Békéscsabai SZC Nemes Tihamér Technikum és Kollégium

Szakképesítés megnevezése: Szoftverfejlesztő és -tesztelő

Azonosító száma: 5 0613 12 03

VIZSGAREMEK

Snakker

Készítették:

Bánszky Balázs Ferenc	Osztály: 13.C	Oktatási azonosító: 7xxxxxxxxxx
Bencsik Attila	Osztály: 13.C	Oktatási azonosító: 72600228125

Békéscsaba, 2024/2025

1. Bevezetés

A Message-App egy valós idejű internetes webalkalmazás amely képes a felhasználókat összekötni üzenet csere céljából. Az alkalmazás ezen túl rendelkezik extra funkciókkal mint például a lehetőség barátok bejelölésére, a felhasználó barátainak valós idejű felhasználói státusz közlése a felhasználóval. Üzeneteket “csatornákon” keresztül lehet váltani, ezeknek a csatornáknak több felhasználó is a tagja lehet. Lehetőség van a csatornákat csoportosítani (akár témájuk szerint, vagy más közös tulajdonság alapján) úgynevezett “csatorna csomagokba”, vagy röviden “csomagokba”. A felhasználók egy meghívó által csatlakozhatnak egy csatornához, vagy egy csomaghoz és annak összes csatornájához. A személyre szabhatóság is egy szempont volt az alkalmazás kifejlesztésében: az oldal témája változtatható, a felhasználók profilképet tölthetnek fel maguknak.

Miért ezt választottuk?

Egy üzenetelő alkalmazás kifejlesztése nagy területet lefed a szoftverfejlesztés ágai közt: a hitelesítés, valós idejű kommunikáció, adatok rendszerezése, fájlok feltöltése és kezelése különböző készségeket igényelt, amelyek továbbtanulása és gyakorlása további jártassággal látott el minket a fejlesztés során.

Úgy gondoljuk, hogy egy valós idejű rendszer megtervezése és megvalósítása sok tapasztalattal láthat el minket a komplexitási miatt, ezzel jobban felkészítve minket jövőre.

2. Indoklás

A mai világban szinte mindenkinek kulcsfontosságú az online kapcsolattartás. Lehet ez kapcsolattartás ismerősökkel, szakmai kommunikáció vagy új kapcsolatok szerzése az online térben. A Snakker alkalmazás ezt az igényt hivatott kiszolgálni. A sok üzenetváltó app közül a Snakker-t kiemeli a könnyű használhatósága, letisztult kezelőfelülete és a felesleges, figyelemelterelő és a munkamenetet lassító funkciók hiánya.

3. Fejlesztői dokumentáció

3.1 Követelmények:

A Snakker webalkalmazásnak képesnek kell lennie a következőkre:

1. Valós idejű (real-time) kapcsolatot létrehozni kettő vagy több felhasználó között.

2. Felhasználói érzékeny adatokat biztonságos és titkosított formában tárolni és kezelni.
3. A kinézet személyre szabhatóságának lehetőségét nyújtani a felhasználónak.
4. Gyors és könnyen kezelhető felhasználói felület (UI).

3.2 Rendszerterv:

3.2.1 Backend:

Backend framework-nek a Node.js-t választottuk. Azért esett erre a választás mivel nagyon elterjedt a használata szoftver rendszerek körében. Továbbá rengeteg modul áll rendelkezésre az npm csomagkezelőn keresztűl ezzel gyorsítva a fejlesztési folyamatot. Végsősorban sok tapasztalatunk van a Node.js keretrendszerrel, így egy szilárdabb terved voltunk képesek készíteni, és tisztább fejjel kezdhettünk neki a munkának.

Az API úgy épül fel, hogy a felhasználó által kezdeményezett esemény minden esetben egy HTTP kéréssel valósuljon meg. Ha egy hiba lép fel a kérés feldolgozása közben, vagy hibás a kapott adat, egy { error: string } üzenettel tér vissza, ahol a “hiba” mező a hibát magyarázza el. A következőkben a várt és visszaküldött adatok TypeScript formátumban vannak definiálva.

3.2.1.1 Felhasználókat érintő API

- POST /api/login

  A várt adat:

  {
  username: string
  password: string
  }

  Bejelentkezteti a felhasználót, vagy hibád küld vissza, ha a felhasználónév és jelszó nem megfelelő

- POST /api/register

  A várt adat:

{

username: string

password: string

}

Regisztrál egy új felhasználót, vagy hibát küld vissza, ha a felhasználó már létezik.

