

# **SPLETNA APLIKACIJA ZA KOMUNIKACIJO WAVECHAT**

Strokovno poročilo

Mentor: Marko Kastelic, univ. dipl. inž.

Avtor: Tim Thuma, R 4. C

Kamnik, april 2023

## **POVZETEK**

V tem strokovnem poročilu je predstavljena spletna aplikacija, narejena z namenom vzpostavljanja komunikacije med uporabniki. Komunikacija poteka v realnem času prek tekstovnih sporočil in slik. V vsebini so predstavljeni zaledni sistem, spletna stran in vmesnik za komunikacijo med njima. Uporabniki si v aplikaciji sporočila lahko izmenjujejo paroma (ena na ena) ali v skupinah. Člani skupin imajo možnost komunikacije po tematsko ločenih komunikacijskih kanalih. Uporabniki so umeščeni v sistem prijateljstva, ki je podoben prijateljstvu na družabnem omrežju. Uporabnike in skupine je možno poiskati z uporabo njihovega imena. Vsa poslana sporočila se trajno hranijo v relacijski podatkovni zbirki. Večina kode aplikacije je spisane v programskem skriptnem jeziku JavaScript, skupaj z različnimi knjižnicami za sprotno osveževanje spletne strani in komunikacijo s podatkovno zbirko. Aplikacija je dostopna na svetovnem spletu.

**Ključne besede:** spletna aplikacija, sporočila, družabno omrežje, JavaScript, GraphQL

## **ABSTRACT**

This report presents my web application for communication among users. Communication is done in real-time by text messages and images. I present the backend, frontend and the interface for communication between the two. Users can send messages in a one-on-one fashion or in groups. Group members can communicate through multiple communication channels with different topics. A system of friendships like one found in social media is implemented among users. Users and groups can be found by searching by their name. All sent messages are permanently stored in a database. Most of the application's code is written in the JavaScript programming language, together with different libraries for creating reactive web pages and communication with databases. The web application is accessible on the World Wide Web.

**Key words:** web application, messaging, social media, JavaScript, GraphQL

# KAZALO

<b>1. UVOD .....</b>	<b>4</b>
<b>2. ZALEDNI SISTEM .....</b>	<b>4</b>
2.1 PODATKOVNI MODEL .....	5
2.2 SEJE .....	7
<b>3. KOMUNIKACIJA MED BRSKALNIKOM IN STREŽNIKOM .....</b>	<b>8</b>
<b>4. SPLETNA STRAN.....</b>	<b>9</b>
4.1 PRIJAVA IN REGISTRACIJA .....	10
4.2 ZAČETNA STRAN APLIKACIJE .....	12
4.3 ISKANJE UPORABNIKOV IN SKUPIN.....	13
4.4 DIREKTNA SPOROČILA .....	13
4.5 SKUPINSKI POGOVORI .....	14
4.6 UPRAVLJANJE LASTNEGA RAČUNA .....	15
<b>5. IZDELEK V PRODUKCIJSKEM NAČINU .....</b>	<b>16</b>
5.1 VZVRATNI POSREDNIK .....	16
5.2 TLS/HTTPS.....	16
<b>6. ZAKLJUČEK.....</b>	<b>17</b>
<b>7. VIRI IN LITERATURA.....</b>	<b>18</b>

# 1. UVOD

Dandanes komunikacijo prek spleta jemljemo za samoumevno. Poleg načinov komunikacije kot so telefonski pogovori, spletna pošta ipd., so v zadnjih desetletjih zelo popularna postala t. i. instantna sporočila, ki nam omogočajo izmenjavo sporočil v realnem času. Storitve za instantno komunikacijo so lahko vgrajene v družabna omrežja ali pa so ločene aplikacije. Sam sem razvil storitev WaveChat, ki omogoča izmenjavo sporočil v stilu ena na ena in v skupinah, vsebuje pa tudi elemente družabnih omrežij, kot je sistem prijateljstva med uporabniki.



*Slika 1: Logotip aplikacije WaveChat (Vir: Lastni, 2023)*

Za razvoj aplikacije sem uporabil razvojno metodo kodiraj in popravi (ang. Code and Fix), pri kateri sem aplikacijo razvijal z minimalnimi načrti vnaprej, dokler nisem bil z izdelkom zadovoljen. Ta metoda se mi je zdela primerna, saj nisem delal v skupini, zaradi česar nisem potreboval dobrih priprav na delo.

Aplikacijo sestavljajo trije deli: zaledni sistem, vmesnik za komunikacijo med brskalnikom in strežnikom ter spletna stran. Zaledni sistem upravlja s podatkovno zbirko in sejami, spletna stran nudi prijazen uporabniški vmesnik za uporabo aplikacije, vmesnik za komunikacijo pa skrbi, da podatke iz brskalnika oz. spletne strani lahko pošiljamo do zalednega sistema in obratno. Posamezne komponente sistema so podrobneje opisane v naslednjih poglavjih.

## 2. ZALEDNI SISTEM

Spletni strežnik sem vzpostavil z izvajalnim okoljem NodeJS, ki mi omogoča, da izvajam JavaScript kodo izven brskalnika. Z njim sem vpeljal knjižnico Express, ki služi kot ogrodje spletnega strežnika, na katerega lahko dodam knjižnice za delo s podatkovno bazo in za komunikacijo med brskalnikom in strežnikom. Pri delu z JavaScript-om sem uporabil TypeScript, ki mi je omogočal, da sem v JavaScript-u določil tipe spremenljivk, kar mi je pomagalo pri zmanjševanju števila napak med programiranjem.

