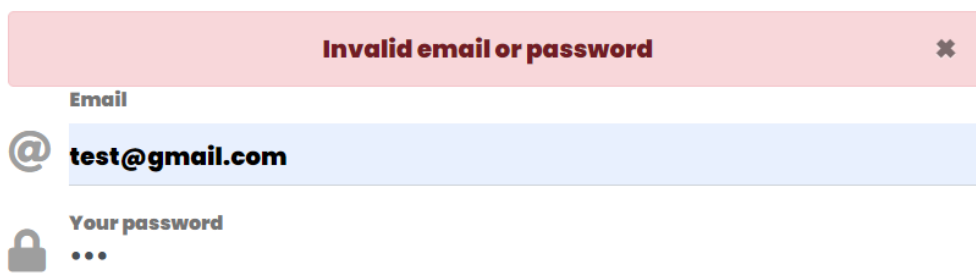


Slika 5 Email

Ukoliko korisnik ne uspe da se uloguje na nalog tj. uneo je pogrešan email ili šifru dobiće poruku o tome (Slika 6). Ali ako se uspešno uloguje biće preusmeren na glavnu stranicu. Korisnik dobije kao rezultat jedan string (JWT) koji se čuva kao kolačić pod imenom „access-token“ koji će kasnije služiti za autorizaciju.

Primer koda kako se čuva kolačić.

```
cookie.save('access-token', response.token, {maxAge: COOKIE_TTL, secure: true});
```



The image shows a login interface with a red error banner at the top that reads "Invalid email or password" with a close icon. Below the banner, there are two input fields. The first is labeled "Email" and contains the text "test@gmail.com" preceded by an @ symbol icon. The second is labeled "Your password" and contains three dots, indicating a password field. The entire form is set against a light gray background.

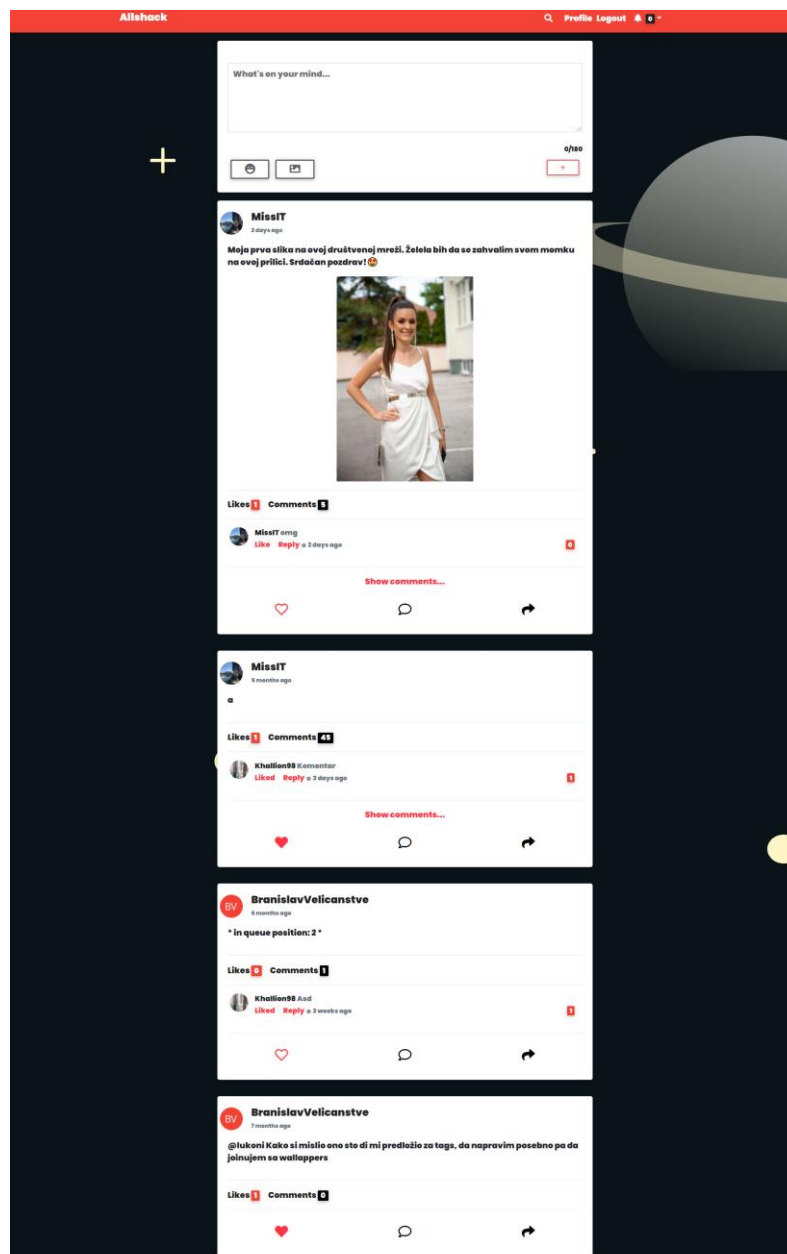
Slika 6 Neuspešan login

3.1.3. Objave

Stranica sa objavama prikazuje sve objave ljudi koje pratite. Sve objave nakon prvih 10 se dinamički učitavaju tako što se „scroll-uje“ do dna stranice. Tako funkcioniše sve dok postoji neki post, nakon toga se prikaže poruka da nema više objava. Na početku se nalazi kartica na kojoj možete postaviti objavu sa tekstem i opciono maksimalno 5 slika (Slika 7).

Prikazivanje objava u zavisnosti od profila i od kojeg rednog broja započinje pretraga uz pomoć „fetchJson“ metode koja je nadogradnja na „fetch“ JS metodu.

```
export const getPosts = (offset, profile) => {  
  const id = profile ? `&profile=${profile}` : ''  
  return fetchJson(`/post?offset=${offset}${id}`, {}, true)  
};
```



Slika 7 Stranica sa objavama

3.1.3.1. Individualna objava

Svaka objava se sastoji od slike (ukoliko nema koriste se inicijali), korisničkog imena, vremena objavljivanja, teksta i opciono slika. Dole su prisutni brojevi lajkova i komentara koji poseduje ta objava i najnoviji komentar ako ima (Slika 8).



Branislav Velicanstve

7 months ago

@lukoni Kako si mislio ono sto di mi predložio za tags, da napravim posebno pa da joinujem sa wallappers

Likes **1** **Comments** **2**



Khallion98 Sa najboljim programskim jezikom JavaScript

Like **Reply** 6 seconds ago

0

Show comments...



Slika 8 „Post“

3.1.3.2. Objave sa slikom

Objave sa slikama su urađene tako što ukoliko su postavljene dve ili više slika prikazuju se u vidu „carousel“. Gde program automatski lista kroz sve slike ili korisnik samostalno može da vidi sve slike klikom na strelice ili „swipe“ na mobilnim uređajima (Slika 9).



