# IZRADA INTERNETSKOG BLOGA

**Masnić, Ivan**

**SVEUČILIŠTE U SPLITU**
**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**
Preddiplomski stručni studij Informacijska tehnologija

**IVAN MASNIĆ**

# Z A V R Š N I  R A D

# IZRADA INTERNETSKOG BLOGA

Split, rujan 2023.

**SVEUČILIŠTE U SPLITU**
**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijska tehnologija

**Predmet:** Baze podataka

# Z A V R Š N I  R A D

**Kandidat:** Ivan Masnić

**Naslov rada:** Izrada internetskog bloga

**Mentor:** mr.sc. Ivica Ružić, viši predavač

Split, rujan 2023.

# Sadržaj

# Sažetak

U ovom završnom radu istražen je proces izgradnje blog web aplikacije nazvane "SocialApp" koristeći najpopularnije i moderne tehnologije: Express.js za izgradnju poslužitelja i React.js za izgradnju klijentske aplikacije. Cilj je ovog rada prikazati izrađivanje aplikacije koja korisnicima omogućuje objavljivanje svojih misli, dijeljenje objava drugim korisnicima, komentiranje objava i instantnu komunikaciju s drugim korisnicima.

Rad analizira osnovne koncepte Javascript radnog okvira Express.js za izgradnju poslužitelja za prihvaćanje i obradu podataka i Javascript biblioteke React.js za izgradnju modernih i responzivnih korisničkih sučelja.

U radu su istražene i opisane osnovne funkcionalnosti vezane za blog stranice poput CRUD operacije objava, dodavanja komentara, pretraživanja korisnika i sučelja za komunikaciju s drugim korisnicima.


Ključne riječi: blog, Express.js, Javascript, React.js



# Summary

## Creating an Internet blog website

This work explores the process of creating a blog web application named "SocialApp" using the most popular and modern technologies: Express.js for creating the web server and React.js for creating the client side application. The goal of this work is to demonstrate the building of an app which enables the user to post their thoughts, sharing their posts with other users, commenting posts and instant communication with other users.

This work analyzes the basic concepts of the Javascript framework Express.js for creating a web server to handle and process data and the Javascript library React.js for creating modern and responsive user interfaces.

Basic functionalities related to blog applications such as CRUD operations of posts, adding comments, searching for other users and instant communication were explained and explored in this work.
Keywords: blog, Express.js, Javascript, React.js

# Uvod

Globalna pandemija koja je zahvatila svijet uvelike je promijenila način na koji ljudi interaktiraju međusobno. Fizička distanca i ograničenja kretanja potaknuli su veću upotrebu digitalnih sredstava komunikacije, koja pružaju alternativni način održavanja međuljudskih odnosa. Internetski blog može stvoriti osjećaj inkluzije jer omogućava korisnicima da se povežu s raznolikim skupinama ljudi koje dijele iste interese ili vrijednosti. Također, internetski blog omogućuje korisnicima da izraze svoje mišljenje i emocije putem sadržaja koji dijele. Naime, treba i napomenuti da postoje tzv. administratori, korisnici koji nadgledaju i upravljanju sadržajem koji se objavljuje na platformi. Administratori imaju više funkcija, no najvažnija je uklanjanje neprimjerenog sadržaja ili prekršaja pravila.

U ovom radu su kroz 5 poglavlja opisane tehnologije korištene za izradu internetskog bloga te sve funkcionalnosti dostupne korisnicima aplikacije. Pri izradi Internetskog bloga korištene su tehnologije trenutno najpopularnije biblioteke programskog jezika Javascript: React.js za izgradnju korisničkog sučelja te Express.js za izgradnju poslužitelja.

U drugom poglavlju su ukratko opisane tehnologije korištene za izradu same aplikacije. U trećem poglavlju su pobliže prikazane funkcionalnosti aplikacije s klijentske strane podijeljene na korisničko iskustvo korisnika i moderatora (administratora). U četvrtom poglavlju su pobliže opisani korišteni modeli i poslužitelj za izradu aplikacije s pridodanim primjerima. Naposljetku, u petom poglavlju je naveden razlog i važnost odabira same teme rada i tehnologije korištene te mogućnosti poboljšanja same aplikacije na osnovu postavljenih ciljeva.

# 2. Tehnologije

## 2.1. Javascript

Javascript je objektno orijentiran programski jezik nastao 1995. godine. Kreirao ga je Brendan Eich za tvrtku Netscape. Sam naziv Javascript nastao je kako bi novi programski jezik privukao što veću pozornost. Iako su Java i Javascript sintaktički slični, imaju mnoge razlike. Java, za razliku od Javascripta, ima čvrste tipove podataka i klasno je orijentiran programski jezik. Javascript podržava primitivne tipove podataka kao što su brojevi, stringovi i booleani. Međutim, tipovi podataka su određeni tijekom vremena izvršavanja kôda (engl. *runtime*) naspram Jave u kojoj su tipovi određeni tijekom pretvaranja u strojni jezik prema deklaracijama tipa podatka u kôdu. Javascript podržava osnovne programske koncepte kao *if* uvjetne izjave, petlje, funkcije, objektno orijentirano programiranje bazirano na prototipima itd.

### 2.1.2. Node.js

Node.js je višeplatformsko poslužiteljsko Javascript *runtime* okruženje koje služi za izgradnju skalabilnih web aplikacija. Node.js podržava i asinkroni način rada pokretan događajima za izgradnju aplikacija koje rade u stvarnom vremenu kao što su: web stranica, browser igara i aplikacija za instant komunikaciju. Izgrađen je na osnovi V8 Javascript pokretača koji je razvio Google.

### 2.1.3. React.js

React.js je besplatna Javascript biblioteka za izradu korisničkih sučelja baziranih na komponentama. Prednost biblioteke React naspram ostalih biblioteka je što pruža jednostavan i fleksibilan način deklarativnog načina programiranja. Programer dizajnira korisnička sučelja i sa svakom promjenom stanja aplikacije, React.js ažurira i ponovno ispisuje korisniku komponente na osnovu promjene stanja. Praćenje promjena unutar React.js komponenti moguće je pomoću ugrađenih "React hooks" funkcija kao npr: `useState`, `useEffect` i `useContext`.

