Razvoj mrežne aplikacije za čavrljanje koristeći Websocket i E2EE

Omazić, Matej

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:818402

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-16**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Matej Omazić

RAZVOJ MREŽNE APLIKACIJE ZA ČAVRLJANJE KORISTEĆI WEBSOCKET I E2EE

Diplomski rad

Pula, rujan 2023. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Matej Omazić

RAZVOJ MREŽNE APLIKACIJE ZA ČAVRLJANJE KORISTEĆI WEBSOCKET I E2EE

Diplomski rad

JMBAG: 0303082472
Studijski smjer: Informatika
Kolegij: Napredni algoritmi i strukture podataka
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: izv. prof. dr. sc. Tihomir Orehovački

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, dolje potpisani Matej Omazić, ovime dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom " Razvoj mrežne aplikacije za čavrljanje koristeći WebSocket i E2EE" koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišta Jurja Dobrile u Puli te kopira javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student Matej Omazić

U Puli, rujan 2023. godine

SAŽETAK

Ovaj diplomski rad opisuje izradu mrežne aplikacije za čavrljanje koristeći WebSocket protokol i end-to-end enkripciju (E2EE). Rad se fokusira na stvaranje novog projekta, uspostavljanje komunikacije putem WebSocket tehnologije i implementaciju sigurnosnih mehanizama za zaštitu privatnosti korisnika. Također se obrađuje korisnički unos, upravljanje korisnicima i sobama za čavrljanje, te implementacija korisničkog sučelja. Istražuju se važne funkcionalnosti WebSocket protokola i primjenjuju sigurnosni kriterij za E2EE. Cilj je pružiti uvid u razvoj mrežne aplikacije s naglaskom na sigurnost, privatnost i funkcionalnost.

KLJUČNE RIJEČI: čavrljanje, sigurnost, aplikacija, internet, privatnost, stvarno vrijeme

ABSTRACT

This master's thesis describes the creation of online chat applications using the WebSocket protocol and end-to-end encryption (E2EE). The work focuses on creating a new project, establishing communication via WebSocket technology and implementing security mechanisms to protect user privacy. Processing user input, managing users and chat rooms, and implementing the user interface is also covered. Important functionalities of the WebSocket protocol are explored and security guidelines for E2EE are applied. The goal is to provide insight into the development of online applications with an emphasis on security, privacy and functionality.

KEY WORDS: chat, saftey, application, internet, privacy, real-time

Sadržaj

1. Uvod 1
2. Korištena razvojna okruženja za izradu aplikacije 2
2.1 React 2
2.2 NextJS 3
2.3 IntelliJ IDEA 4
2.4 MongoDB 5
2.5 Postman 6
2.6 Pusher 7
2.7 E2EE 8
3. Pregled srodnih aplikacija 10
3.1 WhatsApp 10
3.2 Telegram 11
3.3 Viber 12
4. Dijagrami 13
4.1 Dijagram slučajeva korištenja 13
4.2 Klasni dijagram 14
4.3 Sekvencijski dijagram 15
5. Datoteke i skripte u aplikaciji 16
5.1 .next 17
5.2 app 17
5.2.1 site 17
5.2.2 actions 21
5.2.3 api 22
5.2.4 components 28
5.2.5 context 32
5.2.6 conversations 33
5.2.7 hooks 40
5.2.8 libs 42
5.2.9 types 44
5.2.10 users 45
5.2.11 ostale datoteke u mapi app 46
5.3 node_modules 48
5.4 pages 48
5.5 prisma 50
5.6 public 51
5.7 ostale skripte 51
6. Zaključak 52

1. Uvod

U današnjem digitalnom dobu, mrežne aplikacije za čavrljanje su postale iznimno popularne i neizostavan dio našeg svakodnevnog života. Kako bi korisnici mogli komunicirati u stvarnom vremenu, ključno je razviti efikasan i siguran sistem koji podržava brz protok podataka. Ovaj diplomski rad fokusiran je na razvoj mrežne aplikacije za čavrljanje koristeći WebSocket protokol za komunikaciju i end-to-end enkripciju (E2EE) za osiguranje privatnosti korisnika.

WebSocket je protokol koji omogućuje komunikaciju u oba smjera između servera i klijenta putem jedne TCP veze. Uspostavljajući održivu vezu, WebSocket omogućuje slanje poruka u stvarnom vremenu između korisnika. Iskorištavanje ovog protokola za razvoj mrežne aplikacije za čavrljanje pruža korisnicima iskustvo komunikacije bez potrebe za osvježavanjem stranice.

Uzimajući u obzir važnost privatnosti korisnika u današnjem digitalnom okruženju, end-to-end enkripcija (E2EE) postaje neophodna za osiguranje povjerljivosti poruka. E2EE šifrira poruke na klijentovoj strani prije slanja i dešifrira ih samo na primateljevoj strani, osiguravajući da samo korisnici koji sudjeluju u razgovoru mogu čitati sadržaj. Integriranje E2EE u razvoj mrežne aplikacije za čavrljanje pruža korisnicima sigurnost i povjerenje u svoje interakcije.

Za razvoj ove aplikacije koristit će se IntelliJ IDEA, popularan i moćan integrirani razvojni okvir koji pruža brojne alate i mogućnosti za razvoj modernih web aplikacija.

Cilj ovog diplomskog rada je istražiti i demonstrirati postupak razvoja mrežne aplikacije za čavrljanje koristeći WebSocket protokol i end-to-end enkripciju. Ovaj rad pružit će detaljan pregled aplikacije, implementacijskih koraka i izazova koje su stali na put tijekom razvojnog procesa.

2. Korištena razvojna okruženja za izradu aplikacije

2.1 React

React je popularan JavaScript okvir za razvoj korisničkih sučelja (eng. user interface) koji omogućuje izgradnju interaktivnih web aplikacija. Osnovna ideja React-a je koncept komponenata koje su modularni dijelovi korisničkog sučelja koji se mogu ponovno koristiti (Kai, 2022).

Jedna od ključnih značajki React-a je virtualni DOM (Document Object Model), koji predstavlja apstrakciju stvarnog DOM-a. Virtualni DOM omogućuje React-u da efikasno ažurira samo promijenjene dijelove korisničkog sučelja, umjesto cijelog DOM-a što rezultira boljom performansom. React koristi JSX (JavaScript XML) sintaksu koja omogućava pisanje HTML-sličnog koda unutar JavaScript datoteka. Ovo omogućuje jednostavnije kombiniranje logike JavaScripta s korisničkim sučeljem (Webandcrafts, 2021).

React također podržava jednosmjerno vezivanje podataka (eng. one-way data binding) kroz svoj sustav upravljanja stanjem (eng. state management). Kroz upravljanje stanjem, React omogućuje jednostavno praćenje i ažuriranje podataka u aplikaciji. React je fleksibilan i može se koristiti za razvoj različitih vrsta aplikacija, od jednostavnih web stranica do složenih jednostranih aplikacija (eng. single-page applications). Također je moguće kombinirati React s drugim tehnologijama i okvirima kako bi se postigle dodatne funkcionalnosti. Zahvaljujući aktivnoj zajednici i velikom broju dostupnih biblioteka i komponenti, React je postao popularan izbor za razvoj modernih, interaktivnih korisničkih sučelja na webu. Prikaz koda napisanog u Reactu vidljiv je na slici 1 (Kai, 2022).

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

1 usage  momazic
interface DropdownForTripDurationProps {
    onOptionChange: (selectedOption: string) => void;
    className: string;
}

5+ usages  momazic
const DropdownForTripDuration: React.FC<DropdownForTripDurationProps> = ({className : string , onOptionChange }) => {
    const options : string[]  = ['mjesečni dolazak u firmu'];

    const [selectedOption : string , setSelectedOption : React.Dispatch<React.SetStateA... ] = useState( initialState: '');
    const [customInput : string , setCustomInput : React.Dispatch<React.SetStateA... ] = useState( initialState: '');
```

Slika 1. Prikaz React/TypeScript koda

2.2 NextJS

Next.js je open-source, platformski nezavisna sredina koja omogućuje izvođenje JavaScript koda na serverskoj strani (eng. backend). To znači da se Next.js koristi za razvoj backend aplikacija, kao i za izgradnju skalabilnih mrežnih servisa. Next.js se temelji na V8 JavaScript engineu koji je razvila tvrtka Google i koji se također koristi u preglednicima poput Chrome-a i ostalim preglednicima temeljenim na Chromiumu (Copes, 2023).

Jedna od glavnih karakteristika Next.js-a je njegova asinkrona, neblokirajuća priroda. To znači da Next.js može obraditi više zahtjeva istovremeno bez blokiranja izvršavanja drugih zadataka. Ovo je moguće zahvaljujući petlji koja obrađuje zahtjeve i poziva odgovarajuće funkcije kada su spremne za izvršavanje. Ovakav način rada čini Next.js vrlo efikasnim i pogodnim za razvoj visoko skalabilnih aplikacija. Next.js dolazi s bogatim ekosustavom modula i paketa koji se lako mogu instalirati putem npm (Node Package Manager) sustava. Ovaj ekosustav omogućuje razvojne inženjere da koriste velik broj već izgrađenih biblioteka i alata kako bi ubrzali razvoj svojih aplikacija. Next.js je vrlo popularan u razvoju web aplikacija i mikroservisa. Njegova fleksibilnost, visoka performansa i jednostavnost upotrebe privukli su veliki broj programera diljem svijeta. Next.js se također često koristi za izgradnju API-ja (eng. Application Programming Interface) i za obradu podataka u stvarnom vremenu (Panchal, 2023).

Next.js je snažna platforma koja omogućuje izvođenje JavaScript koda na serverskoj strani. Njegove glavne prednosti su asinkrona priroda, visoka skalabilnost i bogat ekosustav modula. Korištenje Next.JS vidljivo je na slici 2.

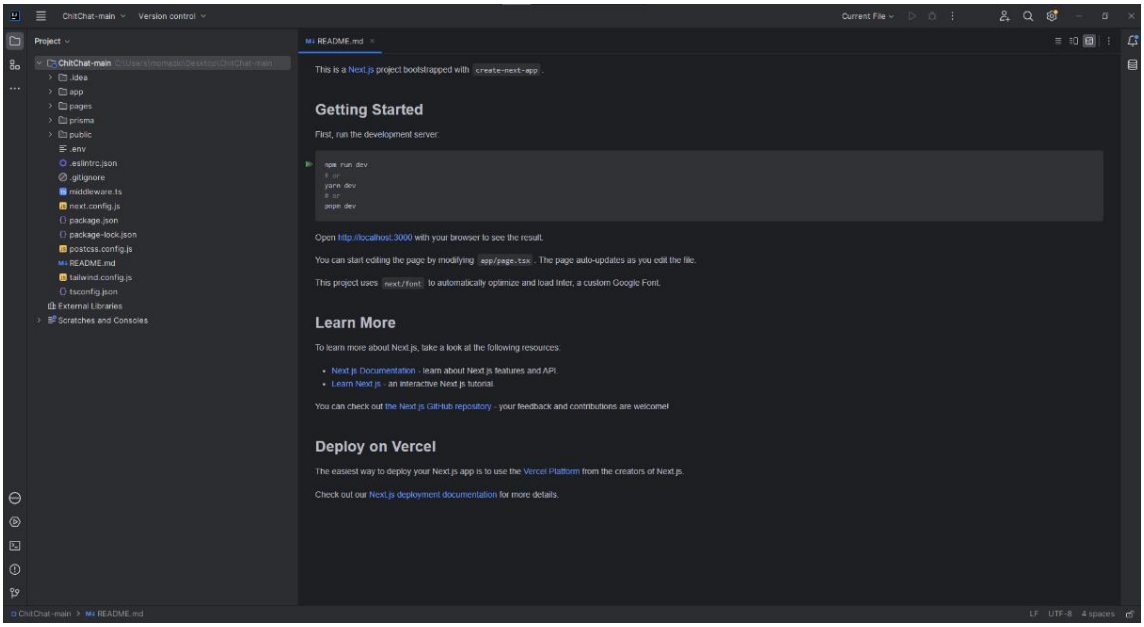
```
import prisma from "@/app/libs/prismadb";

1 usage
interface IParams {
  email?: string;
}

no usages
export async function GET(
  request: Request,
  { params }: { params: IParams }
) : Promise<any>  {
  try {
    const { email : string | undefined  } = params;

    const userDetails = await prisma.user.findUnique({
      where: {
        email
      }
    });

