Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

Chat aplikacija

Seminarski rad iz predmeta Internet tehnologije

Mentor: Studenti:

Beograd, 2023.

Sadržaj

Sadržaj

[1. Korisnički zahtev 1](#_Toc158251148)

[2. Opis sistema 2](#_Toc158251149)

[1. Opis slučajeva korišćenja 2](#_Toc158251150)

[1.1. SK1 - Slanje poruke 2](#_Toc158251151)

[1.2. SK2 - Brisanje poruke 3](#_Toc158251152)

[1.3. SK3 – Registracija korisnika 4](#_Toc158251153)

[2. Opis arhitekture aplikacije 5](#_Toc158251154)

[3. Dijagrami sekvenci 6](#_Toc158251155)

[3.1. DS1: Slanje poruke 6](#_Toc158251156)

[3.2 DS2: Brisanje poruke 7](#_Toc158251157)

[3.3 DS3 – Registracija korisnika 9](#_Toc158251158)

[4. Model podataka 11](#_Toc158251159)

[5. Detaljna specifikacija REST API-ja 15](#_Toc158251160)

[4 Kratak opis tehnologija korišćenih u aplikaciji 22](#_Toc158251161)

[5 Korisničko uputstvo 24](#_Toc158251162)

[6 Prikaz reprezentativnih delova koda 29](#_Toc158251163)

[7 GitHub link ka repozitorijumu 45](#_Toc158251164)

[8 Literatura 46](#_Toc158251165)

# Korisnički zahtev

Potrebno je kreirati inovativnu platformu dizajniranu da omogući korisnicima bogatu interakciju i komunikaciju kroz dinamičan digitalni prostor. Ova aplikacija nudi širok spektar funkcionalnosti počevši od jednostavnog procesa registracije i prijave, koji korisnicima otvara vrata ka personalizovanom iskustvu. Nakon uspešne prijave, korisnici dobijaju priliku da personalizuju svoje profile, dodajući lični pečat svojoj digitalnoj prisutnosti.

Jedan od glavnih aduta naše platforme jeste integracija raznovrsnih chat soba koje omogućavaju korisnicima da se povezuju, dele ideje i komuniciraju u realnom vremenu. Kroz intuitivan interfejs, korisnici mogu lako pretraživati i pridruživati se chat sobama koje odgovaraju njihovim interesovanjima, učestvujući u živopisnim diskusijama i razmeni poruka bez zakašnjenja, zahvaljujući implementaciji web soketa.

Za dodatnu dimenziju interakcije, aplikacija sadrži opciju za pretragu poruka unutar chat soba, omogućavajući korisnicima da lako pronađu razgovore od interesa na osnovu ključnih reči. Osim toga, korisnici imaju mogućnost da uređuju ili brišu svoje poruke, dodajući fleksibilnost i kontrolu nad njihovim komunikacijama.

Kao poseban dodatak, naša platforma nudi segment "Fun Fact" koji pruža korisnicima zanimljive i edukativne činjenice, obogaćujući njihovo iskustvo kroz kontinuirano osvežavanje sadržaja i podsticanje radoznalosti.

Za administratore, aplikacija nudi napredne alate za praćenje i upravljanje popularnošću chat soba. Ova funkcija omogućava administratorima da dobiju uvid u aktivnosti unutar platforme, prepoznaju trendove učešća korisnika i preduzmu odgovarajuće korake za poboljšanje korisničkog iskustva.

# Opis sistema

## Opis slučajeva korišćenja

U ovoj aplikaciji, identifikovano je više slučajeva korišćenja, od kojih ćemo izdvojiti nekoliko.

### 1.1. SK1 - Slanje poruke

**Opis:** Ovaj slučaj korišćenja opisuje postupak kreiranja i slanja poruke putem sistema za razmenu poruka od strane korisnika.

**Akteri:**

* Korisnik
* Sistem (aplikacija za razmenu poruka)

**Preduslovi:**

* Korisnik je prijavljen na svoj nalog u aplikaciji za razmenu poruka.
* Korisnik ima pristup funkcionalnosti za kreiranje i slanje poruka.

**Osnovni scenario:**

1. Korisnik otvara aplikaciju za razmenu poruka na svom uređaju.
2. Nakon prijave, korisnik se nalazi na početnom ekranu aplikacije.
3. Korisnik bira opciju "Nova poruka" ili otvara postojeći razgovor sa kontaktom.
4. Sistem prikazuje formu za unos poruke, uključujući polje za unos teksta.
5. Korisnik unosi željeni tekst poruke u polje za unos.
6. Korisnik bira kontakt ili kontakte kojima želi da pošalje poruku.
7. Korisnik šalje poruku pritiskom na dugme "Pošalji" ili ikonu slanja.
8. Sistem šalje poruku izabranim kontaktima i prikazuje potvrdu o uspešnom slanju poruke.

### 1.2. SK2 - Brisanje poruke

**Opis:** Ovaj slučaj korišćenja opisuje postupak brisanja poruke iz sistema za razmenu poruka od strane korisnika.

**Akteri:**

* Korisnik
* Sistem (aplikacija za razmenu poruka)

**Preduslovi:**

* Korisnik je prijavljen na svoj nalog u aplikaciji za razmenu poruka.
* Korisnik ima pristup funkcionalnosti za brisanje poruka.

**Osnovni scenario:**

1. Korisnik otvara aplikaciju za razmenu poruka na svom uređaju.
2. Nakon prijave, korisnik se nalazi na početnom ekranu aplikacije.
3. Korisnik otvara razgovor u kojem se nalazi poruka koju želi da obriše.
4. Korisnik pronalazi željenu poruku u razgovoru.
5. Korisnik dugim pritiskom na poruku ili izborom odgovarajuće opcije (npr. "Više" ili "Opcije") bira opciju za brisanje poruke.
6. Sistem prikazuje potvrdu ili pita korisnika da potvrdi brisanje poruke.
7. Korisnik potvrđuje brisanje poruke.
8. Sistem briše poruku iz razgovora i ažurira prikaz razgovora na ekranu.