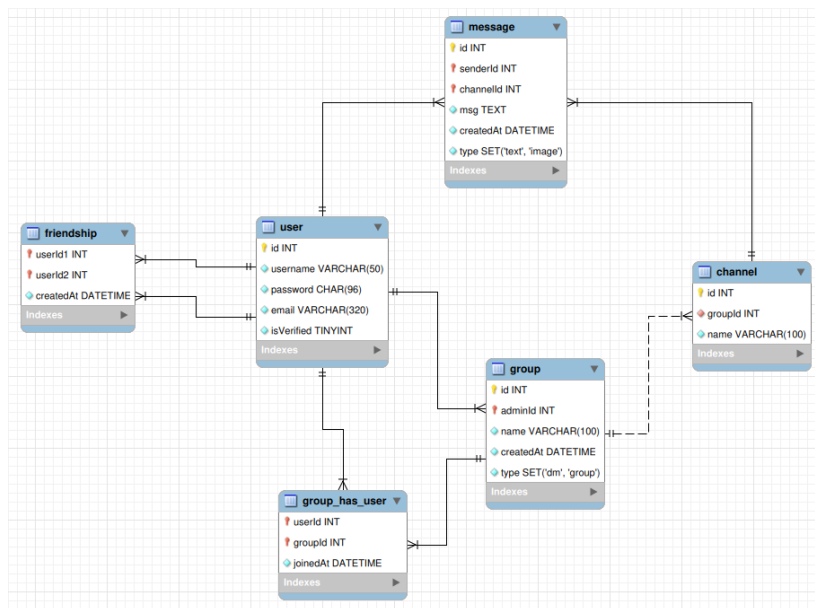
Pri delu z različnimi JavaScript knjižnicami oz. NodeJS paketi sem uporabil upravljalca paketov (ang. package manager) Yarn. Yarn predstavlja novejšo in po mojem mnenju boljše in hitrejšo alternativo upravljalcu paketov NPM, saj pakete na disk naloži na bolj učinkovit način kot to počne NPM, zaradi česar je nalaganje novih paketov hitrejšo in razvoj aplikacij olajšan.

Eden ključnih delov za uspešno pogovarjanje v realnem času je uporaba WebSocket protokola. Ta mi omogoča, da se ob poslanem sporočilu sproži dogodek, na katerega čaka strežnik. Ko strežnik zazna dogodek, to informacijo posreduje odjemalcu, ki ustrezno posodobi svojo vsebino in tako uporabniku takoj pokaže novo sporočilo. Ta način sporazumevanja med brskalnikom in strežnikom je veliko boljši od načina pozivanja (ang. polling), saj od strežnika ne zahteva, da stalno pošilja svoje stanje, kar razbremeni tako strežnik kot tudi omrežje. Za delo s tem protokolom sem uporabil knjižnico SocketIO. Ta ima dve različici, in sicer eno za strežnik in eno za brskalniki. Za njeno uporabo samo določimo naslov strežnika, na katerem je aktiven WebSocket, in dogodke, na katere naj se odzove. Knjižnica definira nekaj svojih dogodkov, kot je dogodek ob povezavi (ang. connect), definiral pa sem tudi lastne dogodke, kot je dogodek, ki se sproži ob novem sporočilu.

## 2.1 PODATKOVNI MODEL

Za hranjenje podatkov sem uporabil MariaDB SUPB, ki je eden izmed bolj uporabljenih relacijskih SUPB-jev. Na strani spletnega strežnika sem uporabil ORM programsko opremo. ORM oz. objektno-relacijska preslikava (ang. object-relational mapping) je programska oprema, ki omogoča pretvorbo razredov iz objektno usmerjenih programskih jezikov v SQL kodo za relacijsko podatkovno bazo. Sam sem uporabil knjižnico TypeORM, s katero sem lahko napisal shemo baze v obliki JavaScript razreda, ki se nato pretvori v ustrezno SQL kodo. TypeORM ustvari tudi tipe za attribute v bazi na podlagi TypeScript tipov. Včasih pa sem želel, da se tip atributa v bazi razlikuje od TypeScript tipa, saj ima MariaDB drugačne možne tipe kot TypeScript. V tem primeru sem eksplicitno navedel kateri tip želim uporabiti v podatkovni bazi. Pri delu z mnogo proti ena relacijami lahko s TypeORM-om v definicijo entitete dodamo JavaScript tabelo, ki hrani povezane vnose iz podatkovne baze. To mi je dalo možnost, da pri delu s stiki nisem rabil pisati lastnih SQL stavkov, ampak sem samo povedal katere relacije želim dobiti, TypeORM pa mi je sam sestavil stavek za stik. Ker TypeORM predstavlja nivo abstrakcije nad SQL jezikom, mu manjkajo nekatere zmožnosti kot so dodajanje CHECK zahtev in kaskadno brisanje tujih ključ, zato sem moral to dodati z lastnimi SQL stavki.

Zanimivo pri delu s TypeORM-om je, da sem lahko pri programiranju uporabil JavaScript okraševalce razredov (ang. class decorator), ki so mi ponudili zelo preprost vmesnik za definiranje sheme podatkovne baze.