- POST /api/logout

  Kijelentkezteti a felhasználót.

- GET /api/user

  A visszaküldött adat:

  {

  id: string

  username: string

  nickname: string

  password: string

  lastChannelId?: string

  }

  Visszaküldi az aktuálisan bejelentkezett felhasználó adatait.

- PATCH /api/user

  A várt adat:

  {

  nickname?: string

  password?: string

  theme?: number

  }

  Módosítja a felhasználó nevét, jelszavát vagy témáját.

- PUT /api/user/avatar

  A várt adat a fájlt tartalma.

  Lecseréli a felhasználó profilképét a feltöltött képre. A kép automatikusan méretezve, vágva és konvertálva lesz WEBP formátumba.

- DELETE /api/user/avatar

  Törli a felhasználó profilképét

- GET /api/user/{id}

  A visszaküldött adat:

  {

  id: string

  username: string

  nickname: string

  isOnline: boolean

  }

  Lekéri az {id} azonosítójú felhasználó adatait.

- GET /api/user/{id}/avatar.webp

  A visszaküldött adat a fájlt tartalma.

  Lekéri az {id} azonosítójú felhasználó profilképét.

- GET /api/user/search

  A várt adat:

  {

  nickname: string

  }

  A visszaküldött adat:

  {

id: string

username: string

nickname: string

isOnline: boolean

}[]

Keresést végez a felhasználók között, majd sorba rendezi a megadott név hasonlósága alapján.

3.2.1.2 Csatornákat érintő API

- GET /api/channels

  A visszaküldött adat:

  {

  id: string

  name: string

  ownerId: string

  }[]

  Lekéri az összes csatornát, aminek a bejelentkezett felhasználó a tagja.

- POST /api/channels

  A visszaküldött adat:

  {

  name: string

  }

  Létrehoz egy új csatornát a megadott néven.

- GET /api/channels/{id}

  A visszaküldött adat:

  {

id: string

name: string

ownerId: string

}

Lekéri az {id} azonosítójú csatorna adatait.

- GET /api/channels/{id}/users

  A visszaküldött adat:

  {

  id: string

  username: string

  nickname: string

  isOnline: boolean

  }[]

  Lekéri az összes felhasználót, akik az {id} azonosítójú csatornának tagja.

- POST /api/channels/{id}/leave

  Eltávolítja a bejelentkezett felhasználót az {id} azonosítójú csatornából. Ha a csatornának nincs több tagja, az összes üzenet, majd a csatorna is törlésre kerül.

3.2.1.3 Csomagokat érintő API

- GET /api/bundles

  A visszaküldött adat:

  {

  id: string

  name: string

  }[]

  Lekéri az összes csatorna-csomagot, amelynek a bejelentkezett felhasználó tagja.

- POST /api/bundles

  A várt adat:

  {

  name: string

  channels: string[]

  }

  Létrehoz egy új csomagot a megadott névvel és csatornákkal.

- GET /api/bundles/{id}

  A visszaküldött adat:

  {

  id: string

  name: string

  }

  Lekéri az {id} azonosítójú csomag adatait.

- POST /api/bundles/{id}/leave

  Kilépteti a bejelentkezett felhasználót az {id} azonosítójú csatorna-csomagból.

- GET /api/bundles/{id}/channels

  A visszaküldött adat:

  {

  id: string

  name: string

  ownerId: string

  }

  Lekéri a csomagot tartalmazó csatornákat.

- POST /api/bundles/{id}/channels

  A várt adat:

  {

  id: string

  }

  Hozzáad egy már meglévő csatornát az {id} azonosítójú csatorna-csomaghoz.

- DELETE /api/bundles/{id}/channels

  Törli a csatornát a csatorna-csomagból. Ez nem törli a csatornát, csupán a csomagból távolítja el.

3.2.1.4 Üzeneteket érintő API

- GET /api/channels/{id}/messages

  A visszaküldött adat:

  {

  id: string

  content: string

  attachmentCount: string

  createdUtc: bigint

  channelId: string

  senderId: string

  senderNickname: string

  attachments: string[]

  }[]

  Lekéri az összes üzenetet az {id} azonosítójú csatornából.

- POST /api/channels/{id}/messages

  A várt adat:

{

content: string

attachmentCount: number

}

Egy új üzenetet küld az {id} azonosítójú csatornába.

- DELETE /api/channels/{id}/messages/{id}

  Törli a megadott üzenetet a csatornából. A csatolmányok fájljai is törlésre kerülnek, ha vannak.