Također, jedna od glavnih značajki i prednosti React.js biblioteke je takozvani Virtual DOM. DOM kod klasičnog web programiranja stvarna je strukturna prezentacija svih HTML čvorišta ispisana korisniku kroz odabrani web pretraživač. Virtualni DOM je kopija stvarnog DOM dokumenta pohranjena u memoriji internetskog pretraživača. React.js biblioteka na efikasan način manipulira Virtual DOM preko funkcija za praćenje stanja HTML čvorišta. U slučaju promjene čvorišta čije se stanje prati, React.js biblioteka vrši promjene samo gdje se dogodila promjena, te nakon toga ažurira aktualno sučelje prikazano korisniku. Stvarna ušteda ovakvog pristupa je smanjenje broja poziva koje klijent radi poslužitelju i ciljane promjene dijelova DOM-a naspram promjene cijelog DOM-a.

### 2.1.4. Express.js

Express.js je minimalistički i najpopularniji Javascript razvojni okvir otvorenog kôda za izradu web aplikacija.Express.js omogućava pisanje raznih izvršitelja za obrađivanje HTTP zahtjeva neovisno o unesenoj putanji. Omogućava integraciju različitih pokretača za ispis korisničkih sučelja te omogućava dodavanje tkz "middleware" biblioteka koje izvršavaju dodatne funkcionalnosti poput manipulacija i obrada zaprimljenih zahtjeva.

## 2.2. MongoDB

MongoDB NoSQL je baza podataka bazirana na dokumentima. Zapis u MongoDB bazi je dokument strukturiran nalik JSON objektu. Polje unutar svakog zapisa je sastavljeno od ključ - vrijednost para gdje vrijednost može biti String, Boolean, Niz, Objekt ili vremenski žig (engl. *Timestamp*). MongoDB podržava i pohranjivanje drugih dokumenata ili niza dokumenata kao vrijednost pojedinog polja vidljivo na slici 1. U polju "`comments`" vidljiv je ugniježđen dokument nalik JSON objektu koji sadrži svoja jedinstvena polja. Ugniježđeni dokument predstavlja jedinstveni komentar vezan za njemu nadređeni dokument što u ovom slučaju predstavlja jedinstvenu objavu.


**Slika 1** : Primjer zapisa u tablici

## 2.2.1. Mongoose.js

Mongoose.js je efektivna i fleksibilna Object Data Modeling (ODM) biblioteka napravljena za Node.js koja koristi MongoDB pokretač (engl. *driver*). Sami dokumenti pohranjeni u MongoDB bazama nalik su JSON objektima bez dodatnih mogućnosti validacije i dublje organizirane strukture. Mongoose pruža programerima efektivan način kreiranja high-level shema koje mongoose.js preko MongoDB drivera mapira u bazu podataka. Mongoose omogućuje napredne tehnike validacije, organizacije odnosa podataka i CRUD operacija. Na ispisu 1 je prikazan primjer Mongoose sheme.

```
const postModel = new mongoose.Schema(
  {
    userId: {
      type: String,
      required: true,
    },
    body: {
      type: String,
      max: 500,
      required: true
    },
    likes: {
      type: Array,
      default: [],
    },
    comments: {
        type: Array,
        default: [],
      },
      isPrivate: {
        type: Boolean,
        default: false
      }
  },
  { timestamps: true }
);
```

**Ispis 1**: Primjer Mongoose.js sheme

# 3. Web aplikacija "SocialApp"

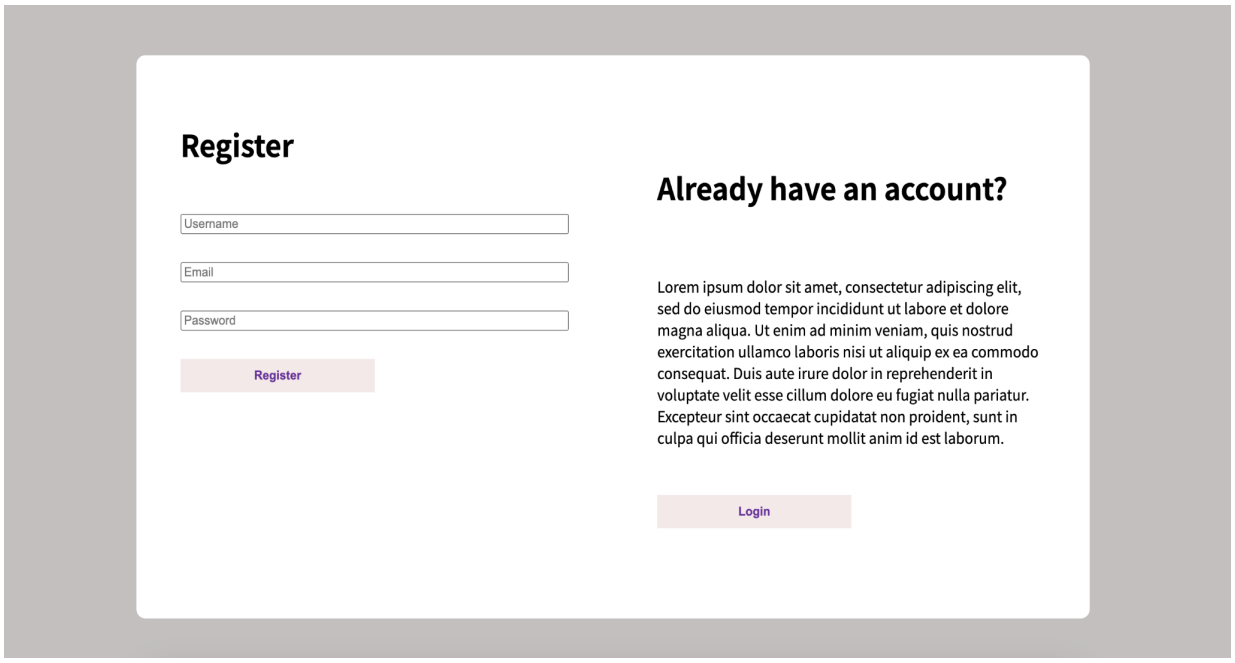
Kroz ovo poglavlje bit će opširno prikazane funkcionalnosti aplikacije, izgled sučelja s priloženim slikama i priloženim izvornim kôdom.