    return NextResponse.json(userDetails)
  } catch (error) {
    return NextResponse.json(null);
  }
}
```

Slika 2. NextJS koji koristi JavaScript kod

2.3 IntelliJ IDEA

IntelliJ IDEA aplikacija vidljiva je na slici 3. Predstavlja jedan od najpopularnijih razvojnih okruženja (eng. IDE) koje se koristi za razvoj softvera. Razvijen od strane JetBrains, IntelliJ IDEA je visoko funkcionalan alat koji pruža programerima mogućnosti za pisanje, uređivanje, otklanjanja neispravnosti (eng. debugging) i ispravljanja (eng. refactoring) koda. IntelliJ IDEA podržava veliki broj programskih jezika, uključujući Java, Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS i mnoge druge. Omogućava programerima da razvijaju raznolike vrste aplikacija, uključujući desktop, web, mobilne i serverske aplikacije. Jedna od glavnih karakteristika IntelliJ IDEA je njen inteligentni kodni editor (eng. code editor). On nudi napredne funkcije poput automatskog dovršavanja koda, refaktoriranja, generiranja koda i analize koda. Također podržava integraciju s alatima za kontrolu verzija, kao što je Git, olakšavajući timski rad i upravljanje projektima. IntelliJ IDEA ima i snažan debugger koji omogućava programerima da lako prate i analiziraju izvršavanje svog koda te podržava izgradnju projekata, automatizirano testiranje i integraciju s popularnim okvirima za testiranje (Tung, 2022).

Pored ovih osnovnih funkcija, IntelliJ IDEA ima bogat ekosistem dodataka i proširenja koji pružaju dodatne funkcionalnosti i podršku za specifične tehnologije i okvire. IntelliJ IDEA je napredno razvojno okruženje koje pruža programerima snažne alate za produktivan i efikasan razvoj softvera. Sa svojim brojnim funkcijama, podrškom za različite jezike i intuitivnim interfejsom, IntelliJ IDEA je izbor mnogih programera širom svijeta (Obregon, 2023).

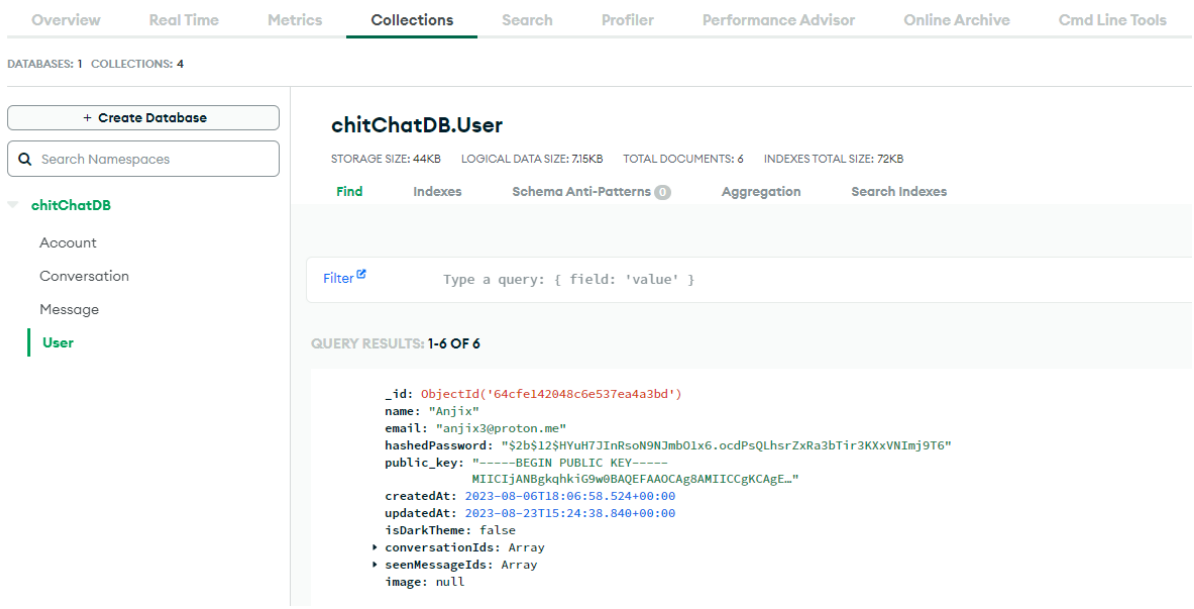


Slika 3. Prikaz IntelliJ IDEA

2.4 MongoDB

MongoDB web aplikacija vidljiva na slici 4 je popularna baza podataka koja se koristi za skladištenje (eng. storing) i upravljanje podacima. Razvijena je kao otvoreni softver i osmišljena je za rukovanje velikim količinama strukturiranih i nestrukturiranih podataka. Jedna od ključnih karakteristika MongoDB-ja je njegov fleksibilni model podataka. Umjesto tradicionalnih relacijskih tablica, MongoDB koristi JSON (eng. JavaScript Object Notation) slične dokumente za pohranu podataka. Dokumenti su grupirani u kolekcije, slično kao tablice u relacijskim bazama podataka. Svaki dokument može imati različitu strukturu, što omogućuje lako dodavanje i mijenjanje polja bez potrebe za promjenom sheme ili predefiniranjem strukture baze podataka (MongoTeam, 2023).

MongoDB podržava razne napredne funkcionalnosti, uključujući replikaciju podataka, fragmentaciju (eng. sharding) i indeksiranje. Replikacija omogućuje stvaranje kopija podataka na različitim čvorovima kako bi se osigurala visoka dostupnost i otpornost na kvarove. Fragmentacija omogućuje horizontalno skaliranje baze podataka tako da se podaci raspodjeljuju preko više čvorova. Indeksiranje omogućuje brz pristup podacima preko indeksa, čime se poboljšava performansa upita. MongoDB je također poznat po svojoj sposobnosti rada s velikim količinama podataka i visokom brzinom upisa i čitanja. Nudi bogat skup upita i operacija za manipulaciju podacima, kao i fleksibilnu mogućnost mapiranja objekata u bazu podataka putem svojih driver-a i ORM (eng. Object-Relational Mappers) alata (MongoTeam, 2023).



Slika 4. Prikaz mongoDB baze pute pretraživača

2.5 Postman

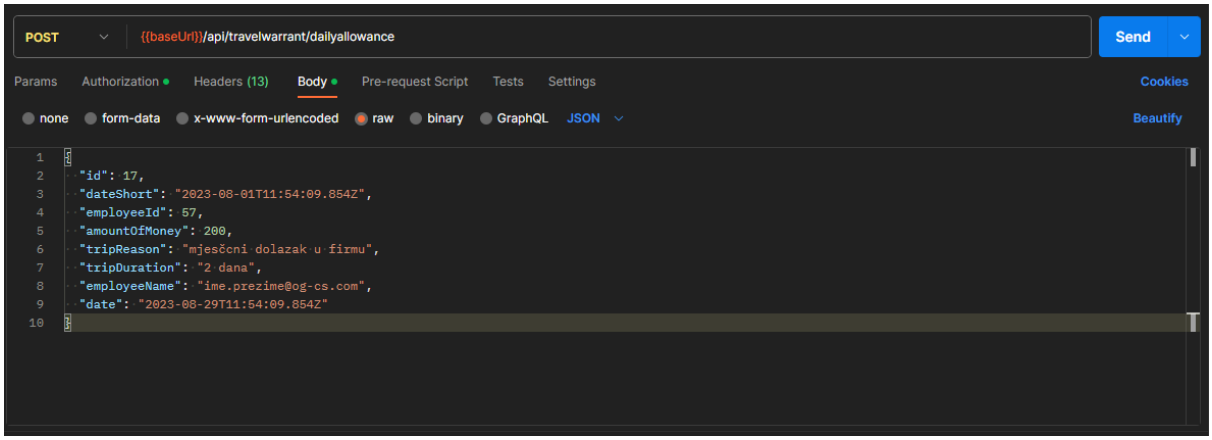
Postman je platforma za razvoj API-ja koja omogućuje programerima da brzo i jednostavno testiraju, dokumentiraju i dijele svoje API-ja. To je alat koji olakšava izgradnju, testiranje i upravljanje HTTP zahtjevima putem korisničkog sučelja (PostmanTeam, 2023).

Postman omogućuje korisnicima da lako stvore HTTP zahtjeve i primaju odgovore od API-ja. Sadrži sučelje koje omogućuje korisnicima da lako upravljaju i uređuju parametre, zaglavlja, tijela zahtjeva i autentifikaciju te podržava različite vrste HTTP metoda poput GET, POST, PUT i DELETE (PostmanTeam, 2023).

Jedna od glavnih značajki Postmana je mogućnost slanja serije zahtjeva, tako da korisnici mogu automatizirati testiranje i testirati API staze s različitim ulaznim podacima. Također omogućuje validaciju odgovora API-ja i upravljanje statusnim kodovima (Mistry, 2023).

Postman ima integraciju s popularnim alatima za upravljanje verzijama kao što su GitHub i GitLab, što olakšava kako timski rad tako i suradnju na projektima. Ima mogućnost generiranja dokumentacije API-ja što je korisno za dokumentiranje i dijeljenje API-ja s drugima (Mistry, 2023).

Postman je sveobuhvatna aplikacija prikazana na slici 5 koja pojednostavljuje proces testiranja, dokumentiranja i dijeljenja API-ja. Olakšava rad programerima i timovima, pomažući im u izgradnji kvalitetnih API-ja i osiguravajući njihovu funkcionalnost i pouzdanost.



Slika 5. Sučelje Postman aplikacije gdje
se koristi POST metoda

2.6 Pusher

Pusher Channel Protocol je protokol koji se koristi za realizaciju komunikacije u trenutnom vremenu između klijentskih aplikacija i Pusher servera. Ovaj protokol omogućava slanje poruka u realnom vremenu čime se ostvaruje brza i efikasna komunikacija između različitih korisnika ili uređaja.

Pusher Channel Protocol se zasniva na WebSocket tehnologiji koja omogućava dvosmjernu komunikaciju između klijenta i servera. Korisnici se mogu preplatiti na određene kanale i primati obavijesti o događajima koji se događaju na tim kanalima. Ovo omogućava da se informacije sinkroniziraju između korisnika u stvarnom vremenu, bez potrebe za osvježavanjem stranice (PusherTeam, 2023).

Protokol podržava slanje poruka u oba smjera, tako da klijentske aplikacije mogu slati poruke serveru i obrnuto. Također, podržava i razne funkcionalnosti poput autentifikacije korisnika, autorizacije pristupa kanalima, grupiranja korisnika u određene kanale i slanja privatnih poruka. Pusher Channel Protocol je dizajniran tako da bude jednostavan za implementaciju i korištenje. Pusher pruža biblioteke i SDK-ove za različite programski jezike, što olakšava integraciju ovog protokola u različite vrste aplikacija, uključujući web aplikacije, mobilne aplikacije i IoT uređaje (PusherTeam, 2023).

Pusher Channel Protocol omogućava razmjenu podataka u stvarnom vremenu između klijenata i servera putem WebSocket veze. To čini ovaj protokol idealnim za implementaciju funkcionalnosti u stvarnom vremenu u aplikacijama koje zahtijevaju brzu i efikasnu komunikaciju između korisnika ili uređaja.

2.7 E2EE

End-to-End Encryption (E2EE) je tehnika šifriranja podataka koja osigurava da samo pošiljatelj i primatelj mogu čitati i razumjeti sadržaj poruke. Ova vrsta šifriranja osigurava privatnost i sigurnost komunikacije tako da se podaci šifriraju na uređaju pošiljatelja i dešifriraju na uređaju primatelja, a putem kojeg se prenose ostaje nečitljiv bilo kojoj trećoj strani/osobi koja bi mogla presresti komunikaciju (Bhardwaj, 2021).

U E2EE sustavu, podaci se šifriraju pomoću ključa koji je poznat samo pošiljatelju i primatelju. Ova tehnika osigurava da se podaci štite tijekom prijenosa preko interneta ili drugih komunikacijskih kanala. Samo primatelj koji posjeduje odgovarajući dešifrirajući ključ može dešifrirati poruku i pročitati sadržaj (CloudFareTeam, 2023).

E2EE se sve više koristi u različitim aplikacijama, poput chat aplikacija i e-pošte, kako bi se korisnicima omogućilo da komuniciraju na siguran način bez straha od neovlaštenog pristupa ili prisluškivanja. Ova tehnologija postaje sve važnija kako raste svijest o potrebi za privatnosti i sigurnosti podataka u digitalnom dobu (CloudFareTeam, 2023).

Važno je napomenuti da iako E2EE pruža visoku razinu sigurnosti i privatnosti, također može predstavljati izazov u smislu provođenja zakona i borbe protiv kriminala. Mnoge zemlje imaju zakone i propise koji uređuju korištenje šifriranja i mogu postojati pravni okviri koji ograničavaju ili zahtijevaju mogućnost dešifriranja podataka u određenim situacijama. U svakom slučaju, E2EE ima ključnu ulogu u zaštiti privatnosti i osiguravanju sigurne komunikacije u digitalnom svijetu (IBMTeam, 2023).