- PUT /api/channels/{id}/messages/{id}/attachments

  A várt adat a fájlt tartalma. Feltölti a fájlt és hozzácsatolja a megadott üzenethez.

- HEAD /api/channels/{id}/messages/{id}/attachments/{id}

  Lekéri a csatolmány típusát és méretét.

- GET /api/channels/{id}/messages/{id}/attachments/{id}

  A visszaküldött adat a fájlt tartalma. Lekérdezi a csatolt fájl tartalmát.

3.2.1.5 Meghívókat érintő API

- GET /api/invitations

  A visszaküldött adat:

  {

  id: string

  userId: string

  targetId: string

  usages: number

  }[]

Lekéri az összes meghívót a {targetId} azonosítójú csatornához vagy csatorna-csomaghoz. Nincs külön meghívó típus csatornáknak vagy csomagoknak, mert az azonosítók globálisan egyedi.

- POST /api/invitations

  A várt adat:

  {

  for: string

  }

  Létrehoz egy új meghívót a {for} azonosítójú csatornához vagy csatorna-csomaghoz.

- DELETE /api/invitations/{id}

  Törli az {id} azonosítójú meghívót

- GET /api/invitations/{id}/use

  Hozzáadja a bejelentkezett felhasználót az {id} azonosítójú csatornához vagy csatorna-csomaghoz.

3.2.1.6 Barátokat érintő API

- GET /api/friends

  A visszaküldött adat:

  {

  id: string

  username: string

  nickname: string

  verified: boolean

  }[]

  Lekéri a bejelentkezett felhasználó összes barátját.

- POST /api/friends/{id}

Elküld egy barát kérelmet az {id} azonosítóval rendelkező felhasználónak. Ha a másik felhasználó már küldött kérelmet, akkor ez azt hitelesíti, így a baráti kapcsolat teljesen kiépül.

- GET /api/friends/{id}

  A visszaküldött adat:

  {

  status: ‘IN’ | ‘OUT’ | ‘FRIEND’ | ‘NONE’

  }

  Lekéri az {id} azonosítójú baráti státuszát.

  - IN: A másik felhasználó kérelmet küldött
  - OUT: A bejelentkezett felhasználó kérelmet küldött
  - FRIEND: Mindkét fél beleegyezett a baráti kapcsolatba
  - NONE: Egyik fél se küldött baráti kérelmet
- DELETE /api/friends/{id}

Törli az {id} azonosítójú felhasználót a bejelentkezett felhasználó barátai közül. Ez a távoli felhasználó részéről is törli a bejelentkezett felhasználót.

- GET /api/friends/{id}/channel

  A visszaküldött adat:

  string

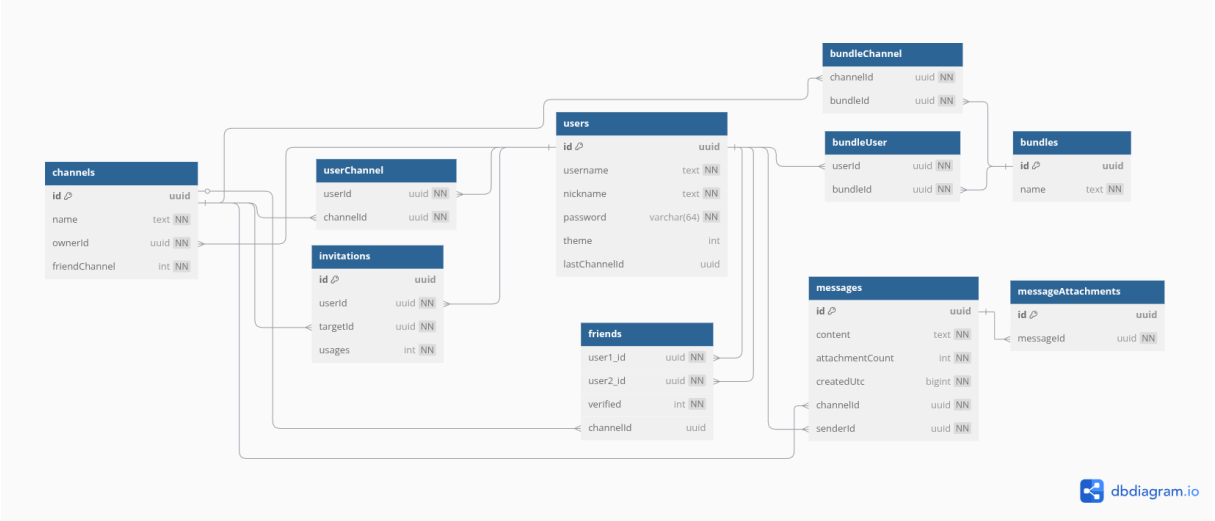
  Lekéri az {id} azonosítójú felhasználó baráti csatornájának azonosítóját.