## 3.1. Kratak opis aplikacije

Aplikacija "SocialApp" omogućuje korisnicima jednostavno izražavanje vlastitih misli putem objava te uspostavljanje veza s novim korisnicima putem funkcionalnosti prijateljstava. Također, omogućuje komentiranje i sviđanje kako vlastitih objava, tako i objava drugih korisnika. Radi održavanja visokih standarda sadržaja na web aplikaciji, prisutna je i uloga administratora koji ima ovlasti ukloniti materijale koji nisu usklađeni s pravilima i smjernicama za korištenje aplikacije.

### 3.2.1. Korisničko sučelje aplikacije - Registracija i prijava

Osobe koje žele koristiti “SocialApp” internetski blog prvo trebaju napraviti korisnički račun. Na slici 2 prikazana je forma za registraciju novog korisnika. Forma se sastoji od polja za korisničko ime, polja za email adresu i polja za lozinku. Pri registraciji korisnik mora unijeti jedinstveno korisničko ime i email adresu. U slučaju zauzeća željenog korisničkog imena ili email adrese, poslužitelj javlja odgovarajuću grešku koju klijentski dio aplikacije ispisuje korisniku ispod botuna “Register”.




**Slika 2**: Forma za registraciju novog korisnika

Ako korisnik već posjeduje korisnički račun na "SocialApp" blog stranici, klikom na botun "Login" može se preusmjeriti na stranicu za prijavu. Stranica za prijavu sadrži formu s poljem za korisničko ime i poljem za lozinku. Nakon unosa traženih vrijednosti klikom na botun "Login" poziva se "`loginHandler`" funkcija prikazana na ispisu 2.

```
const loginHandler = async (e) => {
    setErrorMsg("");
    e.preventDefault();
    const { username, email, password } = values;
    const user = {
      username,
      email,
      password,
    };
    try {
      const response = await axios.post(
        "http://localhost:8080/account/login",
        user
      );
      login(response.data);
      window.location.reload()
    } catch (e) {
      setErrorMsg(e.response.data);
    }
  };
```

**Ispis 2:** Funkcija za prijavu korisnika

Funkcija prosljeđuje poziv poslužitelju i u slučaju pronalaska važećeg korisničkog računa preusmjerava korisnika na naslovnu stranicu. U neuspjelom slučaju pronalaska i autentikacije korisnika, poziva se funkcija "`setErrorMsg`" i ispisuje korisniku poruku s odgovarajućom greškom generiranoj na poslužitelju.

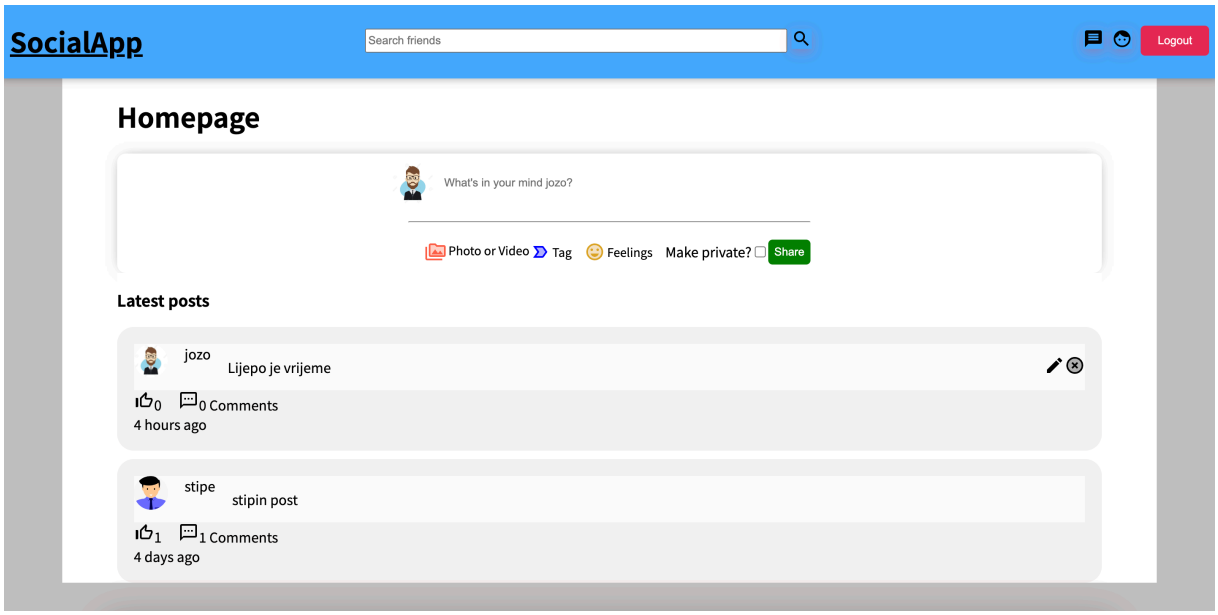
### 3.2.3. Naslovna stranica

Nakon uspješne prijave korisniku je prikazana naslovna stranica. Naslovna stranica se sastoji od navigacijske trake, forme za izradu nove blog objave i prostora gdje su ispisane sve objave od korisnika i korisnikovih prijatelja. Na slici 3 je prikazana navigacijska traka. Traka se sastoji od 3 dijela:

- poveznice za naslovnu stranicu korisnika,
- središnjeg prozora za unos teksta i botuna u obliku povećala
- sličica: "Logout" botuna za odjavljivanje s aplikacije, ikone za profilnu stranicu korisnika i ikone za Messenger.