Koncept enkripcije može se pratiti unazad do drevnih civilizacija, ali E2EE kakav se danas poznaje pojavio se dolaskom modernog računalstva i interneta. Rane metode šifriranja, kao što je Cezarova šifra, uključivale su jednostavne tehnike zamjene, koje su bile lake za probiti. Razvoj robusnijih kriptografskih algoritama poput RSA i AES označio je značajne prekretnice u enkripciji. E2EE, kao specifičan pristup enkripciji, postao je istaknut sredinom 20. stoljeća. Dobio je na snazi s porastom digitalnih komunikacijskih tehnologija, posebice e-pošte i razmjene izravnih poruka. Rad na ovom polju uključuje razvoj protokola šifriranja Pretty Good Privacy (PGP) od strane Phila Zimmermanna 1991. godine, koji je korisnicima omogućio sigurno šifriranje svojih poruka e-pošte. Kasnije je Signal, aplikacija otvorenog koda za razmjenu poruka, popularizirala E2EE u aplikacijama za razmjenu trenutnih poruka (MIT Press, 1994).

U svojoj srži, E2EE djeluje na nekoliko temeljnih načela:

1. Šifriranje podataka: E2EE koristi jake algoritme šifriranja za transformaciju podataka otvorenog teksta u šifrirani tekst. Ovaj šifrirani tekst može dešifrirati samo određeni primatelj koristeći svoj privatni ključ.
2. Upravljanje ključevima: E2EE se oslanja na pažljivo upravljanje ključevima šifriranja. Svaki korisnik ima par ključeva: javni ključ i privatni ključ. Javni ključ se koristi za šifriranje, dok se privatni ključ čuva u tajnosti za dešifriranje.
3. Razmjena ključeva: Prije nego što dvije strane mogu sigurno komunicirati, moraju razmijeniti javne ključeve na siguran način. Ovaj proces osigurava da samo namjeravani primatelj može dešifrirati poruke (CSRC, 2022).

Unatoč svojim prednostima, E2EE se također suočava sa značajnim izazovima:

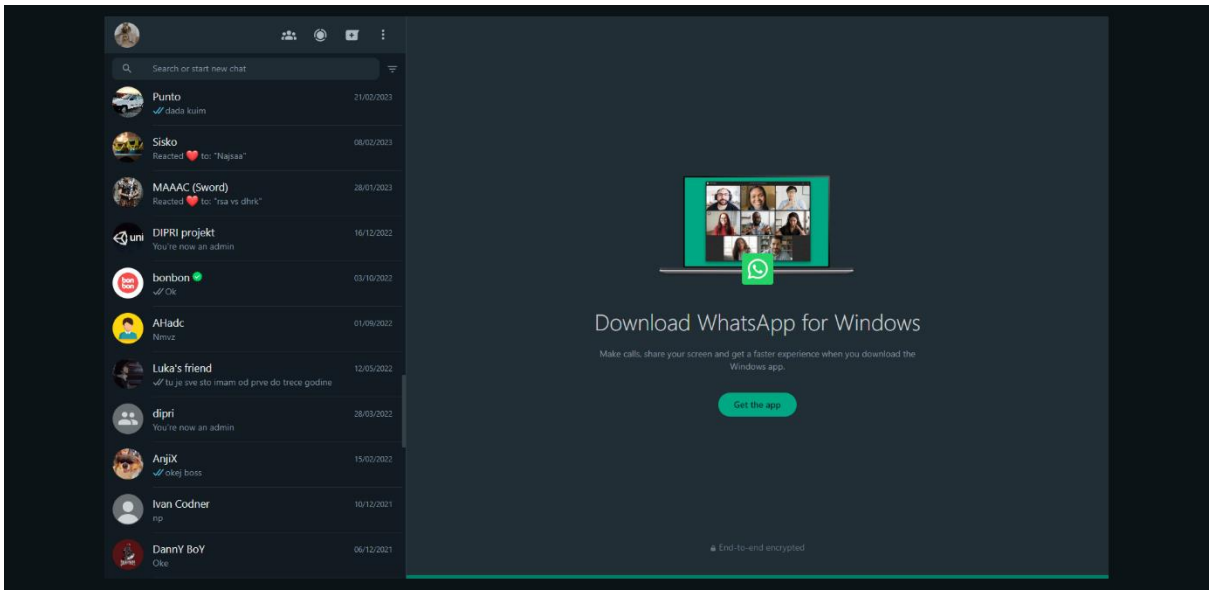
1. Upotrebljivost: Implementacija E2EE može biti složena za netehničke korisnike. Upravljanje ključevima za šifriranje i sigurna provjera javnih ključeva drugih korisnika može biti previše.
2. Zakonodavstvo i propisi: Vlade i agencije za provođenje zakona mogu se suprotstaviti E2EE, tvrdeći da može ometati istrage i napore u pogledu nacionalne sigurnosti.
3. Oporavak ključeva: U slučaju izgubljenih ključeva ili zaboravljenih lozinki, podaci šifrirani s E2EE mogu postati trajno nedostupni.
4. Zlonamjerni softver i sigurnost krajnjih točaka: E2EE štiti podatke u prijenosu, ali ne štiti od zlonamjernog softvera.

E2EE je primjer stalne napetosti između privatnosti i sigurnosti. S jedne strane, osigurava da pojedinci mogu sigurno komunicirati bez straha od neovlaštenog nadzora ili povrede podataka. S druge strane, to predstavlja izazov za agencije za provođenje zakona koje se pokušavaju boriti protiv kibernetičkog kriminala i terorizma. Ravnoteža između ovih zabrinutosti predmet je stalne rasprave. Neki zagovaraju važnost snažne enkripcije kao temeljnog ljudskog prava, dok drugi pozivaju na mehanizme koji omogućuju zakonit pristup šifriranim podacima pod određenim okolnostima (The Guardian, 2016).

3. Pregled srodnih aplikacija

3.1 WhatsApp

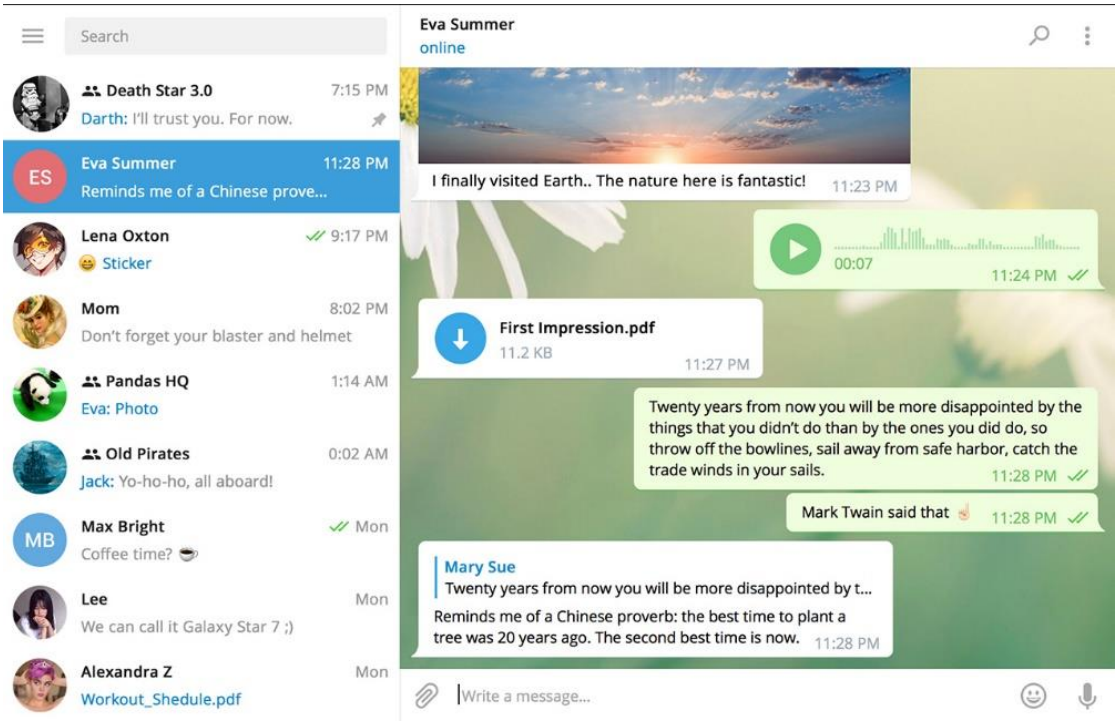
WhatsApp je popularna aplikacija za slanje poruka s end-to-end enkripcijom, primjer je vidljiv na slici 6. Omogućava besplatne razgovore, dijeljenje fotografija, videozapisa i dokumenata. Podržava grupne razgovore, pozive i video pozive. Ima dodatne značajke poput dijeljenja lokacije i statusa. Jednostavna za korištenje i široko dostupna diljem svijeta. Nedostatak WhatsAppa je što ne postoji mogućnost skrivanja od drugih korisnika, također kao Viber i Telegram, tako i WhatsApp traži podjelu broj mobitela kako bi se koristila sama aplikacija.



Slika 6. WhatsApp izgled aplikacije

3.2 Telegram

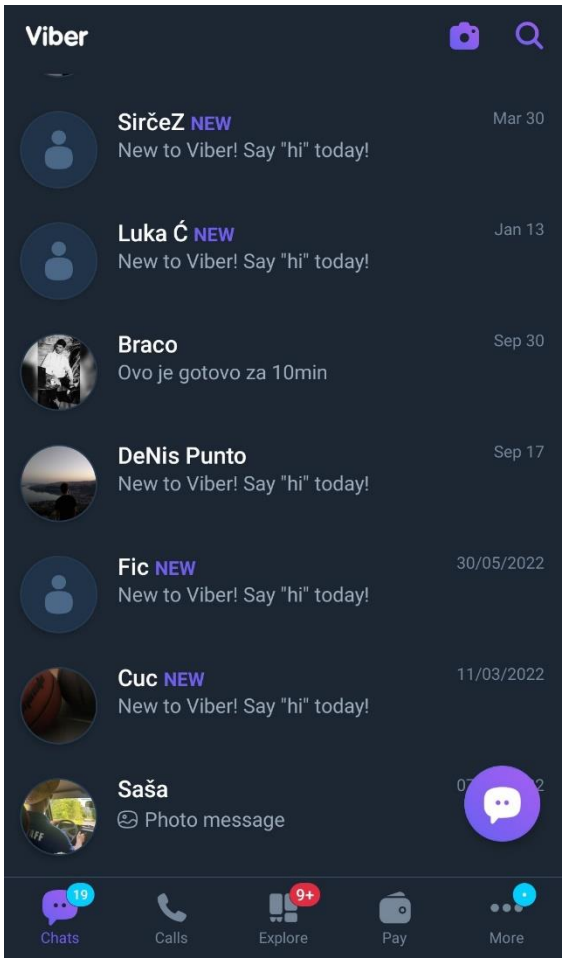
Telegram je sigurna i brza aplikacija za slanje poruka, vidljivo na slici 7. Omogućava slanje tekstualnih poruka, fotografija i videozapisa. Podržava grupne razgovore i javne kanale. Snažna enkripcija osigurava privatnost korisnika. Dostupan na različitim platformama s intuitivnim sučeljem. Najveći problem Telegram je njegova nedostupnost na više platformi, njegove funkcionalnosti se ne mogu u potpunosti koristiti na svim uređajima.



Slika 7. Telegram izgled aplikacije
(The Telegram Team, 2017)

3.3 Viber

Viber je popularna aplikacija za slanje poruka i pozive putem interneta. Korisnicima omogućava besplatno slanje tekstualnih poruka, dijeljenje fotografija, videozapisa i glasovnih poruka. Također podržava pozive i video pozive visoke kvalitete. Viber ima široku dostupnost na različitim platformama, uključujući iOS i Android. Osim toga, nudi i dodatne značajke poput stvaranja grupa, slanja naljepnica, dijeljenja lokacije i igara unutar aplikacije. Viber je jednostavan za korištenje i popularan među korisnicima širom svijeta. Nedostatak Vibera je što se ne može koristiti na internetu kao web aplikacija, samo putem Play Store i App Store (slika 8).

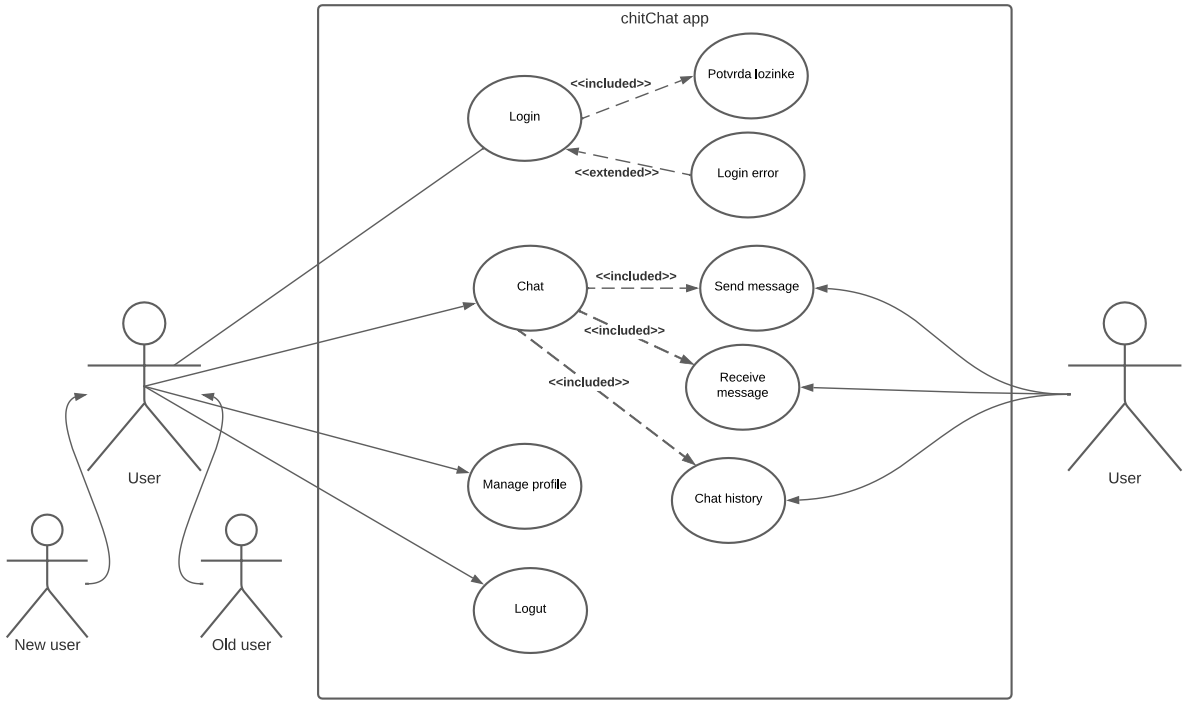


Slika 8. Viber izgled aplikacije

4. Dijagrami

4.1 Dijagram slučajeva korištenja

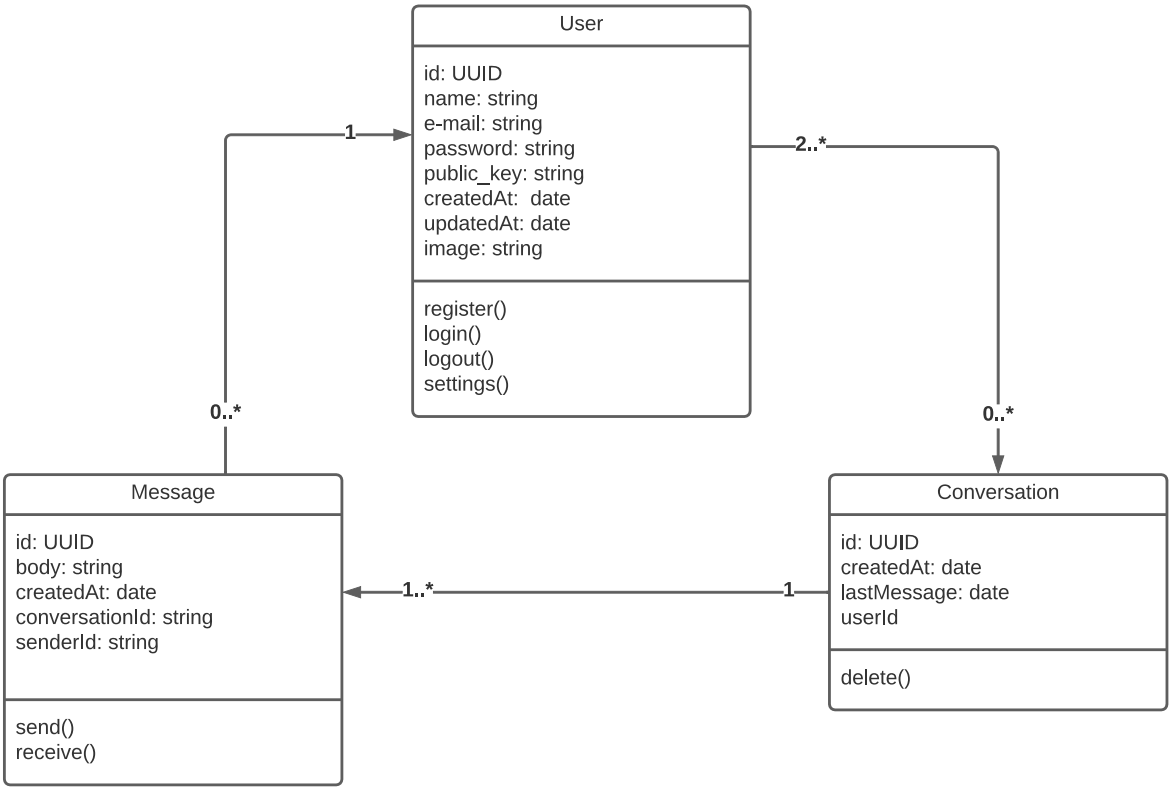
Na slici 9 vidljiv je dijagram slučajeva korištenja. Imamo dva glumca (eng. actor), korisnika, te s druge strane drugog korisnika koji predstavlja osobu s druge strane komunikacije. Prvi „actor“ je korisnik koji je možda novi korisnik i prvi puta koristi aplikaciju ili stari korisnik, tj. korisnik koji se vraća. Drugi „actor“ je korisnik koji zapravo samo prima poruke. Aplikacija započinje registracijom te se ulazi pomoću prijave (eng. login) nakon što se unese lozinka, ona se uspoređuje s onom lozinkom u bazi. Ako su lozinke različite aplikacija javlja grešku u prijavi (eng. login error), no ako je lozinka točna aplikacija vodi na početnu stranicu. Na početnoj stranici može se čavrljati (eng. chat) s drugim osobama, urediti profil ili se odjaviti. Korištenje čavrljanja može se poslati poruka koju će drugi actor primiti, također može se primi poruka od tog drugog actora te oba aktora imaju pregled u povijest razgovara (eng. chat history).



Slika 9. Dijagram slučajeva korištenja

4.2 Klasni dijagram

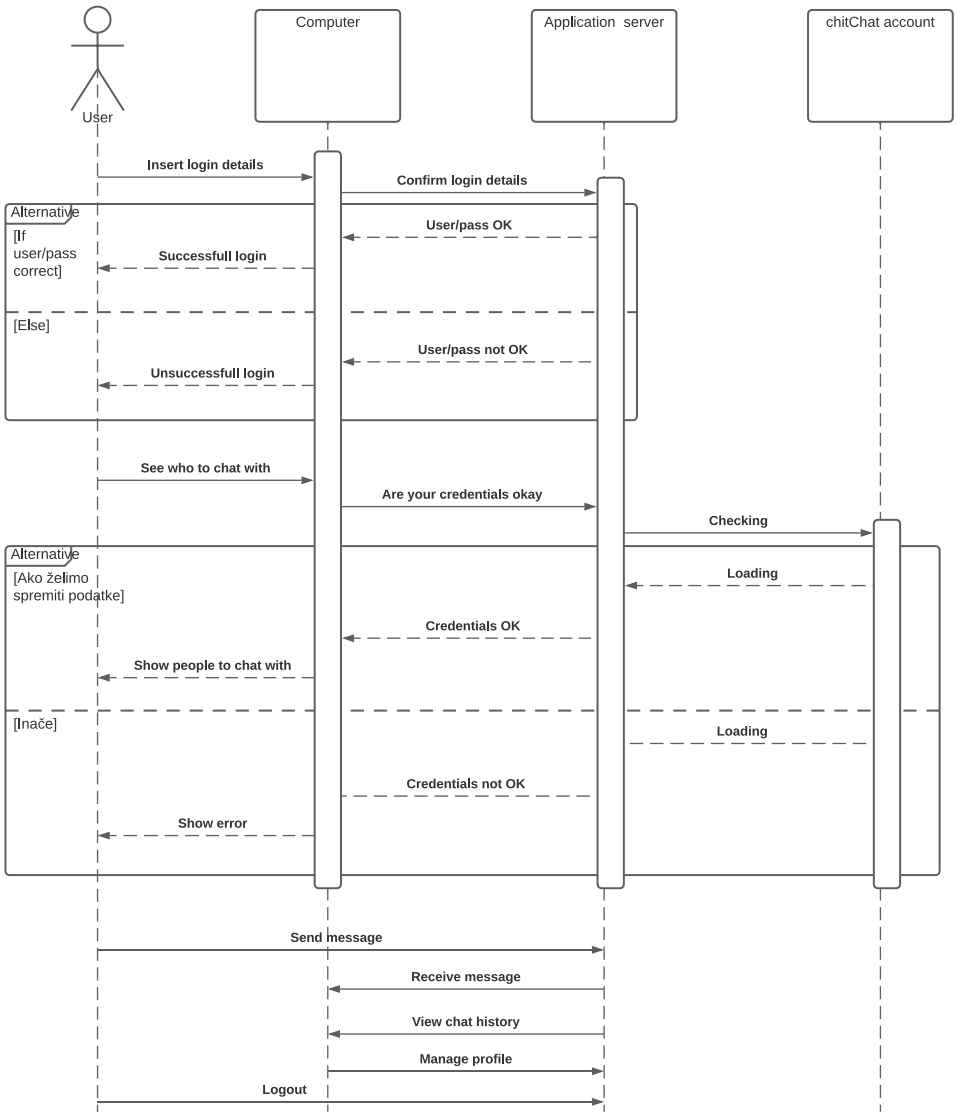
U klasnom dijagramu mogu se vidjeti povezane ključne stvari i klase aplikacije, a to su korisnik i njegovi razgovori. Na slici 10 vidljiva je povezanost između tablica. Na taj način korisnik je povezan s razgovorom te korisnik može imati 0 ili više razgovora, no razgovor mora imati minimalno dva korisnika ili više njih. Razgovor u sebi može sadržavati jednu ili više poruka, dok poruka ima jedan i samo jedan razgovor kojem pripada. Poruka ima jednog korisnika, dok korisnik može imati 0 ili više poruka.



Slika 10. Klasni dijagram

4.3 Sekvencijski dijagram

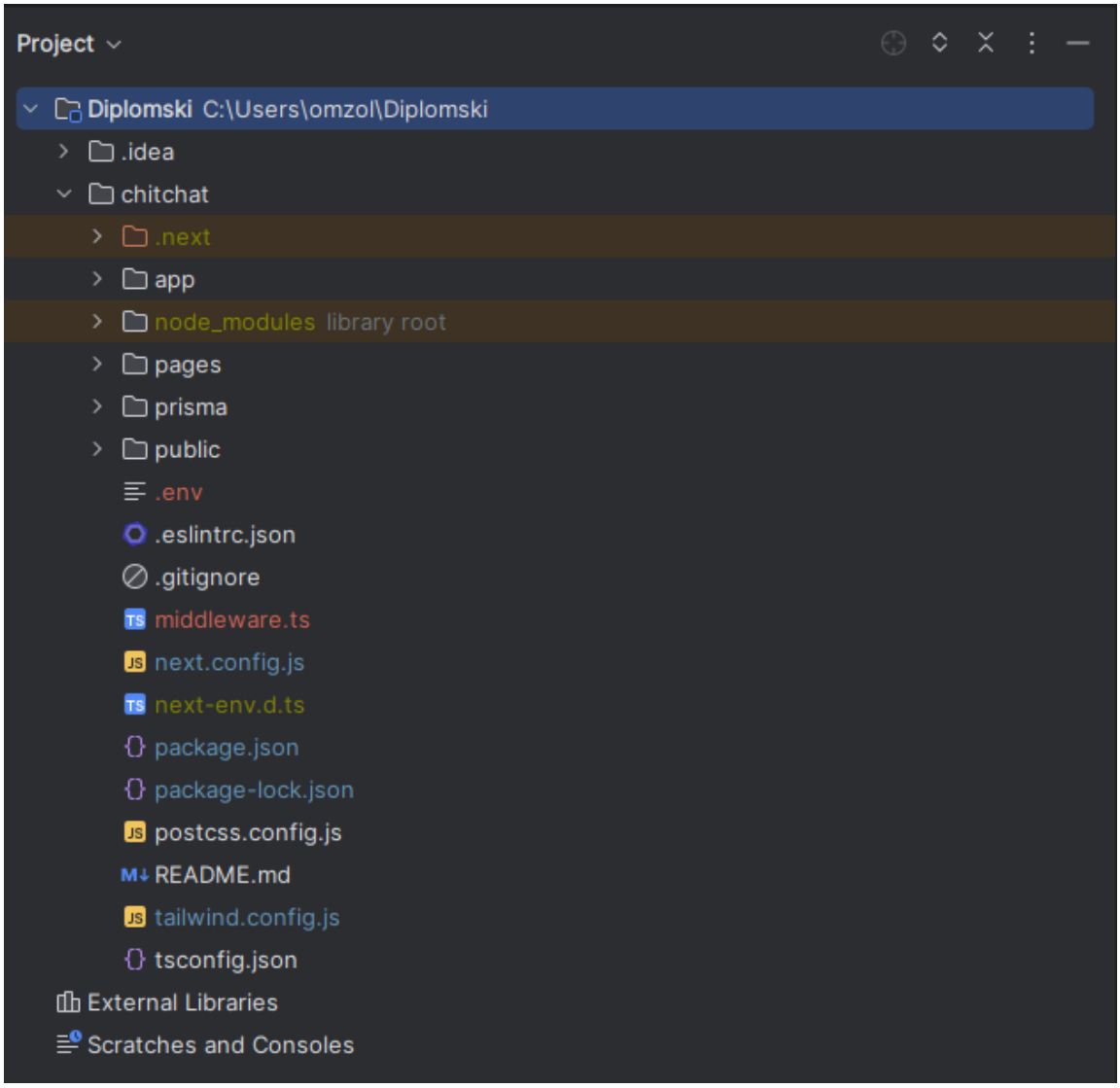
Sekvencijski dijagram vidljiv na slici 11 prikazuje kako izgleda kada se korisnik želi prijaviti u aplikaciju, te što se događa ako se želi čavrljati s drugim korisnicima. Ako korisnik unese točan e-mail i lozinku aplikacija će ga proslijediti na početnu stranicu, no ako na jednome od toga pogriješi aplikacija javlja grešku u prijavi. Kada korisnik poželi čavrljati s nekim, aplikacija prvo provjerava akreditaciju (eng. credentials), koje osobe su sve dostupne na aplikaciji. Događa se provjera, ako ne postoji korisnik u aplikaciji dobiva se poruka u obliku greška (Nema korisnika za čavrljanje), no ako ima korisnika, aplikacija ih prikazuje. Ako se pošalje poruka ona ide do servera te se sprema u bazu i prikazuje krajnjem korisniku, no kada želi primiti poruku ona se čita iz baze, te je isto za povijest cijelog razgovora i za korisničke postavke.



Slika 11. Sekvencijski dijagram

5. Datoteke i skripte u aplikaciji

Koristeći prozor (eng. tab) „Project“ u aplikaciji IntelliJ IDEA dobiva se prikaz svih mapa korištenih u projektu (slika 12). Prva mapa „.idea“ je napravljena od same aplikacije IntelliJ IDEA za otvaranje projekta te chitchat mapa koja sadrži sve stvari od kojih je aplikacija izrađena.



Slika 12. Prikaz datoteka u projektu

5.1 .next

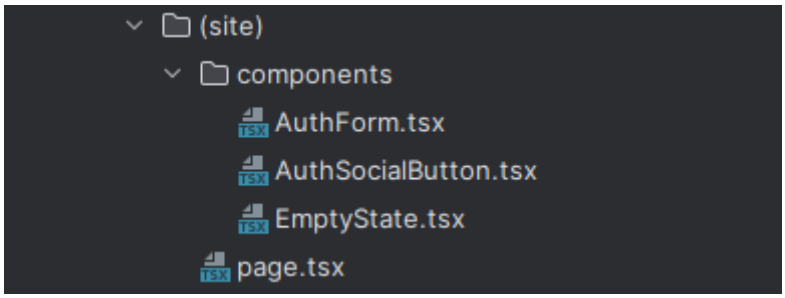
Mapa .next se automatski generira kada se izradi aplikacija pomoću naredbe za izgradnju Next.js. Next.js je popularan okvir za razvoj web aplikacija koji se temelji na React.js. Pruža mogućnost jednostavnog server-side i client-side renderiranja, automatsko generiranje statičkih stranica. Olakšava izradu naprednih aplikacija podrškom za API rute i lako integriranje s drugim alatima i bibliotekama (NextJSTeam, 2023).

5.2 app

„App“ kao što i samo ime govori sadrži najbitnije od aplikacije. Predstavlja glavni direktorij koji sadrži izvorni kod (eng. source code) i različita sredstva za samu web aplikaciju.

5.2.1 site

Unutar mape „App“ nalazi se mnogo podmapa od kojih je jedna (site), slika 13. Sadrži još jednu mapu komponente (eng. components), te skriptu page.tsx.



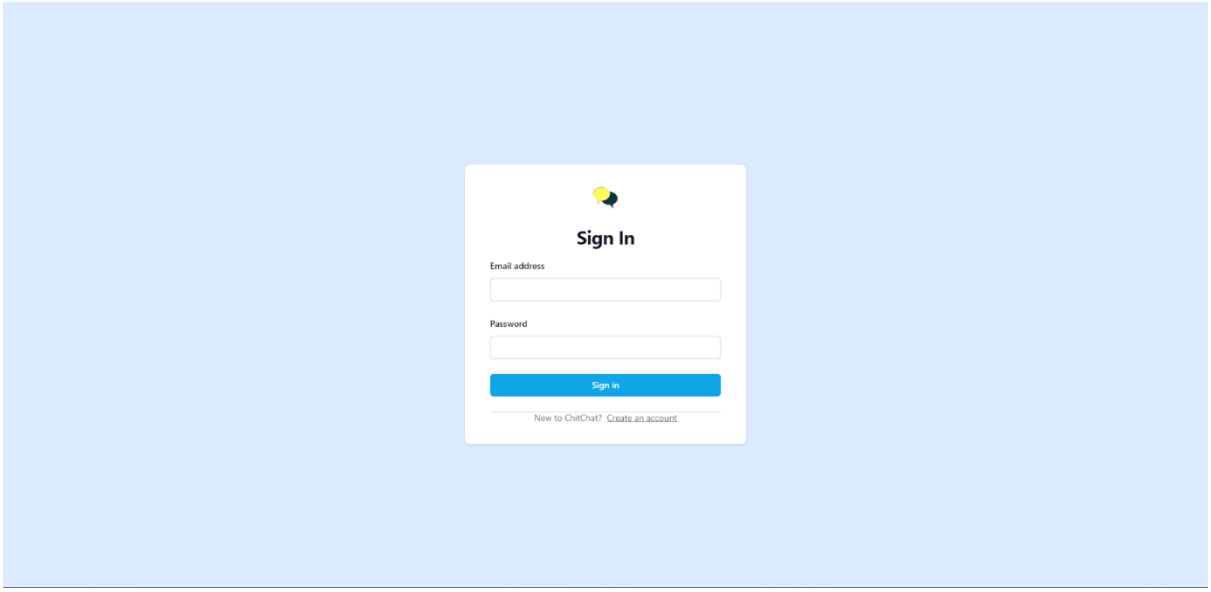
Slika 13. Prikaz mape (site)

Slika 14 prikazuje skriptu page.tsx u kojoj se zapravo poziva skripta AuthForm.tsx zajedno sa CSS oznakama (eng. Cascading Style Sheets tags) koje uređuju tu formu kako bi ona izgledala ljepše.