3.2.2 Adatbázis:

Tárolja a felhasználókat, üzeneteket és a csatornákat. A felhasználók regisztrációja hitelesítve van és az érzékeny adatok titkosított formában kerülnek tárolásra az adatbázisban. Az összes elsődleges kulcs UUID típusú, ezzel kiküszöbölve az “felsorolásos támadásokat” (enumeration attack). Ez megoldható másképp, gyorsabb módszerrel is, miszerint külön tárolunk egy privát kulcsot, illetve egy publikus azonosítót, amelyet az API használ. A jelenlegi megoldás viszont leegyszerűsítette a backend működését, ezzel elkerülve további nemkívánatos sebezhetőségeket, amelyeket a privát-publikus kulcs kezelése okozna.

A backend képes működni mind MySQL illetve SQLite adatbázissal. Az adatbázis interfész lehetővé teszi a különböző protokollokkal való kompatibilitást, illetve a duplikált forráskódokat is redukálja. Az alkalmazás le van tesztelve a XAMPP MySQL, MariaDB, SQLite, illetve in-memory SQLite adatbázis technológiákkal.

Az adatbázis hátteret környezeti változók segítségével lehet meghatározni.



3.2.2.1 “users” tábla

A “users” tábla tartalmazza az összes felhasználó adatait, beállításait és egyéb kisegítő értékeit. A “username” mező tartalmazza a hitelesítéshez szükséges nevet. Ez az azonosító nem változtatható meg, illetve egyedinek kell lennie. A személyre szabhatóság érdekében létezik a “nickname” mező, amely a látható felhasználónevet tartalmazza. A felhasználó bármikor megváltoztathatja ezt, nem kell egyedinek lenni. A “password” mező tárolja az SHA256 titkosított, base64 kódolt jelszót. A titkosítás miatt az esetleges adatszivárgás esetén a felhasználók nem veszítik el a fiókjához való hozzáférést. A “theme” mező tárolja a felhasználó által beállított téma azonosítóját. Azért van szükség a szerver oldali tárolásra, mert így a kliens egyből a helyes stílus definíciókat kapja meg, egy további fetch-et elkerülve. Az utolsó mező, a “lastChannelId” tartalmazza a felhasználó által utoljára látogatott csatorna azonosítóját. Így különböző eszközökön is ugyan ott folytathatja a csevegést, ahol azt abbahagyta, ezzel is növelve a kényelmes és görgülékeny beszélgetést.

3.2.2.2 “channels” tábla

Ez a tábla tárolja az összes csatornát, amelyeken keresztül csevegést lehet folytatni. A “name” mező tartalmazza a csatorna nevét, amely a front-enden jelenik meg. Ezen kívül tárolásra

kerül a csatornát létrehozó felhasználó azonosítója az “ownerId” mezőben, a későbbi visszakeresés érdekében. Továbbá a “friendChannel” mező értéke határozza meg, hogy az adott csatorna automatikusan jött e létre két barát között. Ha igen, akkor a csatorna nem kerül listázásra sem a front-enden, sem az API-ban. Ezeket a csatornákat a barátok listán keresztül lehet elérni, de a későbbiekben lesz leírva.

3.2.2.3 “userChannel” kapcsolótábla

Ez a tábla kapcsolja össze a “users” és a “channels” táblát. Ez lehetővé teszi, hogy egy felhasználó több csatorna tagja is legyen, illetve hogy egy csatornában több felhasználó is részt tudjon venni. A “userId” mező tartalmazza az adott felhasználó azonosítóját, illetve a “channelId” az adott csatorna azonosítóját.

3.2.2.4 “bundles” tábla

Ez a tábla tartalmazza az összes csatorna-csomag adatait. Mivel a csomagok kapcsolótáblák helyes használata által működnek, ebben a táblában elegendő csak a csomag nevét eltárolni a “name” mezőben.

3.2.2.5 “bundleChannel” kapcsolótábla

Ez a tábla hozza létre a kapcsolatot a “bundles” és a “channels” tábla között. Ez által egy csatorna több csomaghoz is tartozhat, és egy csomag több csatornából is állhat. A “channelId” mező tartalmazza az adott csatorna azonosítóját, illetve a “bundleId” mező az adott csomag azonosítóját.

