Békéscsabai SZC Nemes Tihamér Technikum és Kollégium

Szakképesítés megnevezése: Szoftverfejlesztő és -tesztelő

Azonosító száma: 5 0613 12 03

VIZSGAREMEK

Yapper

Készítették:

Baranya Gyula	Osztály: 13.b	Oktatási azonosító: 72567488297
Patai Olivér	Osztály: 13.b	Oktatási azonosító: 72595510237
Szabó Márton	Osztály: 13.b	Oktatási azonosító: 72572159843

Békéscsaba, 2024/2025

Tartalomjegyzék

Ábrajegyzék

Bevezető

Probléma rövid ismertetése

A digitális kommunikáció jelentősége

A 21. században a digitális kommunikáció mindennapjaink szerves részévé vált. Az emberek közötti kapcsolattartás, információcsere, közösségépítés és együttműködés jelentős része már nem személyesen, hanem különböző online platformokon keresztül történik. A gyors, megbízható és biztonságos üzenetküldés iránti igény folyamatosan növekszik, legyen szó magánszemélyekről, baráti társaságokról, családokról vagy akár munkahelyi csapatokról. Az online csevegőalkalmazások (chat appok) ezért napjaink egyik legnépszerűbb és legfontosabb szoftvermegoldásai közé tartoznak.

A jelenlegi helyzet és a felhasználói igények

Az elmúlt években számos csevegőalkalmazás jelent meg a piacon, amelyek különböző funkciókat és szolgáltatásokat kínálnak. Ezek közül sok azonban vagy túl bonyolult, vagy túlzottan egyszerű, esetleg nem felel meg a modern biztonsági elvárásoknak, vagy nem nyújt megfelelő felhasználói élményt. A felhasználók részéről egyre nagyobb az igény egy olyan platform iránt, amely egyszerre könnyen kezelhető, gyors, megbízható, biztonságos, és lehetőséget ad mind egyéni, mind csoportos kommunikációra.

A fejlesztendő alkalmazás célja

A Yapper nevű csevegőalkalmazás fejlesztésének célja, hogy egy korszerű, felhasználóbarát, biztonságos és könnyen bővíthető platformot hozzon létre, amely kielégíti a mai felhasználók igényeit. Az alkalmazás lehetőséget biztosít regisztrált felhasználók számára, hogy egymással valós időben üzeneteket váltsanak, képeket osszanak meg, valamint csoportos beszélgetéseket hozzanak létre. A fejlesztés során kiemelt figyelmet fordítottunk a felhasználói élményre, annak intuitív mivoltára.

A megoldandó problémák részletezése

Valós idejű kommunikáció

A felhasználók elvárják, hogy az üzenetek azonnal megérkezzenek a címzetthez, függetlenül attól, hogy az adott pillanatban milyen eszközt használnak. Ezért a fejlesztés során olyan technológiákat kellett alkalmazni, amelyek lehetővé teszik a valós idejű adatátvitelt (pl. WebSocket, Socket.IO).

Felhasználói azonosítás és jogosultságkezelés

A biztonságos kommunikáció alapfeltétele, hogy csak hitelesített felhasználók férhessenek hozzá az alkalmazás funkcióihoz. A regisztráció, email-hitelesítés, bejelentkezés, jelszó-visszaállítás, valamint a jogosultságok kezelése mind-mind fontos részei a rendszernek.

Adatvédelem és biztonság

A felhasználók által megosztott adatok (üzenetek, képek, profiladatok) védelme kiemelt fontosságú. A fejlesztés során gondoskodni kellett az adatok titkosításáról, a jelszavak biztonságos tárolásáról, a jogosulatlan hozzáférés megakadályozásáról.

Felhasználói élmény és ergonómia

Az alkalmazás sikerességének egyik kulcsa a könnyű kezelhetőség és az átlátható, modern felhasználói felület. A navigáció, az üzenetküldés, a profilbeállítások, a csoportos beszélgetések kezelése mind-mind intuitív módon kell, hogy működjön.

Platformfüggetlenség és bővíthetőség

A felhasználók különböző eszközökről (asztali gép, laptop, tablet, okostelefon) szeretnék elérni az alkalmazást. Ezért a fejlesztés során törekedtünk a reszponzív, platformfüggetlen megvalósításra, valamint arra, hogy a rendszer könnyen bővíthető legyen új funkciókkal.

A fejlesztés során felmerülő kihívások

A valós idejű kommunikáció, a biztonságos adatkezelés, a felhasználói élmény optimalizálása, valamint a rendszer skálázhatósága mind-mind komoly szakmai kihívást jelentettek. A fejlesztőcsapatnak meg kellett találni az egyensúlyt a funkcionalitás, a biztonság, a teljesítmény és a felhasználói élmény között.

A dokumentáció célja

Jelen dokumentáció célja, hogy részletesen bemutassa a Yapper csevegőalkalmazás fejlesztésének hátterét, a megoldandó problémákat, a választott technológiákat, a rendszer felépítését, a biztonsági megfontolásokat, valamint a felhasználói és fejlesztői szempontokat. A dokumentáció segítséget nyújt mind a fejlesztőknek, mind a felhasználóknak az alkalmazás megértésében, használatában és továbbfejlesztésében.

Összefoglalás

A Yapper csevegőalkalmazás fejlesztése egy valós, napjainkban is aktuális problémára kínál megoldást: a gyors, biztonságos és felhasználóbarát online kommunikációra. A projekt során kiemelt figyelmet fordítottunk a modern technológiai és ergonómiai elvárásokra, a biztonságra, valamint a könnyű használhatóságra és bővíthetőségre. A következő fejezetekben részletesen bemutatjuk a választott témát, a fejlesztési folyamatot, a rendszer felépítését, a biztonsági kérdéseket, az adatbázis felépítését, valamint a tesztelési és felhasználói szempontokat.

Téma indoklása

A csevegőalkalmazások szerepe a modern társadalomban

A digitális kommunikáció térnyerése az elmúlt évtizedekben alapvetően átalakította az emberek közötti kapcsolattartás módját. A csevegőalkalmazások, vagyis az online chat platformok, mára a mindennapi élet elengedhetetlen részévé váltak, legyen szó magánéleti, oktatási vagy munkahelyi környezetről. Ezek az alkalmazások lehetővé teszik a gyors, valós idejű információcserét, a csoportos együttműködést, valamint a közösségek építését, fenntartását. A világjárvány időszaka különösen rámutatott arra, mennyire fontosak a megbízható, könnyen használható és biztonságos online kommunikációs eszközök.

A témaválasztás szakmai és gyakorlati indokai

A Yapper csevegőalkalmazás fejlesztésének ötlete több, egymást erősítő szakmai és gyakorlati megfontolásból született:

Szakmai kihívás: Egy valós idejű, többfelhasználós, biztonságos kommunikációs rendszer fejlesztése összetett feladat, amely számos modern technológia (pl. WebSocket, JWT, reszponzív frontend, adatbázis-kezelés, autentikáció, jogosultságkezelés) alkalmazását igényli. Ez kiváló lehetőséget ad a fejlesztőknek arra, hogy elmélyítsék tudásukat a webfejlesztés különböző területein.

Gyakorlati relevancia: A csevegőalkalmazások iránti igény folyamatosan nő, mind a magánszemélyek, mind a vállalkozások körében. Egy jól működő, könnyen bővíthető chat platform fejlesztése nemcsak tanulási célból hasznos, hanem a későbbiekben akár valós projektek alapjául is szolgálhat.

Biztonsági szempontok: A felhasználói adatok védelme, a hitelesítés, az üzenetek titkosítása, valamint a jogosulatlan hozzáférés megakadályozása mind-mind olyan kihívások, amelyek megoldása elengedhetetlen egy modern alkalmazás esetében. A projekt során lehetőség nyílik ezen biztonsági kérdések gyakorlati megvalósítására.

Felhasználói élmény: A mai felhasználók elvárják, hogy egy alkalmazás ne csak funkcionális, hanem esztétikus, gyors és könnyen kezelhető is legyen. A Yapper fejlesztése során kiemelt figyelmet fordítottunk a reszponzív dizájnra, az intuitív navigációra és a modern ergonómiai elvek alkalmazására.

A projekt célcsoportja

A Yapper alkalmazás elsődleges célcsoportja a fiatal felnőttek, diákok, valamint a kis- és középvállalkozások, akik számára fontos a gyors, megbízható és biztonságos online kommunikáció. Emellett a fejlesztés során figyelembe vettük azokat a felhasználókat is, akik kevésbé jártasak a digitális technológiákban, ezért az alkalmazás kezelőfelülete egyszerű, átlátható és könnyen tanulható.

A fejlesztés tanulási értéke

A projekt során a fejlesztőcsapat tagjai számos, a munkaerőpiacon is keresett technológiát és módszertant sajátíthattak el, többek között: - REST API tervezés és implementáció - Valós idejű kommunikáció (Socket.IO) - Felhasználókezelés, autentikáció, jogosultságkezelés (JWT, bcrypt) - Adatbázis-tervezés és -kezelés (MongoDB) - Frontend fejlesztés (React, reszponzív dizájn) - Automatizált tesztelés (Selenium) - Biztonsági alapelvek alkalmazása (input validáció, XSS/CSRF védelem, jelszókezelés)

A választott technológiák indoklása

A projekt során alkalmazott technológiák kiválasztásánál a következő szempontokat vettük figyelembe: - **Skálázhatóság:** A Node.js és a MongoDB lehetővé teszi a rendszer későbbi bővítését, akár nagyobb felhasználószám esetén is. - **Valós idejűség:** A Socket.IO integrációja biztosítja az azonnali üzenetküldést és fogadást. - **Biztonság:** A JWT-alapú autentikáció, a

bcrypt-tel történő jelszóhash-elés, valamint a HTTPS támogatás mind hozzájárulnak a felhasználói adatok védelméhez. - **Fejlesztői élmény:** A React és a modern frontend eszközök gyors fejlesztést, könnyű karbantartást és bővíthetőséget tesznek lehetővé. - **Automatizált tesztelés:** A Selenium segítségével a legfontosabb funkciók automatikusan tesztelhetők, így a hibák gyorsan kiszűrhetők.

Összefoglalás

A Yapper csevegőalkalmazás témájának választása nemcsak a digitális kommunikáció aktuális jelentőségéből fakad, hanem abból is, hogy a projekt komplexitása révén lehetőséget ad a modern webfejlesztési technológiák, módszertanok és biztonsági elvek gyakorlati alkalmazására. A fejlesztés során szerzett tapasztalatok mind a szakmai fejlődést, mind a későbbi munkaerőpiaci érvényesülést támogatják, miközben egy valóban hasznos, a mindennapokban is alkalmazható szoftver született.