```tsx
import Image from "next/image";
import AuthForm from "./components/AuthForm";

1 usage  Matej-Omazic *
const Auth = () => {
  return (
    <div className="flex min-h-full flex-col justify-center py-12 sm:px-6 lg:px-8 bg-blue-100 ">
      <AuthForm />
    </div>
  );
};

no usages  Matej-Omazic
export default Auth;
```

Slika 14. Izgled skripte page.tsx

Skripta AuthForm.tsx je skripta koja prikazuje „dvije“ forme, tj. prikazuje jednu stranicu koja ima dva stanja. Prvo stanje je prijava koja je prikazana na slici 15 te registracija. Te dvije forme su jako slične te ovise o potrebama. Zadana forma je prijava te ako se korisnik odluči registrirati, zapravo se ne prosljeđuje na drugu stranicu već forma prijave dobije dodatno polje za upisati, promijeni joj se ime i ostale potrebne stvari, ista stvar se događa kada se s registracije mijenja na prijavu.



Slika 15. Izgled obrazca za prijavu

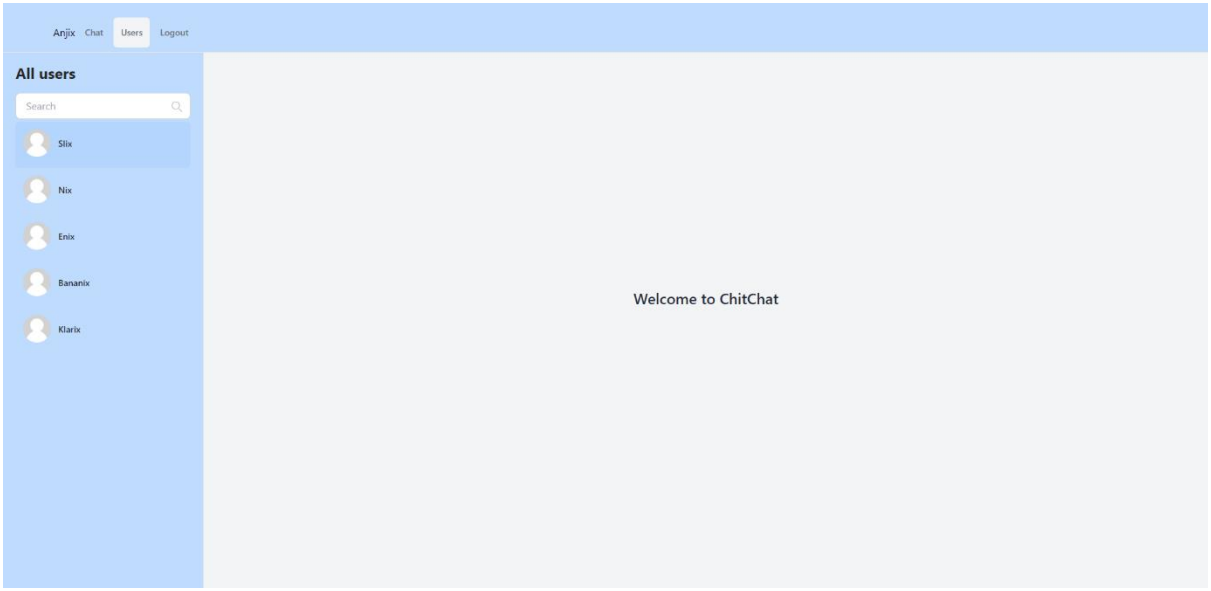
Slika 16 prikazuje kako se korisnik povezuje s bazom kada se želi prijaviti ili registrirati. Ako se korisnik odluči za registraciju dogodit će se samo dio od linije 52 do linije 63, no ako korisnik već ima račun te se odluči prijaviti, dogodit će se dio od 64 linije koda pa sve do 82 linije. Kada korisnik kreira račun njegovi podaci se prosljeđuju u bazu te se njemu automatski vrši prijava (linija 60), gdje se uspoređuju njegovi podaci s podacima iz baze te ako je sve u redu korisnika se prosljeđuje na stranicu /razgovori (eng. /conversations).

Slika 16. Izgled dijela koda iz skripte
AuthForm.tsx

Slika 17 prikazuje skriptu EmptyState.tsx, dok slika 18 prikazuje tu skriptu u aplikaciji koja se zapravo poziva kada se korisnik prijavi ili registrira te prikazuje dobrodošlicu na stranicu sa samim imenom stranice.

```tsx
import getCurrentUser from '@/app/actions/getCurrentUser';


5+ usages  Matej-Omazic
const EmptyState = async () : Promise<Element>  => {

  const currentUser : User | null  = await getCurrentUser();

  let divClassName : string  = `px-4 py-10 sm:px-6 lg:px-8 lg:py-6 h-full flex justify-center items-center`;
  let h3ClassName : string  = `mt-2 text-2xl font-semibold`;

  if (currentUser?.isDarkTheme) {
    divClassName += ` bg-gray-800`;
    h3ClassName += ` text-gray-100`;
  } else {
    divClassName += ` bg-gray-100`;
    h3ClassName += ` text-gray-800`;
  }

  return (
      <div className={divClassName}>
        <div className="text-center items-center flex flex-col">
          <h3 className={h3ClassName}>
          Welcome to ChitChat
        </h3>
      </div>
    </div>
  );
}

```

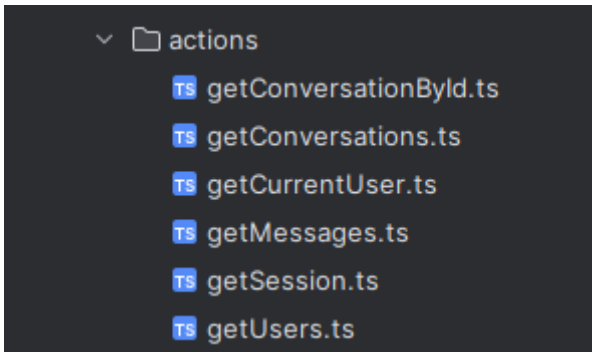
Slika 17. Izgled skripte EmptyState.tsx



Slika 18. Izgled EmptyState.tsx u aplikaciji

5.2.2 actions

Mapa „actions“ koja je vidljiva na slici 19 sadrži pozive tablica za razgovore, poruke i korisnike iz baze koristeći Prismu.



Slika 19. Izgled mape actions

Po samome imenu vidljivo je što koja skripta poziva, dok je na slici 20 vidljiv izgled za samog korisnika. Korisnik se traži po e-mail adresi, jer svaki korisnik mora imati različitu e-mail adresu, dok ostale stvari mogu biti iste. U skripti se poziva i getSession koji je povezan s API-ijem, sučeljem za programiranje aplikacija (eng. Application Programming Interface) te se preko toga dobiva autentifikacija korisnika.

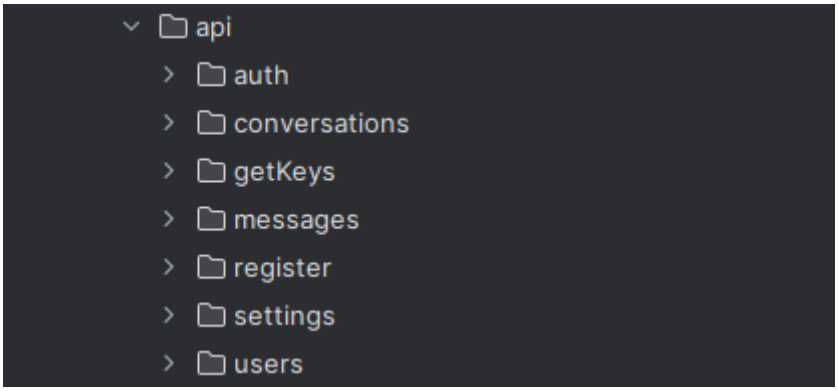
```
1  import prisma from "@/app/libs/prismadb";
2  import getSession from "./getSession";
3
   14+ usages  Matej-Omazic
4  const getCurrentUser = async () : Promise<...>  => {
5    try {
6      const session : Session | null  = await getSession();
7
8      if (!session?.user?.email) {
9        return null;
10     }
11
12     const currentUser : User | null  = await prisma.user.findUnique( args: {
13       where: {
14         email: session.user.email as string
15       }
16     });
17
18     if (!currentUser) {
19       return null;
20     }
21
22     return currentUser;
23   } catch (error: any) {
24     return null;
25   }
26 };
27
   5 usages  Matej-Omazic
28 export default getCurrentUser;
```

Slika 20. Izgled skripte getCurrentUser

5.2.3 api

Api mapa prikazana na slici 21 sadrži sve krajnje točke (eng. endpoint).Svaki endpoint odgovara određenom URL-u i HTTP metodi (GET, POST, PUT, DELETE itd.). Odgovorna je za rukovanje dolaznim zahtjevima i pružanje odgovarajućih odgovora. GET metoda je za dohvaćanje stvari, POST metoda je za objavljivanje podatka, dok PUT isto služi za objavljivanje, ali i za ažuriranje podatka dok DELETE metoda služi za brisanje podatka.

API endpoint je digitalno mjesto gdje API prima zahtjeve o određenom resursu na svom poslužitelju. U API-ima, endpoint je obično uniformni lokator resursa (URL) koji pruža lokaciju resursa na poslužitelju. Klijent šalje zahtjev API-ju aplikacije kako bi zatražio resurs iz baze podataka ili izvršio neku akciju na poslužitelju. Nakon što primi i potvrdi zahtjev klijenta, API izvršava zatraženu akciju, a zatim šalje odgovor natrag klijentu. Ovaj odgovor uključuje status zahtjeva (npr. dovršen ili odbijen) i sve resurse koje je klijent zatražio (Juviler, 2023).



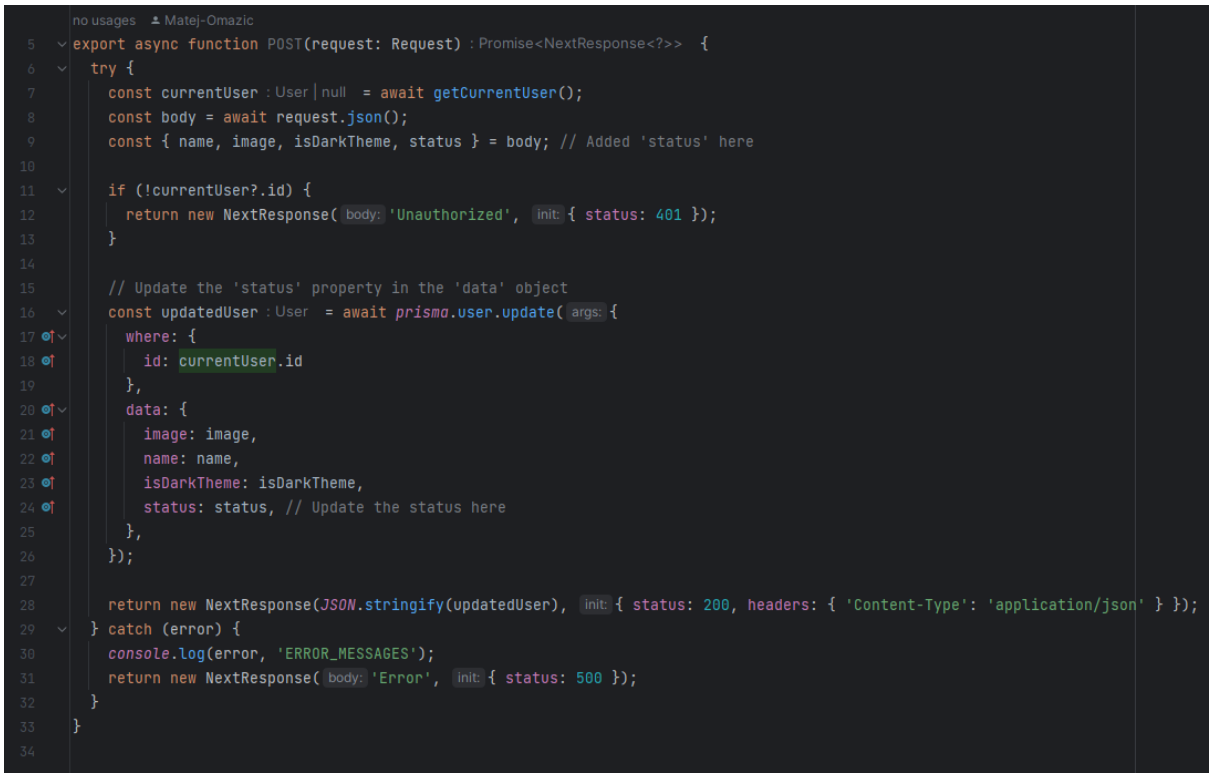
Slika 21. Izgled mape api

Autentifikacija korisnika se vrši preko route.ts, međutim svaka skripta u api mapi se naziva route.ts, te iz tog razloga su sve te route.ts stavljene u različite mape, tako se route.ts za autentifikaciju nalazi u mapi auth. Slika 22 prikazuje route.ts iz auth te se vidi funkcije autoriziranja, kada korisnik pošalje svoje podatke oni se uspoređuju u ovoj funkciji, naravno ako nešto nije u redu, vratit će se greška. Ako je sve u redu vraćaju se korisnikove informacije te se korisnik prosljeđuje na početnu stranicu.