Slika 2: Entitetno-relacijski model baze (Vir: Lastni, 2023)

Med uporabniki sem izvedel rekurzivno mnogo-mnogo relacijo, ki omogoča hranjenje prijateljstev. Vsak uporabnik se lahko pridruži več skupinam, uporabnik, ki je skupino ustvaril, pa postane administrator skupine. Skupine sem razdelil na dva tipa: direktna sporočila in klasične skupine. Ker so direktna sporočila in skupine enake po strukturi, sem za ločevanje med njimi uporabil atribut »type«, namesto da bi naredil dve ločeni entiteti. Vsaka skupina za komunikacijo med uporabniki potrebuje vsaj en komunikacijski kanal. Za direktna sporočila je značilno, da vsebujejo le en komunikacijski kanal z imenom »dm\_default«, ime skupine, pod katero so shranjena, pa je sestavljeno iz identifikacijskih števil obeh uporabnikov, ki se pogovarjata. V tabeli sporočil poleg vsebine sporočila hranim tudi uporabnika, ki ga je poslal, in kanal, v katerega je bilo sporočilo poslano.

Gesla uporabnikov hranim v šifrirani obliki, ustvarjeni z algoritmom Argon2. Ta algoritem je eden izmed novejših algoritmov za šifriranje nizov, ki je nastal med tekmovanjem Password Hashing Competition leta 2015. Izveden je lahko v dveh oblikah: Argon2d, Argon2i. Različica Argon2d je zgrajena na način, da je čim bolj odporna proti napadom z uporabo GPE in je hitrejša od različice Argon2i. Argon2i pa je zgrajen na način, da zmanjša možnost napada z uporabo stranskega kanala (ang. side-channel attack). Ustvarjalci algoritma so v njegovi

specifikaciji zapisali: »Argon2i uporablja podatkovno neodvisen dostop do spomina, kar je zaželeno za šifriranje gesel in izpeljavo ključev na podlagi gesel« (Biryukov, Dinu in Khovratovich, 2017, 3). Zato sem v svoji aplikaciji za šifriranje gesel uporabil različico Argon2i. Algoritem skupaj s šifrirano obliko gesla, dolgo 32 zlogov, shrani še uporabljeno sol.

Če uporabnik namesto tekstovnega sporočila pošlje sliko, v podatkovno bazo shranim le ime slike, slika sama pa se shrani na oblachno storitev Wasabi. To uporabnikom omogoča dostop do slik tudi izven moje aplikacije in razbremeni spletni strežnik. Pri delu z oblachnimi storitvami je bilo pomembno, da zasebnih podatkov, kot so zasebni ključi za dostop do storitve, nisem razkril javnosti, torej da jih nisem zapisal neposredno v kodo, ki sem jo naložil v sistem za nadzor različic, kot je Git, temveč da sem jih shranil v posebni datoteki, namenjeni samo hranjenju skritih spremenljivk, ki ni nikoli na vpogled javnosti. Zelo koristno je tudi dejstvo, da je Wasabi zgrajen na enaki tehnologiji kot Amazonova AWS S3 storitev, zaradi česar mi je bilo na voljo ogromno pomoči in knjižnic za preprosto delo z njim.

Uporabnikom sem omejil največjo dovoljeno velikost slike, ki jo lahko naložijo, s čimer sem razbremenil svoje vire. Uporabnik lahko naloži sliko, veliko največ 10 MB, nato pa z orodjem Sharp velike slike pretvorim v sliko z največjo dovoljeno širino in višino 1000 pikslov, potem pa sliko še stisnem z JPEG stiskanjem. Pri tem sem moral biti pozoren, da sem zmanjšal samo večjo stranico, druga pa se je avtomatsko spremenila tako, da je razmerje med širino in višino ostalo stalno in nisem popačil slike.

## 2.2 SEJE

Za delo s sejami sem uporabil knjižnico Express Session, ki je del Express knjižnice, skupaj s podatkovno bazo Redis. Redis je podatkovna baza, ki hrani podatke v primarnem pomnilniku namesto v sekundarnem, kar omogoča hitrejši dostop za podatkov. To je primerno za hranjenje sej, saj je količina podatkov, ki jih moram shraniti relativno majhna, hkrati pa si želim uporabnike čim hitreje avtorizirati za dejanja, ki potrebujejo avtorizacijo. Za razliko od relacijskih podatkovnih baz, kot je MariaDB, Redis hrani podatke v obliki ključ-vrednost (ang. key-value), torej baza nima nobenih tabel. Na strani brskalnika se seja shrani v obliki piškotka, ki ga uporabnik lahko izbriše z odjavo.

Ustvarjenje nove seje poteka po naslednjem postopku:

1. Ob prijavi v aplikacijo se v Redis podatkovno bazo shrani identifikacijska številka seje.
2. Express Session v brskalnik nastavi piškotek s kodirano obliko številke seje.
3. Ko želimo preveriti identiteto uporabnika, se piškotek pošlje na strežnik. Strežnik ta piškotek dekodira in tako dobi številko seje, ki jo poišče v Redis podatkovni bazi.

### **3. KOMUNIKACIJA MED BRSKALNIKOM IN STREŽNIKOM**

Za komunikacijo med brskalnikom in strežnikom sem uporabil protokol GraphQL. Ta zahteve deli na dve vrsti: poizvedbe in mutacije. Poizvedbe so namenjene pridobivanju podatkov iz podatkovne baze, mutacije pa spreminjanju podatkov.

Prednost tega protokola pred protokoli, kot je REST, je to, da lahko iz strežnika pridobim le podatke, ki jih potrebujem, namesto vseh podatkov, ki jih neka zahteva vrne. To mi omogoča dejstvo, da do različnih podatkov ne dostopam na različnih naslovih (ang. endpoint), ampak do vseh podatkov dostopam na enem skupnem naslovu, kamor pošljem strukturo zahteve. S tem sem lahko na primer namesto vseh podatkov o uporabniku dobil le njegovo uporabniško ime, s čimer sem prihranil na porabi pasovne širine omrežja. Slabost GraphQL protokola pa je, da je novejši in je za delo z njim razvitih manj orodij.