Khallion98

2 weeks ago

Asd



Likes **0** **Comments** **0**



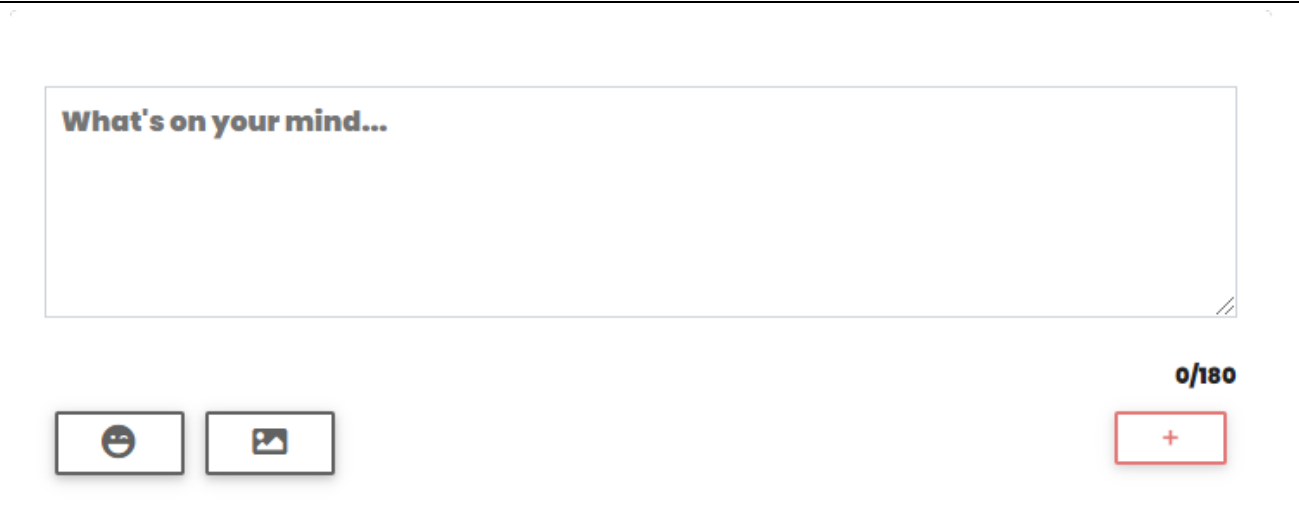
Slika 9 Objava sa slikom

3.1.3.3. Dodavanje objava

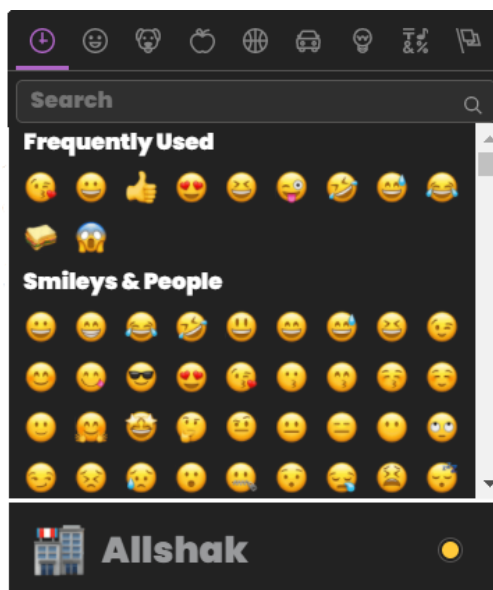
Za dodavanje objave potrebno je uneti tekst do maksimalno 180 karaktera. Dole desno je prisutan indikator koji prikazuje koliko karaktera je uneto. Dugme sa ikonicom + je onemogućeno sve dok se ne unese neki tekst ili ukoliko je prekoračen broj karaktera. Sa leve strane dole se nalaze dva dugmeta, jedno je za „emojis“ a drugo je za dodavanje slika (Slika 10 i 11).

Primer koda za slanje objava na API.

```
sendPostHandler(text, images) {  
  this.setState({loading: true})  
  sendPost(text, images)  
    .then(response => {  
      this.setState({loading: false});  
      this.postCreateFormRef.current.resetText();  
      if (this.props.onlyMe) {  
        this.props.onSendPostHandler(response.post)  
      }  
  
      toastr.success('Post added to timeline!')  
    })  
    .catch(err => {err.response.json().then(err => {  
      this.setState({loading: false});  
      toastr.error(err.error)  
    }}});  
}
```



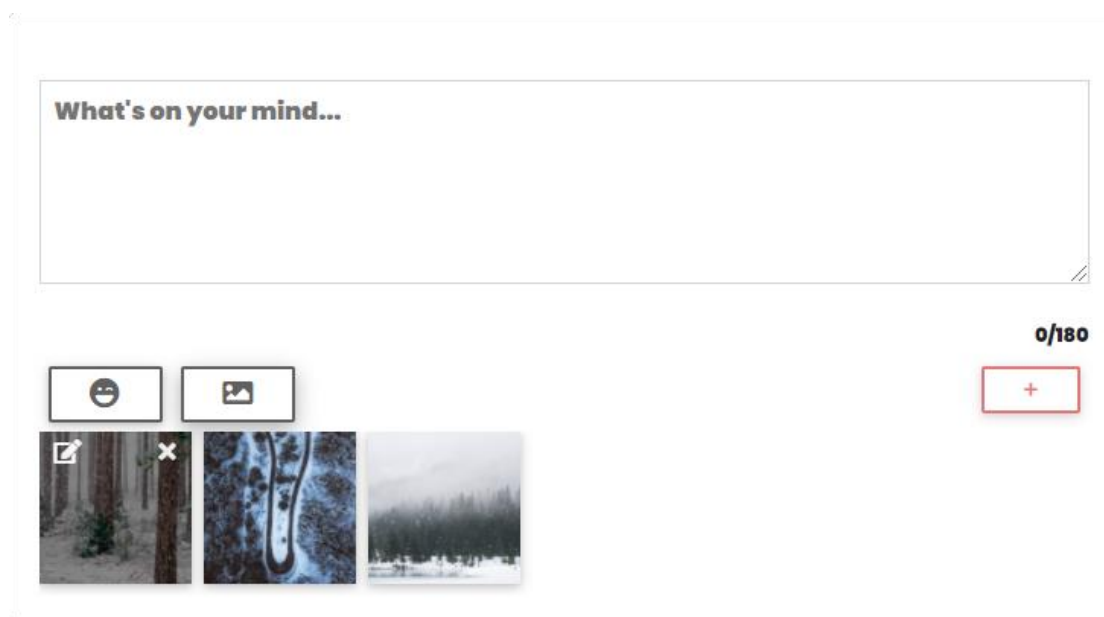
Slika 10 Dodavanje objave



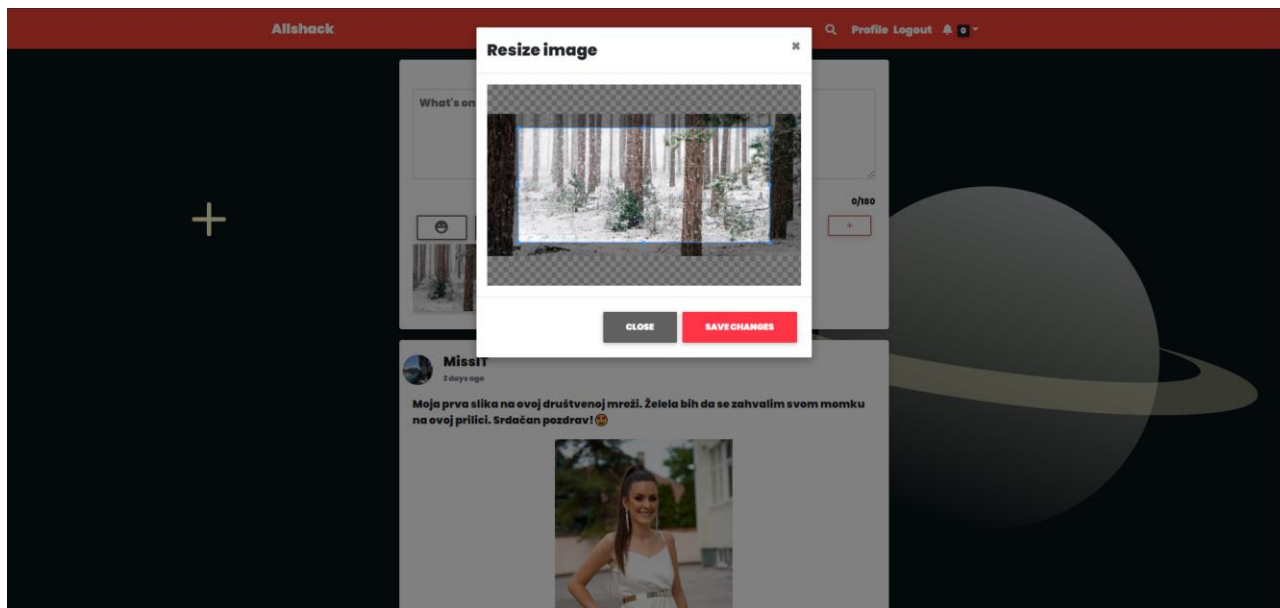
Slika 11 „Emoji picker“

3.1.3.4. Objava uz slike

Slike se dodaju klikom na dugme sa ikonicom slike. Formati koji su dozvoljeni su jpg, png i gif. Maksimalno se može dodati 5 slika. Ukoliko dođe do neke greške dobija se poruka o tome. Nakon uspešnog odabira slika prikazuju se u listi gde se mogu obrisati ili izmeniti klikom na odgovarajuće dugme (Slika 12). Klikom na „edit“ otvara se modal gde se može promeniti veličina slike (Slika 13). Kada je korisnik zadovoljan, klikom na dugme + slike se kompresuju korišćenjem JavaScript-a i tek nakon toga se šalju na server gde se validira da je u pitanju prava slika.