```
    CredentialsProvider( options: {
      name: 'credentials',
      credentials: {
        email: { label: 'email', type: 'text' },
        password: { label: 'password', type: 'password' }
      },
      async authorize(credentials : Record<"email" | "password", s... ) : Promise<Prisma__UserClient<...>> {
        if (!credentials?.email || !credentials?.password) {
          throw new Error( message: 'Invalid credentials');
        }

        const user : User | null = await prisma.user.findUnique( args: {
          where: {
            email: credentials.email
          }
        });

        if (!user || !user?.hashedPassword) {
          throw new Error( message: 'Invalid credentials');
        }

        const isCorrectPassword : boolean = await bcrypt.compare(
          credentials.password,
          user.hashedPassword
        );

        if (!isCorrectPassword) {
          throw new Error( message: 'Invalid credentials');
        }

        return user;
      }
    })
  ],
  debug: process.env.NODE_ENV === 'development',
  session: {
    strategy: "jwt",
  },
  secret: process.env.NEXTAUTH_SECRET,
}
```

Slika 22. Izgled route.ts iz auth mape

Primjer POST metode vidljiv je na slici 23 gdje se mijenjaju neke informacije o korisniku. Na front-end dijelu aplikacije ova se metoda poziva kroz postavke. Za početak se uzima korisnik, te njegove stvari koje se mogu promijeniti (slika i ime). Ako korisnik ne postoji ili postavke nisu od tog korisnika javlja se greška, te da korisnik nema pristup promjeni. Ako je sve u redu, korisnik ima opciju promjene svoje slike i imena, kada ažurira nešto od toga, događa se upiti koji traži korisnika s tim ID-ijem te mu ažurira tu informaciju za novu.



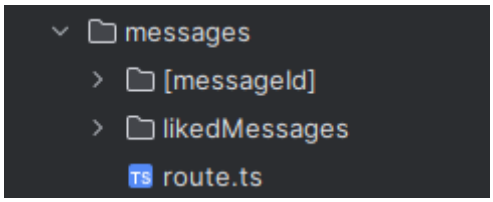
Slika 23. Izgled route.ts iz settingsa

Slika 24 prikazuje možda i najbitniji endpoint, to je dobivanje samog korisnika. Poziva se pomoću GET metode gdje se pretražuje korisnik po njegovom ID-iju (polje koje svaki korisnik ima drugačije, koje se dobije prilikom kreiranja samog korisnika).

```
6
    1 usage   Matej-Omazic
7   interface IParams {
8     id?: string;
9   }
10
    no usages   Matej-Omazic
11  export async function GET(
12    request: Request,
13    { params }: { params: IParams }
14  ) : Promise<...>  {
15    try {
16      const { id : string | undefined  } = params;
17
18      const userDetails : User | null  = await prisma.user.findUnique( args: {
19        where: {
20          id
21        }
22      });
23
24      return NextResponse.json(userDetails)
25    } catch (error) {
26      return NextResponse.json( body: null);
27    }
28  }
```

Slika 24. Izgled route.ts iz user-a

U message mapi imamo dvije podmape i jednu skriptu kao što je vidljivo na slici 25. Route.ts skripta je skripta koja koristi POST metodu za poruke u razgovoru. Dinamička mapa [messageId] i mapa likeMessages u sebi imaju po jednu route.ts skriptu.

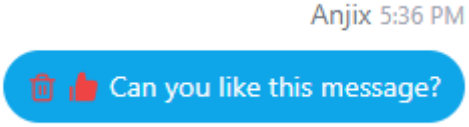


Slika 25. Izgled mape messages

Route.ts iz [messageId] služi za brisanje jedne poruke u razgovoru te korisnik naravno može obrisati samo svoje poruke. Izgled upita u skripti vidljiv je na slici 26, dok u aplikaciji to izgleda kao na slici 27, kada stisnemo na crveni koš poruka se obriše.

```
1 usage   Matej-Omazic
export async function DELETE(
    request: Request,
    { params }: { params: IParams }
) : Promise<...>  {
    try {
        const { messageId : string | undefined  } = params;
        const currentUser : User | null  = await getCurrentUser();

        if (!currentUser?.id) {
            return NextResponse.json( body: null);
        }

        const existingMessage : (Message & {conversation: (Con...  = await prisma.message.findUnique( args: {
            where: {
                id: messageId
            },
            include: {
                conversation: {
                    include: {
                        users: true
                    }
                }
            }
        });
```

Slika 26. Izgled upita za birsanje jedne poruke

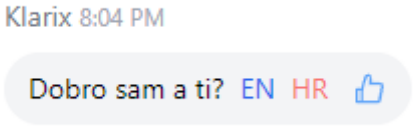


Slika 27. Izgled gumba za birsanje poruke u aplikaciji

Mapa likedMessages zapravo u sebi ima još jednu dinamičku mapu istu kao kod brisanja poruka gdje se koristi messageId te onda slije route.ts pomoću koje se određuje je li se poruka korisniku sviđa ili ne (eng. like). Na slici 28 vidljivo je kako izgleda skripta za sviđanje poruke, dok je na slici 29 vidljivo kako izgleda gumb kada se poruka želi lajkati.

```
import { NextResponse } from "next/server";

import getCurrentUser from "@/app/actions/getCurrentUser";
import { pusherServer } from '@/app/libs/pusher'
import prisma from "@/app/libs/prismadb";


no usages  Matej-Omazic
export async function POST(
    request: Request,
) : Promise<NextResponse<?>>  {
  try {
    const currentUser : User | null  = await getCurrentUser();
    const body = await request.json();
    const {
      message,
      image,
      conversationId,
      encryptedMsg,
      isLiked
    } = body;

    if (!currentUser?.id || !currentUser?.email) {
      return new NextResponse( body: 'Unauthorized',  init: { status: 401 });
    }

```

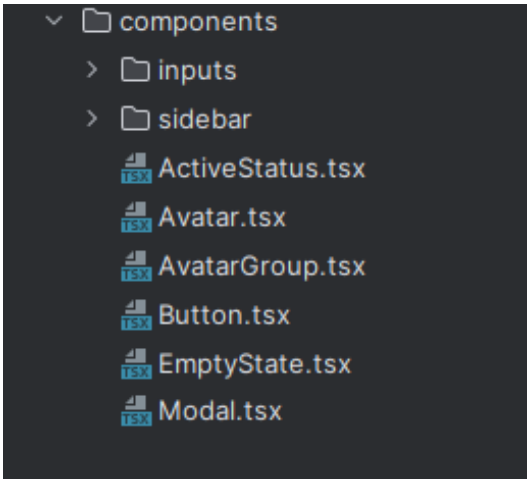
Slika 28. Izgled route.ts skripte za lajkanje poruke



Slika 29. Izgled gumba za lajkanje poruke

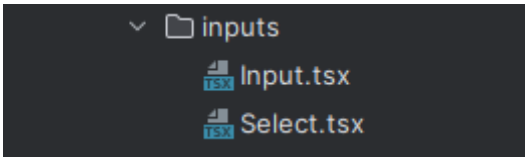
5.2.4 components

Components mapa vidljiva na slici 30 kao što joj i samo ime kaže sadrži komponente aplikacije. To su zapravo dijelovi koji se koriste više puta zbog toga radi posebna skripta i iz razloga smanjivanja koda.



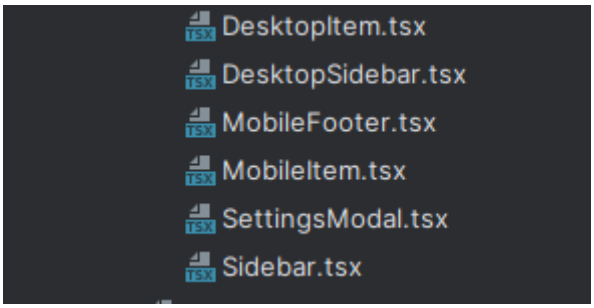
Slika 30. Izgled mape components

Unosi (eng. inputs) su dio mape components te u sebi sadrže dvije skripte koje su vidljive na slici 31. Input.tsx skripta se koristi uvijek kada se nešto unosi u aplikaciji, a to je tijekom prijave i registracije. Kada se pretražuju korisnici, te kada se upisuju poruke. Select komponenta se koristi kada se pravi grupni razgovor (eng. group chat) te se tu dodaje više korisnika koji se odabiru (eng. select).



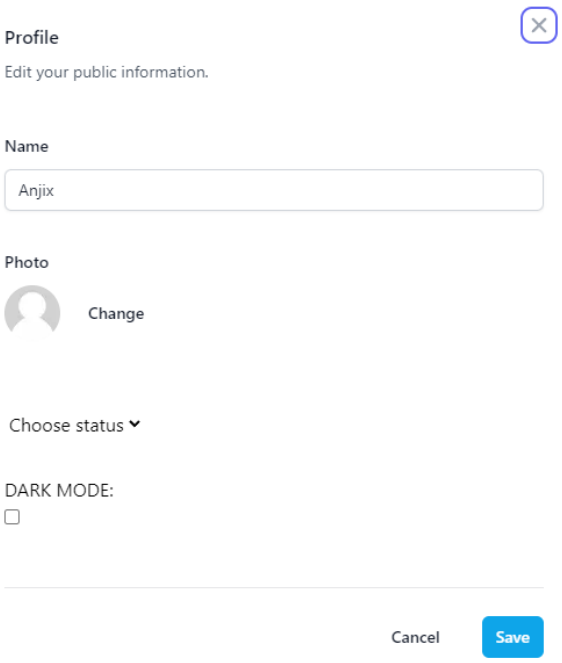
Slika 31. Izgled mape inputs

Bočna traka (eng. sidebar) predstavlja dio mape components te u sebi sadrži još šest skripti. Slika 32 prikazuj listu svih skripti.



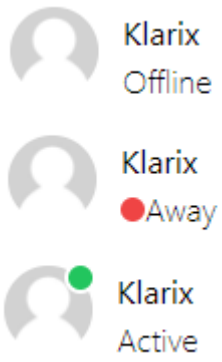
Slika 32. Izgled mape sidebar

Bočna traka se sastoji od glavne skripte u kojoj su pozvane preostale četiri skripte. Skripta se poziva ovisno o veličini prozora aplikacije. Kada je prozor normalno otvoren pozivaju su desktop skripte gdje je napravljen sidebar te su ikone prilagođene za računalni pogled, međutim kada je prozor aplikacije smanjen do određenog dijela, pozivaju se mobile skripte gdje se aplikacije prilagođava izgledu mobitela te su same ikone prilagođene za mobilni pogled. Skripta SettingsModal.tsx je namijenjena za ažuriranje određenih podataka od korisnika te je prikazana na slici 33.



Slika 33. Izgled settingsa u aplikaciji

Na slici 25 vidljive su ostale skripte. Skripta ActiveStatus.tsx prikazuje aktivnost korisnika, tj. ako je korisnik trenutno u aplikaciji uz njegovu ikonu, bit će dodan mali zeleni kružić. Avatar.tsx prikazuje sliku od korisnika te se koristi u razgovoru s tim korisnikom, te također u navigacijskoj traci za pretraživanje svih korisnika te u traci svih razgovora. AvatarGroup.tsx je slika koja se koristi kad je u razgovoru više od dva korisnika te je zadana slika s više korisnika. Button.tsx se koristi za svaki gumb u aplikaciji. Veličina gumba može se mijenjati međutim sve ostalo je isto na svakom gumbu, a on se koristi kod prijave/registracije, te kod dodavanja više korisnika u razgovor te kod brisanja samog razgovora. EmptyState.tsx je prazno stanje koje donosi dobrodošlicu korisniku. Slika 34 prikazuje razne statuse korisnika. Korisnik ima tri statusa koja može odabrati te jedno koje je nepromjenjivo. Može biti online, može biti away, te može biti nevidljiv (eng. invisible), to stanje je prikazano kao Offline, također Offline je prikazano i kada je korisnik stvarno odjavljen iz aplikacije.



Slika 34. Prikaz korisnikovih statusa

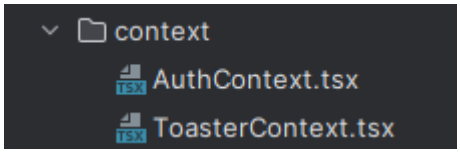
Na slici 35. vidljivo je kako izgleda skripta Avatar.tsx koja zapravo na temelju korisnikovog mail dolazi do samog korisnika, uzima njegovu sliku i stavlja u div. Onda se dodaju različiti CSS stilovi putem Tailwind-a.

```tsx
'use client';

import {User} from "@prisma/client";

import useActiveList from "../hooks/useActiveList";
import Image from "next/image";

1 usage  Matej-Omazic
interface AvatarProps {
  user?: User;
};

8+ usages  Matej-Omazic *
const Avatar: React.FC<AvatarProps> = ({ user : User | undefined  }) => {
  const { members : string[]  } = useActiveList();
  const isActive : boolean  = members.indexOf(user?.email!) !== -1;

  return (
    <div className="relative">
      <div className="relative inline-block rounded-full overflow-hidden h-9 w-9 md:h-11 md:w-11">
        <Image fill src={user?.image || '/images/placeholder.jpg'} alt="Avatar"/>
      </div>
      {isActive ? (
        <span
          className="absolute block rounded-full bg-green-500 ring-2 ring-white top-0 right-0 h-2 w-2 md:h-3 md:w-3"/>
      ) : null}
    </div>
  );
}