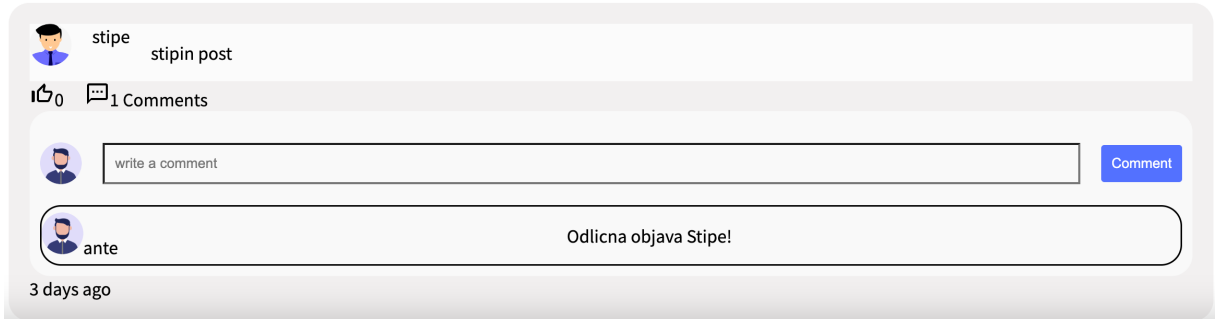

**Slika 3**: Navigacijska traka

Ukoliko korisnik želi napraviti novu objavu, može koristiti formu koja se nalazi ispod alatne trake prikazane na slici 4. Forma se sastoji od polja za unos teksta, Checkboxa kojim korisnik može objavu sakriti od ostalih korisnika koji nisu njegovi prijatelji i botuna “Share” koji služi za objavljivanje unesenog teksta. Nakon klika na botun “Share” objava se pojavi u korisničkom prostoru za objave (eng. *feed*). Korisnik ima mogućnost brisanja i uređivanja svojih objava preko ikona koje se nalaze na desnoj strani objave.




**Slika 4**: Forma za izradu nove objave

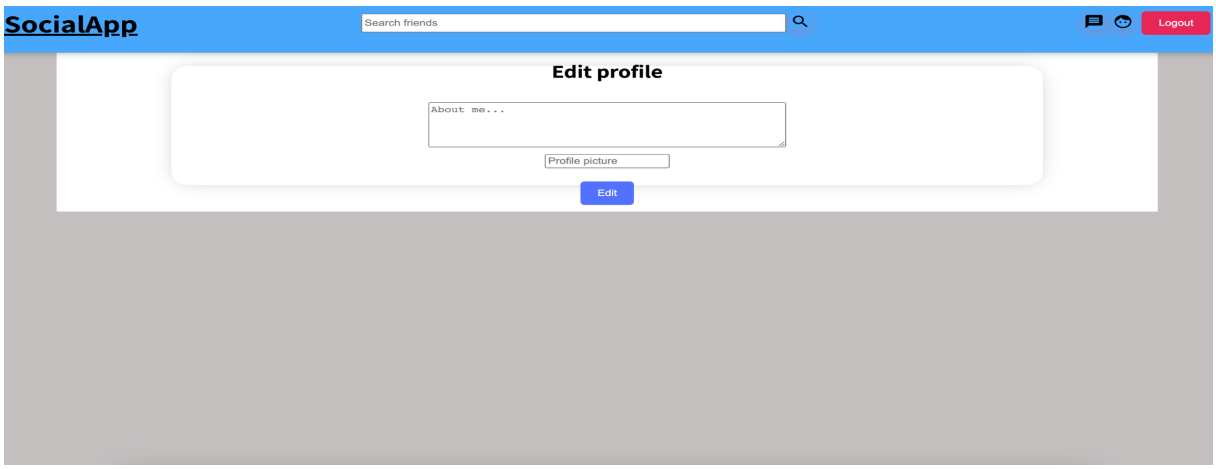
Na svakoj objavi u donjem lijevom kutu postoje dvije ikonice, ikona za *like* svih pojedinih objava prikazanih na *feed* prostoru korisnika i ikona za komentiranje objave. Korisnik može označiti da mu se objava sviđa klikom na ikonicu s podignutim palcem. Također, kod svake objave postoji opcija komentiranja određene objave, odnosno izražavanja svoga mišljenja na tuđe ili svoje objave klikom na chat ikonu prikazanu na slici 5.




**Slika 5**: Forma za pisanje komentara

### 3.2.4. Profilna stranica

Profilna stranica korisnika prikazuje podatke o korisniku i njegove objave. Profilnoj stranici je moguće pristupiti putem alatne trake, klikom na ikonu čovjeka. Na stranici se nalazi botun ”Edit” gdje korisnik klikom na botun otvara formu za uređivanje korisničkog profila prikazana na slici 6. Korisnik može urediti polje “About me” u kojem korisnik može ukratko napisati nešto o sebi što će biti vidljivo drugim korisnicima te polje “Profile picture” gdje može ažurirati poveznicu na svoju profilnu sliku.




**Slika 6**: Forma za ažuriranje profila

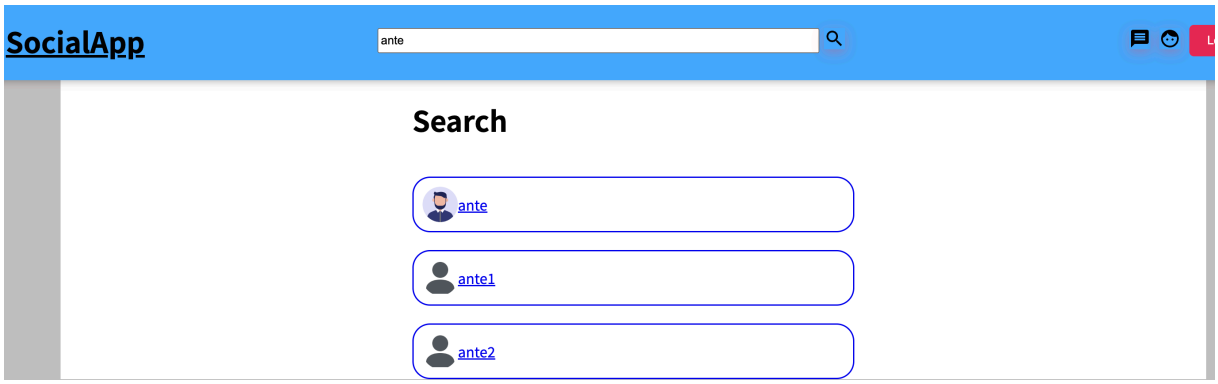
Nakon unosa željenih promjena korisnik klikom na botun “Update” ažurira svoj profil. Klikom na botun poziva se funkcija “`submitHandler`” koja šalje podatke na poslužitelj i u slučaju uspješne promjene preusmjerava korisnika na profilnu stranicu koja je prikazana na ispisu 3.

```
    async () => {
      const response = await axios.put(
        "http://localhost:8080/user/" + id + "/edit",
        {
          id: currentUser._id,
          aboutMe: values.aboutMe,
          profilePicture: values.profilePicture
        }
      );
      return response;
    },
    {
      onSuccess: () => {
        setSucces(!success)
      },
    }
  );
```

**Ispis 3**: Funkcija za ažuriranje podataka korisnika

## 3.2.5. Dodavanje prijatelja

Jedna od najvažnijih značajki modernih društvenih mreža je povezivanje s drugim korisnicima i stvaranje novih prijateljstava. Korisnik na alatnoj traci može pretraživati ostale korisnike po njihovom korisničkom imenu putem polja za pretraživanje. Prikazani rezultati će odgovarati punom ili djelomičnom upitu korisnika. Nakon upisa željenog pojma, klikom ikone povećala korisniku su ispisani svi rezultati koji odgovaraju njegovom upitu, kao što je prikazano na slici 7.