### 1.3. SK3 – Registracija korisnika

**Opis:** Ovaj slučaj korišćenja opisuje postupak registracije korisnika u sistemu za razmenu poruka.

**Akteri:**

* Korisnik
* Sistem (aplikacija za razmenu poruka)

**Preduslovi:**

* Korisnik ima pristup aplikaciji za razmenu poruka.
* Korisnik nema registrovan nalog u sistemu.

**Osnovni scenario:**

1. Korisnik otvara aplikaciju za razmenu poruka na svom uređaju.
2. Na početnom ekranu aplikacije, korisnik odabira opciju za registraciju novog naloga.
3. Sistem prikazuje formu za unos podataka potrebnih za registraciju.
4. Korisnik unosi svoje podatke u formu, kao što su ime, prezime, e-mail adresu i lozinku.
5. Korisnik proverava unete podatke da bi bio siguran da su tačni.
6. Korisnik pritiska dugme za registraciju ili sličnu opciju na formi.
7. Sistem proverava unete podatke da bi utvrdio da li su validni i da li su svi obavezni podaci popunjeni.
8. Ukoliko su svi podaci ispravno uneti, sistem registruje korisnika i kreira nalog u bazi podataka.
9. Nakon uspešne registracije, sistem prikazuje korisniku poruku o uspešnoj registraciji i preusmerava ga na početni ekran aplikacije.

## Opis arhitekture aplikacije

U ovom poglavlju opisaćemo arhitekturu aplikacije koja je razvijena korišćenjem React i Laravel tehnologija. Arhitektura aplikacije igra ključnu ulogu u organizaciji koda, komunikaciji između različitih delova sistema i obezbeđivanju performansi i skalabilnosti aplikacije.

Aplikacija se sastoji od dva glavna dela: frontend (React) i backend (Laravel). Ovaj razvojni stack omogućava razdvajanje odgovornosti između korisničkog interfejsa i poslovne logike, što olakšava održavanje i skaliranje.

Frontend deo aplikacije je implementiran koristeći React, popularni JavaScript framework za izradu korisničkog interfejsa. Evo ključnih komponenti arhitekture React dela:

* Aplikacija je organizovana u više komponenti, od kojih svaka ima svoj specifičan zadatak i odgovornost. To olakšava ponovno korišćenje, testiranje i održavanje koda.
* React Router se koristi za upravljanje rutama i navigacijom unutar aplikacije. Omogućava prikazivanje različitih stranica i komponenata u zavisnosti od URL adrese.
* Za komunikaciju sa backendom koristi se Axios biblioteka za slanje HTTP zahteva prema API-ju servera.
* React omogućava upravljanje stanjem aplikacije korišćenjem stanja i efekata. Ovo omogućava praćenje promena u aplikaciji i ažuriranje korisničkog interfejsa u skladu s tim.

Backend deo aplikacije se bazira na Laravel, PHP frameworku za izradu web aplikacija. Evo ključnih komponenti arhitekture Laravel dela:

* Laravel se koristi za izgradnju API-ja koji omogućava komunikaciju između frontend i backend dela aplikacije. API se često koristi za slanje i primanje JSON podataka.
* Laravel kontroleri se koriste za upravljanje zahtevima, obradu poslovne logike i vraćanje odgovora frontendu.
* Koristi se za definisanje ruta i mapiranje HTTP zahteva na odgovarajuće kontrolere i akcije.
* Laravelov ORM (Object-Relational Mapping) omogućava interakciju sa bazom podataka putem objekata i modela, što olakšava upravljanje podacima.

## Dijagrami sekvenci

U ovom poglavlju ćemo prikazati dijagrame sekvenci za ranije prikazane slučajeve korišćenja.

### DS1: Slanje poruke

1. Korisnik otvara aplikaciju za razmenu poruka na svom uređaju.
2. Nakon prijave, korisnik se nalazi na početnom ekranu aplikacije.
3. Korisnik bira opciju "Nova poruka" ili otvara postojeći razgovor sa kontaktom.
4. Sistem prikazuje formu za unos poruke, uključujući polje za unos teksta.
5. Korisnik unosi željeni tekst poruke u polje za unos.
6. Korisnik bira kontakt ili kontakte kojima želi da pošalje poruku.
7. Korisnik šalje poruku pritiskom na dugme "Pošalji" ili ikonu slanja.
8. Sistem šalje poruku izabranim kontaktima i prikazuje potvrdu o uspešnom slanju poruke.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 1 dijagram sekvenci 1

### 3.2 DS2: Brisanje poruke

1. Korisnik otvara aplikaciju za razmenu poruka na svom uređaju.
2. Nakon prijave, korisnik se nalazi na početnom ekranu aplikacije.
3. Korisnik otvara razgovor u kojem se nalazi poruka koju želi da obriše.
4. Korisnik pronalazi željenu poruku u razgovoru.
5. Korisnik dugim pritiskom na poruku ili izborom odgovarajuće opcije (npr. "Više" ili "Opcije") bira opciju za brisanje poruke.
6. Sistem prikazuje potvrdu ili pita korisnika da potvrdi brisanje poruke.
7. Korisnik potvrđuje brisanje poruke.
8. Sistem briše poruku iz razgovora i ažurira prikaz razgovora na ekranu.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 2 dijagram sekvenci 2

### DS3 – Registracija korisnika

1. Korisnik otvara aplikaciju za razmenu poruka na svom uređaju.
2. Na početnom ekranu aplikacije, korisnik odabira opciju za registraciju novog naloga.
3. Sistem prikazuje formu za unos podataka potrebnih za registraciju.
4. Korisnik unosi svoje podatke u formu, kao što su ime, prezime, e-mail adresu i lozinku.
5. Korisnik proverava unete podatke da bi bio siguran da su tačni.
6. Korisnik pritiska dugme za registraciju ili sličnu opciju na formi.
7. Sistem proverava unete podatke da bi utvrdio da li su validni i da li su svi obavezni podaci popunjeni.
8. Ukoliko su svi podaci ispravno uneti, sistem registruje korisnika i kreira nalog u bazi podataka.
9. Nakon uspešne registracije, sistem prikazuje korisniku poruku o uspešnoj registraciji i preusmerava ga na početni ekran aplikacije.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 3 dijagram sekvenci 3

## Model podataka

Evo kratkog opisa svake klase i njihovih međusobnih veza:

1. **ChatRoom**: Klasa **ChatRoom** predstavlja chat sobu. Sadrži atribute poput imena sobe, privatnosti, maksimalnog broja učesnika i opisa. U vezi je sa dve druge klase: **Message** i **User**. Metoda **messages()** ukazuje na vezu "jedan prema mnogo" sa klasom **Message**, što znači da jedna chat soba može sadržati mnogo poruka. Druga veza je preko metode **participants()**, koja ukazuje na "mnogo-prema-mnogo" vezu sa klasom **User**, što znači da jedna chat soba može imati više učesnika (korisnika).