3.2.2.6 “bundleUser” kapcsolótábla

Ez a tábla kapcsolja össze a “users” és a “bundles” táblát. Így egy felhasználó több csatorna-csomag részese is lehet, és egy csomaghoz több felhasználó is csatlakozhat. A “userId” mező tartalmazza az adott felhasználó azonosítóját, illetve a “bundleId” az adott csomag azonosítóját.

3.2.2.7 “invitations” tábla

Ez a tábla tartalmazza az összes meghívó adatait és statisztikáit. Az egyszerűség érdekében nincs külön meghívó tábla a csatornáknak és a csomagoknak. A “channelId” tartalmazza az adott csatorna vagy éppen csomag azonosítóját. Mivel az azonosítók globálisa egyediek, lehetőség van tesztelni, hogy az adott azonosító éppen csatorna vagy csatorna-csomag. A

“usages” mező egy számlálót tartalmazza, amely számolja, hogy eddig hányszor használta egy felhasználó az adott meghívót.

3.2.2.8 “friends” tábla

A “friends” tábla tartalmazza a felhasználók közötti baráti kapcsolatokat. A “user1_id” és a “user2_id” mezők tartalmazzák a két felhasználó azonosítóját, amely között a kapcsolat létezik. A “verified” mező jelzi, hogy a kapcsolat mindkét fél felől hitelesítve van-e. A két barát között egy csatorna automatikusan létrejön, amelynek az azonosítója a “channelId” mezőben kerül tárolásra.

3.2.2.9 “messages” tábla

A legnagyobb tábla a “messages”, amely az összes üzenetet tárolja amelyeket a felhasználók küldtek különböző csatornákban. A “content” mező tárolja el az üzenet szöveges tartalmát. Az üzenet küldési időpontját a “createdUtc” mező tartalmazza. Ez az időpont az UTC+0 időzónában van, így elkerülve az esetleges, időzónák által keltett hibákat és kellemetlenségeket. A “channelId” mező tartalmazza a csatorna azonosítóját, amiben az üzenet elküldésre került. Az üzenet írójának azonosítója a “senderId” mezőben van. Ezen kívül tárolásra kerül az üzenet mellékleteinek a számát az “attachmentCount” mezőben. Ez által előre lehet látni az esetleges adatsérülést, illetve a front-end is fel tud készülni a csatolmányok betöltésére.

3.2.2.10 “messageAttachments” tábla

Ebben a mezőben az adott csatolmány azonosítóját kivéve csak a fájlhoz tartozó üzenet azonosítója van eltárolva a “messageId” mezőben. A fájl tartalma a szerver fájlrendszerében van eltárolva, ahol a fájl neve a csatolmány azonosítója, illetve a fájl kiterjesztése a csatolmány típusa.

3.2.3 Valós idejű kapcsolat:

A valós idejű kapcsolat megvalósításához backend a kliens felé WebSocket kapcsolattal kommunikál. Erre a megoldásra azért van szükség, hogy a backend is tudjon kezdeményezni üzenetküldést a felhasználó felé. Akkor amikor a backend próbál közölni adatot a felhasználóval HTTP kérések nem használhatóak mivel a felhasználó nem küld kérést amire a backend válaszolni tudna. Erre a problémára nyújt megoldást a WebSocket. A WebSocket egy bidirectional kapcsolatot hoz létre, mi viszont csak a szervertől a felhasználó felé való üzenetküldésre használjuk. A felhasználótól szerver felé történő üzenetküldés pedig HTTP

kéréseken keresztül valósítottuk meg. Ezzel nagyrészt unidirectional WebSocket kapcsolatot létrehozva.