**Slika 7**: Primjer ispisa rezultata pretraživanja

Klikom na željeni rezultat pretraživanja korisnik je preusmjeren na profilnu stranicu željenog korisnika. Na profilnoj stranici korisnika prikazani su: profilna slika korisnika, kratki osvrt o korisniku i botun "Add friend" ili "Unfriend", ovisno jesu li trenutni korisnik aplikacije i korisnik čija je profilna stranica ispisana već prijatelji. U *feed* prostoru korisnika čija je profilna stranica ispisana, prikazane su objave tog korisnika. Ako su korisnici prijatelji, sve će objave korisnika biti prikazane. U protivnom, biti će prikazane samo objave koje nisu označene kao privatne. Ako korisnik želi dodati novog prijatelja, mora kliknuti botun "Add friend". Klikom na botun poziva se funkcija "`submitHandler`" prikazana na ispisu 4. Prilikom slanja zahtjeva poslužitelju unutar tijela POST zahtjeva šaljemo `id` korisnika kojeg želimo dodati. U slučaju uspješnog dodavanja novog prijatelja šalje se novi zahtjev za profilnu stranicu korisnika gdje će biti prikazani svi postovi korisnika.

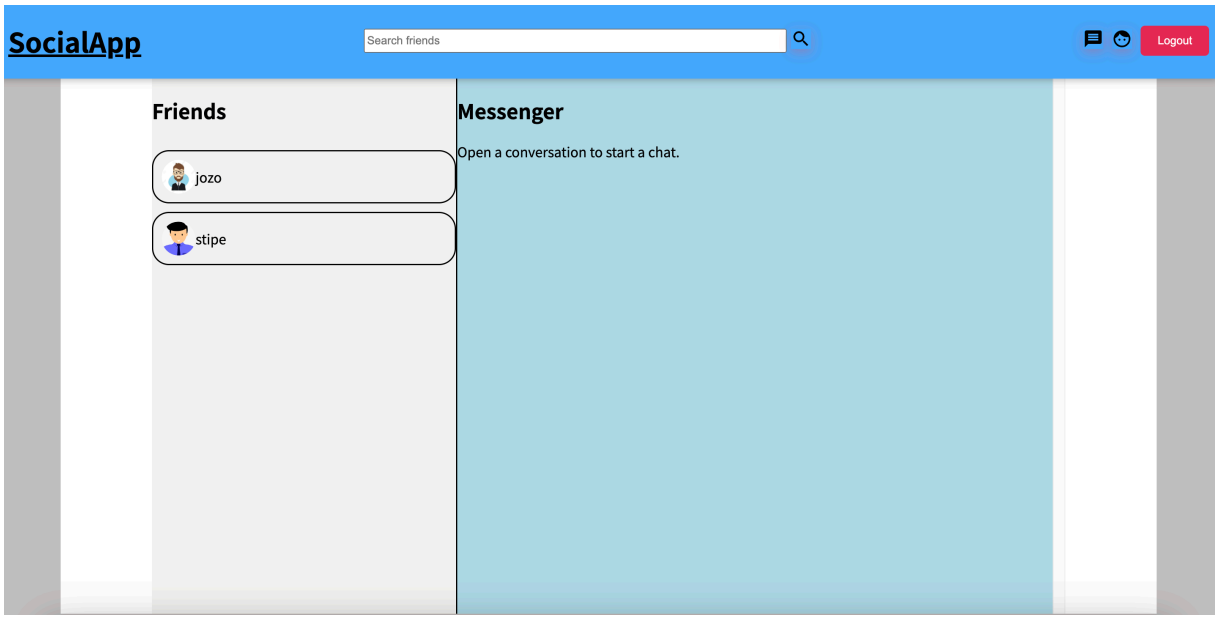
```
    async () => {
      const response = await axios.post(
        "http://localhost:8080/user/" + id + "/addFriend",
        {
          userId: currentUser._id,
        }
      );
      return response;

    },
    {
      onSuccess: () => {
        queryClient.invalidateQueries(["profilepage"]);
      },
    }
```

**Ispis 4**: Prikaz funkcije za dodavanje prijatelja

## 3.2.6. Messenger

U sklopu aplikacije dostupna je trenutna komunikacija između korisnika. Poveznica za pristupu stranice za instant komunikaciju nalazi se na desnoj strani alatne trake u obliku ikone prozorčića s tekstom. Stranica, prikazano na slici 8, sastoji se od 2 prozorčića. S lijeve strane, u “Friends“ prozorčiću, korisniku su prikazani svi njegovi prijatelji. Klikom na bilo kojeg prijatelja otvara se prozor s prijašnjim razgovorom s tim korisnikom. Na desnoj strani ekrana pod Messenger prozorčićem prikazane su sve prijašnje poruke između korisnika i odabranog prijatelja. Ukoliko korisnik želi poslati novu poruku drugom korisniku, klikom na botun “Send” šalje se trenutna poruka prisutna u prozorčiću za pisanje Messenger prozora.