Slika na kojoj se nalazi tekst, snimak ekrana

Opis je automatski generisan

1. **ChatRoomUser**: Iako sadržaj ove klase nije potpuno prikazan, iz komentara se može zaključiti da je to pivot klasa koja upravlja "mnogo-prema-mnogo" vezom između **ChatRoom** i **User** klasa. U Laravelu, pivot klase se koriste za manipulaciju dodatnim podacima u pivot tabeli, što sugeriše da ova klasa omogućava dodatnu logiku ili atribute koji se odnose na odnos između chat soba i korisnika.

Slika na kojoj se nalazi tekst, snimak ekrana, softver

Opis je automatski generisan

1. **Message**: Klasa **Message** predstavlja poruku unutar chat sobe. Sadrži atribute kao što su tekst poruke, ID korisnika koji ju je poslao, ID chat sobe u kojoj je poslata, vreme slanja, čitanja i brisanja. Ova klasa ima dve veze: **user()** i **chatRoom()**. Metoda **user()** ukazuje na "jedan prema jedan" vezu sa klasom **User**, što znači da svaka poruka ima tačno jednog autora. Metoda **chatRoom()** takođe ukazuje na "jedan prema jedan" vezu, ali sa klasom **ChatRoom**, označavajući chat sobu u kojoj je poruka poslata.

Slika na kojoj se nalazi tekst, snimak ekrana, softver

Opis je automatski generisan

1. **User**: Klasa **User** predstavlja korisnika u chat aplikaciji. Sadrži standardne atribute za korisnika kao što su ime, email i lozinka. Ova klasa ima veze sa **ChatRoom** i **Message** klasama. Veza sa **ChatRoom** klasom je realizovana preko pivot klase **ChatRoomUser**, koja upravlja "mnogo-prema-mnogo" vezom, dok je veza sa **Message** klasom direktna, gde svaki korisnik može biti autor više poruka.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Font

Opis je automatski generisan

Slika na kojoj se nalazi tekst, snimak ekrana, Font, dokument

Opis je automatski generisan

Slika 4 dijagram klasa

## Detaljna specifikacija REST API-ja

U ovom poglavlju ce biti prikazani detalji o api rutama za kontrolere.

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih caskanja |
| HTTP metoda | GET |
| URL | /api/chat-rooms |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {[      {          "id": 1,          "name": "maiores",          "is\_private": 0,          "max\_participants": 9,          "description": "Ad eos et id esse ea suscipit et.",          "created\_at": "2024-01-26T22:50:35.000000Z",          "updated\_at": "2024-01-26T22:50:35.000000Z"      },      {          "id": 2,          "name": "in",          "is\_private": 1,          "max\_participants": 5,          "description": "Illum odio voluptas facilis fugiat et cum.",          "created\_at": "2024-01-26T22:50:35.000000Z",          "updated\_at": "2024-01-26T22:50:35.000000Z"      },      ..    } |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje chat rooma |
| HTTP metoda | POST |
| URL | /api/chat-rooms |
| URL parametri | ?name=soba&is\_private=0&max\_participants=5 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "name": "soba",      "is\_private": "0",      "max\_participants": "5",      "updated\_at": "2024-01-30T20:48:39.000000Z",      "created\_at": "2024-01-30T20:48:39.000000Z",      "id": 31  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje chat rooma |
| HTTP metoda | PUT |
| URL | /api/ chat-rooms |
| URL parametri | 31?name=soba&is\_private=0&max\_participants=5 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 31,      "name": "soba",      "is\_private": "0",      "max\_participants": "5",      "description": **null**,      "created\_at": "2024-01-30T20:48:39.000000Z",      "updated\_at": "2024-01-30T20:48:39.000000Z"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje chat rooma |
| HTTP metoda | DELETE |
| URL | /api/ chat-rooms |
| URL parametri | 31 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | -No Content- |
| Format izlaznih parametara | application/json |



|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih poruka |
| HTTP metoda | GET |
| URL | /api/ messages |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | [      {          "id": 1,          "body": "Nostrum a sint debitis non corporis eum autem. Voluptas aperiam aut ut soluta voluptates id culpa. Iste ut eius enim necessitatibus. Dolores quidem dolor iure totam ut ut similique.",          "sent\_at": "2024-01-24 07:20:54",          "read\_at": "2024-01-19 18:56:47",          "deleted\_at": **null**,          "created\_at": "2024-01-26T22:50:35.000000Z",          "updated\_at": "2024-01-26T22:50:35.000000Z",          "user\_id": 11,          "chat\_room\_id": 11      },      {          "id": 5,          "body": "In laudantium possimus et. Repellendus non et excepturi et accusamus voluptatem. Sed molestiae quis velit aut saepe suscipit praesentium cupiditate. Libero consequuntur sit dolorem et accusantium et.",          "sent\_at": "2024-01-14 17:46:44",          "read\_at": "2024-01-09 21:32:33",          "deleted\_at": **null**,          "created\_at": "2024-01-26T22:50:35.000000Z",          "updated\_at": "2024-01-26T22:50:35.000000Z",          "user\_id": 15,          "chat\_room\_id": 15      }, |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje poruke |
| HTTP metoda | POST |
| URL | /api/messages |
| URL parametri | ?body=body poruke&user\_id=1&chat\_room\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "body": "body poruke",      "user\_id": "1",      "chat\_room\_id": "1",      "updated\_at": "2024-01-30T20:51:36.000000Z",      "created\_at": "2024-01-30T20:51:36.000000Z",      "id": 11  } |
| Format izlaznih parametara | application/json |