Slika 12 Lista slika



Slika 13 Promena veličine slike

3.1.3.5. Lajk objave

Klikom na lajk dugme, menja se ikonica iz punog srca u prazno ili obrnuto u zavisnosti od toga da li ste već lajkovali objavu ili ne (Slika 14).

Primer koda koji je korišćen za prikazivanje lajk dugmeta.

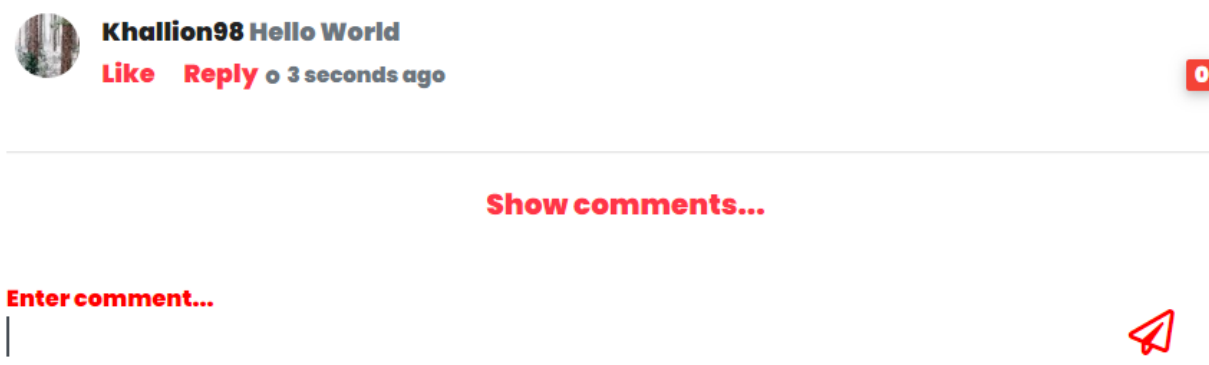
```
render() {
  const {liked,onHandlePostLike,id} = this.props;
  return (
    <MDBTooltip>
      <MDBBtn color={'white'} block={true} style={{boxShadow: 'none'}} onClick={()
=> onHandlePostLike(id,liked)}>
        <MDBIcon className={'text-danger'} far={!liked} icon={'heart'}
size={'2x'}/><br/>
      </MDBBtn>
      <div>{liked ? 'Liked':'Like'}</div>
    </MDBTooltip>
  );
}
```



Slika 14 Lajk

3.1.3.6. Komentar objave

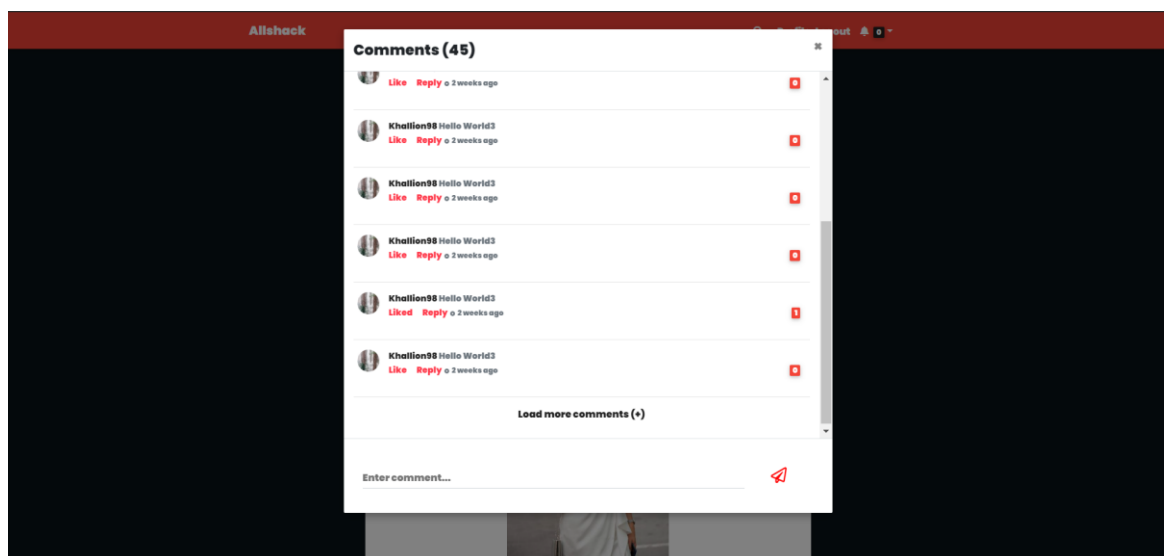
Klikom na dugme komentar, pojavi se mesto gde korisnik može da unese svoj komentar. Nakon unošenja komentara korisnik dobija vizuelni pokazatelj da se komentar dodao tako što se menja broj komentara i njegov komentar je sada aktuelan kao najnoviji (Slika 15).



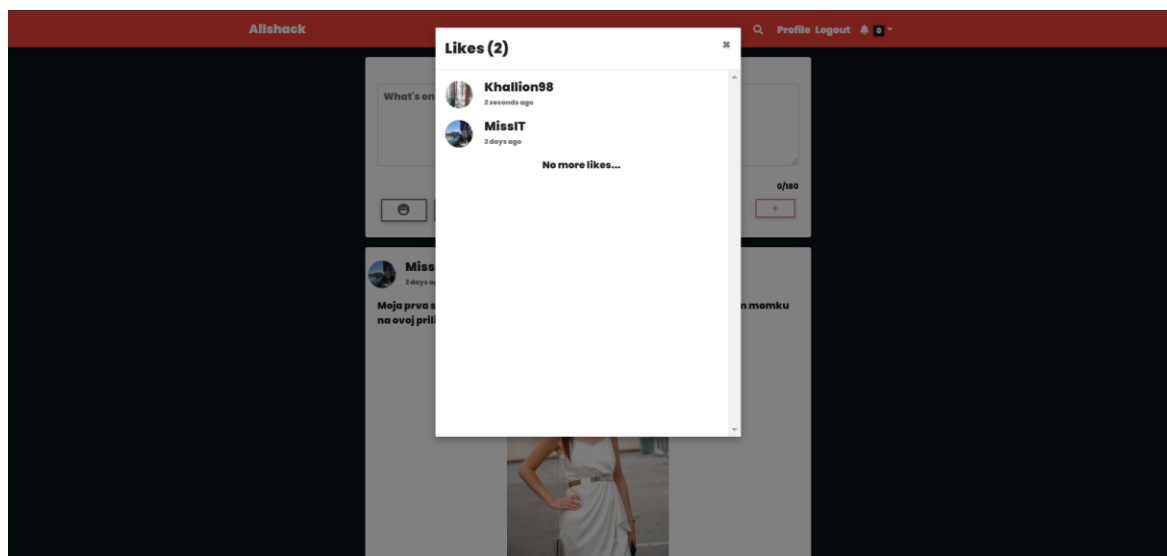
Slika 15 Dodavanje komentara

3.1.3.7. Lista lajkova i komentara

Ukoliko korisnici žele da vide sve korisnike koji su komentarisali ili lajkovali objavu mogu to da urade klikom na „Likes“ i „Comments“ gde će se otvoriti modal sa listom svih korisnika koji su lajkovali ili listom svih komentara vezanih za tu objavu. Takođe se sve dinamički učitava sa setom od po 10 komentara ili lajkova (Slika 16 i 17).