Téma kifejtése

Követelmények

A Yapper szoftveralkalmazás fejlesztése során az alábbi elvárásoknak kellett megfelelni:

- **Életszerű, valódi problémára nyújt megoldást:**
  A Yapper egy modern, valós idejű csevegőalkalmazás, amely a mindennapi online kommunikáció igényeire ad választ. Segítségével a felhasználók gyorsan, biztonságosan és kényelmesen tarthatják a kapcsolatot egymással, legyen szó magánéleti, tanulmányi vagy munkahelyi kommunikációról.

- **Adattárolási és -kezelési funkciók:**
  Az alkalmazás MongoDB adatbázist használ, amelyben a felhasználók, üzenetek, kapcsolatok és egyéb metaadatok strukturáltan, biztonságosan kerülnek tárolásra. Az adatok kezelése RESTful API-n keresztül történik, amely lehetővé teszi a CRUD (Create, Read, Update, Delete) műveleteket.

- **RESTful architektúra:**
  A backend Node.js és Express alapokon készült, a REST architektúra elveit követve. Az API végpontok logikusan szervezettek, jól dokumentáltak, és minden fő funkció (regisztráció, bejelentkezés, üzenetküldés, profilkezelés stb.) elérhető HTTP kéréseken keresztül.
  A frontend React-alapú SPA, amely a backenddel HTTP-n és Socket.IO-n keresztül kommunikál.

- **Platformfüggetlen, reszponzív kliens:**
  A felhasználói felület teljes mértékben reszponzív, így asztali számítógépen, laptopon,

tableten és mobiltelefonon is kényelmesen használható. A dizájn mobilbarát, a navigáció és az ergonómia minden eszközön optimális felhasználói élményt biztosít.

- **Tiszta kód elvei:**
  A forráskód megfelel a tiszta kód (Clean Code) alapelveinek:
  - Átlátható, jól strukturált, olvasható és könnyen karbantartható.
  - A komponensek és függvények egyértelmű, beszédes neveket kaptak.
  - A logika moduláris, a felelősségi körök jól elkülönülnek.
  - A hibakezelés és input validáció mindenhol megvalósul.

- **Dokumentáció:**
  Jelen dokumentáció részletesen bemutatja a szoftver célját, a komponensek technikai leírását, a működés műszaki feltételeit, valamint a használat rövid bemutatását. A dokumentáció a fejlesztői és felhasználói szempontokat egyaránt lefedi.

- **Önálló szellemi termék:**
  A Yapper alkalmazás és annak teljes forráskódja, valamint a dokumentáció a fejlesztőcsapat saját, önálló szellemi terméke, külső forrásból származó kódot csak indokolt esetben, megfelelő hivatkozással tartalmaz.

1. Funkcionális követelmények

- A felhasználók regisztrálhatnak és bejelentkezhetnek email-címmel és jelszóval.
- Email-verifikáció szükséges a regisztráció után.
- Elfelejtett jelszó esetén jelszó-visszaállítás emailben.
- A felhasználók szerkeszthetik a profiljukat (név, profilkép).
- Privát üzenetek küldése és fogadása két felhasználó között.
- Csoportok létrehozása, tagok hozzáadása/eltávolítása, admin jogok kezelése.
- Csoportos üzenetküldés, csoportüzenetek megtekintése.
- Barátküldés, elfogadás, visszautasítás, törlés.
- Push értesítések támogatása (új üzenet, csoportesemények).
- Üzenetekhez kép csatolása, reakciók hozzáadása.
- Felhasználók keresése, barátlista megtekintése.
- Minden funkció elérhető mobilon és asztali böngészőben is.

2. Nem-funkcionális követelmények

- Az alkalmazás legyen reszponzív, mobilbarát.
- Felhasználóbarát, letisztult és átlátható felület.
- Gyors válaszidő, valós idejű kommunikáció (WebSocket/SSE).
- Biztonságos adatkezelés (jelszavak hash-elése, JWT tokenek, HTTPS támogatás).
- Megfelelő hibakezelés, informatív visszajelzések.
- Adatvédelmi szempontok betartása (GDPR kompatibilitás).
- A rendszer legyen könnyen bővíthető és karbantartható.
- Automatikus tesztelés támogatása (pl. Selenium, Postman).

3. Technikai követelmények / Függőségek

Backend

- **Node.js** (ajánlott: 18.x vagy újabb)
- **Express** – REST API szerver
- **Mongoose** – MongoDB ODM
- **MongoDB** – NoSQL adatbázis
- **bcryptjs** – Jelszó hash-elés
- **jsonwebtoken** – Auth tokenek kezelése
- **nodemailer** – Email küldés (verifikáció, jelszó-visszaállítás)
- **cloudinary** – Képfeltöltés és tárolás
- **socket.io** – Valós idejű kommunikáció
- **web-push** – Push értesítések
- **dotenv** – Környezeti változók kezelése
- **cors, cookie-parser** – Biztonság és session kezelés
- **nodemon** – Fejlesztői környezethez

Frontend

- **React** – Felhasználói felület
- **Vite** – Fejlesztői szerver és build tool
- **react-router-dom** – Oldalak közötti navigáció
- **zustand** – Állapotkezelés
- **axios** – HTTP kérések
- **socket.io-client** – Valós idejű kommunikáció
- **tailwindcss, daisyui** – Stílus és UI komponensek
- **react-hot-toast** – Értesítések
- **Selenium WebDriver** – Automatizált UI tesztelés
- **Postman** – API tesztelés

Szükséges fiókok és hozzáférések

A projekt futtatásához és fejlesztéséhez az alábbi fiókokra és hozzáférési adatokra van szükség:

1. MongoDB adatbázis

- **Felhasználónév:** *mongodb_username*
- **Jelszó:** *mongodb_password*
- **Adatbázis URL:** *mongodb+srv://:@/?retryWrites=true&w=majority*

2. Nodemailer (email küldéshez)

- **Email cím:** *sajat_email_cim@example.com*
- **Jelszó / App password:** *email_app_password*
- **Kétlépcsős azonosítás:**
  Az email fiókhoz kétlépcsős azonosítás (2FA) szükséges. A Nodemailer csak olyan email fiókkal működik, ahol alkalmazásjelszó (app password) van beállítva.

3. Cloudinary (képfeltöltéshez)

- **Cloud name:** *cloudinary_cloud_name*
- **API Key:** *cloudinary_api_key*
- **API Secret:** *cloudinary_api_secret*

4. VAPID kulcsok (push értesítésekhez)

- **VAPID_PUBLIC_KEY:** *vapid_public_key*
- **VAPID_PRIVATE_KEY:** *vapid_private_key*

.env fájlok részletezése

A projekt működéséhez két .env fájl szükséges: egy a backendhez, egy a frontendhez. Ezek tartalmazzák a környezeti változókat, amelyek nélkülözhetetlenek a helyes működéshez.

A Vapid kulcsok generálása helyileg történik a fejlesztőnél, a README.md állományba részletezve van annak generálása.

backend/.env

```
MONGODB_URI=mongodb+srv://yapperhello:OxVd9wlVoJKvuc9d@cluster0.my6sker.mon
godb.net/yapper_db?retryWrites=true&w=majority&appName=Cluster0
PORT=5001
JWT_SECRET=WWFwcGVySW5jcmVkaWJseVNlY3JldEtleTcz

CLOUDINARY_CLOUD_NAME=dwt1lgisu
CLOUDINARY_API_SECRET=oQ4sVjM1qSLow8M78e7nlOmi-6I
CLOUDINARY_API_KEY=742141846288734

NODE_ENV=development

#NODE_ENV=production

FRONTEND_URL=http://localhost:5173
BACKEND_URL=http://localhost:5001

EMAIL_USER=yapper.hello@gmail.com
EMAIL_PASS=yayn ehuz gvtz xday

VAPID_PUBLIC_KEY=<frissen generált kulcs>

VAPID_PRIVATE_KEY=<frissen generált kulcs>
```

frontend/.env

```
VITE_FRONTEND_URL=http://localhost:5173
VITE_BACKEND_URL=http://localhost:5001
#MODE=production
MODE=development
```

MongoDB Atlas és Cloudinary

A projekt fejlesztése és üzemeltetése során két fontos külső szolgáltatás webes felületén is szükség lehet bejelentkezésre: **MongoDB Atlas** (az adatbázis kezelőfelülete) és **Cloudinary** (a képfeltöltések kezelése).

> **Megjegyzés:**
> Az adatbázis nem része a projektnek, hanem egy online szerveren fut, így külön adatbázis dump nem szükséges.

Rendszerterv

Architektúra áttekintése

A Yapper egy klasszikus kliens-szerver architektúrát követ, amely három fő komponensből áll:

- **Frontend (React SPA):**
  A felhasználói interakciók kezelése, az adatok vizuális megjelenítése, valamint a REST API és a valós idejű kommunikáció (Socket.IO) kezelése.
- **Backend (Node.js + Express):**
  Az üzleti logika, az adatkezelés, a hitelesítés, a jogosultságkezelés, valamint a valós idejű üzenetküldés megvalósítása.
- **Adatbázis (MongoDB):**
  A felhasználók, üzenetek, kapcsolatok, csoportok és egyéb adatok tárolása.

Fő komponensek és azok feladatai

- **Felhasználókezelés:**
  Regisztráció, email-verifikáció, bejelentkezés, jelszó-visszaállítás, profilmódosítás, jogosultságkezelés.
- **Chat modul:**
  Privát és (ha van) csoportos üzenetküldés, üzenetlista, chat előzmények, olvasottsági státuszok.
- **Értesítés modul:**
  Böngésző értesítések kezelése, push notification támogatás.

- **Teszt modul:**
  Automatizált végpont- és UI tesztek (Selenium), amelyek a fő funkciók helyes működését ellenőrzik.

Biztonsági kérdések

Jelszókezelés

- A jelszavakat bcrypt-tel hash-eljük, soha nem tároljuk őket tiszta szövegként.
- Jelszó-visszaállítás során időkorlátos, egyszer használatos tokeneket alkalmazunk.

Autentikáció és jogosultságkezelés

- A felhasználók JWT tokennel azonosítják magukat minden védett végponton.
- A tokenek érvényességi ideje korlátozott, a backend minden kérésnél ellenőrzi a jogosultságot.