|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje poruke |
| HTTP metoda | PUT |
| URL | /api/ messages |
| URL parametri | 11?body=body poruke&user\_id=1&chat\_room\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 11,      "body": "body poruke",      "sent\_at": **null**,      "read\_at": **null**,      "deleted\_at": **null**,      "created\_at": "2024-01-30T20:51:36.000000Z",      "updated\_at": "2024-01-30T20:51:36.000000Z",      "user\_id": "1",      "chat\_room\_id": "1"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Brisanje poruke |
| HTTP metoda | DELETE |
| URL | /api/ messages |
| URL parametri | 11 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | --No Content-- |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | r?name=korisnik&email=korisnik@gmail.com&password=password&date\_of\_birth=2020-02-02 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "name": "korisnik",      "email": "korisnik@gmail.com",      "profile\_image": **null**,      "date\_of\_birth": "2020-02-02",      "status": "offline",      "last\_online\_at": "2024-01-30T20:57:55.471075Z",      "bio": **null**,      "location": **null**,      "updated\_at": "2024-01-30T20:57:55.000000Z",      "created\_at": "2024-01-30T20:57:55.000000Z",      "id": 32  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | email=korisnik@gmail.com&password=password |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "token": "4|7hPvAbxpbvswspwHiqA5VXbO0MBXhAlBGc15dhf8d3e09e76"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Odjava korisnika |
| HTTP metoda | POST |
| URL | /api/logout |
| URL parametri | nema |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Logged out"  } |
| Format izlaznih parametara | application/json |

# Kratak opis tehnologija korišćenih u aplikaciji

JavaScript (JS) je široko usvojen jezik za programiranje koji se koristi za razvoj dinamičkih i interaktivnih web stranica i aplikacija. Glavna prednost JavaScripta je njegova sposobnost izvršavanja na strani klijenta, što znači da se kod izvršava unutar web pregledača korisnika, a ne na serveru. Ovo omogućava implementaciju dinamičkih funkcija na web stranicama, kao što su animacije, igre i osvežavanje podataka bez potrebe za ponovnim učitavanjem stranice.

React, koji predstavlja jedan od najpopularnijih JavaScript biblioteka za razvoj korisničkih interfejsa, koristi JSX, sintaksnu proširenje koje dozvoljava uključivanje HTML i CSS koda direktno unutar JavaScript fajlova. Zahvaljujući Reactu, developeri mogu efikasno graditi jednostranične aplikacije (SPA) i kompleksne korisničke interfejse sa komponentama kao što su liste, grafikoni i interaktivni elementi.

PHP je server-side programski jezik koji se često koristi za razvoj web aplikacija. Zbog svoje lakoće za učenje, širokoj dostupnosti i obilju ugrađenih funkcija, PHP je idealan za izradu složenih web aplikacija. PHP skripte se izvršavaju na serveru, omogućavajući obradu podataka prije nego što se rezultati pošalju klijentu, te se primenjuje u raznim oblastima uključujući obradu obrazaca, upravljanje sesijama i slanje e-pošte.

Laravel, kao visoko cenjeni PHP okvir, nudi širok spektar ugrađenih funkcionalnosti i alata za efikasnu i pojednostavljenu izradu web aplikacija. Laravel pojednostavljuje mnoge aspekte web razvoja, uključujući rad sa bazama podataka, autentifikaciju, rutiranje i slanje e-pošte, pružajući razvojnim timovima moćne alate za brz i siguran razvoj.

CSS (Cascading Style Sheets) je esencijalan jezik za stilizaciju web stranica, omogućavajući definisanje vizuelnog izgleda i formata. CSS se koristi za kontrolu boja, fontova, rasporeda i drugih stilskih aspekata web stranica, što omogućava kreiranje estetski privlačnih i vizuelno konzistentnih korisničkih interfejsa.

HTML (Hypertext Markup Language) je osnovni jezik za kreiranje i strukturiranje web sadržaja. HTML omogućava definisanje strukture web stranica, organizovanje sadržaja u logičke segmente kao što su zaglavlje, telo i podnožje, te uključivanje raznih elemenata kao što su tekst, slike i obrasci.

Axios je popularna JavaScript biblioteka dizajnirana za upravljanje HTTP zahtevima između klijenta i servera, često se koristi zajedno sa React aplikacijama za efikasnu komunikaciju sa API-ima i obradu podataka. Nudi napredne funkcionalnosti poput automatskog konvertovanja odgovora u JSON, konfigurisanja zahteva, upravljanja greškama, kao i podršku za asinhrone operacije kao što su Promise i async/await.

# Korisničko uputstvo

Prilikom pokretanja aplikacije korisnicima se prikazuje početna stranica kaon a slici ispod.

Slika na kojoj se nalazi tekst, snimak ekrana, Četvorougao, dizajn

Opis je automatski generisan

Slika 5 početna

Odabirom opcije autentifikacija se korisniku prikazuje forma za login I registraciju kaon a slici ispod.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Operativni sistem

Opis je automatski generisan

Slika 6 login

Slika na kojoj se nalazi tekst, snimak ekrana, dizajn

Opis je automatski generisan

Slika 7 register

Kada se uloguje administrator njemu se prikazuje stranica sa grafikonom, kao I statistike vezane za ukupan broj korisnika I dostupnih chat soba

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 8 prikaz statistika na admin delu

Prijavljeni korisnici imaju mogućnost da pregledaju svoj profil odabirom opcije Moj profil.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Operativni sistem

Opis je automatski generisan

Slika 9 moj profil

Klikom na dugme uredi profil, korisnicima se otvara modal kao na slici ispod.

Slika na kojoj se nalazi tekst, snimak ekrana, Operativni sistem, softver

Opis je automatski generisan

Slika 10 update profila

Klikom na dugme otvori korisniku se otvara chat soba u kojoj može da ćaska sa drugim korisnicima.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Ikona na računaru

Opis je automatski generisan

Slika 11 prikaz caskanja

Korisnik može da obriše ili izmeni svoje poruke.

Prijavljeni korinisici takođe mogu da se pridruže određenim chat sobama, kao I da ih pretražuju.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, broj

Opis je automatski generisan

Slika 12 prikaz chat soba

# Prikaz reprezentativnih delova koda

Komponenta AuthForm se koristi za implementaciju funkcionalnosti prijave i registracije korisnika u aplikaciji. Komponenta se koristi za interakciju sa korisničkim podacima kao što su ime, email, šifra, slika profila, datum rođenja, biografija i lokacija.