**Slika 8**: Prikaz sučelja za instant komunikaciju

Trenutna komunikacija omogućena je "`useEffect`" React hookom, koji je prikazan na ispisu 5, gdje klijent svako 5000 milisekundi šalje novi upit poslužitelju za poruke između dva korisnika.

```
useEffect(() => {
  setTimeout(() => {
    queryClient.invalidateQueries(["chat"]);
  }, 5000);
}, [messages]);
```

**Ispis 5**: Funkcija za dohvat novih poruka

## 3.3. Administratorsko sučelje aplikacije

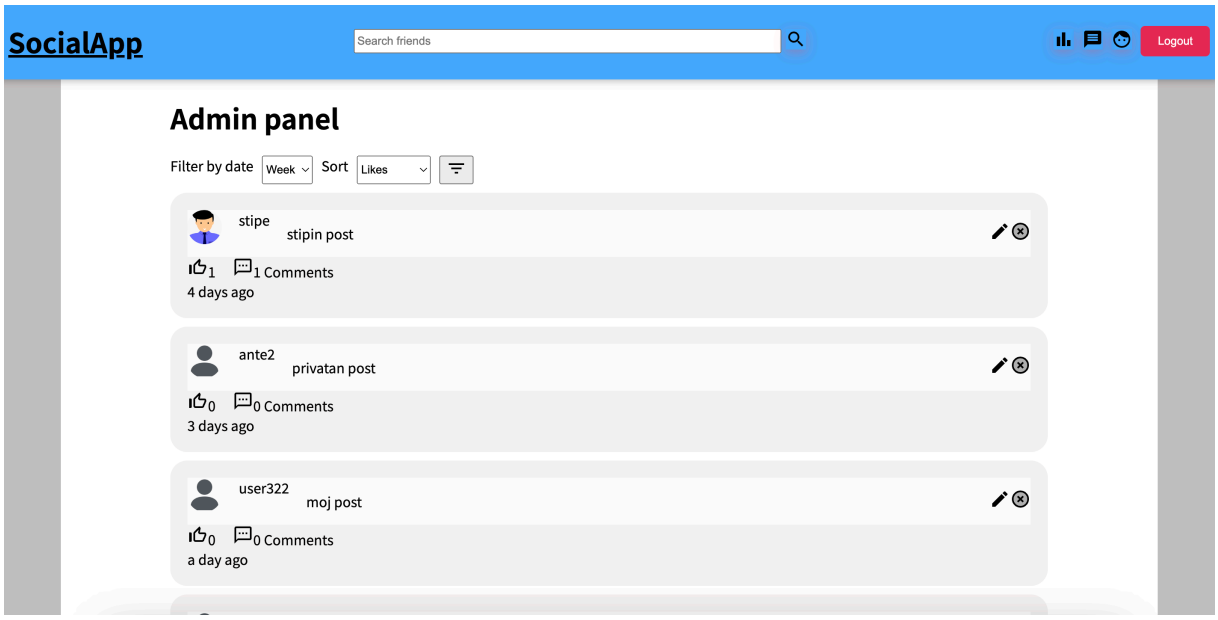
Zbog prirode aplikacije kao društvene mreže, potrebna je neka vrsta nadzora nad sadržajem koji se objavljuje od strane korisnika. Tu ulogu popunjava Administrator za kojeg je polje "isAdmin" unutar tablice popunjeno s *true*. Administrator posjeduje sve mogućnosti kao i obični korisnik, no treba naglasiti da uz to ima i mogućnost statističkog pregleda svih objava, neovisno jesu li te objave označene kao privatne ili ne. Administrator ima i mogućnost brisanja sadržaja drugih korisnika ukoliko je sadržaj uvredljiv, vulgaran, prijeteći, rasistički ili štetan na bilo koji drugi način.

### 3.3.1. Analitika i pretraživanje objava

Administrator panelu pristupa putem alatne trake. Na alatnoj traci postoji dodatna ikona u obliku tablice slijeda koja je jedino vidljiva korisnicima koji imaju administratorske ovlasti kao što je prikazano na slici 9.

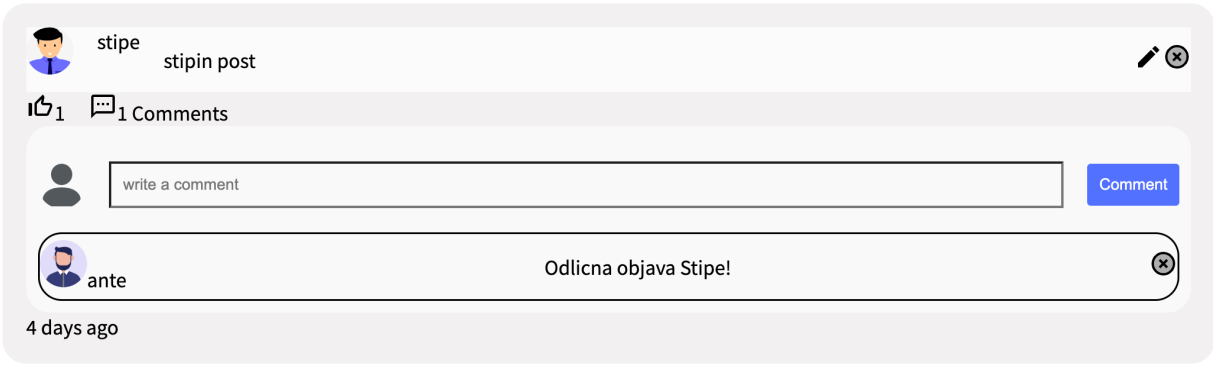

**Slika 9**: Izgled administratorske alatne trake

Klikom na ikonu  u obliku statističke tablice, administrator  je preusmjeren na Admin panel stranicu prikazanu na slici 10. Administrator može pretražiti sve objave napravljene od svih korisnika aplikacije. Kao način naprednog pretraživanja sadržaja moguće je pretražiti objave napravljene u zadnjih tjedan dana, zadnjih mjesec dana, te zadnju godinu dana. Moguće ih je i dodatno sortirati po broju lajkova ili broju komentara. Pokraj filtera postoji ikona filtera koja šalje zahtjev klijentu za traženi sadržaj na osnovi vrijednosti odabrane iz pripadajućih padajućih izbornika.




**Slika 10**: Prikaz administratorskog sučelja

Mogućnosti i zaduženja Administratora su uklanjanje i mijenjanje svog objavljenog sadržaja. Korisnici koji nemaju administratorska prava mogu mijenjati i brisati samo sadržaj koji su oni stvorili koristeći ikone pokraj njihovih objava i komentara. Iz tog razloga su administratoru uvijek vidljive ikone za uređivanje ili brisanje sadržaja aplikacije vidljivo na slici 11 neovisno o autoru objave.