5+ usages  Matej-Omazic
export default Avatar;
```

Slika 35. Izgled skripte Avatar.tsx

5.2.5 context

Context mapa sadrži skripte koje su vidljive na slici 36. Context je metoda za prosljeđivanje rekvizita (eng. props) od roditelj do dijete komponente, bez stvarnog ručnog prosljeđivanja na svakoj razini komponente. Context je dizajniran kako bi olakšao dijeljenje podataka u aplikacijama. Podaci koji se ne moraju često ažurirati trebaju biti postavljeni na React context. Zato što context nije napravljen kao cjelovit sustav upravljanja stanjem. Napravljen je kako bi olakšao konzumaciju podataka (Kaushika, 2023).



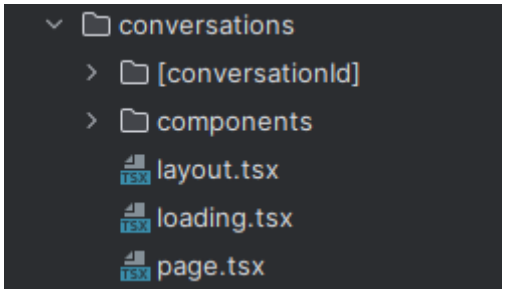
Slika 36. Izgled mape context

ToasterContext.tsx prosljeđuje tj. samo prima Toaster obavijesti (slika 37), kao npr. ako dođe do uspješne prijave, doći će poruka s naslovom „Logged in“ ili ako dođe do pogreške „Login error“. AuthContext koristi SessionProvider, a to je komponenta koju pruža NextJS i omogućava dijeljenje podataka o sesiji između komponenti pomoću React Context-a. Koristeći SessionProvider, instance useSession() mogu dijeliti objekt sesije između komponenti.

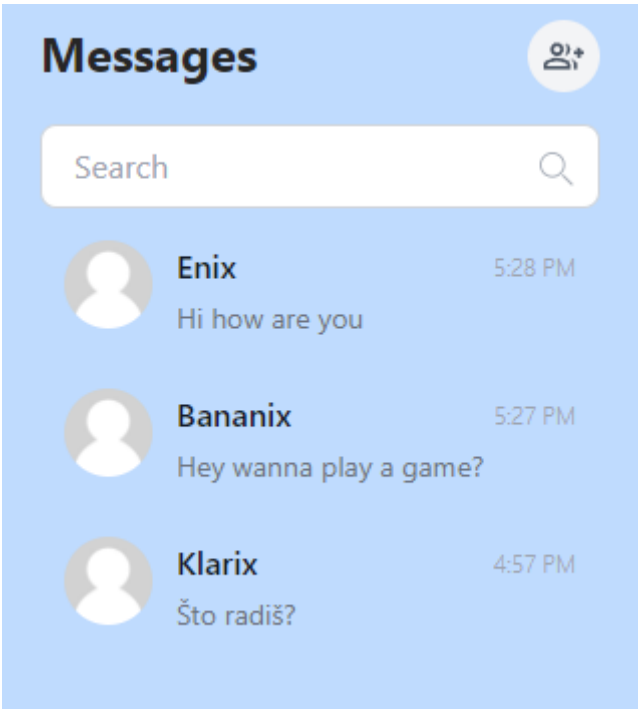
Slika 37. Prikaz Toast obavijest za uspješnu prijavu

5.2.6 conversations

Slika 38 prikazuje mapu razgovori, to je mapa koja sadrži sve bitno o razgovorima i čavrljanju. Page.tsx skripta koristi prije spomenuti EmptyState.tsx koji korisniku želi dobrodošlicu. Loading.tsx je mali loading ekran koji se koristi po cijeloj aplikaciji, no ovdje je korišten samo pri učitavanju u sam razgovor, te layout.tsx (slika 39) koji se sastoji od trake za pretraživanje (eng. search bar) te svih korisnika s kojima postoji minimalno jedna poruka, tj. sa svim korisnicima s kojima je započeta komunikacija.

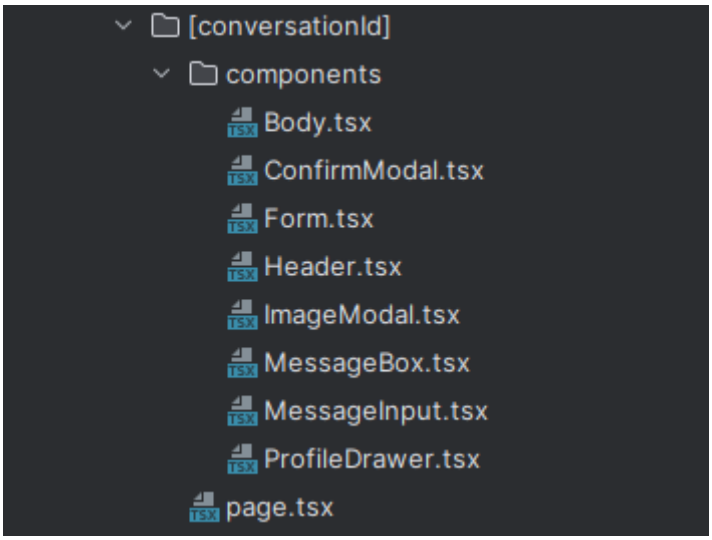


Slika 38. Izgled mape conversations



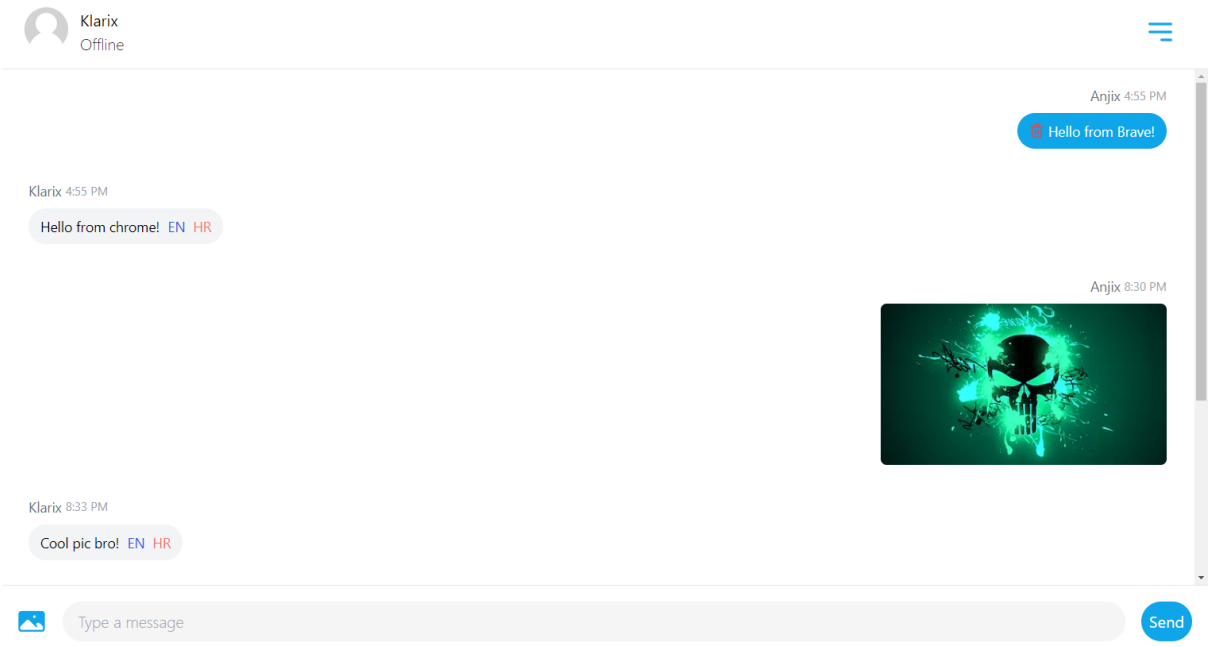
Slika 39. Izgled skripte layout.tsx u aplikaciji

Slika 40 prikazuje dinamičku mapu conversationsId. Dinamička mapa se koristi kada se unaprijed ne zna točan naziv segmenata, a želi se kreirati ruta iz dinamičkih podataka, koja se popunjava u trenutku zahtjeva. Kada korisnik odabere razgovor s nekim drugim korisnikom automatski se kreira id za taj razgovor, no kako je on kod svakog razgovora drugačiji, zato se koristi dinamička mapa (NextJSTeam, 2023). Dinamička mapa se koristi kod endpointa kako bi se pozvao API, a ona se „stvara“ tako da kad kreiramo mapu. Naziv mora biti u uglatim zagradama. Page.tsx skripta je jedina skripta koja se ne nalazi u komponentama, već u samoj components mapi, te ona pokazuje EmptyState.tsx.



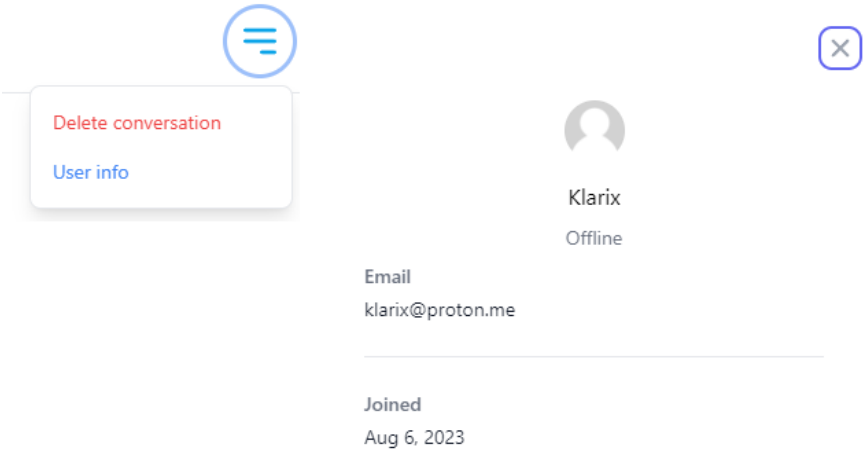
Slika 40. Izgled mape conversationsId

Mapa components u mapi conversationsId sadrži dosta dodatnih skripti od kojih je najbitnija Body.tsx. U toj skripti se pozivaju sve ostale skripte iz mape components. Slika 41 prikazuje skriptu Body.tsx.



Slika 41. Izgled Body.tsx skripte

Skripta Header.tsx koja je prikazana na slici 42 u sebi sadrži izbornik pomoću kojega možemo obrisati razgovor s korisnikom ili vidjeti njegove informacije.



Slika 42. Izgled Header.tsx u aplikaciji

Form.tsx je forma gdje se upisuje poruka, tj. ona poziva MessageInput.tsx u koji se zapravo upisuje poruka, te se šalje u bazu. Prije nego što se poruka pošalje bazi, koristi se paket E2EE, koji uzima poruku i javni ključ korisnika, te enkriptira poruku kako ona ne bi bila vidljiva u bazi (slika 43).

```
1 usage  Matej-Omazic
const onSubmit: SubmitHandler<FieldValues> = async (data : FieldValues ) : Promise<void>  => {
  setValue( name: 'message',  value: '',  options: { shouldValidate: true });
  const encryptedMsg : {cipher_text: string, aes_key:...  = await E2EE.encrypt( public_key: localStorage.getItem(recipientId) || "", data.message)
  axios.post( url: '/api/messages',  data: {
    ...data,
    encryptedMsg,
    conversationId: conversationId
  })
}
```

Slika 43. Funkcija enkripcije poruke

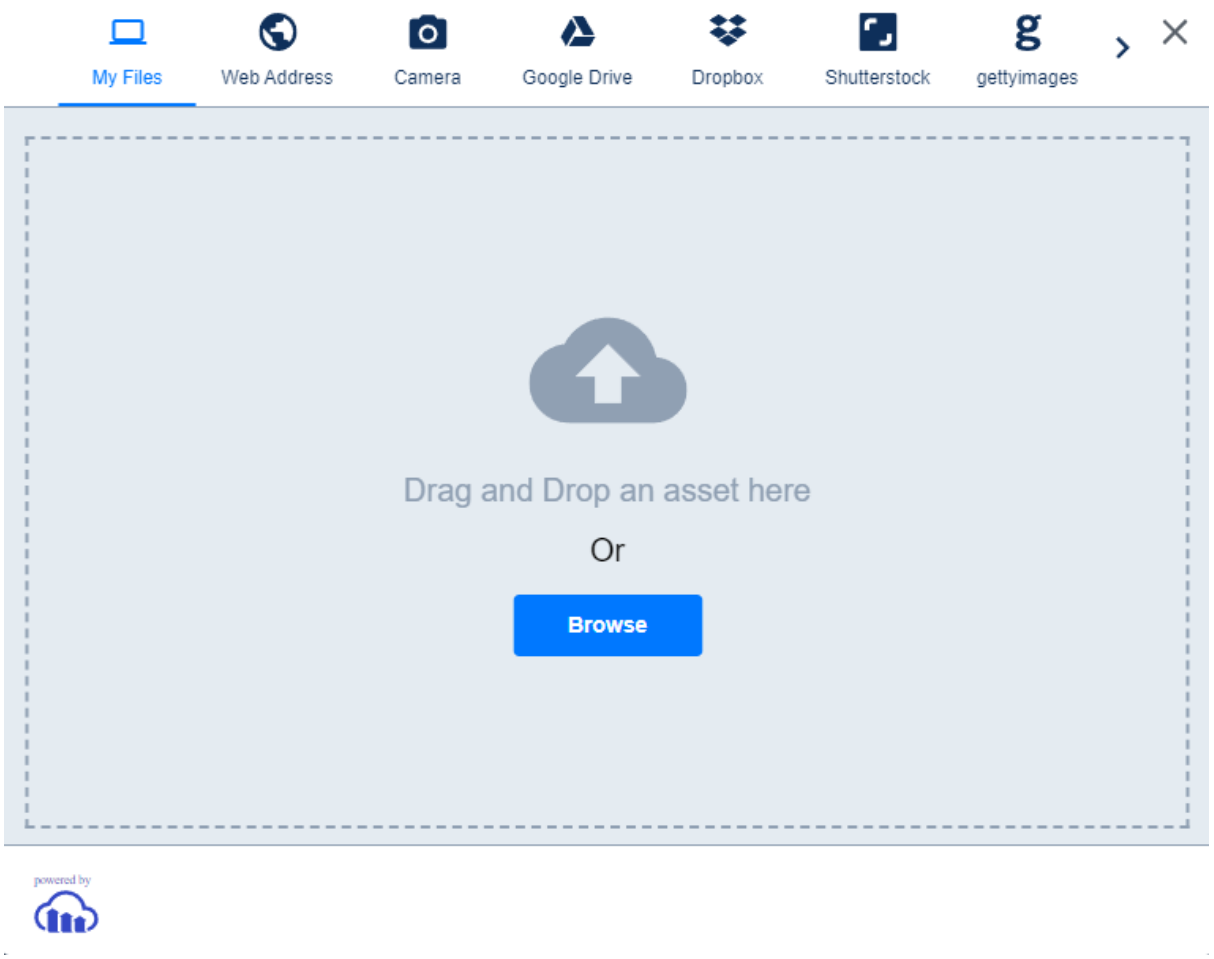
MessageBox.tsx je skripta koja prikazuje sve poruke iz određenog razgovora na ekran. Prije pokazivanja same poruke, dohvaćamo poruku iz baze, te prije nego se prikaže poruka se mora dešifrirati koristeći paket E2EE (slika 44), privatni ključ korisnika, te poruku iz baze. Nakon toga poruka se prikazuje u aplikaciji.

```
    console.log("MSG: ", data.body)
    console.log("MINE: ", isOwn)

    if(isOwn){
      setRawMessage( value: data.body || "")
    }
    else{
    E2EE.decrypt(data.encryptedMsg?.aes_key!, data.encryptedMsg?.iv!,
        localStorage.getItem( key: "private_key")! ,data.encryptedMsg?.cipher_text!).then(msg : string  =>{
      setRawMessage(msg)

    })
  }
  }, deps: [])
```

Slika 44. Funkcija dekripcije poruke

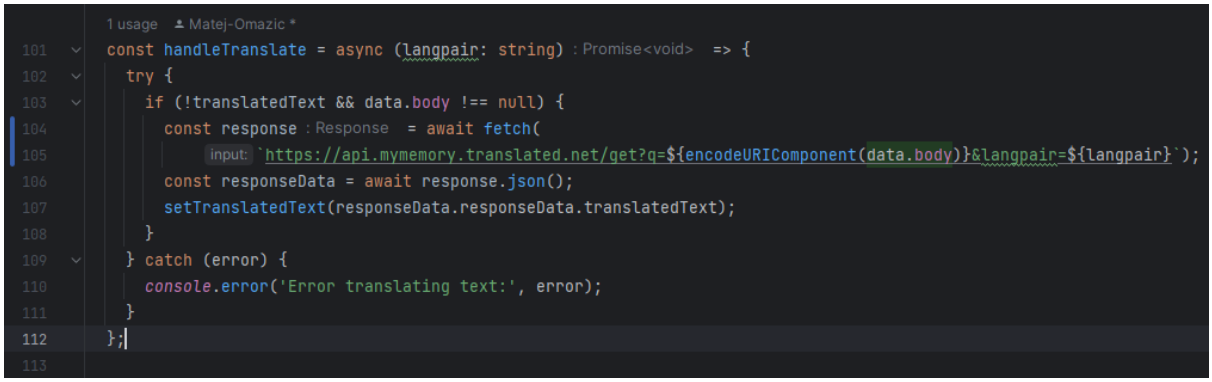
Kako se Form.tsx služi za slanje poruka, osim poruka u aplikaciji mogu se poslati i datoteke. Tu se koristi ekspanzija Cloudinary. Cloudinary je end-to-end rješenje za upravljanje slikama i video zapisima za web-mjesta i mobilne aplikacije, koje pokriva sve od učitavanja slika i videa, pohranjivanja, manipulacija, optimizacija do isporuke (CloudinaryTeam,2023). Slika 45 prikazuj formu za slanje poruka, dok slika 46 prikazuje Cloudinary ekspanziju.

Slika 45. Izgled forme za poruke

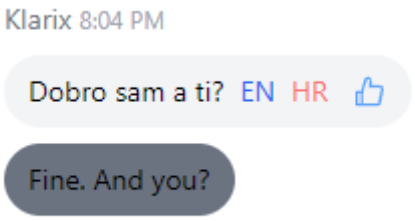


Slika 46. Izgled Cloudinaryija u aplikaciji

MessageBox.tsx skripta zadužena je za pokazivanje poruke u obliku male kutijice. Kada se poruka prikazuje na zaslonu ona se ujedno i dekriptira. U MessageBox.tsx dolazi sama poruka te ovisno ako je poruka poslana uz nju dolazi gumb za lajkanje tj. je li poruka dobila reakciju ili nije te s gumbom za brisanje poruke. Međutim ako je poruka dobivena ta poruka također prikazuje gumb za lajkanje gdje se njoj daje reakcija te dolazi s dva gumba koja prevode poruke. Ovisno ako je poruka na hrvatskome ili engleskome jeziku. Ako je poruka na engleskome koristit će se gumb za prijevod na hrvatski i obrnuto. Na slici 47 vidljiva je funkcija koja uzima poruku i stavlja ju u API poziv kako bi se ona prevela. Na slici 48 vidljivo je izgled u aplikaciji.

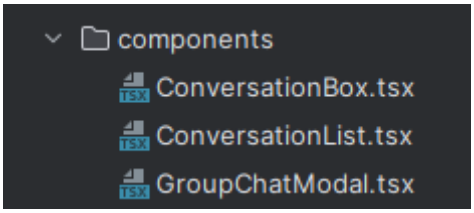


Slika 47. Izgled funkcije za prijevod poruke



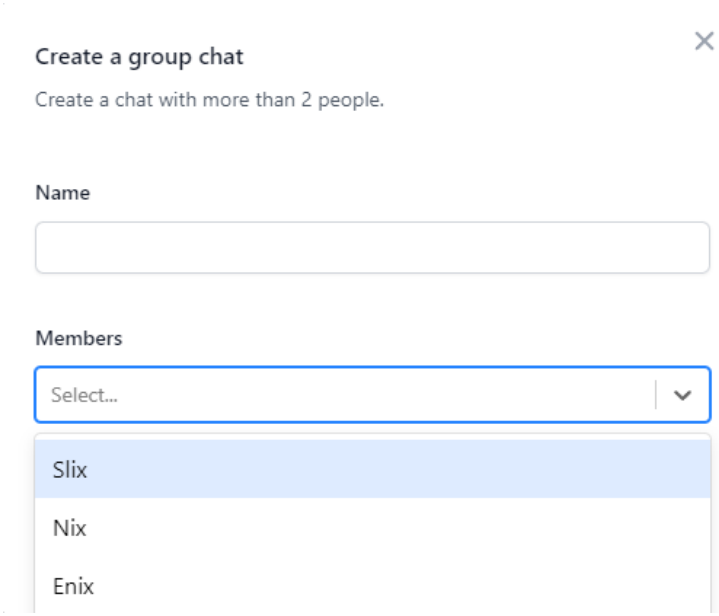
Slika 48. Prikaz prevođenja u aplikaciji

Preostala je mapa components koja se nalazi unutar mape conversations te 49 prikazuje 3 skripte koje su unutar mape components.



Slika 49. Izgled mape components

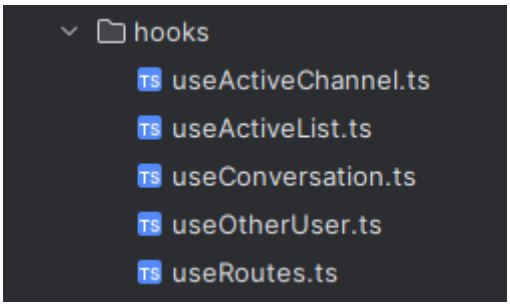
Skripta ConversationBox.tsx u sebi poziva korisnika s kojim se komunicira, njegovu sliku te zadnju poruku u razgovoru s tim korisnikom. ConversationList.tsx u sebi poziva ConversationBox.tsx i tako dobiva listu svih korisnika s kojima je razmijenjena minimalno jedna poruka te na temelju toga dobijemo malu kutiju kao na slici 45. GroupChatModal.tsx je skripta koja nam prikazuje prozor u kojem postoji mogućnost izrade razgovora s minimalno 2 dodatna korisnika te taj prozor se vidi na slici 50.



Slika 50. Izgled izrade razgovora sa više korisnika

5.2.7 hooks

Slika 51 prikazuje mapu hooks sa svim svojim skriptama. Kuke (eng. hooks) su JavaScript funkcije koje upravljaju ponašanjem stanja tako što ih izoliraju od komponente. Kuke su način na koji se funkcijske komponente "prikače" na React-ovo stanje. Predstavljeni su u Reactu 16.8.0. Ranije su samo komponente temeljene na klasi mogle koristiti React-ovo stanje. Osim što omogućuju funkcijskim komponentama da to učine, hook-ovi nevjerojatno olakšavaju ponovnu upotrebu logike stanja između komponenti (Tapas, 2022).



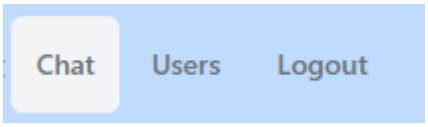
Slika 51. Izgled mape hooks

React ima svoje već napravljene (eng. built in) hook-ove, no nekada potrebno je koristiti sličnu logiku za svoje funkcije kako bi se dobili ono željeno. UseActiveList.ts ima par napravljenih funkcija koje se koriste u UseActiveChannel.ts. On služi za korištenje Pushera (WebSocketa) u aplikaciji. Pretplaćuje (eng. subscribe) se na korisnika ovisno gdje se on nalazi u aplikacija, kako bi se sve događalo u stvarnom vremenu. UseConversation.ts se brine za razgovore između korisnika, kada nađemo korisnika, te kada otvorimo razgovor s njime. UseOtherUser.ts se povezuje na UseConversation zato što kada se otvori razgovor s korisnikom, preko UseOtherUser, lakše se koriste podatci tog drugog korisnika.

UseRoutes.ts se koristi za kretanje po aplikaciji. Tu se nalaze sve fiksne rute u aplikaciji. Slika 52 prikazuje izgled koda te skripte te je na slici 53 vidljiv izgled u aplikaciji. Postoji ruta za razgovore (eng. conversations) koja pokazuje sve korisnike s kojima se razmijenila minimalno jednu poruku, ruta za korisnike (eng. users) koja prikazuje sve korisnike koji imaju račun u aplikaciji, te ruta za odjavu (eng. logout).