Komponenta koristi React-ov Hook "useState" za praćenje trenutnog stanja prijave ili registracije (označeno varijablom "isLogin") i podataka unetih u formi (spremljenih u "formData"). Takođe se koristi "useNavigate" iz "react-router-dom" za preusmeravanje korisnika na određenu stranicu nakon uspešne prijave ili registracije.

Kod takođe uključuje funkcije za obradu promena unosa (inputa) i selektovanja slike profila. Za prijavu, šalje se HTTP zahtev prema serveru sa unetim emailom i šifrom. Za registraciju, koristi se "FormData" objekat koji omogućava slanje različitih tipova podataka, uključujući i sliku profila.

import React, { useState } from 'react';

import axios from 'axios';

import './AuthForm.css';

import InputField from './InputField';

import { useNavigate } from 'react-router-dom';

const AuthForm = ({setToken}) => {

  let navigate= useNavigate();

  const [isLogin, setIsLogin] = useState(true);

  const [formData, setFormData] = useState({

    name: '',

    email: '',

    password: '',

    confirmPassword: '',

    date\_of\_birth: '',

    bio: '',

    location: '',

  });

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setFormData({ ...formData, [name]: value });

  };

  const handleFileChange = (e) => {

    setFormData({ ...formData, profile\_image: e.target.files[0] });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (isLogin) {

      try {

        const response = await axios.post('http://127.0.0.1:8000/api/login', {

          email: formData.email,

          password: formData.password,

        });

        console.log(response.data);

        localStorage.setItem("auth\_token",response.data.token);

        localStorage.setItem("auth\_id",response.data.user.id);

        setToken(response.data.token)

        navigate('/chatrooms')

      } catch (error) {

        console.error('Login error:', error.response.data);

        alert(error.response.data.message.toString());

      }

    } else {

      try {

        const registerFormData = new FormData();

        Object.keys(formData).forEach((key) => {

          if (key !== 'confirmPassword') {

            registerFormData.append(key, formData[key]);

          }

        });

        const response = await axios.post('http://127.0.0.1:8000/api/register', registerFormData, {

          headers: {

            'Content-Type': 'multipart/form-data',

          },

        });

        console.log(response.data);

      } catch (error) {

        console.error('Registration error:', error.response.data);

        alert(error.response.data.message.toString());

      }

    }

  };

  return (

    <div className="auth-form">

      <h2>{isLogin ? 'Login' : 'Register'}</h2>

      <form onSubmit={handleSubmit}>

        {!isLogin && (

          <>

            <InputField

              type="text"

              name="name"

              placeholder="Name"

              value={formData.name}

              onChange={handleInputChange}

            />

            <InputField

              type="date"

              name="date\_of\_birth"

              placeholder="Date of Birth"

              value={formData.date\_of\_birth}

              onChange={handleInputChange}

            />

            <InputField

              type="text"

              name="bio"

              placeholder="Bio"

              value={formData.bio}

              onChange={handleInputChange}

            />

            <InputField

              type="text"

              name="location"

              placeholder="Location"

              value={formData.location}

              onChange={handleInputChange}

            />

            <InputField

              type="file"

              name="profile\_image"

              onChange={handleFileChange}

            />

          </>

        )}

        <InputField

          type="email"

          name="email"

          placeholder="Email"

          value={formData.email}

          onChange={handleInputChange}

        />

        <InputField

          type="password"

          name="password"

          placeholder="Password"

          value={formData.password}

          onChange={handleInputChange}

        />

        {!isLogin && (

          <InputField

            type="password"

            name="confirmPassword"

            placeholder="Confirm Password"

            value={formData.confirmPassword}

            onChange={handleInputChange}

          />

        )}

        <button type="submit">{isLogin ? 'Login' : 'Create Account'}</button>

      </form>

      <button onClick={() => setIsLogin(!isLogin)}>

        {isLogin ? "Don't have an account? Register" : 'Have an account? Login'}

      </button>

    </div>

  );

};

export default AuthForm;

UserProfile" se koristi za prikaz i uređivanje korisničkog profila. Korisnicima se omogućava pregled informacija o profilu, poput imena, email adrese, datuma rođenja, lokacije i biografije, kao i pregled chat soba koje korisnik poseduje. Takođe, korisnicima je omogućeno uređivanje informacija o profilu i promena lozinke.

Komponenta koristi različite React Hooks kao što su "useState" za praćenje stanja, "useEffect" za asinhrono dohvaćanje podataka o chat sobama i "useNavigate" za upravljanje rutama. Takođe, koristi se "axios" za komunikaciju sa serverom i "react-icons/fa" za prikaz ikona.

Kada korisnik klikne na dugme "Uredi Profil", komponenta omogućava uređivanje informacija o profilu, uključujući ime, email, datum rođenja, biografiju i lokaciju. Nakon uređivanja, korisnik može sačuvati promene klikom na dugme "Ažuriraj Profil".

Takođe, komponenta omogućava promenu lozinke kroz formu "Promena Lozinke", gde korisnik unosi trenutnu lozinku, novu lozinku i potvrdu nove lozinke.

Ukoliko korisnik poseduje chat sobe, komponenta prikazuje listu tih soba i omogućava korisniku da otvori određenu sobu klikom na dugme "Otvori".