**Slika 11**: Prikaz botuna dostupnih administratoru

# 4. Poslužitelj

Poslužiteljska strana aplikacije bazirana je na Express.js radnom okruženju. Express.js omogućava brzo podizanje poslužitelja za web aplikacije, pruža mogućnost preusmjeravanja zahtjeva na web poslužitelj i rad s bazama kao što su MongoDB i MySQL.

## 4.1. Modeli

Za potrebe izrade aplikacije sa svim opisanim funkcionalnostima izrađena su 3 modela: "`User`", "`Post`" i "`Conversation`". Na ispisu 6 prikazan je model korisnika. Model korisnika baziran je na mongoose Javacript biblioteci koja pruža način modeliranja i strukturiranja podataka na shematski način. Mongoose shema je nacrt za dokumente korištene unutar MongoDB kolekcija. Model "`User`" sastoji se od "`username`", "`email`", "`password`", "`profilePicture`", "`aboutMe`", "`friends`" i "`isAdmin`" polja.

Polja "`username`", "`email`" i "`password`" imaju dodatne mongoose.js opcije unutar svojih definicija kao što su "`required`", "`unique`", "`min`" i "`max`" koje omogućuju dodatna pravila za validaciju podataka tijekom spremanja dokumenta u kolekciju. Opcija "`default`" za pojedina polja postavlja vrijednost na neku određenu početnu vrijednost u slučaju da tu vrijednost nije zadao korisnik. Sam "`id`" nije zadan niti eksplicitno definiran u definiciji samog modela, već je generiran od strane mongoose biblioteke tijekom spremanja novog dokumenta.

```
const userModel = new mongoose.Schema(
  {
    username: {
      type: String,
      require: [true, "Username required"],
      unique: [true],
      minlength: [3, "Username must be at least 3 characters long"],
      max: [15, "Username can not exceed 15 characters"],
    },
    email: {
      type: String,
      required: [true, "Email required"],
      unique: [true],
      max: [50, "Email can not exceed 50 characters"],
    },
    password: {
      type: String,
      required: [true, "Password required"],
      min: [6, "Password must be at least six characters"],
    },
    profilePicture: {
      type: String,
      default:"https://upload.wikimedia.org/wikipedia/commons/thumb/5/59/Useravatar.svg/240px-
User-avatar.svg.png",
    },
    aboutMe:{
      type: String,
      default: "Lorem ipsum dolor sit amet, consectetur adipiscing elit,ed do eiusmod tempor
incididunt ut labore et dolore magna aliqua."
    },
    friends: {
      type: Array,
      default: [],
    },
    isAdmin: {
      type: Boolean,
      default: false,
    },
  },
  { timestamps: true }
);
```

**Ispis 6**: Model Korisnika

Polje "`isAdmin`" određuje razinu prava koja korisnik ima. Prilikom stvaranja novog korisnika polje je postavljeno na početku vrijednost neistina (engl. *false*). Prije nego što se novi dokument spremi u bazu podataka, lozinka korisnika se šifrira pomoću biblioteke bcrypt prikazane ispisu 7. Osnovni način pohranjivanja lozinke bez šifriranja je u obliku čitljivog teksta i ne pruža nikakve dodatne zaštite u slučaju curenja podataka. Šifriranje podataka služi kao zaštita osjetljivih podataka korisnika u slučaju hakiranja baze podataka. Prvi korak šifriranja je generiranje tkz. salt-a, nasumičnog niza alfanumeričkih znakova. Drugi korak je generiranje šifrirane lozinke u kojoj se prethodno generirani salt dodaje nizu alfanumeričkih znakova generiranih algoritmom za šifriranje.

```
exports.register = async (req, res) => {
  try {
    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(req.body.password, salt);

    const newUser = new User({
      username: req.body.username,
      email: req.body.email,
      password: hashedPassword,
    });
    const user = await newUser.save();
    res.status(200).json(user);
  } catch (err) {
    if (err.name === 'ValidationError') {
      let errors
        console.error(errors = Object.values(err.errors).map(val => val.message))
        return res.status(500).json(errors);
      }
    return res.status(500).json("Username or password taken");
  }
};
```

**Ispis 7**: Funkcija za registraciju novog korisnika

Model "Post" prikazan na ispisu 8 sastoji se od polja: "userId", "body", "likes", "comments", "isPrivate". Polje "userId" podatkovnog je tipa string. Nova objava ne može biti spremljena bez "userId" zadane vrijednosti zbog opcije "required". Polje "body" podatkovnog je tipa string i određeno je opcijom "max", tj. maksimalne veličine od 500 znakova. Također, kao i polje "userId", nova objava ne može biti spremljena u slučaju prazne vrijednosti "body" zbog opcije "required". Polje "isPrivate" je tipa boolean i osnovna vrijednost mu je *false*.

```
const postModel = new mongoose.Schema(
  {
    userId: {
      type: String,
      required: true,
    },
    body: {
      type: String,
      max: 500,
      required: true
    },
    likes: {
      type: Array,
      default: [],
    },
    comments: {
        type: Array,
        default: [],
      },
      isPrivate: {
        type: Boolean,
        default: false
      }
  },
  { timestamps: true }
);
```

**Ispis 8**: Model Objave

Polja “likes” i “comments” su tipa niz i nemaju zadanu očekivanu vrijednost koja će biti pohranjena u nizu. Navedena polja su generirana u kontrolerima tijekom pohranjivanja dokumenta i nisu obavezna. Na ispisu 9 prikazana je “likePost” funkcija koja se poziva kada korisnik želi označiti objavu određenog korisnika kao ‘Sviđa mi se’.

```
exports.likePost = async (req, res) => {
  try {
    const post = await Post.findById(req.body.postId);
    if (!post.likes.includes(req.body.userId)) {
      await post.updateOne({ $push: { likes: req.body.userId } });
      res.status(200).json("The post has been liked");
    } else {
      await post.updateOne({ $pull: { likes: req.body.userId } });
      res.status(200).json("The post has been disliked");
    }
  } catch (err) {
    res.status(500).json(err);
  }
};
```

**Ispis 9**: Funkcija za označavanje objave kao ‘’Sviđa mi se’’

Funkcija prihvaća dvije vrijednosti u tijelu zahtjeva, “postId” za pronalazak objave unutar baze podataka i “userId” kao id korisnika koji označava da mu se objava sviđa. U slučaju da id korisnika ne postoji unutar polja “likes” kao vrijednost u nizu, bit će dodana u niz i objava će biti označena da se sviđa korisniku. U slučaju postojanja, id korisnika će biti uklonjen iz niza i objava će prestati biti označena da se sviđa korisniku.