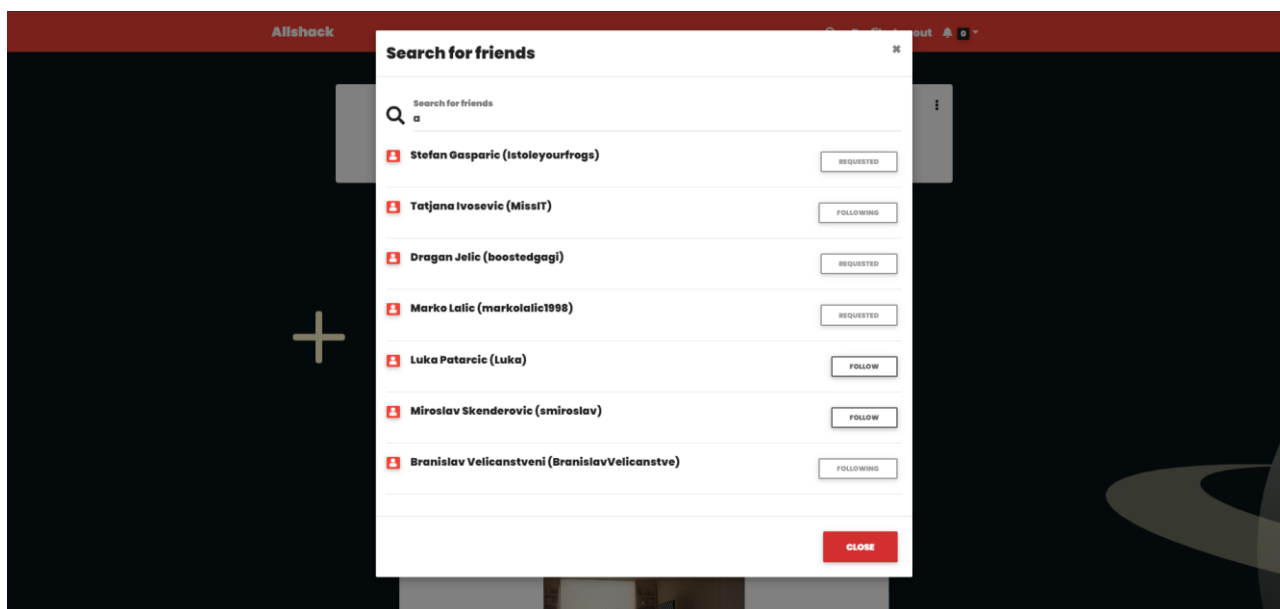
Slika 16 Lista komentara



Slika 17 Lista lajkova

3.1.4. Pretraga

Pretraga funkcioniše na taj način da se otvara modal i korisnik unosi ime osobe koju želi da pronade. Na svaku promenu vrednosti polja se pretražuje u bazi. Da bismo zaustavili da svaki put šalje zahtev koristi se „debounce“ što samo nakon prestanka kucanja i kratkog intervala šalje zahtev. Kada se dobije rezultat sa leve strane je prikazano ime i prezime koje klikom vodi do profila tog korisnika a sa desne strane status prijateljstva. Ukoliko nije zapraćen korisnik, klikom na dugme se šalje zahtev tom korisniku (Slika 18).

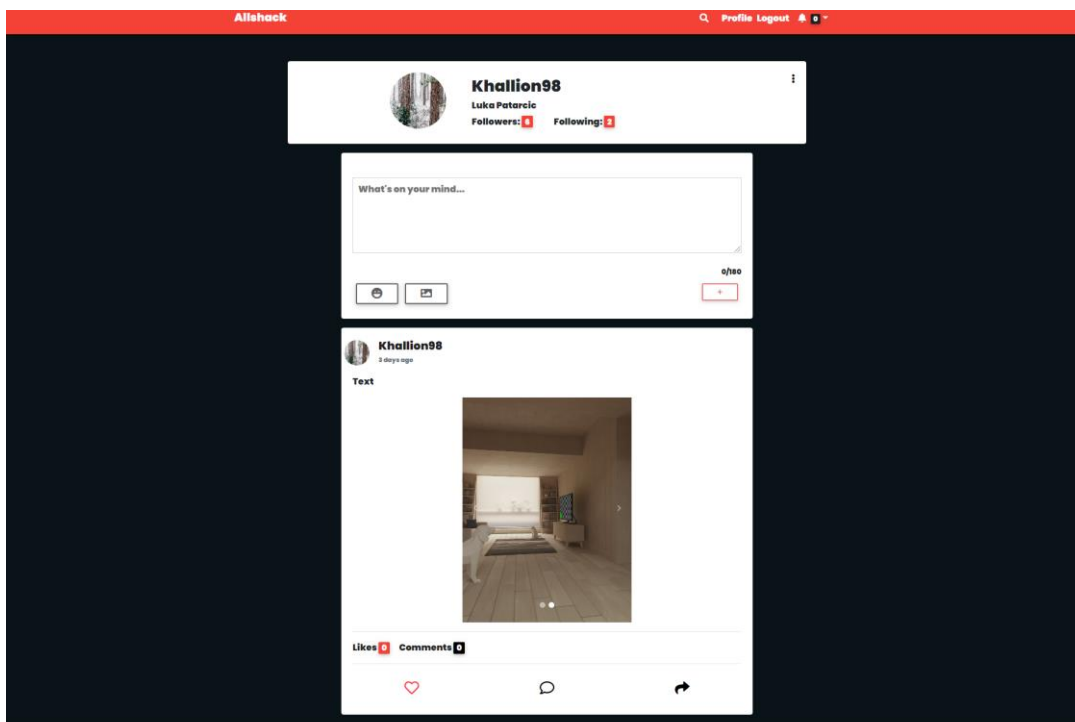


Slika 18 Pretraga

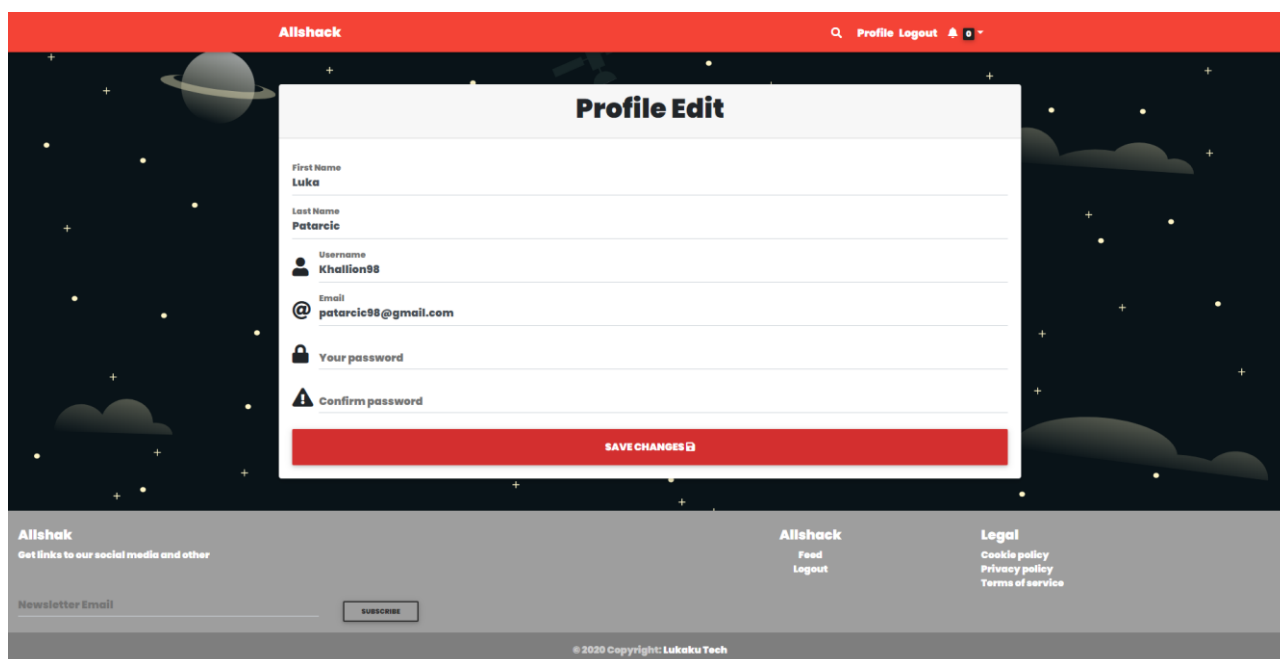
3.1.5. Profil

Profil sadrži glavne informacije o osobi čiji je profil (Slika 19). Pored toga, ako je to profil korisnika koji je i ulogovan prikazuje se deo za dodavanje objave. Klikom na profilnu sliku moguće je promeniti korisničku profilnu sliku. Klikom na ikonicu sa tri vertikalne tačke se odlazi na stranicu gde se mogu promeniti osnovne