Za realizacijo GraphQL protokola sem uporabil orodji Apollo in TypeGraphQL. Z orodjem TypeGraphQL sem dosegel konsistentnost med obliko zahtev in kodo za njihovo obdelavo. Ta mi je tudi omogočil, da sem določil, do katerih atributov v podatkovni bazi lahko uporabnik dostopa. Te podatki, skupaj z vhodnimi in izhodnimi parametri zahtev, so bili nato posredovani orodju Apollo, ki je na njihovi podlagi omogočal izvajanje zahtev le v primerni obliki.

Na strani strežnika sem v Express ogrodje dodal knjižnico Apollo Server, ki omogoča vzpostavitev razreševalca (ang. resolver), ki obdela prejete zahteve. Na strani brskalnika pa sem uporabil knjižnico Apollo Client, s katero lahko pošljem zahteve na strežnik. Zelo uporabna lastnost knjižnice Apollo Server je, da samodejno ustvari t. i. razvojno igrišče, na katerem sem lahko preizkusil, če moje zahteve res delajo tako, kot sem si želel.



TypeGraphQL mi je omogočil zaščito občutljivih podatkov tako, da so do njih lahko dostopali le pooblašчени uporabniki. Tako sem onemogočil dostop do gesel uporabnikov, e-poštni naslov posameznika pa lahko vidi le lastnik računa.

Orodje Apollo privzeto ne podpira nalaganja slik, lahko pa bi to omogočil z uporabo knjižnice GraphQL Upload. Na žalost je ta knjižnica slabo razvita in mi je ni uspelo usposobiti za delovanje, zato sem za nalaganje slik postavil ločeno storitev, na kateri lahko prek REST protokola naložim sliko. Slika se na tej storitvi obdela z orodjem Sharp in naloži na storitev Wasabi.

## 4. SPLETNA STRAN

Za izdelavo spletne strani sem uporabil knjižnici ReactJS in NextJS. ReactJS je JavaScript knjižnica, ki omogoča, da se vsebina spletne strani avtomatsko posodobi, ko se posodobijo spremenljivke, ki jih spletna stran prikazuje. NextJS pa je ogrodje za ReactJS aplikacije, ki mi omogoča funkcionalnosti, kot so upodabljanje na strežniku (ang. server side rendering), uporaba dinamičnih poti, izdelava komponent za večkratno uporabo ipd., hkrati pa mi določi strukturo projekta, da med razvijanjem aplikacije ne rabim sam vzdrževati smiselne hierarhije datotek. Uporaba ogrodja NextJS mi omogoča tudi, da ob navigaciji po spletni aplikaciji ni potrebno osveževanje celotnega okna, ampak se osvežijo samo deli, ki se zamenjajo, kar naredi aplikacijo prijaznejšo uporabniku in bolj odzivno.

Z NextJS lahko spletno stran poženeš v razvojnem ali v produkcijskem načinu. Razvojni način mi nudi sprotno osveževanje strani, ko spremenim kodo (ang. hot reload), kar pospeši razvoj strani. V produkcijskem načinu pa se celotna spletna stran vnaprej prevede iz ReactJS sintakse v JavaScript sintakso, razen strani, za katere ni možno vnaprej določiti strukture, saj se spreminja glede na vhodne parametre. Za te strani mi NextJS nudi upodabljanje na strežniku. To pomeni, da ko uporabnik pošlje zahtevo na spletni strežnik, ta upodobi (ang. render) spletno stran in prevedeno kodo pošlje kot odgovor. S tem razbremenim uporabnikov brskalnik, hkrati pa je nalaganje strani hitrejše, saj je spletni strežnik običajno bolj zmogljiv od uporabnikove naprave.

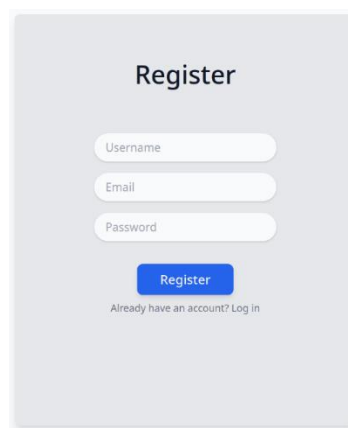
Za oblikovanje strani sem uporabil knjižnico TailwindCSS, s katero sem namesto klasične CSS kode uporabljal vnaprej pripravljene CSS razrede. S tem sem lahko hitro oblikoval posamezne elemente, hkrati pa me je prisililo, da so različnimi deli aplikacije skladno oblikovani. Na

spletni strani uporabljam primarno barvo modro in sekundarno barvo vijolično. TailwindCSS pa tudi omogoča, da uporabnik lahko preklaplja med temno in svetlo temo. Aplikacijo sem konfiguriral na način, da privzeto uporabi temo, ki jo uporablja uporabnikov operacijski sistem.

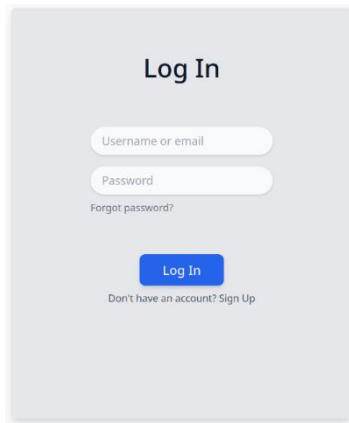
## 4.1 PRIJAVA IN REGISTRACIJA

Ob prvem obisku spletne aplikacije si nov uporabnik ustvari uporabniški račun. Za registracijo novega računa mora uporabnik navesti tri podatke: unikatno uporabniško ime, unikatni e-poštni naslov in geslo. Da je registracija računa uspešna, mora uporabnik potrditi lastništvo podanega e-poštnega naslova tako, da klikne na potrditveno povezavo, ki mu je bila poslana v nabiralnik. Ob ponovnem obisku storitve se lahko prijavi z uporabo ali uporabniškega imena ali e-poštnega naslova.