Na ispisu 10 prikazan je model “Conversation”. Sastoji se od polja “users” i “messages”. Oba su polja tipa niz i nemaju zadan tip podatka koji će biti pohranjen u nizu.

```
const conversationModel = new mongoose.Schema(
  {
    users: {
      type: Array,
    },
    messages: {
      type: Array,
      default: [],
    },
  },
  { timestamps: true }
);
```

**Ispis 10**: Model Razgovora

Rad i manipulacija podacima su određeni u kontrolerima. Na ispisu 11 prikazana je funkcija "`newConversation`". Funkcija je pozvana kada korisnik želi započeti razgovor s novim prijateljem. Poslužitelj prima dva podatka unutar zahtjeva, "`senderId`" i "`receiverId`".

```
exports.newConversation = async (req, res) => {
  const newConversation = new Conversation({
    users: [req.body.senderId, req.body.receiverId],
  });

  try {
    const savedConversation = await newConversation.save();
    res.status(200).json(savedConversation);
  } catch (err) {
    res.status(500).json(err);
  }
};
```

**Ispis 11**: Funkcija za izradu novog Razgovora

## 4.2. Putanje i kontroleri

Na ispisu 12 prikazane su sve putanje na poslužitelju. Ovisno o zahtjevu koji poslužitelj primi, poziva se odgovarajuća funkcija koja sadrži poslovnu logiku. Poslužitelj može obraditi PUT, POST, DELETE i GET zahtjeve.

```
//account routes
router.post("/account/login", userController.login)
router.post("/account/register", userController.register)

//user routes
router.get("/user/:id", userController.getUser)
router.post("/user/:id/addFriend", userController.addFriend)
router.get("/user/search/:keyword", userController.getUsers)
router.put("/user/:id/edit", userController.updateUser)

//post routes
router.post("/post/new", postController.newPost)
router.put("/post/:id/update", postController.updatePost)
router.delete("/post/delete", postController.deletePost)
router.get("/post/:id", postController.getPost)
router.post("/post/like", postController.likePost)
router.post("/post/comment", postController.commentPost)
router.put("/post/comment/delete", postController.deleteComment)
router.get("/timeline", postController.getAllPosts)
router.get("/timeline/feed", postController.getUserFeed)
router.get("/timeline/user", postController.getUserPosts)

//conversation routes
router.get("/messenger/:firstUserId/:secondUserId",
conversationController.getConversation)
router.get("/messenger/:userId", userController.getFriends)
router.post("/messenger/newConversation", conversationController.newConversation)
router.post("/messenger/:conversationId/newMessage",
conversationController.newMessage)
```

**Ispis 12**: Putanje korištene u aplikaciji

Na ispisu 13 je primjer funkcije za pretraživanje postojećih korisnika. Kada poslužitelj primi GET zahtjev na putanji "`localhost:8080/user/search/:keyword`", poziva se "`getUsers`" funkcija. Prema ključnoj riječi koju poslužitelj primi stvara se novi objekt za regularne izraze. Biblioteka mongoose.js podržava pretraživanje baze prema regularnim izrazima pomoću ključne riječi `$regex`. U slučaju pronalaska rezultata poslužitelj vraća klijentu listu rezultata sa statusom 200 - OK. U slučaju nepronalaska ijednog rezultata poslužitelj klijentu vraća poruku "Users not found" sa statusom 404 - Not Found

```
exports.getUsers = async (req, res) => {
  const regex = new RegExp(req.params.keyword, 'i')
  try{
    const searchResult = await User.find({username: {$regex: regex}})
    console.log(searchResult)

    return res.status(200).json(searchResult)
  }catch(err){
    return res.status(404).json("User not found")
  }
}
```

**Ispis 13**: Funkcija za pretraživanje korisnika koji odgovaraju upitu

# 5. Zaključak

Pandemija koja je obuhvatila cijeli svijet pokazala je važnost socijalnog aspekta ljudskog bića. Ljudi su zbog ograničenja kretanja i socijalizacije razvili psihičke probleme poput tjeskobe i depresije. Cilj ovog rada bio je stvoriti novu platformu za digitalnu interakciju ljudi. Sama aplikacija zamišljena je kao jednostavna za korištenje pružajući korisnicima moderan i dinamičan način izrade objava i povezivanje s drugim korisnicima platforme. Zbog potrebe izrade modernog sučelja odabrane su trenutno najpopularnije tehnologije, između ostalog i Javascript biblioteka React.js kao sveobuhvatan i izrazito moćan alat za izradu samih sučelja.

Aplikacija kao platforma društvene mreže stvorena je kao alternativa postojećim društvenim mrežama. Aplikacija sadrži sve osnovne i najbitnije građevne elemente društvene mreže: izrada objava, povezivanje s ljudima, trenutačna komunikacija i način moderiranja objavljenog sadržaja kroz ulogu administratora.

Uz sve glavne opisane funkcionalnosti koje su omogućene u aplikaciji postoje dodatne stavke koje bi poboljšale korisničko iskustvo i zelju korištenja. Jedna od takvih je izrada obavijesti koje korisnik primi prilikom novih aktivnosti, na primjer kad jedan od njegovih prijatelja napravi objavu.

# 6. Literatura

[1] Javascript (posjećeno 3.9.2023):

https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[2] Node.js (posjećeno 21.8.2023):

https://developer.mozilla.org/en-US/docs/Glossary/Node.js

[3] React.js (posjećeno 21.8.2023):

https://legacy.reactjs.org/

[4] Express.js (posjećeno 21.8.2023):

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

[3] MongoDB (posjećeno 3.9.2023):

https://www.mongodb.com/what-is-mongodb

[3] Mongoose.js (posjećeno 3.9.2023):

https://mongoosejs.com/docs/guides.html