Input validáció és védelem

- Minden bemeneti adatot validálunk (pl. email formátum, jelszó hossza).
- Védekezünk XSS, CSRF és egyéb tipikus webes támadások ellen.

Adatbázis leírása és modell-diagram

A Yapper alkalmazás MongoDB-re épülő adatbázisa több, egymással kapcsolatban álló dokumentumot (modellt) tartalmaz. Az alábbiakban bemutatjuk a főbb adatmodelleket és azok mezőit.

Felhasználó (User) modell

- **_id**: Egyedi azonosító (ObjectId)
- **email:** Email cím (string, kötelező, egyedi)
- **fullName:** Teljes név (string, kötelező)
- **password:** Jelszó hash (string, kötelező, minimum 4 karakter)

- **profilePicture:** Profilkép URL vagy elérési út (string, alapértelmezett: üres)
- **resetPasswordToken:** Jelszó-visszaállítás token (string, opcionális)
- **resetPasswordExpires:** Token lejárati ideje (Date, opcionális)
- **createdAt, updatedAt:** Létrehozás és módosítás dátuma (Date, automatikus)

Üzenet (Message) modell

- **_id:** Egyedi azonosító (ObjectId)
- **senderId:** Feladó felhasználó azonosítója (User._id, kötelező)
- **receiverId:** Címzett felhasználó azonosítója (User._id, opcionális – privát üzenetnél)
- **groupId:** Csoport azonosítója (Group._id, opcionális – csoportos üzenetnél)
- **text:** Üzenet szövege (string, opcionális)
- **image:** Csatolt kép URL vagy elérési út (string, opcionális)
- **createdAt, updatedAt:** Küldés és módosítás időpontja (Date, automatikus)

Csoport (Group) modell

- **_id:** Egyedi azonosító (ObjectId)
- **name:** Csoport neve (string, kötelező)
- **avatar:** Csoport profilképe (string, opcionális)
- **members:** Tagok listája (User._id tömb, kötelező)
- **admins:** Adminisztrátorok listája (User._id tömb, opcionális)
- **createdBy:** Létrehozó felhasználó (User._id, kötelező)
- **createdAt:** Létrehozás dátuma (Date, automatikus)

Barátság (Friendship) modell

- **_id:** Egyedi azonosító (ObjectId)
- **requester:** Barátságot kezdeményező felhasználó (User._id, kötelező)
- **recipient:** Barátságot fogadó felhasználó (User._id, kötelező)

- **status:** Barátság státusza (string: “pending”, “accepted”, “rejected”; alapértelmezett: “pending”)
- **createdAt, updatedAt:** Létrehozás és módosítás dátuma (Date, automatikus)

Push értesítési feliratkozás (PushSubscription) modell

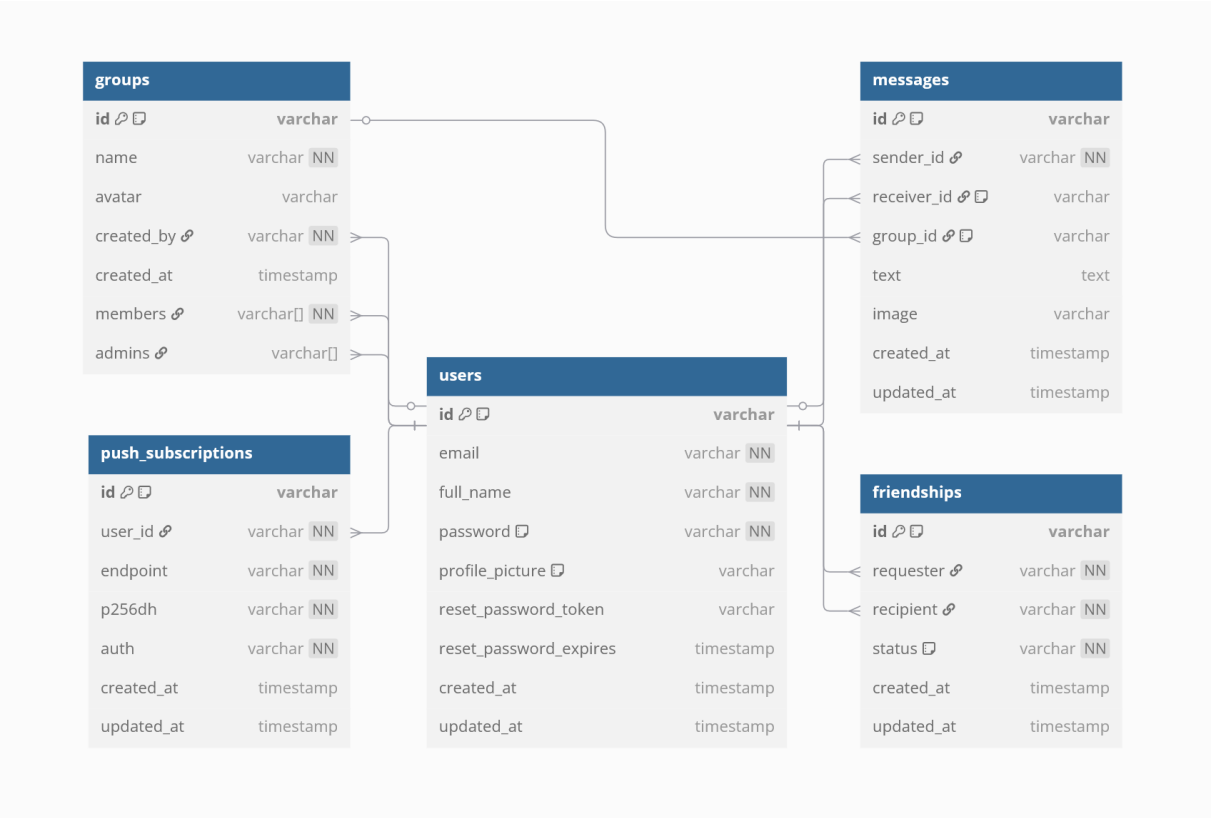
- **_id:** Egyedi azonosító (ObjectId)
- **userId:** Felhasználó azonosítója (User._id, kötelező)
- **subscription:**
  - **endpoint:** Feliratkozás végpontja (string, kötelező)
  - **keys:**
    - **p256dh:** Kulcs (string, kötelező)
    - **auth:** Kulcs (string, kötelező)
- **createdAt, updatedAt:** Létrehozás és módosítás dátuma (Date, automatikus)

Adatbázismodell-diagram (szövegesen)

- Egy **User** több **Group**-ban lehet tag és admin.
- Egy **User** több **Friendship**-ben szerepelhet requester vagy recipient szerepben.
- Egy **User** több **Message**-et küldhet (senderId), és kaphat (receiverId).
- Egy **Group**-nak több **User** tagja és adminja lehet.
- Egy **Message** vagy egy **User**-höz (privát), vagy egy **Group**-hoz (csoportos) tartozik.
- Egy **User**-nek lehet több **PushSubscription** rekordja (pl. több eszközről).

Adatbázismodell-diagram (vizuálisan)

Az alábbi ábra szemlélteti az adatbázis főbb tábláit (kollekcióit) és azok kapcsolatát:



1. ábra Adatbázismodell-diagram

Navigáció és ergonómia

A Yapper alkalmazás felhasználói felülete a modern ergonómiai elveknek megfelelően készült, hogy minden eszközön könnyen kezelhető és átlátható legyen.

- **Fő navigációs elemek:**
  - **Főoldal:** Bejelentkezés vagy regisztráció után a felhasználó az üzenetlistát látja.
  - **Csevegések:** Bal oldali sávban a privát és csoportos beszélgetések listája.
  - **Csoportok:** Külön szekcióban jelennek meg a felhasználó által létrehozott vagy adminisztrált csoportok.
  - **Barátok:** Barátlista, barátságkezelés (elfogadás, visszautasítás, törlés).
  - **Profil:** Profilkép, név, email, jelszó módosítása, push értesítések kezelése.
  - **Kijelentkezés:** Egyértelműen elérhető gomb minden nézetből.
- **Ergonómiai szempontok:**

  - **Reszponzív dizájn:** Minden funkció elérhető mobilon, tableten és asztali gépen is.
  - **Átlátható elrendezés:** A legfontosabb funkciók (üzenetküldés, csoportkezelés) gyorsan elérhetők.
  - **Visszajelzések:** Minden művelet után (pl. üzenet elküldése, barátság elfogadása) azonnali vizuális visszajelzés.
  - **Hozzáférhetőség:** Színek, kontrasztok, gombméretek megfelelnek a hozzáférhetőségi ajánlásoknak.

- **Hozzáférhetőség:** Színek, kontrasztok, gombméretek megfelelnek a hozzáférhetőségi ajánlásoknak.
- **Progresszív webalkalmazás (PWA) előnyei:**
  - **Telepíthetőség:** A Yapper PWA-ként működik, így a felhasználók egyszerűen hozzáadhatják az alkalmazást a mobiljuk vagy számítógépük kezdőképernyőjéhez, mintha natív alkalmazás lenne.
  - **Natív élmény:** A PWA offline módban is működik, gyorsan indul, és teljes képernyős, zavaró böngészőelemek nélküli élményt nyújt.
  - **Push értesítések:** A böngésző push notification támogatásának köszönhetően a felhasználók valós időben értesülhetnek az új üzenetekről vagy eseményekről, akkor is, ha az alkalmazás nincs megnyitva.
  - **Automatikus frissítés:** A PWA automatikusan frissül a háttérben, így a felhasználók mindig a legújabb verziót használják, külön telepítés vagy frissítés nélkül.
  - **Platformfüggetlenség:** Ugyanaz a felhasználói élmény biztosított minden modern böngészőn, legyen szó Androidról, iOS-ről vagy asztali rendszerről.

Ezek az ergonómiai és technológiai előnyök hozzájárulnak ahhoz, hogy a Yapper használata kényelmes, gyors és megbízható legyen minden eszközön, a natív alkalmazásokhoz hasonló élményt nyújtva.