V primeru, da uporabnik pozabi svoje geslo, lahko zahteva spremembo svojega gesla tako, da vnese svoj e-poštni naslov in nanj prejme povezavo do strani za spremembo gesla. Ko uporabnik zahteva spremembo gesla, se ustvari enolični žeton za spremembo njegovega gesla, ki je vgrajen v poslano povezavo. Za generiranje žetona sem uporabil knjižnico UUID, ki ustvari naključen niz znakov, dolg 128 bitov. Ta žeton se izteče, če si uporabnik v roku treh dni ne zamenja gesla. Žeton sem shranil v podatkovni bazi Redis, skupaj z identifikacijsko številko uporabnika, ki mu ta žeton pripada. E-poštno sporočilo se pošlje s pomočjo knjižnice Nodemailer, ki mi ponuja preprost vmesnik, s katerim določim prijavnne podatke za administratorski e-poštni račun, iz katerega se pošlje sporočilo.

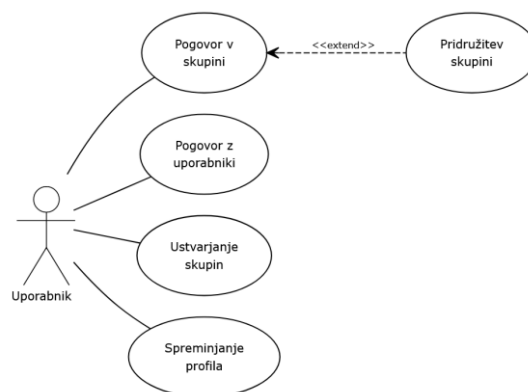
The image shows a registration form with a light gray background. At the top, the word "Register" is centered in a bold, dark font. Below the title, there are three white input fields with rounded corners, each containing a placeholder label: "Username", "Email", and "Password". Underneath these fields is a blue button with the word "Register" in white. At the bottom of the form, there is a small, light gray link that says "Already have an account? Log in".

Slika 3: Forma za registracijo (Vir: Lastni, 2023)

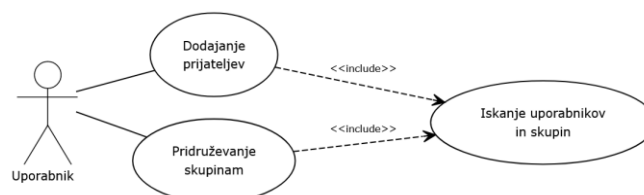


Slika 4: Forma za prijavo (Vir: Lastni, 2023)

Ko je uporabniški račun ustvarjen, uporabnik dobi dostop do vseh funkcionalnosti aplikacije. Druge uporabnike in skupine lahko poišče s poznavanjem njihovega uporabniškega imena ali imena skupine. Za komunikacijo ena na ena ni potrebno, da uporabnika dodamo kot prijatelja, medtem ko je za komunikacijo v skupini prej potrebna pridružitvev tej skupini.



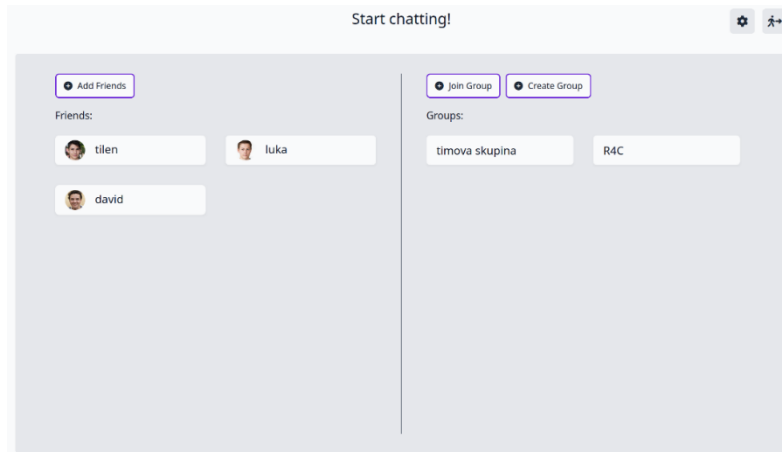
Slika 5: Diagram primerov uporabe 1 (Vir: Lastni, 2023)



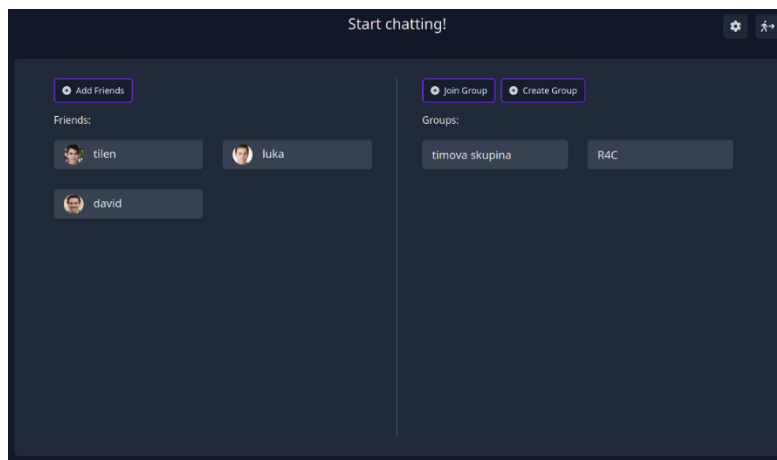
Slika 6: Diagram primerov uporabe 2 (Vir: Lastni, 2023)

## 4.2 ZAČETNA STRAN APLIKACIJE

Uporabniku se na začetni strani izpiše seznam dodanih prijateljev in skupin, s katerimi se lahko pogovarja. Iz tu pa lahko tudi dostopa do strani za iskanje drugih uporabnikov in skupin ter do nastavitve svojega računa.