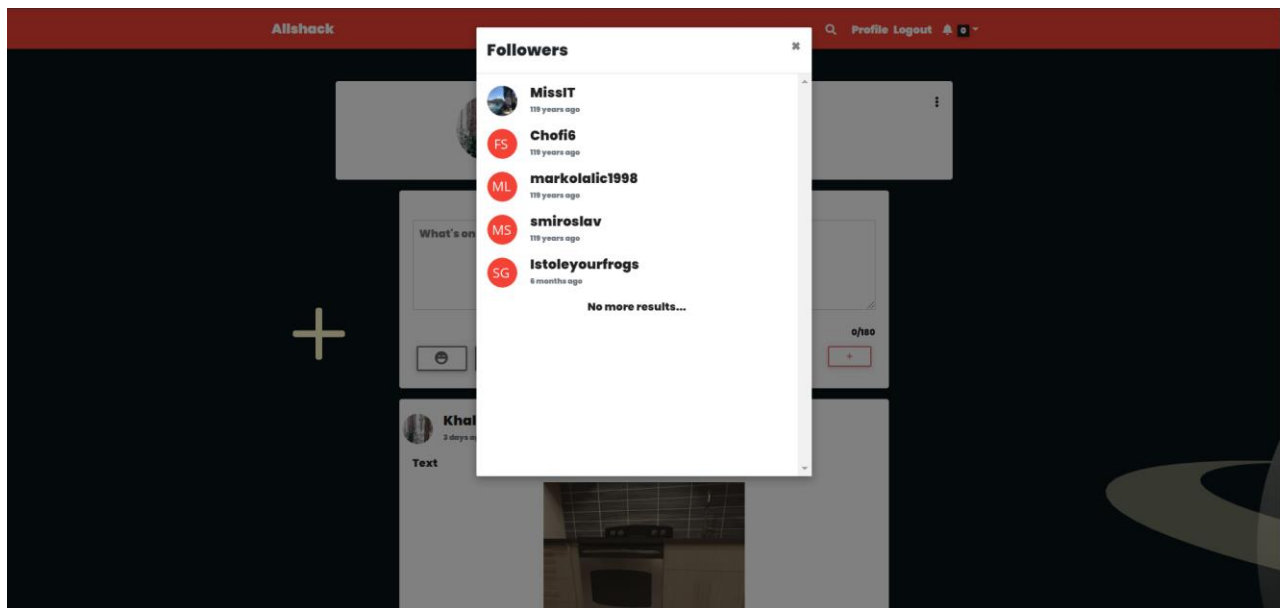
informacije o korisniku kao i ime, email lozinka itd (Slika 20). Lista ljudi koje Vas prate i koje Vi pratite mogu se videti klikom na „Followers“ iii „Following“ gde se otvara modal sa listom korisnika (Slika 21).



Slika 19 Profil korisnika



Slika 20 Izmena profila



Slika 21 Lista pratioca

3.2. Mobilna Aplikacija

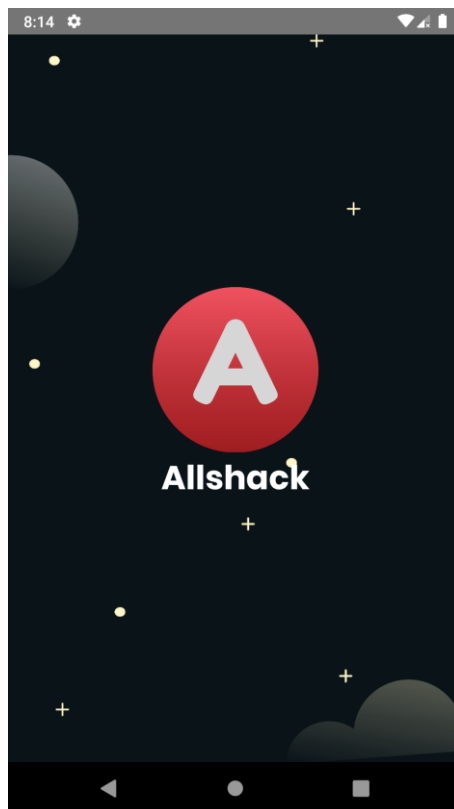
Mobilna aplikacija je napravljena uz pomoć React Native biblioteke. Pošto je web deo napisan u React bilo je potrebno većinski napraviti dizajn i dodatne funkcionalnosti koje mobilna aplikacija pruža. Sastoji se od 2 glavna dela. Prvi deo je kada korisnik nije autentifikovan što su Login i Registracija stranice. Drugi deo predstavlja celu aplikaciju kada se korisnik prijavi. Na dnu se nalazi „Bottom Tabs Navigation“ koji predstavlja navigaciju između glavnih delova aplikacije što su „Feed“, Pretraga, Notifikacije i Profil.

3.2.1. „Splash Screen“

„Splash Screen“ predstavlja deo gde se autentifikuje korisnik tako što šalje „token“ iz AsyncStorage ka serveru. Ukoliko nije ulogovan ili autentifikovan preusmeren je na stranicu login, ukoliko jeste ima pristup celoj aplikaciji (Slika 22).

Primer koda gde se proverava da li je korisnikov JWT validan

```
if(data.error === 'Authentication Required' || data.error === 'Invalid JWT Token') {  
    this.setState({isAuth:false})  
} else {  
    AsyncStorage.setItem("id",data.id.toString())  
    this.setState({isAuth:true})  
}
```