Főbb funkcionális tesztesetek

Az alkalmazás fejlesztése során automatizált és manuális teszteket is végeztünk. Az alábbiakban néhány főbb funkcionális tesztesetet sorolunk fel:

- **Regisztráció és email-verifikáció:**
  - Helyes adatokkal sikeres regisztráció és verifikáció.
  - Hibás email vagy jelszó esetén hibaüzenet jelenik meg.
- **Bejelentkezés:**
  - Helyes adatokkal sikeres bejelentkezés.
  - Hibás adatokkal hibaüzenet jelenik meg.
- **Üzenetküldés:**
  - Privát és csoportos üzenetek küldése, fogadása.
  - Kép csatolása üzenethez.
- **Barátságkezelés:**
  - Barátküldés, elfogadás, visszautasítás, törlés.
- **Csoportkezelés:**
  - Új csoport létrehozása, tagok hozzáadása/eltávolítása, admin jogok kezelése.
- **Push értesítések:**
  - Böngésző értesítések engedélyezése, tesztelése több eszközön.
- **Jelszó-visszaállítás:**
  - Elfelejtett jelszó funkció működése, tokenes visszaállítás.

Teszteredmények dokumentációja

Az automata tesztek a `frontend/tests` mappában találhatók, futtatásukról és eredményeikről a `README.md` is tartalmaz információt.

Frontend tesztelés (Selenium)

A frontend automatizált tesztelését Seleniummal végeztük, amely során a következő fő funkciókat ellenőriztük:

- **Regisztráció:**
  - **Input:** Helyes email, név, jelszó megadása.
  - **Elvárt válasz:** Sikeres regisztráció, visszaigazoló üzenet.
- **Bejelentkezés:**
  - **Input:** Létező felhasználó email és jelszó.
  - **Elvárt válasz:** Sikeres bejelentkezés, átirányítás a főoldalra.
- **Üzenetküldés:**
  - **Input:** Üzenet szövegének beírása és elküldése.
  - **Elvárt válasz:** Az üzenet megjelenik a beszélgetésben.
- **Barátságkezelés:**
  - **Input:** Barátküldés, elfogadás, törlés gombok használata.
  - **Elvárt válasz:** Barátlista frissül, státusz változik.
- **Csoportkezelés:**
  - **Input:** Új csoport létrehozása, tagok hozzáadása.
  - **Elvárt válasz:** Csoport megjelenik a listában, tagok láthatók.
- **Push értesítések:**
  - **Input:** Értesítések engedélyezése a böngészőben.
  - **Elvárt válasz:** Új üzenet érkezésekor értesítés jelenik meg.

Backend tesztelés (Postman)

A backend API végpontjait Postmannel teszteltük, minden végpontnál ellenőriztük a helyes működést különböző inputokkal. Az alábbiakban részletesen bemutatjuk az egyes végpontokat, a teszteléshez használt inputokat és az elvárt válaszokat:

/auth

- **POST /signup**
  - **Input:**

```
{ "email": "teszt@example.com",
"fullName": "Teszt Elek",
"password": "jelszo123" }
```

  - **Elvárt válasz:**
    201 Created, visszaadott user objektum, email verifikáció szükséges.

- **POST /verify-email**
  - **Input:**

    ```
    { "token": "<helyes_verifikacios_token>" }
    ```

  - **Elvárt válasz:**
    200 OK, email igazolva.
- **POST /login**
  - **Input:**

    ```
    { "email": "teszt@example.com", "password": "jelszo123" }
    ```

  - **Elvárt válasz:**
    200 OK, JWT token, user adatok.
- **POST /logout**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, kijelentkezés megerősítése.
- **PUT /update-profile**
  - **Input:**
    Auth token a headerben, módosítandó adatok pl.

    ```
    { "fullName": "Teszt Elek Új",
     "profilePicture": "https://..." }
    ```

  - **Elvárt válasz:**
    200 OK, frissített user adatok.
- **GET /check**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, user adatok visszaadása.
- **POST /request-password-reset**
  - **Input:**

    ```
    { "email": "teszt@example.com" }
    ```

  - **Elvárt válasz:**
    200 OK, email elküldve a reset linkkel.
- **POST /reset-password**
  - **Input:**

    ```
    { "token": "<helyes_reset_token>", "password": "ujjelszo123" }
    ```

  - **Elvárt válasz:**
    200 OK, jelszó sikeresen módosítva.

/friendship

- **POST /request**
  - **Input:**
    Auth token a headerben,

    ```
    { "recipientId": "<másik_user_id>" }
    ```

  - **Elvárt válasz:**
    200 OK, barátkérelem elküldve.
- **POST /accept**
  - **Input:**
    Auth token a headerben,

    ```
    { "requestId": "<friendship_request_id>" }
    ```

  - **Elvárt válasz:**
    200 OK, barátság elfogadva.
- **POST /reject**
  - **Input:**
    Auth token a headerben,

    ```
    { "requestId": "<friendship_request_id>" }
    ```

  - **Elvárt válasz:**
    200 OK, barátság elutasítva.
- **GET /friends**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, barátok listája.
- **GET /pending**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, függőben lévő barátkérelmek.
- **POST /unfriend**
  - **Input:**
    Auth token a headerben,

    ```
    { "friendId": "<barát_user_id>" }
    ```

  - **Elvárt válasz:**
    200 OK, barátság megszüntetve.

/message

- **GET /users**

  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, üzenetküldésre elérhető felhasználók listája.
- **GET /:id**
  - **Input:**
    Auth token a headerben, :id a másik felhasználó vagy csoport azonosítója.
  - **Elvárt válasz:**
    200 OK, beszélgetés üzenetei.
- **POST /:id**
  - **Input:**
    Auth token a headerben,

    ```
    { "text": "Szia!", "image": null }
    ```

  - **Elvárt válasz:**
    201 Created, elküldött üzenet objektuma.

- **POST /:messageId/reactions**
  - **Input:**
    Auth token a headerben,

    ```
    { "reaction": "👍" }
    ```

  - **Elvárt válasz:**
    200 OK, reakció hozzáadva.

/group

- **POST /**
  - **Input:**
    Auth token a headerben,

    ```
    { "name": "Teszt csoport",

     "members": ["<user_id1>", "<user_id2>"] }
    ```

  - **Elvárt válasz:**
    201 Created, létrehozott csoport objektuma.

- **GET /**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, felhasználó csoportjai.
- **GET /:groupId**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, csoport részletes adatai.

- **PUT /:groupId**
  - **Input:**
    Auth token a headerben,

    ```
    { "name": "Új csoportnév" }
    ```

  - **Elvárt válasz:**
    200 OK, frissített csoport adatok.
- **DELETE /:groupId**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, csoport törölve.
- **POST /:groupId/members**
  - **Input:**
    Auth token a headerben,

    ```
    { "userId": "<új_tag_id>" }
    ```

  - **Elvárt válasz:**
    200 OK, tag hozzáadva.
- **DELETE /:groupId/members**
  - **Input:**
    Auth token a headerben,

    ```
    { "userId": "<tag_id>" }
    ```

  - **Elvárt válasz:**
    200 OK, tag eltávolítva.
- **POST /:groupId/messages**
  - **Input:**
    Auth token a headerben,

    ```
    { "text": "Csoportos üzenet", "image": null }
    ```

  - **Elvárt válasz:**
    201 Created, elküldött csoportüzenet.
- **GET /:groupId/messages**
  - **Input:**
    Auth token a headerben.
  - **Elvárt válasz:**
    200 OK, csoport üzenetei.
- **POST /messages/:messageId/reactions**
  - **Input:**
    Auth token a headerben,

    ```
    { "reaction": "😂" }
    ```

- **Elvárt válasz:**
  200 OK, reakció hozzáadva.

/push

- **GET /vapid-key**
  - **Input:**
    Nincs szükség autentikációra.
  - **Elvárt válasz:**
    200 OK, VAPID public key.
- **POST /subscribe**
  - **Input:**
    Auth token a headerben,

    ```
    { "endpoint": "...", "keys": { "p256dh": "...", "auth": "..." }
    }
    ```

  - **Elvárt válasz:**
    201 Created, feliratkozás mentve.
- **POST /send-notification**
  - **Input:**
    Auth token a headerben,

    ```
    { "userId": "<címzett_id>", "title": "Új üzenet", "body": "Üzen
    et szövege" }
    ```

  - **Elvárt válasz:**
    200 OK, értesítés elküldve.

Hibakezelés

Minden végpontnál teszteltük a hibás inputokat is (pl. hiányzó mezők, hibás token, nem létező azonosító), ilyenkor a szerver megfelelő hibaüzenetet adott vissza (pl. 400 Bad Request, 401 Unauthorized, 404 Not Found).

A tesztek eredményei alapján az összes végpont a dokumentált módon, helyesen működik.

Manuális tesztek

A manuális tesztelést rendkívül egyszerűen tudtuk végrehajtani, hiszen az alkalmazás jellegéből adódóan mindannyian napi szinten használunk chat alkalmazásokat – így csak át kellett térnünk a saját fejlesztésű rendszerünkre. Osztálytársak is segítettek a tesztelésben, kipróbálták az alkalmazást különböző eszközökön, és visszajelzéseikkel hozzájárultak a hibák feltárásához és a felhasználói élmény javításához.

Teszteredmények:

A tesztek sikeres futtatása igazolja, hogy az alkalmazás fő funkciói helyesen működnek fejlesztői környezetben. A tesztek futtatása során minden fő funkciót ellenőriztünk, hibátlan működés esetén a tesztek zöld státuszt adnak vissza. Ezek az automatizált tesztesetek futtatásakor

Összefoglalás

A fenti fejezetek részletesen bemutatták a Yapper alkalmazás fejlesztői oldalát: a követelményeket, a rendszertervet, a biztonsági szempontokat, az adatbázis felépítését, a navigációt, ergonómiát, valamint a főbb teszteseteket és azok eredményeit. A következő fejezetben a felhasználói dokumentáció olvasható.