Slika 7: Začetna stran aplikacije v svetli temi (Vir: Lastni, 2023)

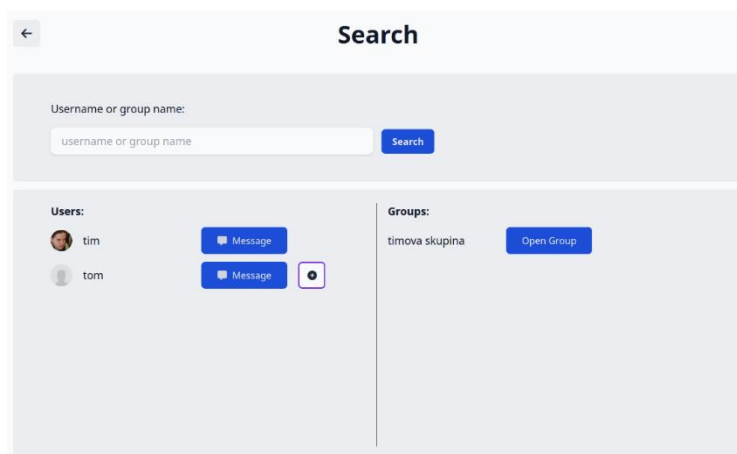


Slika 8: Začetna stran aplikacije v temni temi (Vir: Lastni, 2023)

### 4.3 ISKANJE UPORABNIKOV IN SKUPIN

Uporabnik v iskalno polje vnese iskano ime in dobi seznam uporabnikov in skupin s tem ali podobnim imenom. Prikažejo se mu vsi uporabniki oz. skupine, ki imajo iskani niz v svojem imenu in tisti, katerih ime se od iskanega niza razlikuje za največ dve spremembi. Za iskanje razlike v znakih med iskanim in končnim imenom sem uporabil algoritem Levenshteinove oz. urejevalne razdalje (ang. edit distance), kar omogoča, da uporabnik dobi primerne rezultate iskanja tudi v primeru, če se je zatipkal. Algoritem Levenshteinove razdalje sem izvedel v obliki SQL procedure na podatkovnem strežniku.

Na tej strani lahko uporabnik doda nove prijatelje, se pridruži novim skupinam ali odpre pogovor z drugim uporabnikom.



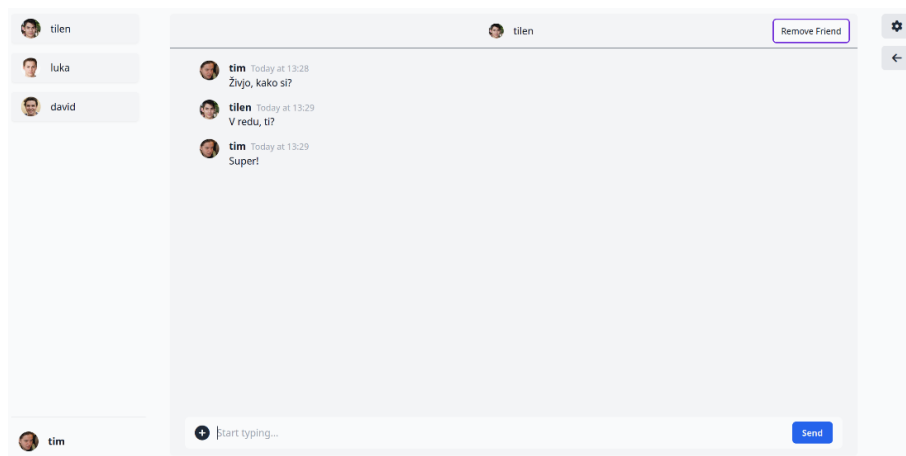
Slika 9: Stran za iskanje uporabnikov in skupin ob vnesenem iskalnem nizu »tim« (Vir: Lastni, 2023)

### 4.4 DIREKTNA SPOROČILA

Direktna sporočila oz. sporočila ena na ena so sporočila, ki si jih med seboj izmenjujeta točno dva uporabnika. Na tej strani lahko skačemo med pogovori s svojimi prijatelji.

Uporabnik najprej vidi le najnovejša sporočila. Ta se nalagajo po sistemu ostranjevanja, in sicer se najprej naloži le najnovejših 15 sporočil, nato pa se s pomikom do vrha prikazanih sporočil naložijo tudi starejša, kar omogoča t. i. pomikanje v neskončnost (ang. infinite scrolling). Če je hkrati na strani prikazanih zelo veliko sporočil, to lahko upočasni delovanje strani, saj obremeni vire odjemalca. Zato sem sporočila prikazal s pomočjo virtualnega seznama, ki na spletno stran uprizori (ang. render) samo tista sporočila, ki so na vidnem polju zaslona.

Da sem vedel, kateri pogovor sem moral naložiti, sem v URL naslov strani zapisal identifikacijsko številko drugega uporabnika. NextJS mi je omogočal, da sem zaznal to številko in na njeni podlagi naložil pogovor.