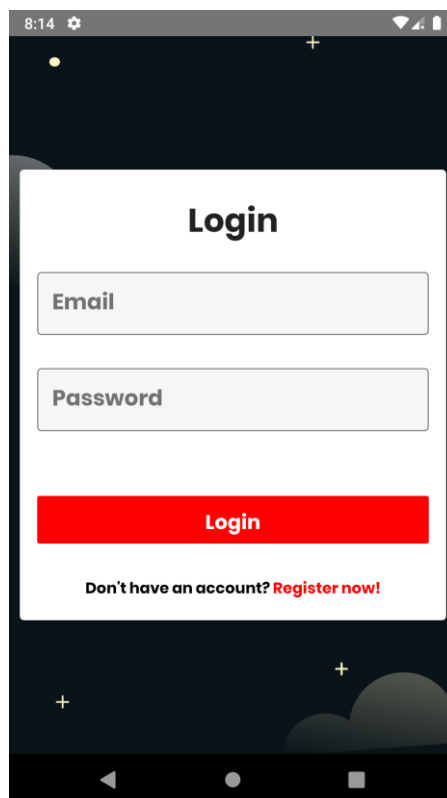


Slika 22 „Splash Screen“

3.2.2. Login/Registracija

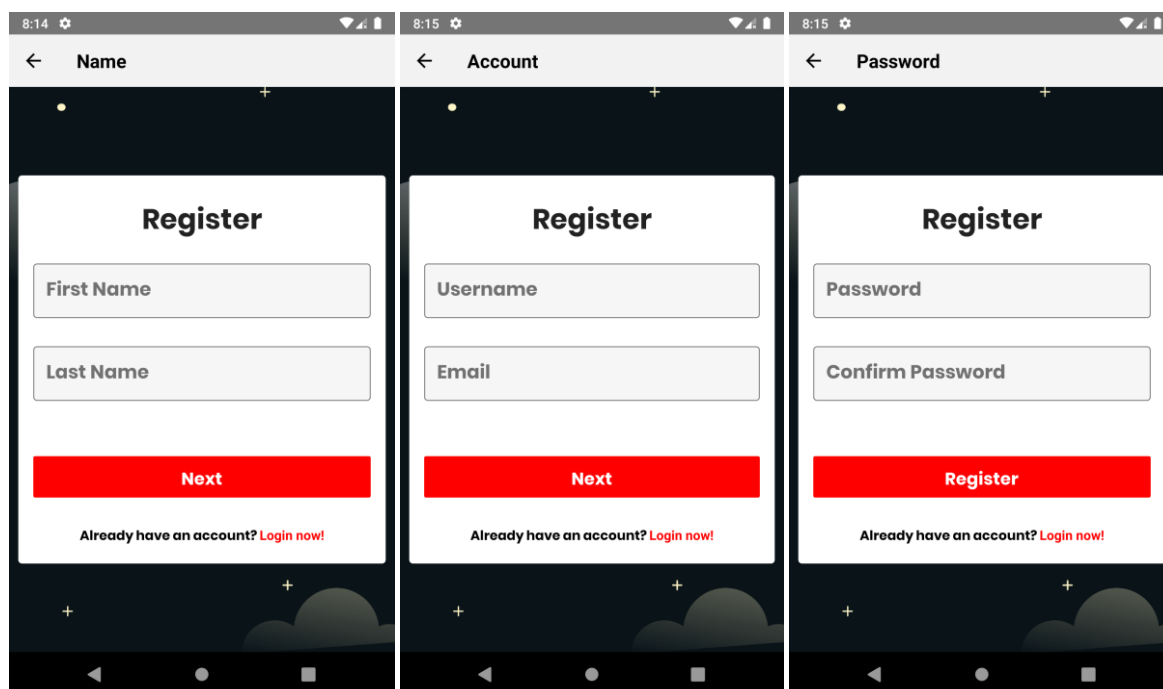
Login stranica potpuno identično funkcioniše kao na web sajtu samo što je dizajn optimizovan za mobilne uređaje (Slika 23). Registracija je podeljena na 3 ekrana. Na prvom ekranu se upisuje ime i prezime, klikom na „Next“ prelazi se na drugi ekran gde se upisuje korisničko ime i email, nakon klika na „Next“ proverava se u bazi da li već postoji korisničko ime i email. Na kraju se upisuje šifra i klikom na dugme „Register“ šalje se verifikacioni email (Slika 24).

```
AsyncStorage.setItem('access-token', data.token);  
AsyncStorage.setItem("id", data.user.id.toString())  
this.context.setIsAuth();
```



A mobile app login screen with a dark background and a white login form. The form has two input fields: 'Email' and 'Password'. Below the fields is a red 'Login' button. At the bottom of the form, there is a link: 'Don't have an account? Register now!'. The screen shows a status bar at the top with the time 8:14 and a navigation bar at the bottom with three icons.

Slika 23 Login



Three mobile app registration screen mockups shown side-by-side. Each screen has a dark background and a white registration form. The first screen is titled 'Name' and has 'First Name' and 'Last Name' input fields, a red 'Next' button, and a link 'Already have an account? Login now!'. The second screen is titled 'Account' and has 'Username' and 'Email' input fields, a red 'Next' button, and a link 'Already have an account? Login now!'. The third screen is titled 'Password' and has 'Password' and 'Confirm Password' input fields, a red 'Register' button, and a link 'Already have an account? Login now!'. Each screen shows a status bar at the top with the time (8:14, 8:15, 8:15) and a navigation bar at the bottom with three icons.

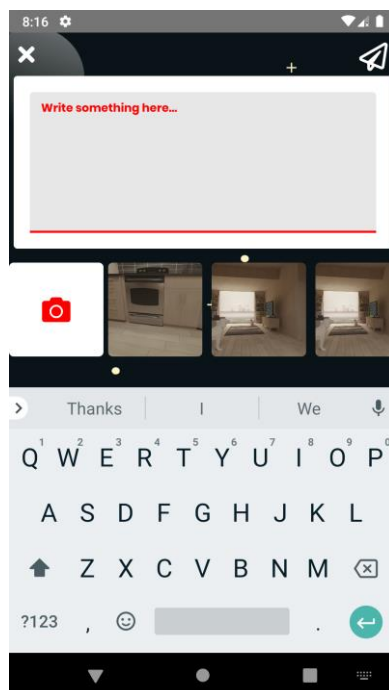
Slika 24 Registracija

3.2.3. Početna stranica

Početna stranica funkcioniše identično kao i na web sajtu samo ponovo je optimizovan za mobilne uređaje (Slika 25). Razlikuje se to da se klikom na komentar preusmerava na novu stranu gde se pokazuje lista komentara. Takođe prisutan je FAB koji kada se stisne preusmeri se na stranicu za dodavanje nove objave. Tu je prisutno mesto za dodavanje teksta objave kao i lista slika sa telefona i dugme koje otvara kameru za slikanje i dugme za otvaranje foto albuma telefona (Slika 26).



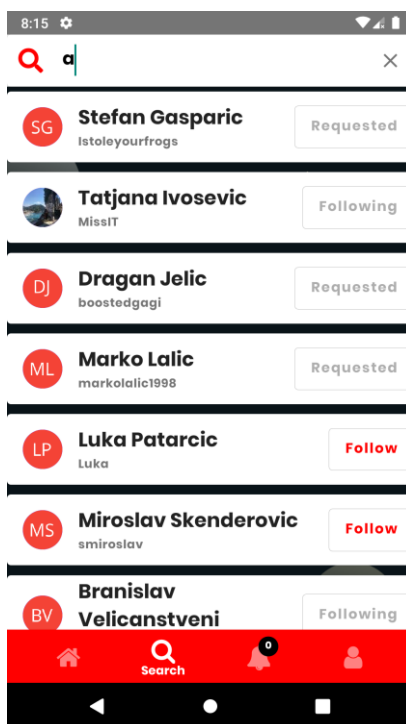
Slika 25 Početna stranica



Slika 26 Postavljanje objave

3.2.4. Pretraga

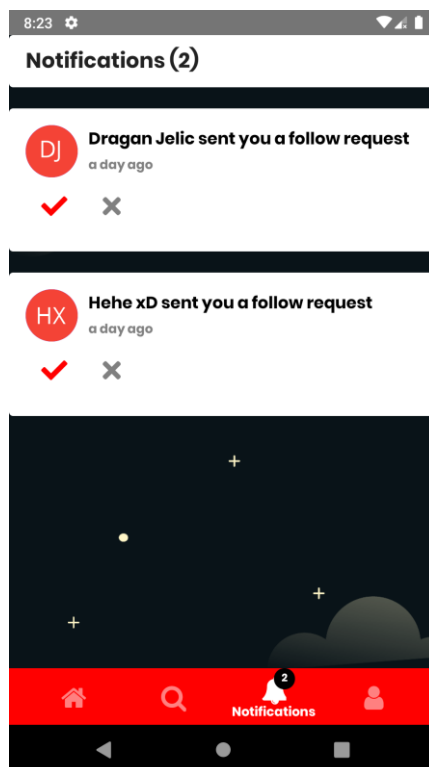
Pretraga poseduje isti princip rada kao i web aplikacija, sem naravno redizajna koji je prilagođen mobilnim uređajima. Kada se klikne na „Search“ automatski se otvara tastatura kako bi korisnik mogao odmah da pretražuje (Slika 27).



Slika 27 Pretraga

3.2.5. Notifikacije

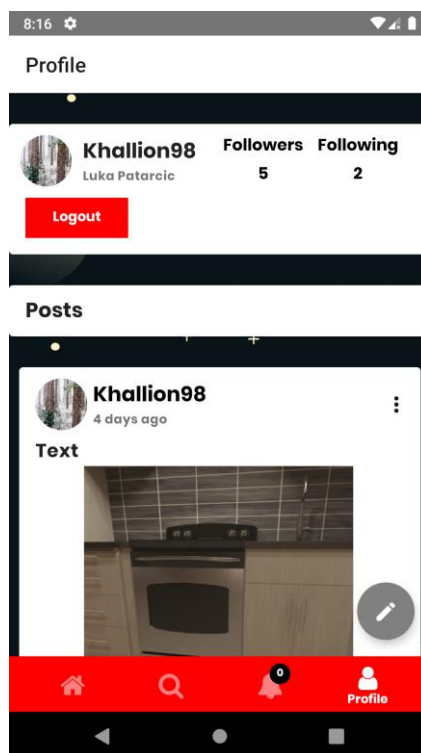
Ekran sa notifikacijama predstavlja listu svih zahteva koji je korisnik dobio. On može da prihvati zahtev za praćenje ili odbije. Ukoliko prihvati, korisnik koji je poslao zahtev će videti sve objave na glavnoj stranici od osobe koju je zapratio (Slika 28).



Slika 28 Notifikacije

3.2.6. Profil

Najveća promena u odnosu na web sajt je to što je moguće obrisati objavu i otpratiti osobu. Sve ostale funkcionalnosti ostaju iste sa redizajnom (Slika 29).