```
5+ usages  Matej-Omazic
const useRoutes = () => {
    const pathname : string | null  = usePathname()
    const {conversationId : string } = useConversation()

    const routes : ({label: string, href: string,...  = useMemo( factory: () => [
        {
          label: 'Chat',
          href: '/conversations',
          icon: HiChat,
          text: "Chat",
          active: pathname === '/conversations' || !!conversationId
        },
        {
          label: 'Users',
          href: '/users',
          icon: HiUsers,
          text: "Users",
          active: pathname === '/users'
        },
        {
          label: 'Logout',
          onClick: () => signOut(),
          href: '#',
          text: "Logout",
          icon: HiArrowLeftOnRectangle,
        }
      ], deps: [pathname, conversationId]);

      return routes;
}

```

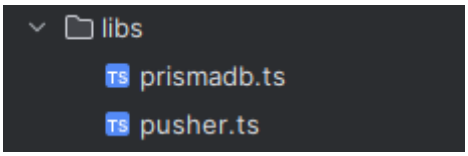
Slika 52. Izgled skripte useRoutes.ts



Slika 53. Izgled useRoutes u aplikaciji

5.2.8 libs

Mapa lib mjesto je gdje se drže sve skripte specifične za aplikaciju koje se globalno koriste u cijeloj aplikaciji. Slika 54 prikazuje izgled mape libs, te se u njoj mogu vidjeti još dvije skripte, jedna za Prisma Client i druga za Pusher (Tarik, 2023).



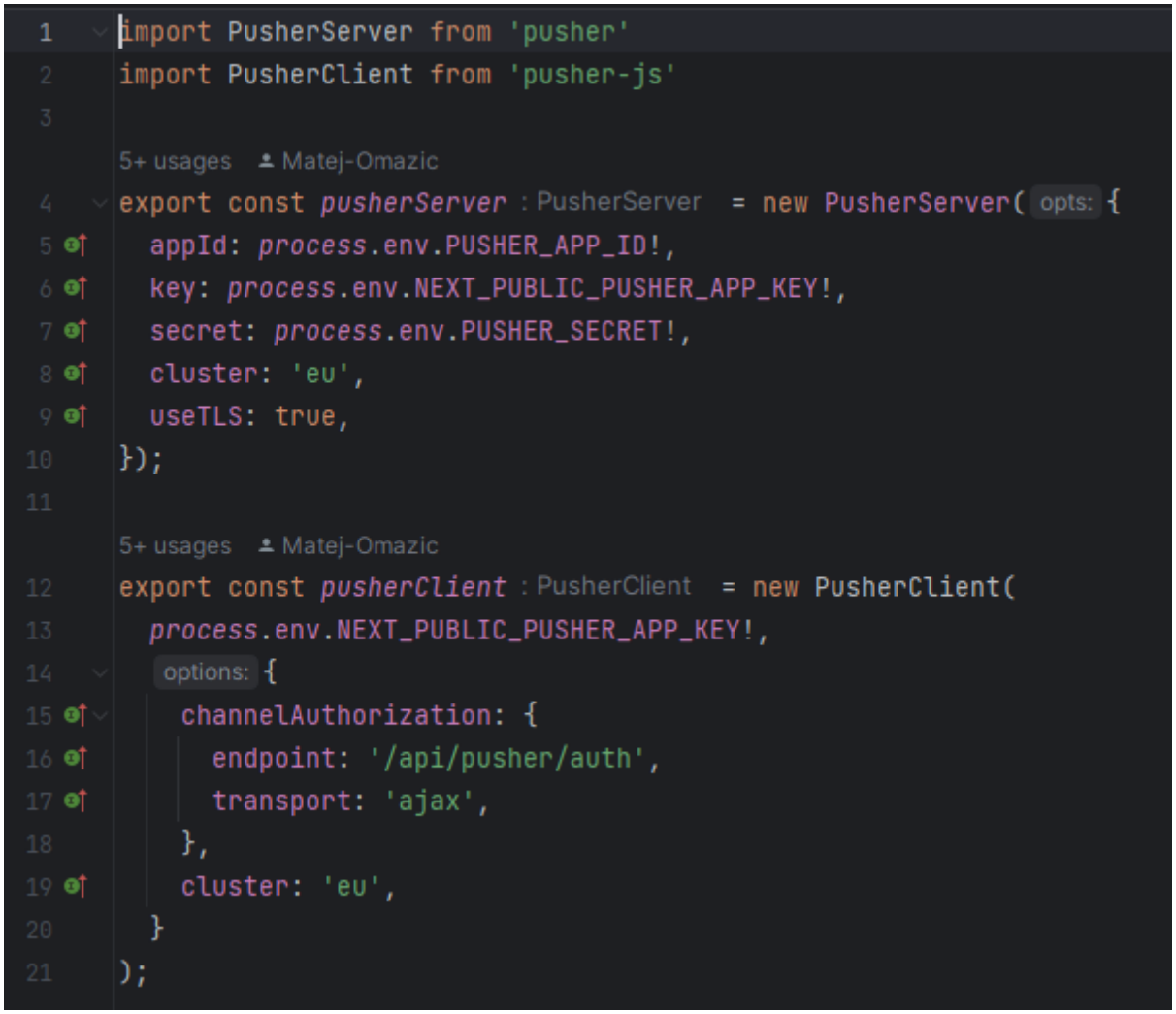
Slika 54. Izgled mape libs

Slika 55 prikazuje izgled koda za skriptu prismadb.ts. Po kodu se vidi da se povezuje na server od Prisma Client-a, te se to koristi u određenim dijelovima aplikacije.

```
1  import { PrismaClient } from "@prisma/client"
2
3  declare global {
     1 usage   Matej-Omazic
4    var prisma: PrismaClient | undefined
5  }
6
7  const client : PrismaClient<Prisma.PrismaClie...  = globalThis.prisma || new PrismaClient()
8  if (process.env.NODE_ENV !== "production") globalThis.prisma = client
9
   5+ usages   Matej-Omazic
10 export default client
```

Slika 55. Izgled koda za prismadb.ts

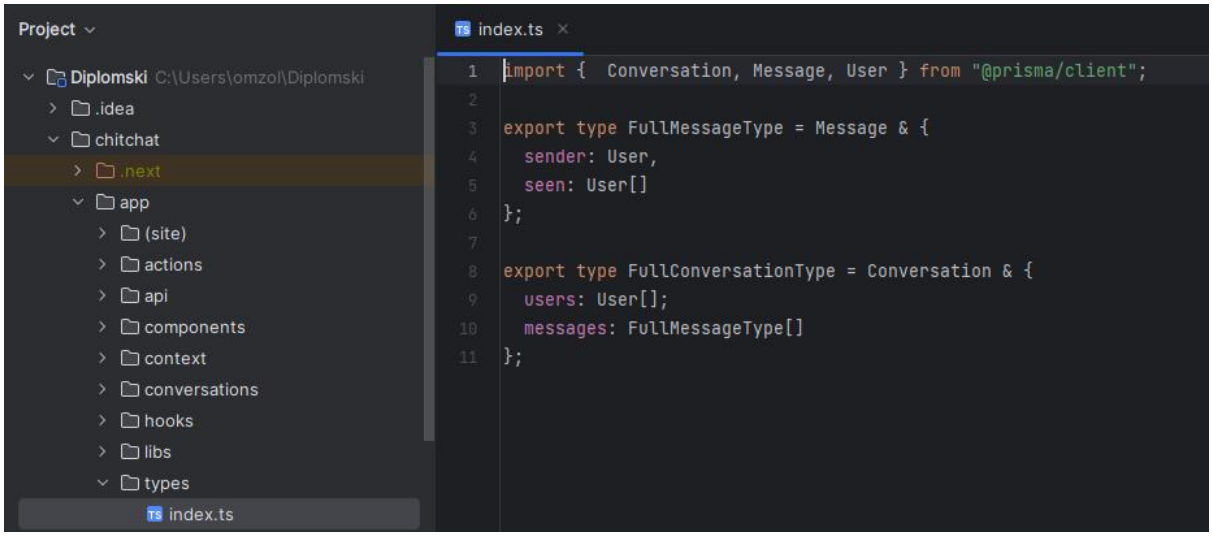
Slika 56 prikazuje izgled koda za pusher.ts. Povezuje se na pusher-ov server te na client. Prilikom spajanje koriste se određene opcije, kao npr. spajanje na grozd (eng. cluster, grupa servera) koji se nalazi u Europi, korištenje TLS-a (Transport Layer Security).



Slika 56. Izgled koda za pusher.ts

5.2.9 types

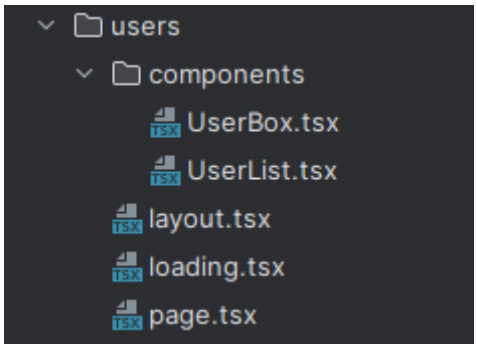
Slika 57 prikazuje mapu types u kojoj se nalazi samo skripta index.ts. Index.ts služi za vrstu razgovora. FullMessageType vraća jednog pošiljatelja poruke, međutim ako je vrsta razgovora grupna, može biti više korisnika koji će pogledati poruku, isto vrijedi i za FullConversationType. Može imati više korisnika, a ne samo jednog.



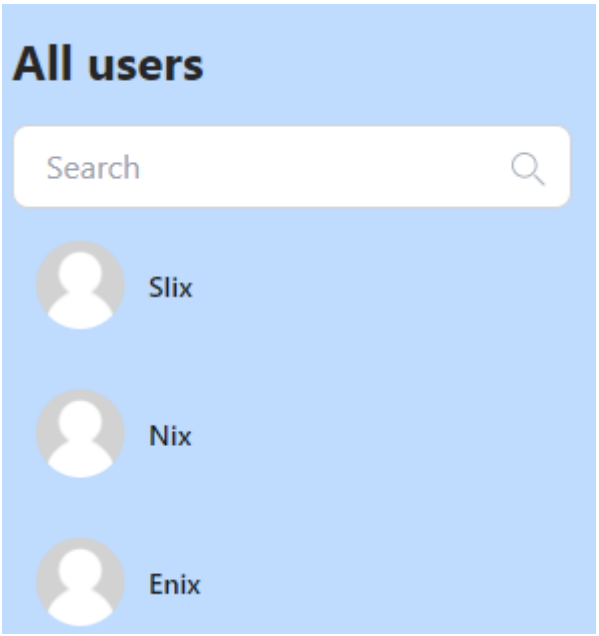
Slika 57. Izgled mape types i prikaz koda index.ts

5.2.10 users

Slika 58 prikazuje users mapu te sve unutar nje, jako slična mapi conversations, gdje u ovom slučaju su samo korisnici. Loading.tsx služi za animaciju učitavanja pri odlasku na stranicu za sve korisnike. Page.tsx koristi EmptyState.tsx, layout.tsx poziva sve ostale komponente uključujući i UserList.tsx koji poziva UserBox, te se na slici 59 vidljivo je kako izgleda u samoj aplikaciji s ispisom svih korisnika, te trakom za pretraživanje koja filtrira korisnike.



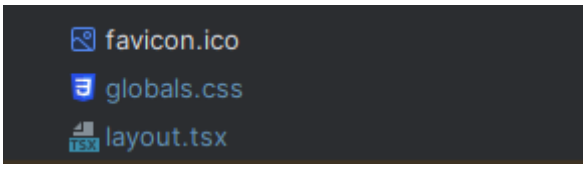
Slika 58. Izgled mape users



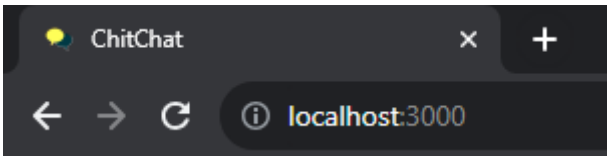
Slika 59. Izgled skripte layout.tsx u aplikaciji

5.2.11 ostale datoteke u mapi app

Preostale mape prikazane su na slici 60, favicon.ico služi za prikazivanje ikone u prozoru pretraživača (slika 61.)



Slika 60. Preostale mape u datoteci app



Slika 61. Izgled ikone u prozoru pretraživača

Global.css skripta se koristi kao što joj i samo ime kaže po cijelo aplikaciji te je zadužena za css. Slika 62 prikazuje izgled koda u skripti global.css. Body dijelovi koda će imati veličinu 100% (po zadanome), te će se koristiti Tailwind. Tailwind je CSS okvir koji pruža jednonamjenske klase koje su većinom samostalne i koje pomažu da se web stranica izravno iz oznaka (De Roy, 2022).

```
1  @tailwind base;
2  @tailwind components;
3  @tailwind utilities;
4
5  html,
6  body,:root{
7    height: 100%
8  }
```

Slika 62. Izgled skripte global.css

Slika 63 prikazuj glavni layout.tsx koji se sastoji od imena i opisa same aplikacije, koji su vidljivi kao i favicon u prozoru pretraživača, layout.tsx također poziva sve glavne funkcije i obavijesti.

```tsx
import './globals.css'
import AuthContext from './context/AuthContext'
import ActiveStatus from './components/ActiveStatus'
import ToasterContext from './context/ToasterContext'

no usages  Matej-Omazic
export const metadata : Metadata  = {
  title: 'ChitChat',
  description: 'Chat app',
}

no usages  Matej-Omazic
export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <html lang="en">
      <body>
        <AuthContext>
          <ToasterContext />
          <ActiveStatus />
          {children}
        </AuthContext>
      </body>
    </html>
  )
}
```

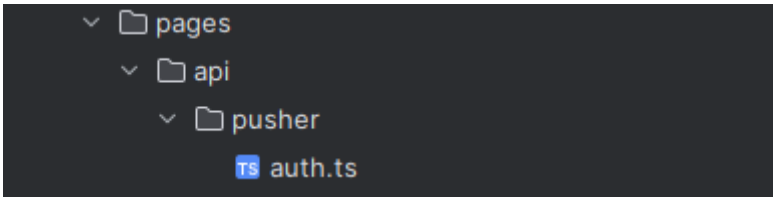
Slika 63. Izgled glavnog layout.tsx

5.3 node_modules

Node_modules mapa može se zamisliti kao predmemorija za vanjske module o kojima ovisi projekt. Kada se koristi npm install (eng. node package manager), moduli se preuzimaju s weba i kopiraju u mapu node_modules, a Node.js je „obučen“ da ih tamo traži kada ih se uveze (eng. import). To se naziva „predmemorijom“ jer se mapa node_modules može potpuno ponovno stvoriti u bilo kojem trenutku samo ponovnim instaliranjem svih zavisnih modula (Mouskhelichvili, 2022).

5.4 pages

Izgled mape pages prikazan je na slici 64. Mapa ima dodatne mape kako bi bilo jasnije. Koristi se API, te se na WebSocket povezuje putem Pushera.



Slika 64. Izgled mape pages

Slika 65 prikazuje izgled skripte auth.ts, gdje se poziva email od korisnika te se preko email-a povezuje na server od pushera. Nakon toga se pozivaju built in funkcije na mjestima gdje se žele koristiti WebSocket-i.