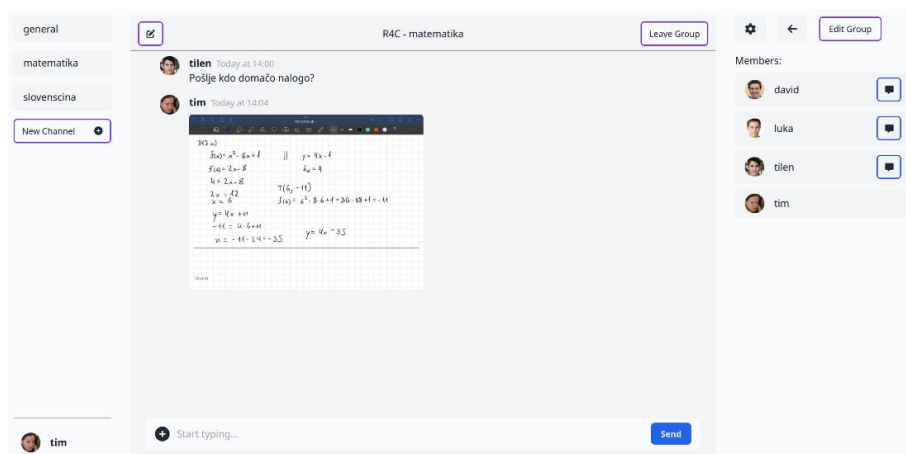


Slika 10: Primer pogovora ena na ena (Vir: Lastni, 2023)

## 4.5 SKUPINSKI POGOVORI

Poleg direktnih sporočil imajo uporabniki na voljo tudi tvorjenje skupin, v katerih se lahko pogovarja več ljudi hkrati. Za pogovor v skupini se je najprej potrebno tej skupini pridružiti. Vsem uporabnikom je na voljo pregled ostalih ljudi, ki so vključeni v skupino, s katerimi iz te strani lahko tudi začnemo pogovor. Skupinski pogovori imajo lahko več različnih kanalov za pogovor, kar uporabnikom omogoča, da med sabo ločijo različne teme pogovorov.

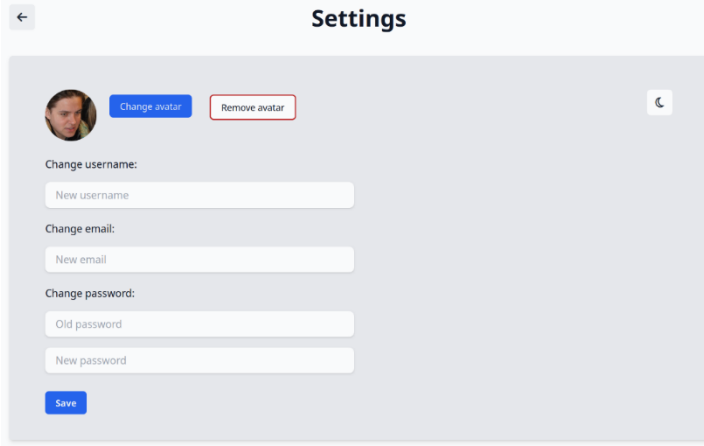
Podobno kot pri direktnih sporočilih, se tudi tu identifikacijska številka kanala zapiše v URL naslov strani, ki jo nato zaznam z NextJS.



Slika 11: Primer pogovora v skupini (Vir: Lastni, 2023)

## 4.6 UPRAVLJANJE LASTNEGA RAČUNA

Uporabnik ima vedno na voljo spreminjanje parametrov profila, ki jih je podal ob registraciji. Lahko si spremeni svojo profilno sliko, uporabniško ime, e-poštni naslov in geslo. Za spremembo svojega gesla mora uporabnik navesti svoje staro in novo geslo.

The image shows a mobile application interface for account settings. At the top, there is a back arrow and the title 'Settings'. Below this, there is a profile picture of a man. To the right of the picture are two buttons: 'Change avatar' (blue) and 'Remove avatar' (red). Further right is a small circular icon. Below the profile picture, there are three sections for changing account information: 'Change username:' with a text input field labeled 'New username'; 'Change email:' with a text input field labeled 'New email'; and 'Change password:' with two text input fields labeled 'Old password' and 'New password'. At the bottom left of the form area is a blue 'Save' button.

Slika 12: Stran za upravljanje lastnega računa (Vir: Lastni, 2023)

## 5. IZDELEK V PRODUKCIJSKEM NAČINU

Svojo spletno aplikacijo sem gostil na navideznem zasebnem strežniku (ang. virtual private server ali VPS) na platformi Contabo. Za razliko od razvojne različice storitve, kjer se aplikacija na novo zgradi ob vsaki spremembi datoteke, sem na produkcijskem strežniku vso TypeScript in ReactJS kodo vnaprej prevedel v JavaScript kodo, saj se tako spletna stran hitreje naloži.

### 5.1 VZVRATNI POSREDNIK

Moja spletna aplikacija deluje na vratih 3000 in 4000, kar je moteče pri uporabi, saj bi moral uporabnik vedno ob obisku spletne strani v brskalnik vpisati tudi številko vrat. Zato sem z orodjem Nginx postavil vzvratni posrednik (ang. reverse proxy), ki omogoča dostop do aplikacije na vratih 80 oz. 443, ki sta vrata za HTTP oz. HTTPS protokol. Pri tem sem moral na prava vrata prestaviti tako spletno stran, kot tudi vmesnik za komunikacijo z zalednim sistemom, kar sem dosegel tako, da sem v nastavitvah Nginx strežnika določil, na katero pot se prenese določen del aplikacije. Spletno stran sem prenesel na korensko pot, vmesnik pa na pot »/graphql«.