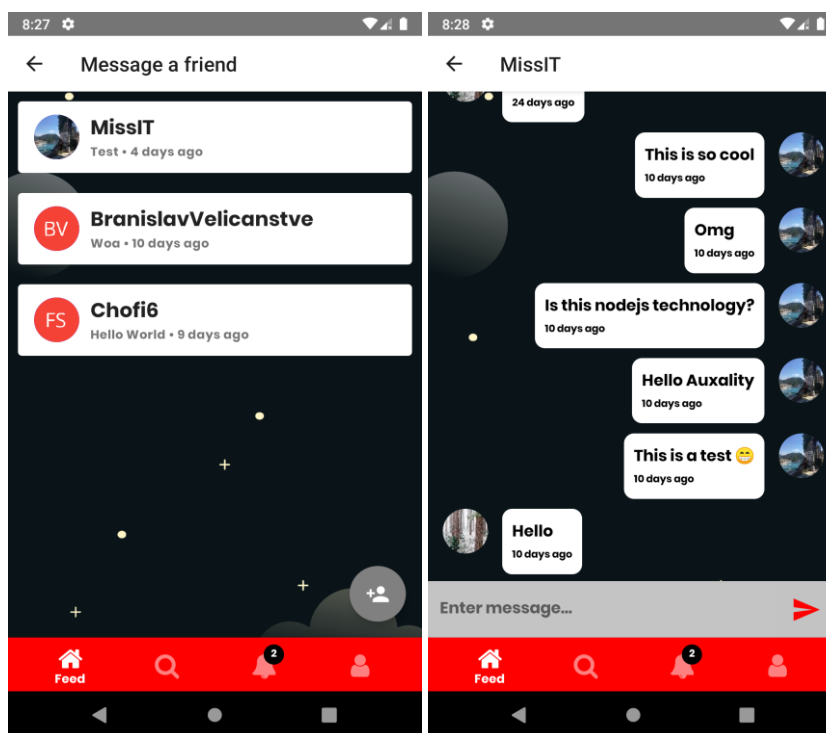
Slika 29 Profil

3.2.7. Čet

Na glavnoj stranici u gornjem desnom uglu prisutno je dugme koje kada se stisne vodi na stranu gde je lista svih osoba sa kojima ste imali neki „Chat“. U donjem desnom uglu se nalazi FAB koje kada se stisne vodi ka novom korisniku kojem želite da pošaljete poruku. Na strani sa porukama drugog korisnika, sa leve strane, nalaze se Vaše poruke, a sa desne strane, korisnika sa kojim četujete. Pored teksta poruke nalazi se podatak o tome pre koliko vremena je poslata poruka, a sa leve tj. desne strane se nalazi slika korisnika. Na donjem delu korisnik može da unese poruku koju želi da pošalje (Slika 30).

Primer koda kako klijent prima poruke i prikazuje ih na ekranu.

```
ws.io.on("message", (message) => {  
  this.setState({  
    messages: [message, ...this.state.messages],  
    sendingMessage: false,  
  })  
})
```



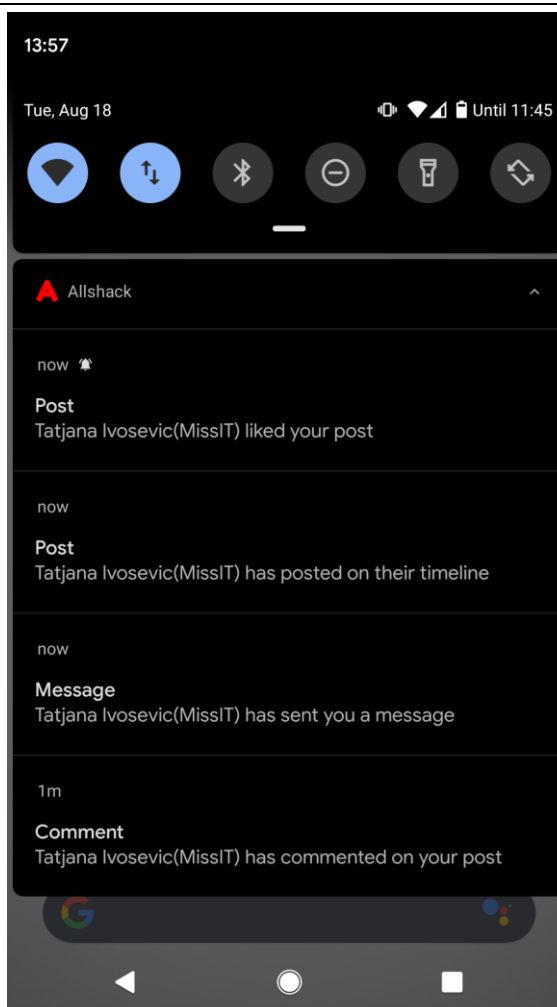
Slika 30 Čet stranice

3.2.8. „Push“ notifikacije

Nativne notifikacije funkcionišu tako što kada se korisnik uloguje na mobilni uređaj, njegov ključ za notifikacije se sačuva u bazu. Bez obzira na koliko uređaja je korisnik ulogovan na svaki će mu stići notifikacija. Notifikacije šalje API na Firebase server koji posle šalje notifikacije na sve uređaje koji su nabrojani u zahtevu. Događaji za koje se šalje notifikacija su: nova objava, lajk objave, komentarisanje na objavu, lajk komentara, poruka od korisnika, zahtev za praćenje itd. (Slika 31).

Primer koda kako se uređaj povezuje na FCM i čuva ključ na uređaju.

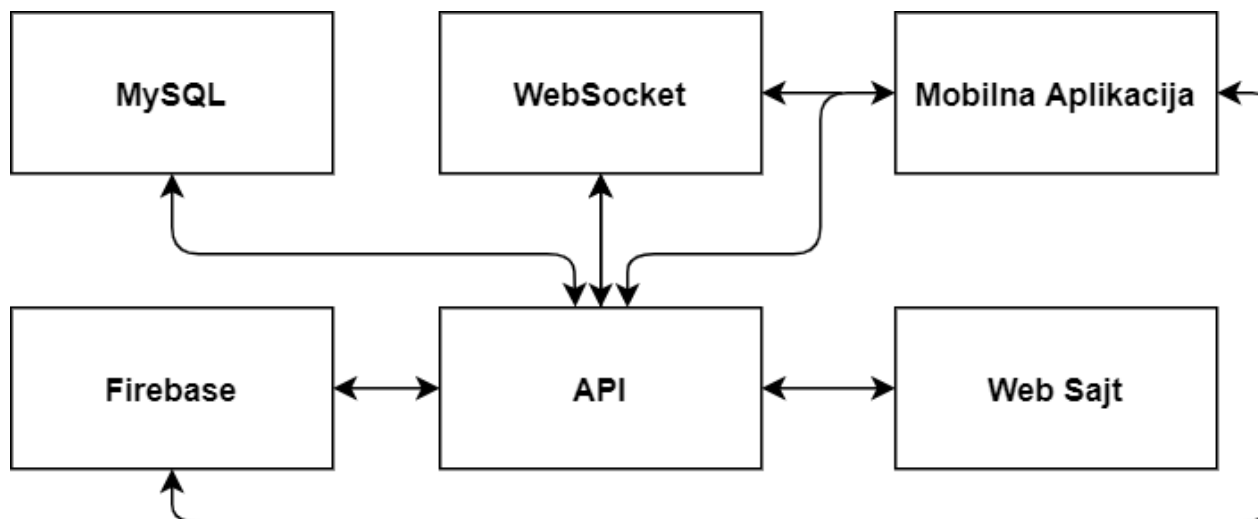
```
messaging().registerDeviceForRemoteMessages();  
messaging()  
  .getToken()  
  .then(token => {  
    AsyncStorage.setItem('notification-key', token);  
  });
```



Slika 31 „Push“ notifikacije

3.3. Šema sistema

Prikaz sistema kako funkcioniše i koji su delovi sistema povezani (Slika 32).