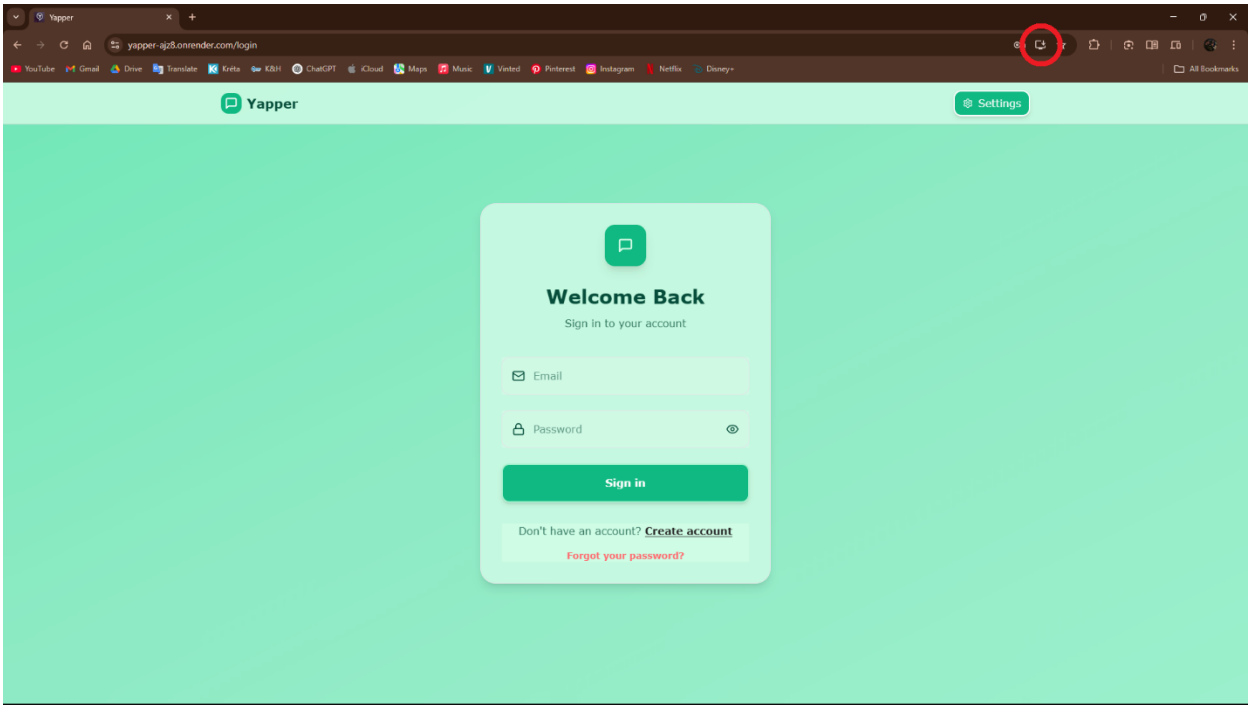
Felhasználói útmutató

A Yapper webes chat-alkalmazás felületén mobil és asztali eszközökön tudunk alapvető internetes kommunikációkat fenntartani ismerőseinkkel. Barátainkkal.

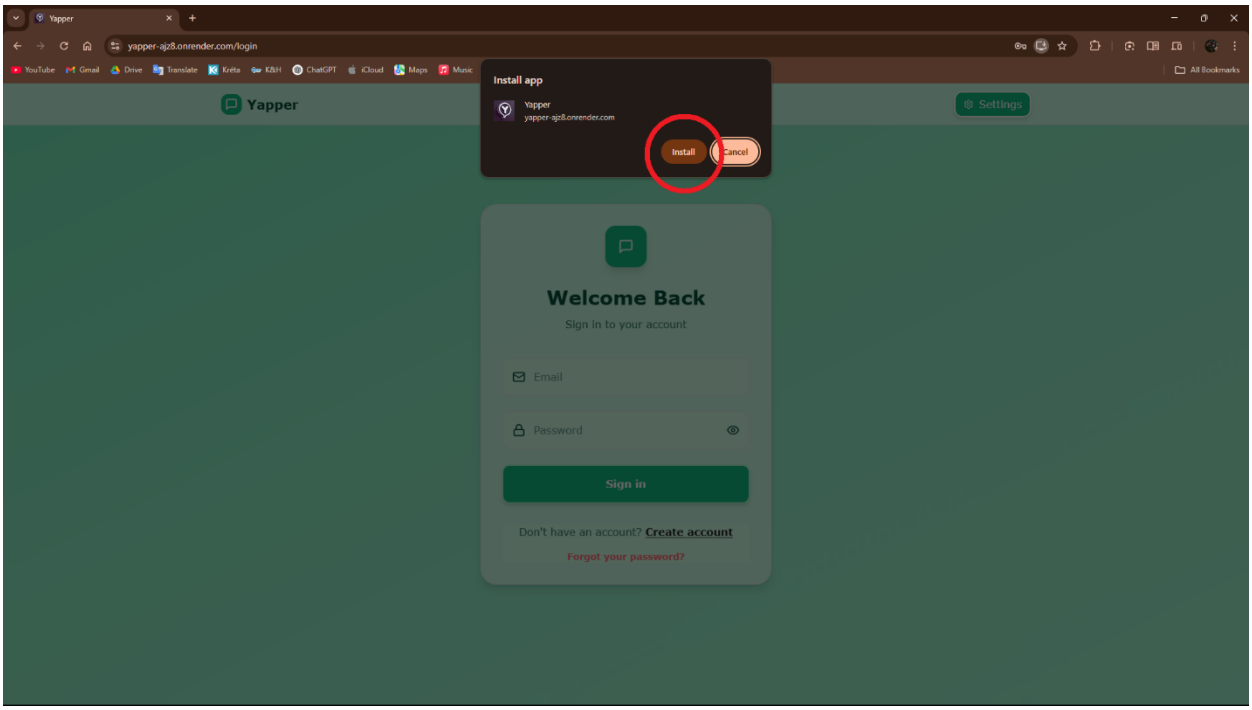
A Yapper intuitiv, egyedi felhasználói felületének köszönhetően anélkül navigálhatunk mind a beszélgetéseink, mind a beállításaink között, ennek bemutatására szolgál ez a fejezet.

Regisztráció/Bejelentkezés és Beállítások

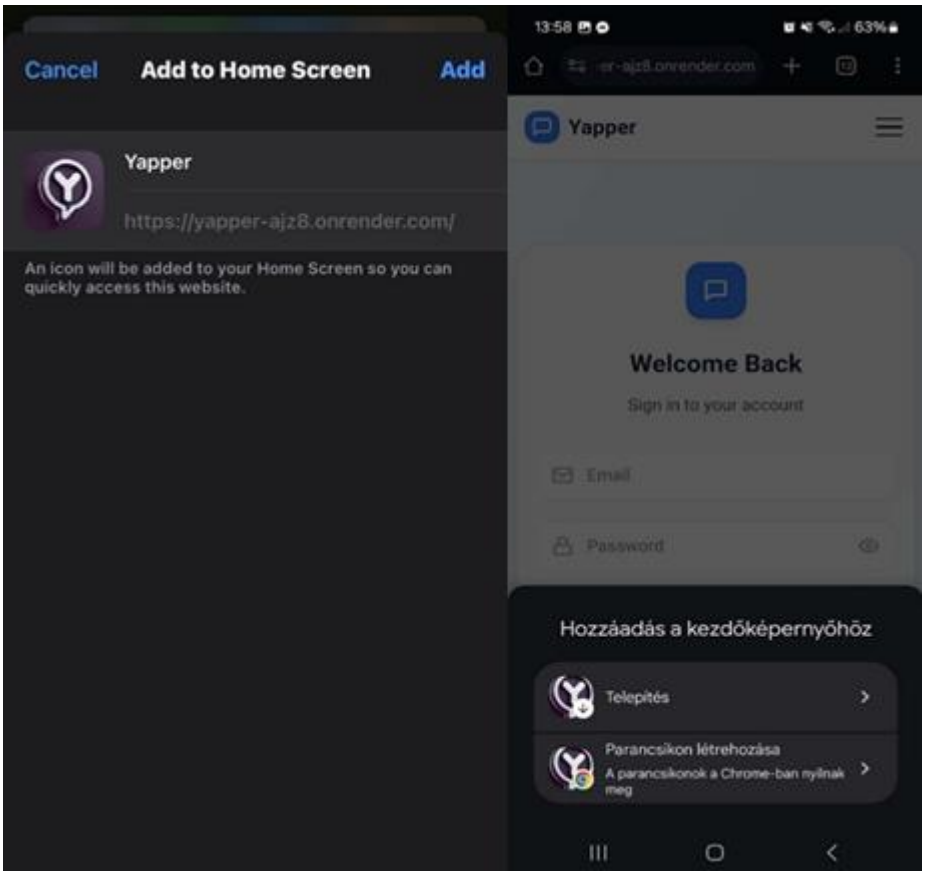
A weboldal betöltésekor, a garantált stabil működés érdekében érdemes Windows és MacOS rendszereken letölteni, Android és IOS rendszereken kezdőképernyőhöz adni a webes alkalmazást.



2. ábra Yapper asztalhoz adása Desktop



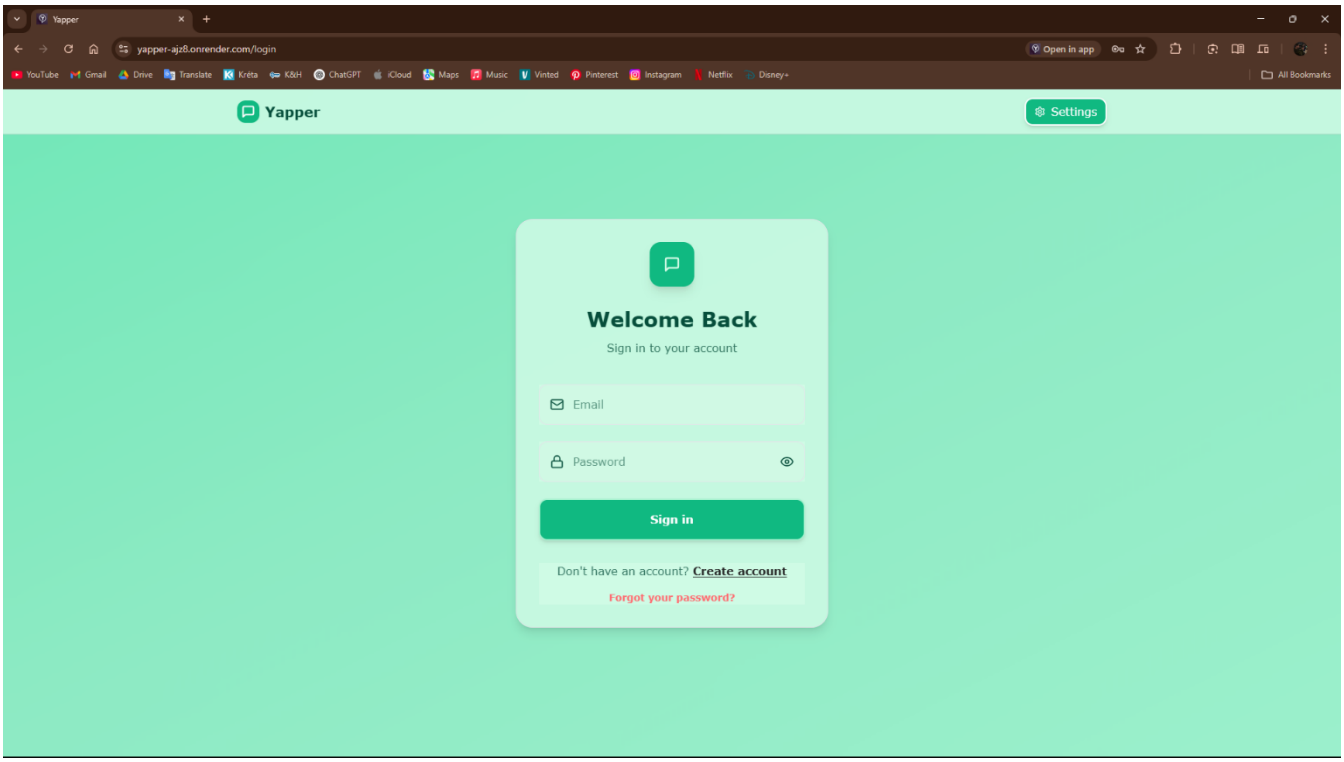
3. ábra Yapper asztalhoz adása Desktop 2