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import './UserProfile.css';

import { useNavigate } from 'react-router-dom';

import { FaBirthdayCake, FaEnvelope, FaMapMarkerAlt, FaUser, FaUserEdit } from 'react-icons/fa';

import useUserData from '../korDefKuke/useUserData';

const UserProfile = () => {

  const auth\_token = localStorage.getItem('auth\_token');

  const { userData, loading, error, setUserData } = useUserData(auth\_token);

  const [newPasswordData, setNewPasswordData] = useState({

    current\_password: '',

    new\_password: '',

    new\_password\_confirmation: '',

  });

  const [chatRooms, setChatRooms] = useState([]);

  const navigate = useNavigate();

  const [isEditing, setIsEditing] = useState(false);

  const handleUserInputChange = (e) => {

    const { name, value } = e.target;

    setUserData({ ...userData, [name]: value });

  };

  const handlePasswordChange = (e) => {

    const { name, value } = e.target;

    setNewPasswordData({ ...newPasswordData, [name]: value });

  };

  const handleProfileUpdate = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.put('http://127.0.0.1:8000/api/user/update', userData, {

        headers: {

          Authorization: `Bearer ${localStorage.getItem('auth\_token')}`,

        },

      });

      console.log('Profile updated:', response.data);

      setIsEditing(false);

    } catch (error) {

      console.error('Profile update error:', error.response.data);

    }

  };

  const handleChangePassword = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post('http://127.0.0.1:8000/api/change-password', newPasswordData, {

        headers: {

          Authorization: `Bearer ${localStorage.getItem('auth\_token')}`,

        },

      });

      console.log('Password changed:', response.data);

      setNewPasswordData({

        current\_password: '',

        new\_password: '',

        new\_password\_confirmation: '',

      });

    } catch (error) {

      console.error('Change password error:', error.response.data);

    }

  };

  useEffect(() => {

    const fetchChatRooms = async () => {

      try {

        const response = await axios.get('http://127.0.0.1:8000/api/my-chat-rooms', {

          headers: {

            Authorization: `Bearer ${auth\_token}`,

          },

        });

        setChatRooms(response.data);

      } catch (error) {

        console.error('Error fetching chat rooms:', error.response.data);

      }

    };

    fetchChatRooms();

  }, [auth\_token]);

  if (loading) {

    return <div>Loading...</div>;

  }

  if (error) {

    return <div>Error: {error}</div>;

  }

  const handleOpenRoom = (roomId) => {

    navigate(`/chatrooms/${roomId}`);

  };

return (

  <div className="user-profile-container">

  <div className="user-info-display">

      {/\* <img

      src={userData.profile\_image ? `http://127.0.0.1:8000/storage/${userData.profile\_image}` : <FaUser />}

      alt="Profile"

      className="profile-image"

    /> \*/}

    <div  className="profile-image">

      {userData.profile\_image ? null : <FaUser />}

    </div>

        <h2><FaUser /> {userData.name}</h2>

        <p><FaEnvelope /> {userData.email}</p>

        <p><FaBirthdayCake /> {userData.date\_of\_birth}</p>

        <p><FaMapMarkerAlt /> {userData.location}</p>

        <p>{userData.bio}</p>

        <button onClick={() => setIsEditing(true)} className="edit-button">

          <FaUserEdit /> Uredi Profil

        </button>

      </div>

      <div className="user-chat-rooms">

        <h2>Moje Chat Sobe</h2>

        <ul>

          {chatRooms.map((room) => (

            <li key={room.id}>

              {room.name}

              <button onClick={() => handleOpenRoom(room.id)}>Otvori</button>

            </li>

          ))}

        </ul>

      </div>

    {isEditing ? ( // Prikazivanje forme za uređivanje samo ako je isEditing true

      <>

        <h2>Profil</h2>

        <form onSubmit={handleProfileUpdate} className="profile-form">

          <input

            type="text"

            name="name"

            placeholder="Name"

            value={userData.name}

            onChange={handleUserInputChange}

          />

          <input

            type="email"

            name="email"

            placeholder="Email"

            value={userData.email}

            onChange={handleUserInputChange}

          />

          <input

            type="date"

            name="date\_of\_birth"

            placeholder="Date of Birth"

            value={userData.date\_of\_birth}

            onChange={handleUserInputChange}

          />

          <input

            type="text"

            name="bio"

            placeholder="Bio"

            value={userData.bio}

            onChange={handleUserInputChange}

          />

          <input

            type="text"

            name="location"

            placeholder="Location"

            value={userData.location}

            onChange={handleUserInputChange}

          />

          <button type="submit">Ažuriraj Profil</button>

          <button onClick={() => setIsEditing(false)} className="close-button">

            Zatvori

          </button>

        </form>

      </>

    ) : null}

      {/\* <h2>Promena Lozinke</h2>

      <form onSubmit={handleChangePassword}>

        <input

          type="password"

          name="current\_password"

          placeholder="Trenutna Lozinka"

          value={newPasswordData.current\_password}

          onChange={handlePasswordChange}

        />

        <input

          type="password"

          name="new\_password"

          placeholder="Nova Lozinka"

          value={newPasswordData.new\_password}

          onChange={handlePasswordChange}

        />

        <input

          type="password"

          name="new\_password\_confirmation"

          placeholder="Potvrda Nove Lozinke"

          value={newPassword}

            onChange={handlePasswordChange}

            />

            <button type="submit">Promeni Lozinku</button>

          </form>

         \*/}

         </div>

      );

    };

    export default UserProfile;

"ChatRooms" se koristi za prikazivanje dostupnih chat soba i omogućava korisnicima da pretražuju sobe, pridruže se njima i pregledaju ih. Komponenta koristi različite React Hook-ove kao što su "useState" za praćenje stanja, "useEffect" za asinhrono dohvaćanje podataka o sobama, i "useCallback" za optimizaciju performansi prilikom poziva funkcije za dohvaćanje soba.

Korisnicima je omogućeno da unesu pojam za pretragu chat soba, pri čemu se pri svakoj promeni termina automatski ažurira lista prikazanih soba. Takođe, komponenta podržava paginaciju, gde se prikazuju samo određeni broj soba po stranici, a korisnik može da prelazi na sledeće i prethodne stranice.

Sobe se prikazuju u tabeli sa informacijama kao što su ime, da li su privatne, maksimalni broj učesnika i opis. Korisnik može da se pridruži sobi klikom na dugme "Pridruži se".