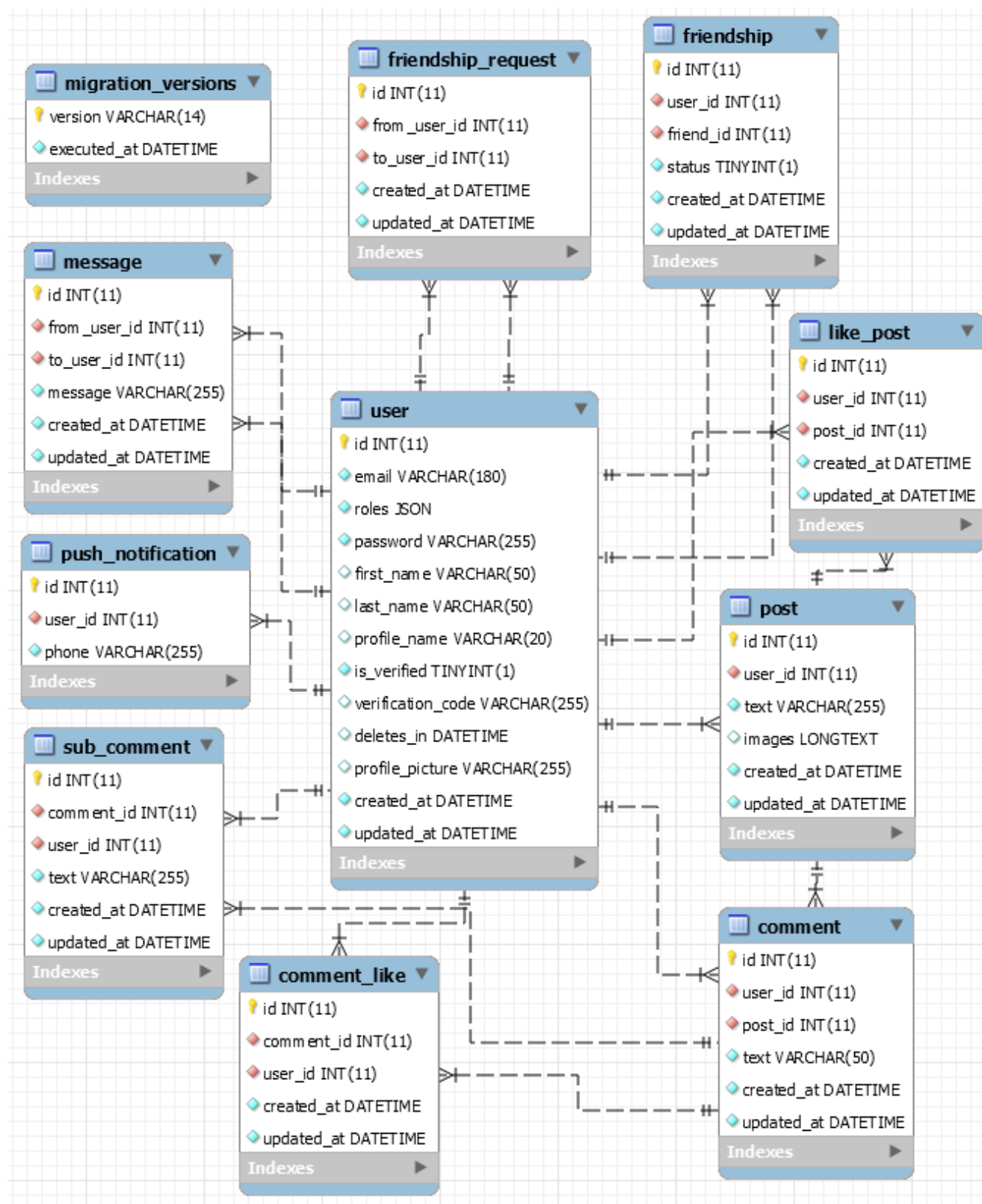


Slika 32 Šema sistema

3.4. Baza podataka

Baza podataka je urađena pomoću MySQL. Sastoji se od 10 tabela (Slika 33).

- User koja je glavna tabela i koja se nadovezuje na sve ostale tabele.
- Post koja sadrži sve informacije o objavama
- Post-Like među tabela za čuvanje koji korisnik je lajkovo koju objavu
- Comment koja sadrži sve komentare za jedan post
- Comment-Like među-tabela za čuvanje podatka koji korisnik je lajkovao koji komentar
- SubComment sadrži sve podkomentare jednog komentara
- Message sadrži sve poruke između dva korisnika
- Push Notifications sadrži ključ ka mobilnom uređaju ulogovanog korisnika za notifikacije
- Friendship sadrži sve korisnike koji prate druge korisnike
- Friendship Request sadrži sve zahteve za praćenje od jednog korisnika ka drugom
- Migration Version sadrži sve migracije koje je Symfony izvršio na bazu



Slika 33 Baza podataka

3.5. Testiranje u Realnom okruženju

Nakon završetka razvoja softvera potrebno je bilo postaviti na internet u vidu produkcione verzije. Svi domeni rade na serverima sa Apache 2.2 i SSL enkripcijom i .htaccess fajlom koji podešava preusmeravanje na HTTPS, keširanje i redirekcije za rute.

3.5.1. API produkcija

Za API deo postavljen je kod na domen <https://api.allshack.lukaku.tech/>. Kako bi proradio kod potrebno je bilo pokrenuti komandu „composer install“ pomoću koje se izgeneriše „autoload“ i instaliraju se sve biblioteke potrebne za projekat. Na kraju je izmenjen .env fajl gde su promenjeni parametri baze podataka, SMTP servera i okruženje na produkciju.

3.5.2. Web sajt produkcija

Za web deo potrebno je bilo pokrenuti komandu „yarn build“ sa kojom React pravi optimizovanu produkcionu verziju aplikacije. Sve potrebne fajlove stavi u „build“ folder koji se samo postavi na internet i besprekorno radi. Projekat je postavljen na domen <https://allshack.lukaku.tech/>.

3.5.3. Mobilna produkcija

Za mobilnu aplikaciju je malo kompleksnija produkciona verzija. Prvobitno je bilo potrebno napraviti „keytool“ sa sledećom komandom

```
Keytool -genkey -v -keystore your_key_name.keystore -alias you_key_alias -keyalg RSA -  
keysize 2048 -validity 10000
```

Posle je potrebno pokrenuti komandu

```
./gradlew assembleRelease
```

Gotov produkcionni .apk fajl se nalazi u „build“ folderu koji je postavljen na internet na domenu <https://downloads.allshack.lukaku.tech/Allshack.apk>

3.5.4. Node.js produkcija

Za Node.js cPanel ima mogućnost postavljanja Node.js aplikacija. One funkcionišu na malo specifičan način, koriste Apache Passanger za pokretanje JavaScript-a. Domen na koji je postavljen „web socket“ Node.js projekat je <https://ws.allshack.lukaku.tech/>

4. Zaključak

Čitav proces od razvoja do implementacije projekta je bilo zadovoljavajući i izazovan. Svi ciljevi su uspešno postignuti i ostvareni. Napravljen je jedan uniformni sistem koji funkcioniše na više operativnih sistema i uređaja. Neverovatno sam puno naučio i napredovao u tom periodu i nastaviću da radim na sebi, svojim veštinama i proširiću još više znanje u sferi web programiranja. Ekspanzija ovog projekta je naravno lako moguća i neke od bitnijih funkcionalnosti koje bih dodao su:

- Poboljšani čet sistem
- Dodavanje video snimaka u objave
- Administrativni deo za nadzor

Literatura

- [1] <https://symfony.com/>
- [2] <https://symfonycasts.com/>
- [3] <https://reactjs.org/>
- [4] <https://reactnative.dev/>
- [5] <https://nodejs.org/en/>
- [6] <https://firebase.google.com/>
- [7] <https://people.vts.su.ac.rs/~chole/>
- [8] <https://github.com/>
- [9] <https://socket.io/>
- [10] Francois Zaninotto „The Definitive Guide to Symfony“, 2007
- [11] Adam Freeman „Pro React 16“, 20019
- [12] Devin Abbott „Fullstack React Native: Create beautiful mobile apps with javaScript and React Native“, 2019

5. Prilozi

5.1. Elektronski materijali

DVD koji sadrži završni rad u *docx* ili *doc* i *pdf* formatu i ostale materijale u elektronskoj formi.