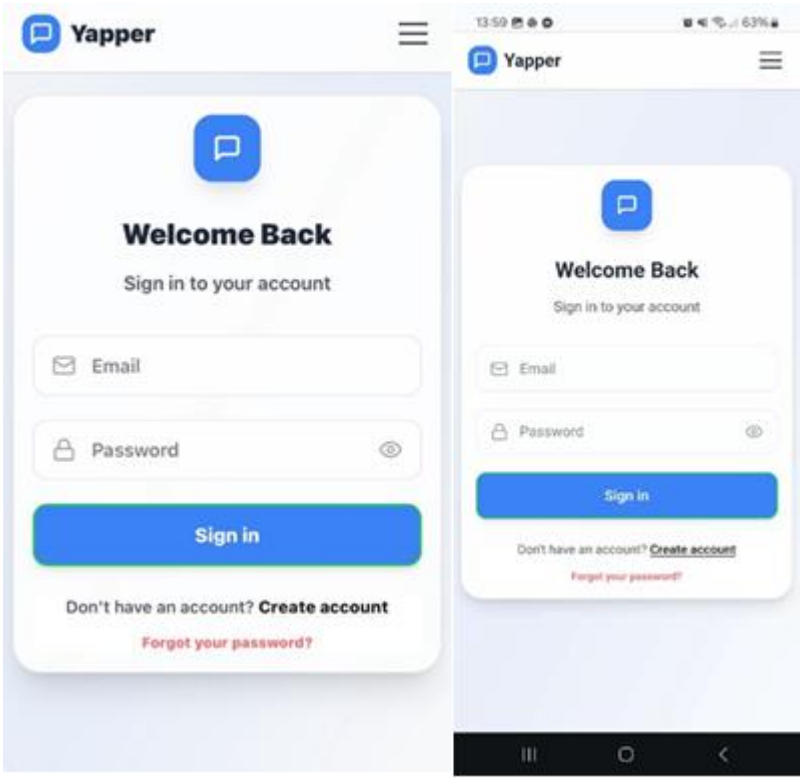
4. ábra Yapper főképernyőhöz adása IOS/Android

Most, hogy megfelelő eszközünkön elvégeztük ezt a műveletet, külön ablakban/alkalmazásban nyílik meg a Yapper. A Bejelentkezés (Login) ablak kerül elénk elsőként, ugyanakkor, ha még új felhasználók vagyunk, egyszerűen regisztrálhatunk magunknak egy új fiókot a beviteli

mezők alatti linkre kattintva. Ekkor kerülünk át a regisztrációs (Create account) oldalra, ahol megadva hiteles adatainkat léphetunk tovább.

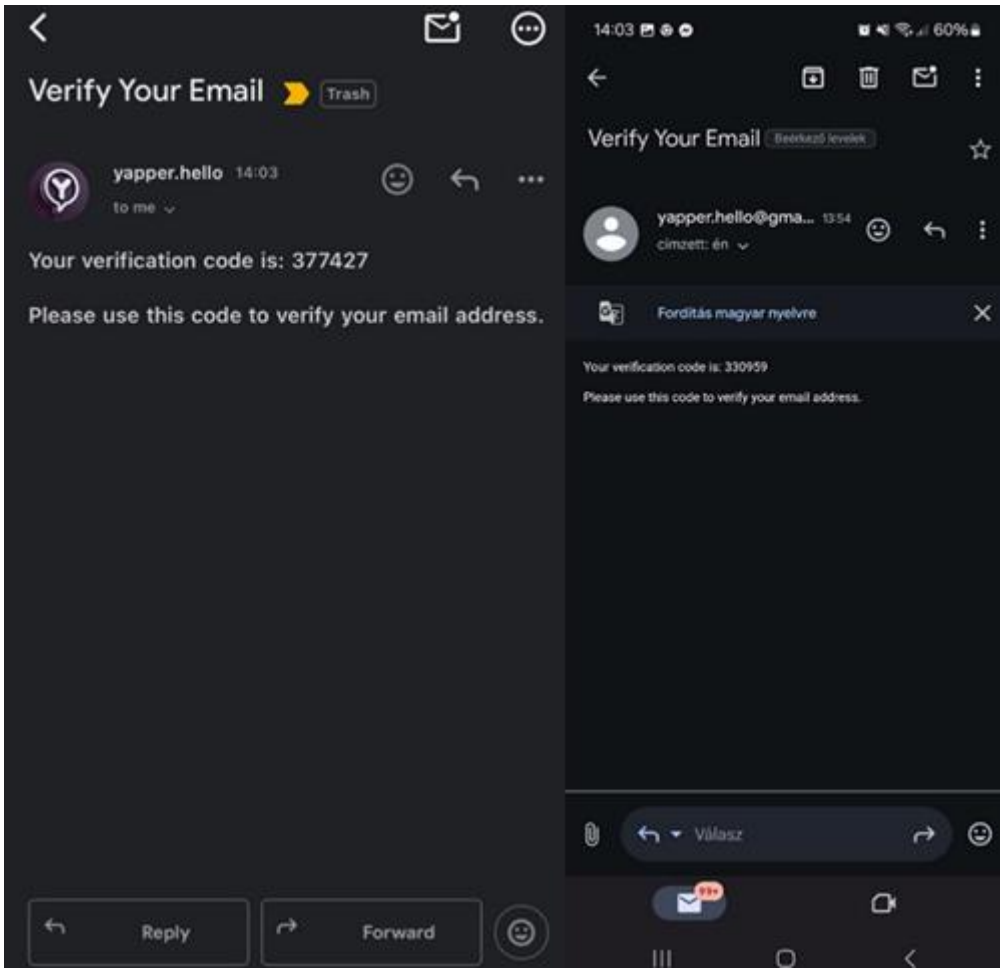


5. ábra Create Account Desktop



6. ábra Create Account Mobil

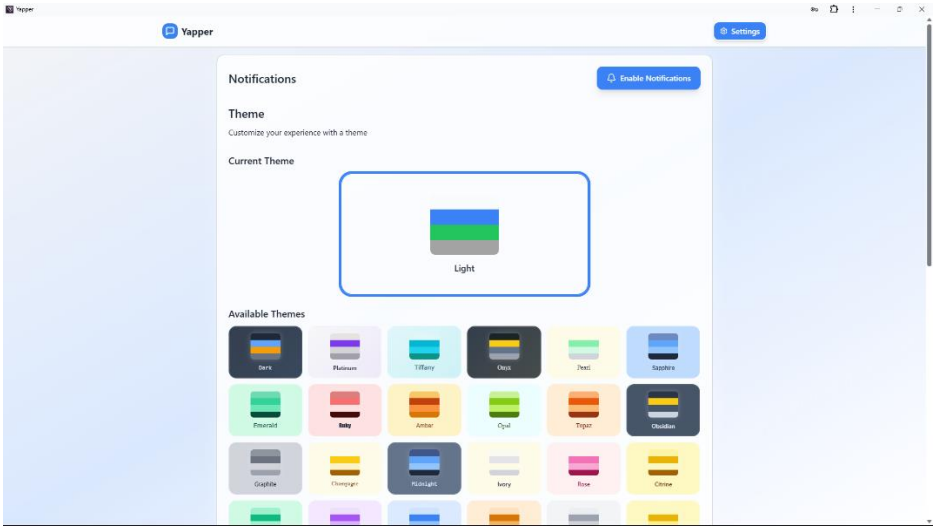
Ekkor a rendszer a megadott e-mail címre küld a frissen regisztrált felhasználónak egy elektronikus levelet, amely a regisztráció hitelesítéséhez szükséges számsorozatot tartalmazza.