```ts
import { NextApiRequest, NextApiResponse } from "next"
import { getServerSession } from "next-auth";

import { pusherServer } from "@/app/libs/pusher";
import { authOptions } from "@/app/api/auth/[...nextauth]/route";

no usages  Matej-Omazic
export default async function handler(
  request: NextApiRequest,
  response: NextApiResponse
) : Promise<void | NextApiResponse...  {
  const session : Session | null  = await getServerSession(request, response, authOptions);

  if (!session?.user?.email) {
    return response.status( statusCode: 401);
  }

  const socketId = request.body.socket_id;
  const channel = request.body.channel_name;
  const data : {user_id: string}  = {
    user_id: session.user.email,
  };

  const authResponse : Pusher.ChannelAuthResponse  = pusherServer.authorizeChannel(socketId, channel, data);
  return response.send(authResponse);
};
```

Slika 65. Izgled auth.ts iz mape pages

5.5 prisma

Prisma mapa se koristi za skriptu schema.prisma gdje se zapravo radi shemu baze za podatke. Kodiranje na silu (eng. hardcoding) se koristi u ovoj skripti gdje mi zapravo u bazu šaljemo (Slika 66.) ono što će User imati, a to su id, ime, e-mail, potvrđen e-mail, sliku, lozinku, vrijeme kreiranje, te vrijeme ažuriranja. User će također biti povezan s različitim razgovorima i porukama.

```
13  model User {
14    id String @id @default(auto()) @map("_id") @db.ObjectId
15    name String?
16    email String? @unique
17    emailVerified DateTime?
18    image String?
19    hashedPassword String?
20    public_key String?
21    createdAt DateTime @default(now())
22    updatedAt DateTime @updatedAt
23    isDarkTheme Boolean @default(false)
24    status String?
25
26
27    conversationIds String[] @db.ObjectId
28    conversations Conversation[] @relation(fields: [conversationIds], references: [id])
29
30    seenMessageIds String[] @db.ObjectId
31    seenMessages Message[] @relation("Seen", fields: [seenMessageIds], references: [id])
32
33    accounts Account[]
34    messages Message[]
35  }
```

Slika 66. Ubacivanje user-a u bazu pomoću prisme

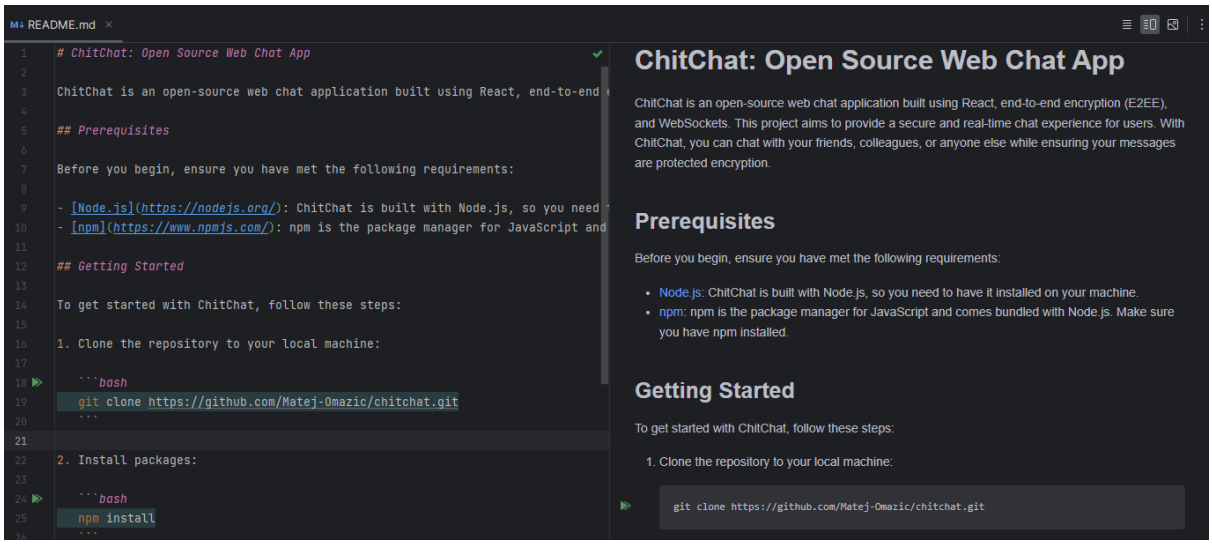
5.6 public

Public mapa sadrži uglavnom medij kao što su slike, međutim to nisu slike koje korisnik može razmjenjivati s drugim korisnicima, već su to slike koje se koriste u aplikaciju, kao što su logo aplikacije, zadana slika profila (eng. default profile picture)

5.7 ostale skripte

Od ostalih skripti bitno je spomenuti .env (dotenv) skriptu koja sadrži varijable okoline (eng. environment variables). Varijable okoline u Nodeu koriste se za pohranjivanje osjetljivih podataka kao što su lozinke, API akreditacije i druge informacije koje se ne smiju pisati izravno u kodu. Varijable okruženja moraju se koristiti za konfiguriranje svih varijabli ili konfiguracijskih detalja koji se mogu razlikovati između okruženja (Parthibakumar, 2022).

Skripta README.md prikazana na slici 67 namijenjena je pružanju orijentacije inženjerima koji pregledavaju kod, posebice korisnicima koji prvi put dolaze. README.md je vjerojatno prva skripta na koju čitatelj naiđe kada pregledava direktorij koji sadrži tuđi kod. Na taj način djeluje kao odredišna stranica za imenik (Google, 2023).



Slika 67. Prikaz README.md datoteke u običnom načinu i u prikazivačkom načinu

6. Zaključak

Cilj ovog diplomskog rada je izrada web aplikacije koja omogućuje komunikaciju u stvarnom vremenu pri tom koristeći end-to-end enkripciju. predstavlja jedan dio i jedan proces puno većeg sustava. Svrha aplikacije jest imati aplikaciju za čavrljanje koja omogućuje da poruke korisnika budu tajne, a da se pri tome ne mora koristiti broj mobitel kao kod drugih aplikacija. U današnjem vremenu svijet bez interneta je nezamisliv, preko njega se obavlja većina stvari, a tako i komunikacija s drugim ljudima. Jedna od bitnijih stvari kada ljudi čuju riječ „internet“ je sigurnost. Kako biti siguran na internetu, te kako da podaci budu/ostanu sigurni?

Front-end aplikacije je napravljen u React. React je učinkovita i fleksibilna JavaScript biblioteka za izgradnju jednostavnih, brzih sučelja web aplikacija. Zbog broja ljudi koji koriste React, podrška je ogromna. Back-end dio aplikacije je napravljen u JavaScriptu uz pomoć NextJS okvira. NextJS je zapravo okvir izgrađen na ReactJS-u koji proširuje mogućnosti ReactJS-a pružajući prikazivanje na back-end strani i pojednostavljuje razvoj samih web aplikacija. Za WebSockets u planu je bio koristiti Socket.io, međutim kako on ima velikih problema u povezivanju sa NextJS, te njegova kompatibilnost sa svim pretraživačima nije najbolja, za WebSockets je korišten Pusher.

Aplikacija „ChitChat“ koja je razvijena u ovom diplomskom radu nastoji korisniku zadržati njegove podatke i omogućiti sigurnost kako samog korisnika tako i njegovih podataka. To je omogućeno preko end-to-end enkripcije. Kako bi se sve brže odvijalo korišten je WebSocket. Protokol koji omogućuje da komunikacija bude u stvarno vremenu. Aplikacija je napravljena na engleskom jeziku kako bi ju mogla koristiti i šira publika. Aplikacija omogućuje komunikaciju u stvarnom vremenu između najmanje dva korisnika. ChiChat je web aplikacija otvorenog koda (eng. open source), što znači da je kod aplikacije dostupan svima za korištenje, modificiranje i distribuciju. Na poveznici se može pogledati cijeli kod aplikacije https://github.com/Matej-Omazic/ChitChat.

Aplikacija ima mnogo potencijala za rast i razvoj, od samog unaprjeđenja trenutnih, do dodavanja posve novih funkcionalnosti. Jedna od bitnih stvari koja bi se mogla dodati je mogućnost prijave/registracije koristeći neku drugu platformu kao npr. Google račun, to bi korisniku olakšalo vrijeme za samo prijavu/registraciju te bi mu računi bili međusobno povezani.

Literatura

Bhardwaj P. (2021) What is End-to-End Encryption (E2EE)? Tutorialspoint. Dostupno na: https://www.tutorialspoint.com/what-is-end-to-end-encryption-e2ee [10. sprnja 2023.]

CloudFareTeam (2023) What is end-to-end encryption (E2EE)? Cloudfare. Dostupno na: https://www.cloudflare.com/learning/privacy/what-is-end-to-end-encryption/ [10. srpnja 2023.]

CloudinaryTeam (2023) Frequently Asked Questions Cloudinary. Dostupno na: https://cloudinary.com/faq [2. rujna 2023.]

Copes F. (2023) Introduction to Node.js NodeJSDev. Dostupno na: nodejs.dev/en/learn/

CSRC (2022) Key Management Guidelines National Institute of Standards and Technology. Dosutpno na: https://csrc.nist.gov/Projects/Key-Management/Key-Management-Guidelines [20. rujna 2023.]

De Roy S. (2022) What is Tailwind CSS? A Beginner's Guide FreeCodeCamp. Dostupno na: https://www.freecodecamp.org/news/what-is-tailwind-css-a-beginners-guide/ [17. kolovoza 2023.]

Google (2023) styleguide Google.github. Dostupno na: https://google.github.io/styleguide/docguide/READMEs.html [10. srpnja 2023.]

IbmTeam (2023) What is end-to-end encryption? IBM. Dostupno na: https://www.ibm.com/topics/end-to-end-encryption [10. srpnja 2023.]

Juviler. J (2023) What Is an API Endpoint? HubSpot. Dostupno na: https://blog.hubspot.com/website/api-endpoint [7. rujna 2023.]

Kai B. (2022) Top 6 Features in React Dev.to. Dostupno na: https://dev.to/brayan_kai/top-6-features-in-react-3b18 [7. srpnja 2023.]

Kaushika S. (2023) Context in React GeeksforGeeks. Dostupno na: https://www.geeksforgeeks.org/context-in-react/ [9. kolovoza 2023.]

Mistry P. (2023) Making the switch: empowering manual testers to automate their API tests Blog.Postman. Dostupno na: https://blog.postman.com/switching-from-manual-to-automated-api-testing/ [9. srpnja 2023.]

MIT Press (1994) The Official PGP User's Guide MIT Dostupno na: https://web.pa.msu.edu/reference/pgpdoc1.html [20. rujna 2023.]

MongoTeam (2023) What Is MongoDB? MongoDB. Dostupno na: https://www.mongodb.com/what-is-mongodb [9. srpnja 2023.]

Mouskhelichvili T. (2022) How to ignore the node_modules folder in Git? TimMousk. Dostupno na: https://timmousk.com/blog/git-ignore-node-modules/ [10. srpnja 2023.] NextJSTeam (2023) Dynamic Routes NextJS. Dostupno na: https://nextjs.org/docs/pages/building-your-application/routing/dynamic-routes [10. kolovoza 2023.]

NextJSTeam (2023) What's in Next.js? NextJS. Dostupno na: https://nextjs.org/ [10. srpnja 2023.]

Obregon A. (2023) IntelliJ IDEA for Android Development: Exploring Its Features and Benefits Medium. Dostupno na: https://medium.com/@AlexanderObregon/intellij-idea-for-android-development-exploring-its-features-and-benefits-4f55dc218726 [8. srpnja 2023.]

Panchal R. (2022) Is Node.JS Popular in 2022? Dzone. Dostupno na: https://dzone.com/articles/is-nodejs-popular-in-2022 [8. srpnja 2023.]

Parthibakumar M. (2022) What is .env ? How to Set up and run a .env file in Node? Codementor. Dostupno na: https://www.codementor.io/@parthibakumarmurugesan/what-is-env-how-to-set-up-and-run-a-env-file-in-node-1pnyxw9yxj [10. srpnja 2023.]

PostmanTeam (2023) What is Postman? Postman. Dostupno na: https://www.postman.com/product/what-is-postman/ [9. srpnja 2023.]

PusherTeam (2023) Pusher Channels Protocol Pusher. Dostupno na: https://pusher.com/docs/channels/library_auth_reference/pusher-websockets-protocol/ [10. srpnja 2023.]

Tapas A. (2022) React Hooks Fundamentals for Beginners FreeCodeCamp. Dostupno na: https://www.freecodecamp.org/news/react-hooks-fundamentals/ [15. kolovoza 2023.]

Tarik (2023) Organizing Your React Native Codebase LevelUp. Dostupno na: https://levelup.gitconnected.com/how-to-structure-react-native-application-679e36c63efd [15. kolovoza 2023.]

The Guardian (2016) 1MDB: The inside story of the world’s biggest financial scandal TheGuardian. Dostupno na: https://www.theguardian.com/world/2016/jul/28/1mdb-inside-story-worlds-biggest-financial-scandal-malaysia [20. rujna 2023.]

The Telegram Team (2017) What else makes Telegram Desktop Cool? Telegram. Dostupnoa na: https://telegram.org/blog/desktop-1-0 [5. rujna 2023.]

Tung V. (2022) What is IntelliJ IDEA? TrustRadius. Dostupno na: https://www.trustradius.com/products/intellij-idea/reviews?qs=pros-and-cons#overview [8. srpnja 2023.]

Webandcrafts (2021) Top Features and Benefits of Using React JS for Web Development Webandcrafts. Dostupno na: https://webandcrafts.com/blog/react-js-features/ [7. srpnja 2023.]

Popis slika

Slika 1. Prikaz React/TypeScript koda....2
Slika 2. NextJS koji koristi JavaScript kod....3
Slika 3. Prikaz IntelliJ IDEA....4
Slika 4. Prikaz mongoDB baze pute pretraživača……....5
Slika 5. Sučelje Postman aplikacije gdje se koristi POST metoda....6
Slika 6. WhatsApp izgled aplikacije....10
Slika 7. Telegram izgled aplikacije....11
Slika 8. Viber izgled aplikacije....12
Slika 9. Dijagram slučajeva korištenja....13
Slika 10. Klasni dijagram....14
Slika 11. Sekvencijski dijagram....15
Slika 12. Prikaz datoteka u projektu....16
Slika 13. Prikaz mape (site)....17
Slika 14. Izgled skripte page.tsx....18
Slika 15. Izgled obrazca za prijavu....18
Slika 16. Izgled dijela koda iz skripte AuthForm.tsx....19
Slika 17. Izgled skripte EmptyState.tsx....20
Slika 18. Izgled Empty u aplikaciji....20
Slika 19. Izgled mape actions....21
Slika 20. Izgled skripte getCurrentUser....21
Slika 21. Izgled mape api....22
Slika 22. Izgled route.ts iz auth mape....23
Slika 23. Izgled route.ts iz settingsa....24
Slika 24. Izgled route.ts iz user-a....25
Slika 25. Izgled mape messages....26
Slika 26. Izgled upita za birsanje jedne poruke....26
Slika 27. Izgled gumba za birsanje poruke u aplikaciji 26
Slika 28. Izgled route.ts skripte za lajkanje poruke....27
Slika 29. Izgled gumba za lajkanje poruke 27
Slika 30. Izgled mape components....28
Slika 31. Izgled mape inputs....28
Slika 32. Izgled mape sidebar....29

Slika 33. Izgled settingsa u aplikaciji....29
Slika 34. Prikaz korisnikovih statusa....30
Slika 35. Izgled skripte Avatar.tsx....31
Slika 36. Izgled mape context....32
Slika 37. Prikaz Toast obavijesti za uspješnu prijavu....32
Slika 38. Izgled mape conversations....33
Slika 39. Igled skripte layout.tsx u aplikaciji....33
Slika 40. Izgled mape conversationsId....34
Slika 41. Izgled Body.tsx skripte....35
Slika 42. Izgled Header.tsx u aplikaciji....35
Slika 43. Funkcija enkripcije poruke....36
Slika:44. Funkcija dekripcije poruke....36
Slika 45. Izgled forme za poruke 37
Slika 46. Izgled Cloudinaryija u aplikaciji 37
Slika 47. Izgled funkcije za prijevod poruke 38
Slika 48. Prikaz prevođenja u aplikaciji....38
Slika 49. Izgled mape components....39
Slika 50. Izgled izrade razgovora sa više korisnika....39
Slika 51. Izgled mape hooks....40
Slika 52. Izgled skripte useRoutes.ts....41
Slika 53. Izgled useRoutes u aplikaciji....41
Slika 54. Izgled mape libs....42
Slika 55. Izgled koda za prismadb.ts....42
Slika 56. Izgled koda za pusher.ts....43
Slika 57. Izgled mape types i prikaz koda index.ts....44
Slika 58. Izgled mape users....45
Slika 59. Izgled skripte layout.tsx u aplikaciji....45
Slika 60. Preostale mape u datoteci app....46
Slika 61. Izgled ikone u prozoru pretraživača....46
Slika 62. Izgled skripte global.css....46
Slika 63. Izgled glavnog layout.tsx....47
Slika 64. Izgled mape pages....48
Slika 65. Izgled auth.ts iz mape pages....49
Slika 66. Ubacivanje user-a u bazu pomoću prisme....50

Slika 67. Prikaz README.md datoteke u običnom načinu i u prikazivačkom načinu rada.....51