Komponenta takođe obrađuje greške koje mogu da se pojave prilikom komunikacije sa serverom i prikazuje odgovarajuće poruke o greškama ili poruku o učitavanju dok se podaci dohvataju.

import React, { useState, useEffect, useCallback } from 'react';

import axios from 'axios';

import './ChatRooms.css';

import ReactPaginate from 'react-paginate';

const ChatRooms = () => {

  const authToken = localStorage.getItem('auth\_token');

  const userId = localStorage.getItem('auth\_id');

  const [chatRooms, setChatRooms] = useState([]);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  const [searchTerm, setSearchTerm] = useState('');

  const [currentPage, setCurrentPage] = useState(0);

  const perPage = 5;

  const handlePageClick = (event) => {

    setCurrentPage(event.selected);

  };

  const handleSearchChange = (e) => {

    setSearchTerm(e.target.value);

    setCurrentPage(0);

  };

    const filteredChatRooms = chatRooms.filter(

      room =>

        room.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

        room.description.toLowerCase().includes(searchTerm.toLowerCase())

    );

  const fetchChatRooms = useCallback(async () => {

    setLoading(true);

    try {

      const response = await axios.get('http://127.0.0.1:8000/api/chat-rooms', {

        headers: {

          Authorization: `Bearer ${authToken}`,

        },

      });

      setChatRooms(response.data);

    } catch (err) {

      setError(err);

    } finally {

      setLoading(false);

    }

  }, [authToken]);

  useEffect(() => {

    fetchChatRooms();

  }, [fetchChatRooms]);

  const handleJoinRoom = async (chatRoomId) => {

    try {

      await axios.post(

        'http://127.0.0.1:8000/api/chat-room-users',

        {

          user\_id: userId,

          chat\_room\_id: chatRoomId,

          joined\_at: new Date().toISOString(),

          role: 'member',

        },

        {

          headers: {

            Authorization: `Bearer ${authToken}`,

          },

        }

      );

      fetchChatRooms();

    } catch (error) {

      console.error('Error joining room:', error.response.data);

    }

  };

  if (loading) return <div>Loading...</div>;

  if (error) return <div>Error: {error.message}</div>;

  const displayedChatRooms = filteredChatRooms.slice(currentPage \* perPage, (currentPage + 1) \* perPage);

  return (

    <div className="chat-rooms-container">

      <h1>Available chat rooms</h1>

      <input

        type="text"

        placeholder="Pretraga chat soba"

        value={searchTerm}

        onChange={handleSearchChange}

        className="search-input"

      />

      <table className="chat-rooms-table">

        <thead>

          <tr>

            <th>Name</th>

            <th>Private</th>

            <th>Max Participants</th>

            <th>Description</th>

            <th>Action</th>

          </tr>

        </thead>

        <tbody>

         {displayedChatRooms.map((room) => (

            <tr key={room.id}>

              <td>{room.name}</td>

              <td>{room.is\_private ? 'Yes' : 'No'}</td>

              <td>{room.max\_participants}</td>

              <td>{room.description || 'N/A'}</td>

              <td>

                <button onClick={() => handleJoinRoom(room.id)} className="join-button">

                  Pridruži se

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

      <ReactPaginate

        previousLabel={'Prethodna'}

        nextLabel={'Sledeća'}

        breakLabel={'...'}

        pageCount={Math.ceil(chatRooms.length / perPage)}

        marginPagesDisplayed={2}

        pageRangeDisplayed={5}

        onPageChange={handlePageClick}

        containerClassName={'pagination'}

        activeClassName={'active'}

      />

    </div>

  );

};

export default ChatRooms;

ChatRoom" se koristi za prikazivanje i upravljanje chat sobom u aplikaciji. Korisnicima se omogućava slanje poruka u chat sobi, pretraga poruka po ključnim rečima i prikazivanje poslednjih pet poruka. Takođe, komponenta prikazuje "Fun Fact" koji se periodično osvežava i menja.

Komponenta koristi različite React Hook-ove kao što su "useState" za praćenje stanja, "useEffect" za asinhrono dohvatanje poruka iz chat sobe i periodično osvežavanje, i "useParams" za dobijanje identifikatora chat sobe iz URL adrese.

Kada korisnik unese poruku i klikne na dugme "Pošalji", poruka se šalje na server putem HTTP zahteva, a zatim se dodaje u listu prikazanih poruka. Takođe, korisnici mogu da pretražuju poruke po ključnim rečima pomoću polja za pretragu.

"Fun Fact" se dohvata sa spoljnog API-ja i prikazuje se u okviru komponente.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { useParams } from 'react-router-dom';

import './ChatRoom.css';

import Message from './Message';

const ChatRoom = () => {

  const [message, setMessage] = useState('');

  const [messages, setMessages] = useState([]);

  const [searchTerm, setSearchTerm] = useState('');

  const [funFact, setFunFact] = useState(''); // Dodajemo state za fun fact

  const userId = localStorage.getItem('auth\_id');

  const authToken = localStorage.getItem('auth\_token');

  const { id: chatRoomId } = useParams();

  const fetchChatRoomMessages = async () => {

    try {

      const response = await axios.get(

        `http://127.0.0.1:8000/api/chat-rooms/messages/${chatRoomId}`,

        {

          headers: {

            Authorization: `Bearer ${authToken}`,

          },

        }

      );

      setMessages(response.data.data);

    } catch (error) {

      console.error('Greška pri dobijanju poruka:', error);

    }

  };

  const fetchFunFact = async () => {

    const apiKey = '7xiJG3ZG/DVXBFQcpnUANw==DCKsOuWEdluVhptV';

    const apiUrl = 'https://api.api-ninjas.com/v1/facts?limit=1';

    try {

      const response = await axios.get(apiUrl, {

        headers: {

          'X-Api-Key': apiKey

        }

      });

      if (response.data) {

        setFunFact(response.data[0].fact);

      }

    } catch (error) {

      console.error('Error fetching Fun Fact:', error);

    }

  };

  useEffect(() => {

    fetchChatRoomMessages();

    // Postavljanje intervala za periodično osvežavanje svakih 5 sekundi

    const intervalId = setInterval(fetchChatRoomMessages, 5000);

    // Očisti interval pri povlačenju komponente

    return () => clearInterval(intervalId);

  }, [chatRoomId]);

  useEffect(() => {

    fetchFunFact(); // Pozivamo funkciju za dohvatanje fun fact-a

  }, []); // Prazan niz znači da će se pozvati samo jednom prilikom montiranja komponente