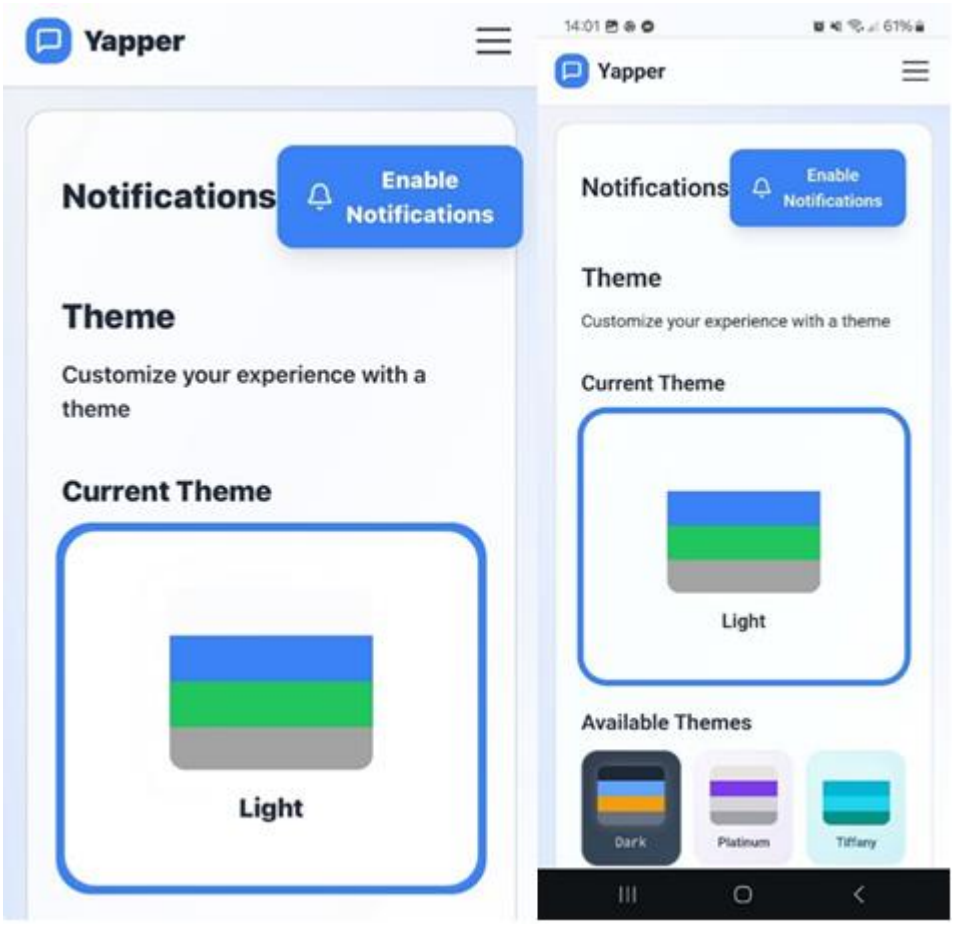
7. ábra Email hitelesítés

FONTOS: ÉRDEMES MEGTEKINTENI A SPAM TÁROLÓT A LEVELET KERESVE

Miután sikeresen elfogadta a rendszer új fiókunk regisztrációját, nincs más dolgunk, mint bejelentkezni a megadott e-mail címmel, jelszóval. Ugyanakkor még mielőtt használatba vehetnénk új fiókunkat, az előbbi elvégzett lépsek bármelyike előtt beléphetünk a Beállítások (Settings) oldalra, ahol is elsősorban engedélyezhetjük a rendszer értesítéseket (Enable Push Notifications), illetve személyre szabhatjuk a Yapper-élményünket különféle témákkal.



8. ábra Settings oldal Desktop

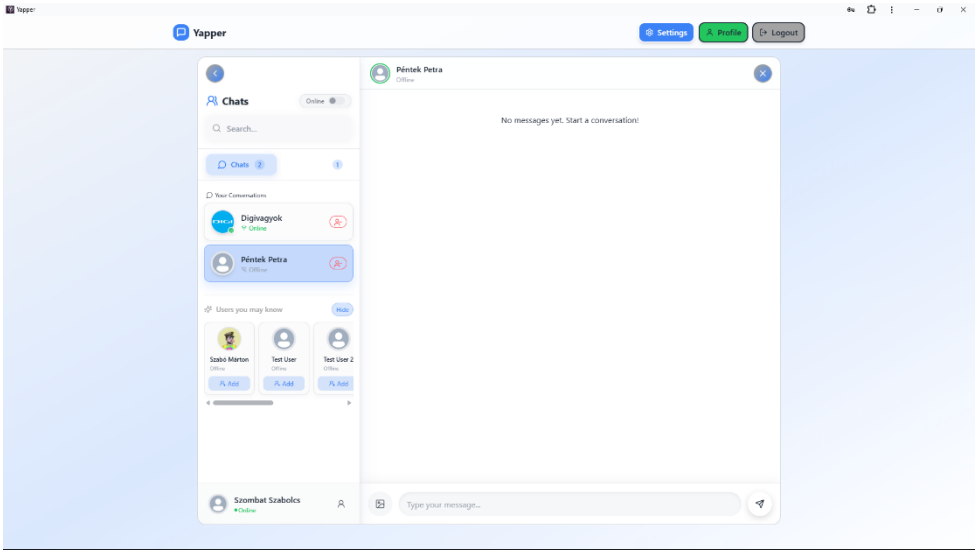


9. ábra Settings oldal Mobil

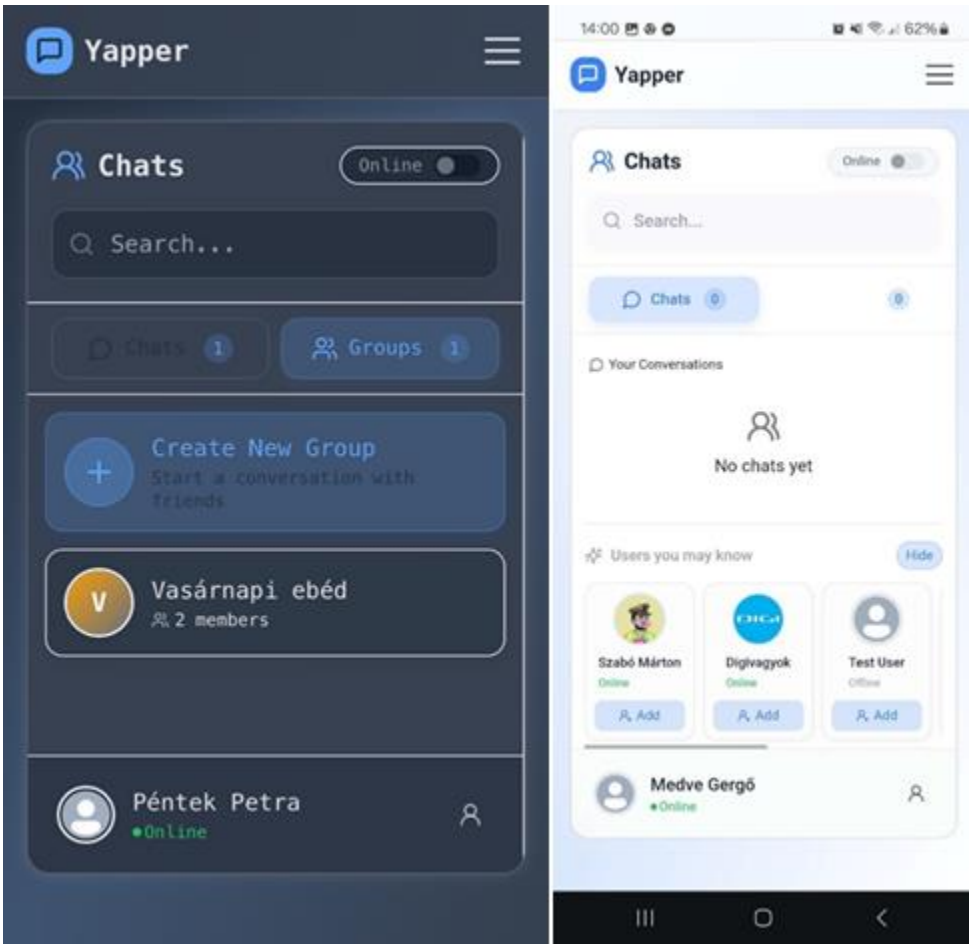
Ezekkel a lépésekkel el is készültünk a szükséges fázisokkal, ahhoz, hogy használatba vehessük a Yapper-t.

Chat-ek (DM és Group)

Ha bejelentkezünk a létrehozott fiókunkkal, láthatjuk egy sidebar-on, hogy kik a Yapper felhasználói, ezáltal könnyedén küldhetünk ismerőseinknek bárat kérelmet, amit ők elfogadhatnak, vagy elutasíthatnak.



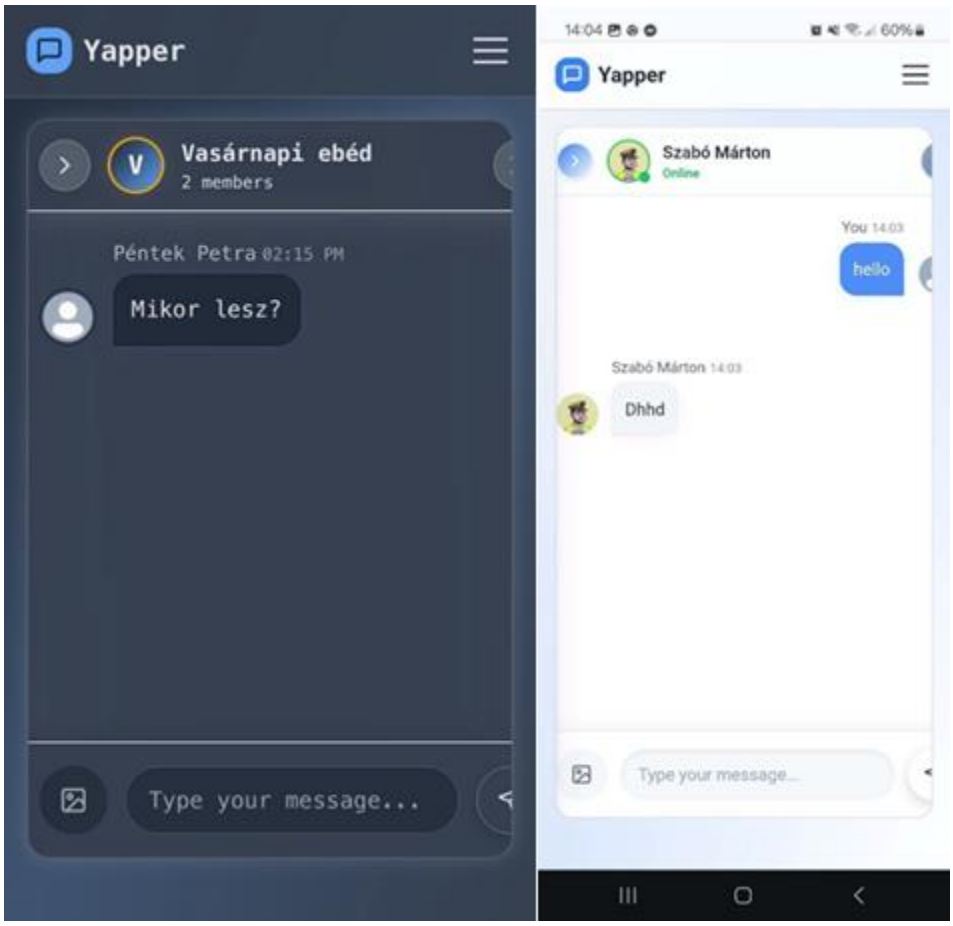
10. ábra Chat nézet Desktop



11. ábra Chat és group nézet Mobil

Ha elfogadják a kérelmünket, megnyílik a lehetőség, hogy beszélgetést kezdeményezhessünk vele. A Yapper felületén a felhasználók szöveges üzenet (beleértve hangulatjelek, emojik) illetve képes formában tudnak egymással kommunikálni, egymás üzeneteire válaszolni, hangulatjellel reagálni.