### 5.2 TLS/HTTPS

Za zasebnost pri uporabi spletnih strani je pomembno, da je komunikacija zakodirana s TLS protokolom, ki omogoča, da do spletne strani dostopam prek HTTPS protokola. Za generiranje TLS certifikata sem uporabil orodje Certbot, s katerim sem ustvaril certifikat, ki ga je izdala certifikatna agencija Let's Encrypt. Certbot je zelo priročen, saj mi ta certifikat avtomatsko posodobi, ko se približa roku izteka. Agencija Let's Encrypt mi lahko izda le certifikate, vezane na domeno, ne pa na IP naslov, zaradi česar sem registriral tudi domeno wavechat.si, prek katere je bila dostopna moja spletna aplikacija. Če uporabnik želi dostopati do aplikacije prek HTTP protokola, ga Nginx strežnik avtomatsko preusmeri na uporabo HTTPS protokola.



## 6. ZAKLJUČEK

Pri izdelavi naloge sem se naučil dela z različnimi JavaScript knjižnicami, ki so mi omogočale, da sem z enim programskim jezikom napisal kodo tako za brskalnik kot tudi za strežnik in za komunikacijo med njima. Težava teh knjižnic je bila, da so včasih zaradi njihove novosti slabše razvite. Veliko težav mi je povzročalo tudi delo s piškotki v razvojnem okolju, saj je pred njihovo uporabo potrebno pravilno nastaviti CORS (Cross-Origin Resource Sharing) spremenljivke in naložiti dodatne vtičnike.

Zelo zanimivo je bilo opaziti izboljšavo v hitrosti delovanja spletne strani v produkcijski različici, saj mi NextJS omogoča vnaprejšnjo izgradnjo strani, zaradi česar so naložni časi občutno krajši.

Menim da sta sistem prijateljstva in izmenjava sporočil dva široka koncepta, ki bi ju v prihodnosti lahko veliko bolj razvil. Prijateljstvo bi nadgradil tako, da bi uporabniki dobivali predloge za skupine in druge uporabnike na podlagi že prej dodanih prijateljev ali pa na podlagi svojih interesov, kar bi izdelek približalo družabnemu omrežju. Sporočilom pa bi poleg tekstovnih sporočil lahko dodal klice in video klice.

## 7. VIRI IN LITERATURA

- [1] Abba, Ihechikara Vincent. 2022. *What is an ORM – The Meaning of Object Relational Mapping Database Tools*. Dostopno prek: <https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/>
- [2] *Apollo documentation*. Dostopno prek: <https://www.apollographql.com/docs/>
- [3] Awad, Ben. 2020. *Fullstack React GraphQL TypeScript Tutorial*. Dostopno prek: <https://youtu.be/I6ypD7qv3Z8S>
- [4] *Behind a reverse proxy*. Dostopno prek: <https://socket.io/docs/v3/reverse-proxy/>
- [5] Biryukov, Alex; Dinu, Daniel in Khovratovich, Dmitry. 2017. *Argon2: the memory-hard function for password hashing and other applications*. Dostopno prek: <https://raw.githubusercontent.com/P-H-C/phc-winner-argon2/master/argon2-specs.pdf>
- [6] *Computer messaging before the Web – A visual timeline (1960-1990)*. 2009. Dostopno prek: <https://www.pingdom.com/blog/computer-messaging-before-the-web-a-visual-timeline-1960-1990/>
- [7] Desjardins, Jeff. 2016. *The Evolution of Instant Messaging*. Dostopno prek: <https://www.visualcapitalist.com/evolution-instant-messaging/>
- [8] Findlay, Louise. 2020. *How to Develop and Deploy Your First Full-Stack Web App Using A Static Site and Node.js*. Dostopno prek: <https://www.freecodecamp.org/news/develop-deploy-first-fullstack-web-app/>
- [9] *Password Hashing Competition*. 2019. Dostopno prek: <https://www.password-hashing.net/>
- [10] Price, Ward. 2021. *Image upload with Node.js and Typescript*. Dostopno prek: <https://wardprice.medium.com/image-upload-with-node-js-and-typescript-c9e2ccec874b>
- [11] Seifert, Tom. *How to host your API/backend on a virtual private server (VPS)*. Dostopno prek: <https://medium.com/@Syex/how-to-host-your-rest-api-backend-on-a-virtual-private-server-vps-eb005118d59a>
- [12] Stemmler, Khalil. 2020a. *Apollo Server File Upload Best Practices*. Dostopno prek: <https://www.apollographql.com/blog/backend/file-uploads/file-upload-best-practices/>
- [13] Stemmler, Khalil. 2020b. *GraphQL File Uploads with Apollo Server 2, React Hooks, TypeScript & Amazon S3*. Dostopno prek: <https://www.apollographql.com/blog/graphql/file-uploads/with-react-hooks-typescript-amazon-s3-tutorial/>
- [14] Thakkar, Raj. 2020. *Integrate Socket.IO with Node.js + Express*. Dostopno prek: [https://medium.com/@raj\\_36650/integrate-socket-io-with-node-js-express-2292ca13d891](https://medium.com/@raj_36650/integrate-socket-io-with-node-js-express-2292ca13d891)

[15] *WebSocket documentation*. Dostopno prek: <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>

# IZJAVA O AVTORSTVU

Izjavljam, da je strokovno poročilo Spletna aplikacija za komunikacijo WaveChat v celoti moje avtorsko delo, ki sem ga izdelal samostojno s pomočjo navedene literature in pod vodstvom mentorja Marka Kastelica.

Kamnik, 8. 4. 2023

Tim Thuma