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post(

        'http://127.0.0.1:8000/api/messages',

        {

          body: message,

          user\_id: userId,

          chat\_room\_id: chatRoomId,

        },

        {

          headers: {

            Authorization: `Bearer ${authToken}`,

          },

        }

      );

      console.log('Poruka uspešno kreirana:', response.data);

      setMessages([...messages, response.data]);

      setMessage('');

    } catch (error) {

      console.error('Greška pri kreiranju poruke:', error);

    }

  };

  const handleSearchChange = (e) => {

    setSearchTerm(e.target.value);

  };

  const filteredMessages = messages.filter((msg) => {

    const userMatch = msg.user.name.toLowerCase().includes(searchTerm.toLowerCase());

    const bodyMatch = msg.body.toLowerCase().includes(searchTerm.toLowerCase());

    return userMatch || bodyMatch;

  });

  return (

    <div className="chat-room-container">

      <h1>Chat Room</h1>

      <div className="fun-fact">

        <p>{funFact}</p>

      </div>

      <div className="search-input-container">

        <label>Pretraga:</label>

        <input

          type="text"

          value={searchTerm}

          onChange={handleSearchChange}

          className="search-input"

        />

      </div>

      <table className="chat-room-table">

        <thead>

          <tr>

            <th>ID</th>

            <th>Poruka</th>

            <th>Korisnik</th>

          </tr>

        </thead>

        <tbody>

          {filteredMessages.slice(-5).map((msg) => (

            <Message

              key={msg.id}

              id={msg.id}

              body={msg.body}

              userName={msg.user.name}

            />

          ))}

        </tbody>

      </table>

      <br />

      <form className="chat-room-form" onSubmit={handleSubmit}>

        <div>

          <label>Poruka:</label>

          <input

            type="text"

            value={message}

            onChange={(e) => setMessage(e.target.value)}

            className="chat-room-input"

          />

        </div>

        <button type="submit" className="chat-room-submit-button">

          Pošalji

        </button>

      </form>

    </div>

  );

};

export default ChatRoom;

Ovaj kod predstavlja komponentu Admin kreiranu pomoću React biblioteke, namenjenu za prikaz statistika u administrativnom panelu aplikacije. Komponenta koristi axios za izvršavanje HTTP zahteva za dobijanje statističkih podataka sa servera i react-chartjs-2 zajedno sa chart.js za vizualizaciju tih podataka kroz bar grafikone.

U okviru komponente Admin, koriste se dva stanja: statistics, koje čuva informacije o chat sobama dobijene sa servera, i brojKorisnika sa brojChatRooms, koji čuvaju ukupan broj korisnika i chat soba, respektivno. Ove vrednosti se inicijalizuju praznim nizom ili nulom i ažuriraju se na osnovu podataka dobijenih iz odgovora servera.

React hook useEffect se koristi za izvršavanje HTTP GET zahteva pomoću axios biblioteke odmah nakon što se komponenta montira. Zahtev se šalje na definisanu URL adresu (http://127.0.0.1:8000/api/statistics), a nakon uspešnog dobijanja odgovora, podaci o chat sobama, ukupan broj korisnika i ukupan broj chat soba se izvlače iz odgovora i postavljaju u odgovarajuća stanja komponente.

Za vizualizaciju statistika, koristi se Bar komponenta iz react-chartjs-2 biblioteke, kojoj se prosleđuju podaci o chat sobama i broju korisnika u svakoj sobi. Ovi podaci se konfigurišu unutar chartData objekta, koji sadrži labele (imena chat soba) i skup podataka uključujući broj korisnika po sobi, boju pozadine i boju ivica barova, kao i širinu ivica.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { Bar } from 'react-chartjs-2';

import {

    Chart as ChartJS,

    CategoryScale,

    LinearScale,

    BarElement,

    Title,

    Tooltip,

    Legend,

  } from 'chart.js';

  ChartJS.register(

    CategoryScale,

    LinearScale,

    BarElement,

    Title,

    Tooltip,

    Legend

  );

  const Admin = () => {

    const [statistics, setStatistics] = useState([]);

    const [brojKorisnika, setBrojKorisnika] = useState(0);

    const [brojChatRooms, setBrojChatRooms] = useState(0);

    useEffect(() => {

      axios.get('http://127.0.0.1:8000/api/statistics')

        .then(response => {

          setStatistics(response.data.chat\_rooms);

          setBrojKorisnika(response.data.total\_users)

          setBrojChatRooms(response.data.total\_chat\_rooms)

        })

        .catch(error => {

          console.error('Error fetching statistics:', error);

        });

    }, []);

    const chartData = {

      labels: statistics.map(room => room.name),

      datasets: [

        {

          label: 'Broj korisnika',

          data: statistics.map(room => room.participants\_count),

          backgroundColor: 'rgba(75,192,192,0.2)',

          borderColor: 'rgba(75,192,192,1)',

          borderWidth: 1,

        },

      ],

    };

    return (

      <div className="container">

        <h2>Admin Panel - Statistike</h2>

        <div className="stat-info">

          <p>Ukupan broj korisnika: {brojKorisnika}</p>

          <p>Ukupan broj chat soba: {brojChatRooms}</p>

        </div>

        <div className="chart-container">

          <Bar

            data={chartData}

            options={{

              maintainAspectRatio: false,

              scales: {

                yAxes: [{

                  ticks: {

                    beginAtZero: true,

                  },

                }],

              },

            }}

          />

        </div>

      </div>

    );

  };

  export default Admin;

# GitHub link ka repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-chatapp_2020_0023>

# Literatura

* Vlajić, S. (2020). Projektovanje softvera. FON.
* W3C. (s.d.). HTML i CSS specifikacije. https://www.w3.org/standards/webdesign/htmlcss
* MySQL. (s.d.). Zvanična web stranica MySQL baze podataka. https://www.mysql.com/
* Lazarević, B., Marjanović, Z., Aničić, N., & Babarogić, S. (n.d.). Baze podataka. FON.
* Tešić, I. (2017, 27. decembar). Uvod u React ekosistem [Članak]. Medium. <https://medium.com/@ITesic/uvod-u-react-ekosistem-8ccfad0a1030>
* React. (n.d.). Learn React [Web stranica]. <https://react.dev/learn>
* Laravel. (n.d.). Laravel 10.x Documentation [Web stranica]. https://laravel.com/docs/10.x