Amennyiben szeretnénk a barátainknak egy külön csoportot - *group chatet* – létrehozni, a teendőnk csupán annyi, hogy a sidebar-on található ’Chats’ és ’Groups’ fülek közül kiválasztjuk a Groups fület, majd a Create New Group gombra kattintunk. Így lehetőséget kapunk elnevezni a csoportot és a barátainkat is invitálni. Ha létrehoztuk a csoportot a kívánt névvel és tagokkal, az összes csoportot melynek az aktív felhasználó a létrehozója és/vagy a tagja ezentúl ezen a fülön találja. Belépve egy csoport beszélgetésébe láthatjuk a chat fejlécében a kiválasztott csoport nevét és tagjainak a számát, illetve a csoport adminjának lehetősége van azt törölni is. Az üzenetküldés lehetőségei és folyamatai pedig megegyeznek a privát üzenetekével.

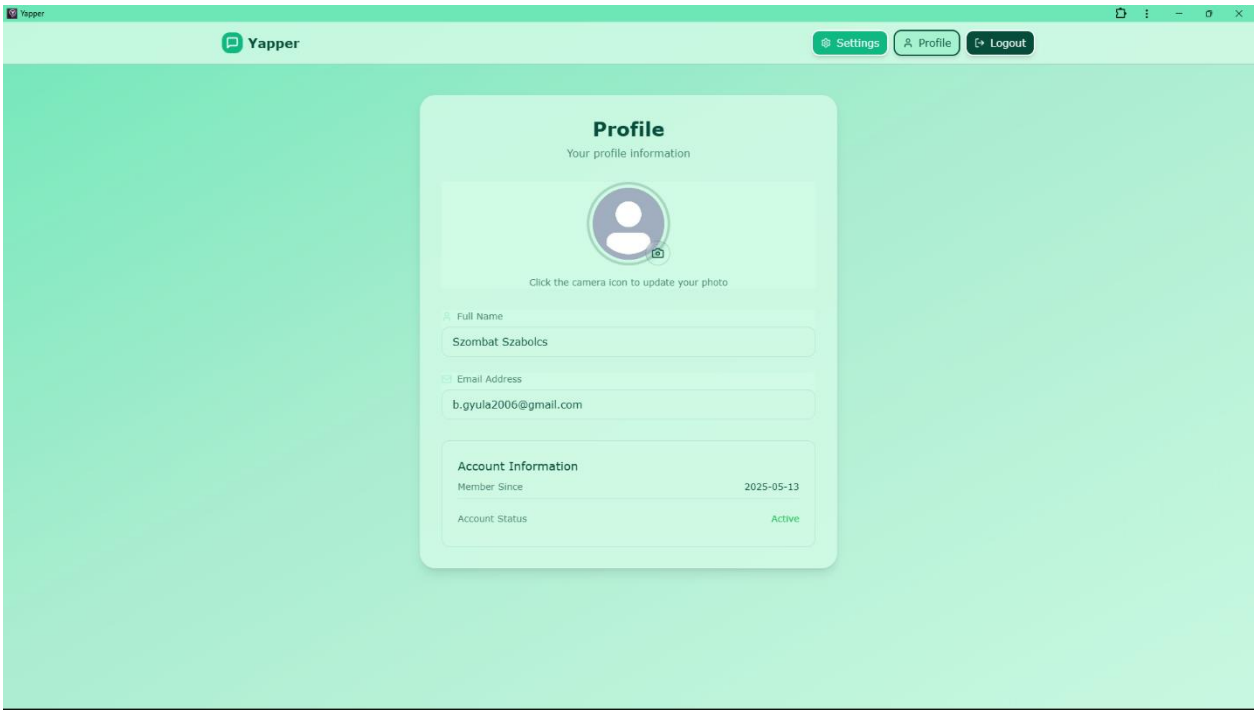


12. ábra Chat nézet Mobil

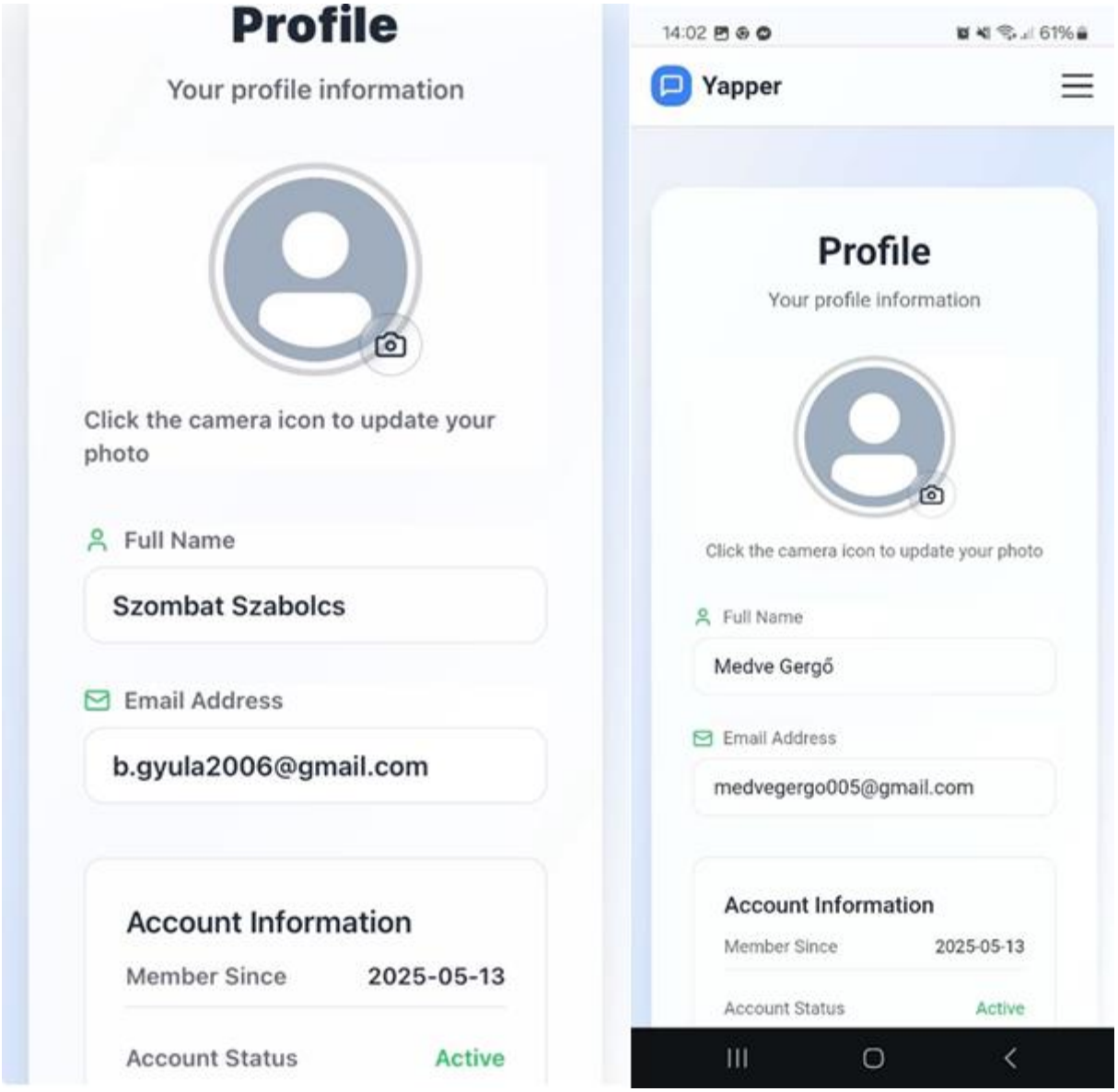
Navigation Bar

A navigation bar-on láthatjuk az alkalmazás nevét és egy üzenet ikont. Ez az ikon kezdőlap gombként funkcionál. Ha esetleg a navigation bar egy eltérő eleme aktív, a beszelgetéseinkhez való visszatérés érdekében erre kell kattintanunk.

Megtalálható továbbá egy, a beállításokhoz vezető gomb (Settings), a Profile gombbal megnézhetjuk a felhsználónk adatait (név és e-mail cím), továbbá egyedi profilképet tudunk beállítani.



13. ábra Profil oldal Desktop



14. ábra Profil oldal Mobil

Az utolsó gomb pedig a kijelentkezéshez használatos gomb (Logout), ez a gomb felelős a kiléptetésért és a bejelentkezési oldalra irányításért, ahonnan a felhasználó újra beléphet, vagy be is zárhatja a programot

Irodalomjegyzék, hivatkozásjegyzék

A fejlesztés során az alábbi forrásokat, oktatóanyagokat és dokumentációkat használtuk inspirációként vagy technikai referenciaként:

- YouTube – Build and Deploy a Full Stack Chat App with React, Node, Socket.io and MongoDB
- burakorkmez/fullstack-chat-app
- YouTube – Real Time Chat App with React, Node.js, Socket.io, MongoDB
- daisyUI dokumentáció
- YouTube – React Chat App with Socket.io and MongoDB
- YouTube – Push Notifications in React with Service Workers

Bizonyos elemeket, ötleteket és megoldásokat ezekből a forrásokból merítettünk, illetve a fejlesztés során ezek segítettek a technikai problémák megoldásában és a modern felhasználói élmény kialakításában.

Összegzés

A Yapper projekt fejlesztése során egy modern, valós idejű chat alkalmazást hoztunk létre, amely mind funkcionálisan, mind technikailag megfelel a mai elvárásoknak. A fejlesztés során számos új technológiát és eszközt ismertünk meg, különös hangsúlyt fektetve a biztonságra, a felhasználói élményre és a reszponzivitásra. Az alkalmazás sikeresen teljesíti a kitűzött követelményeket: támogatja a privát és csoportos üzenetküldést, a barátkezelést, a push értesítéseket, valamint a képek és reakciók kezelését is.

Tapasztalatok

A projekt során értékes tapasztalatokat szereztünk a teljes stack fejlesztésben, a valós idejű kommunikáció (socket.io), a felhőalapú szolgáltatások (MongoDB Atlas, Cloudinary) és a PWA technológiák használatában. Megtanultuk, hogyan lehet egy komplex rendszert modulárisan, jól tesztelhetően és biztonságosan felépíteni. A csapatmunka és a folyamatos tesztelés révén sikerült egy stabil, könnyen bővíthető alkalmazást létrehozni, amely a mindennapi chat alkalmazásokhoz hasonló élményt nyújt.

A fejlesztés során felmerülő problémákat közösen oldottuk meg, és a visszajelzések alapján folyamatosan javítottuk az alkalmazást. A projekt hozzájárult szakmai fejlődésünkhöz, és megerősítette a modern webes technológiákban való jártasságunkat.