*(COMMUNICATION FLOW CHART HERE)

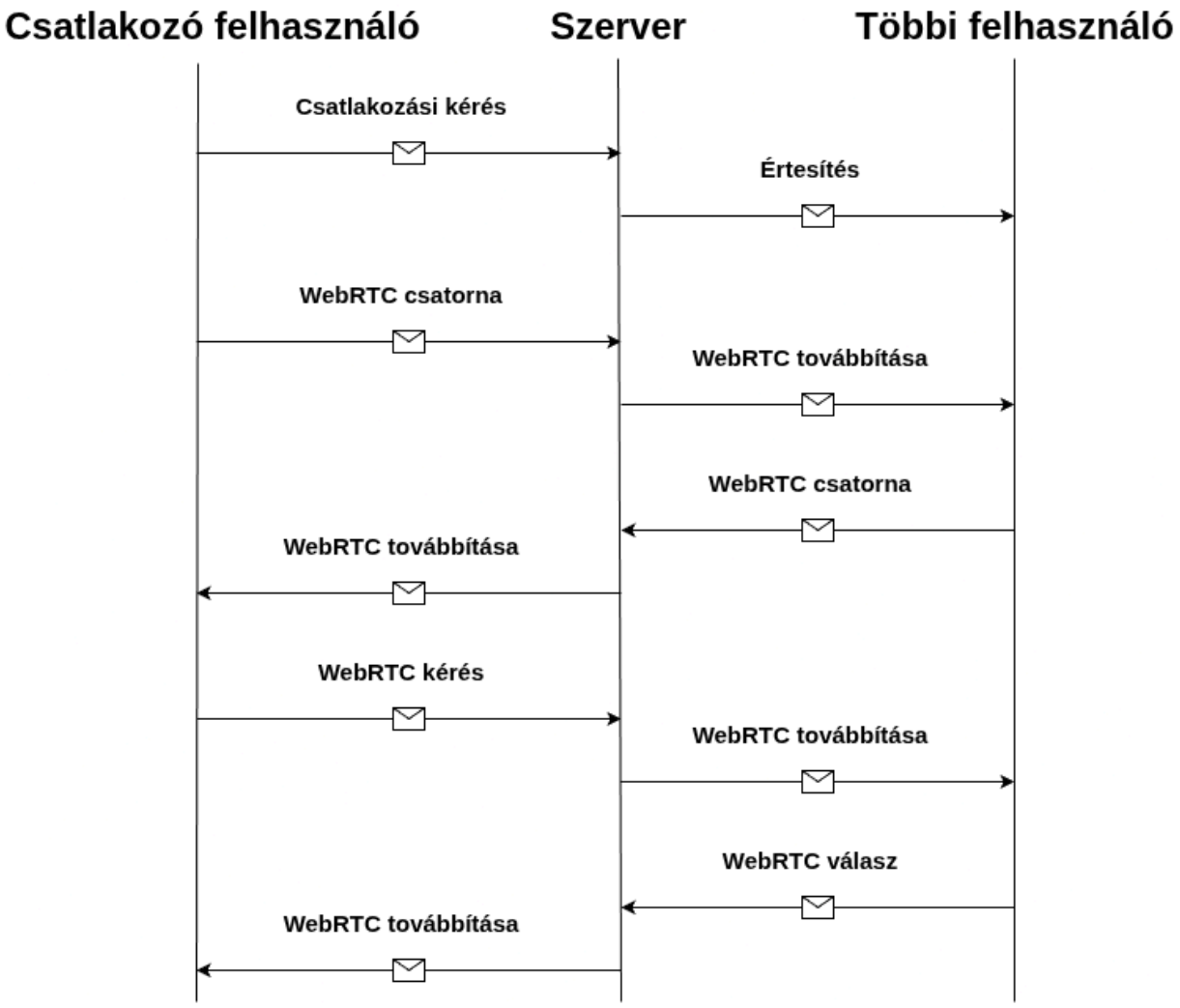
3.2.4 Szinkronizáció:

Minden alkalommal amikor hitelesített kliens csatlakozik vagy újra csatlakozik a backendre szinkronizációnak kell történjen az adatbázissal, a legfontosabb adatok amelyek szinkronizációra szorulnak azok a felhasználó eddigi beszélgetései és ismerősei. Ez a folyamat azzal kezdődik meg, hogy a kliens küld egy HTTP kérést a backend felé amely tartalmazza a kérelmet a felhasználó ismerőseinek lekérésére és/vagy eddigi üzenetek lekérésére emelett tartalmazza a felhasználó hitelesítéshez szükséges információkat, ezzel biztonságban tartva a felhasználó adatait, megakadályozva illetéktelenek hozzáférését a felhasználó érzékeny adataihoz.

Ha kliens és a szerver között megszakad a kapcsolat hálózati probléma miatt a kliens oldal időszerűen megpróbál újra összekapcsolódni a backendel. Abban az esetben ha az újrakapcsolódás sokadik alkalomra se sikeres időtúllépési(timeout) hiba lép fel.

3.2.5 Valós idejű, több végpont közötti videó vagy hanghívás

A felhasználók videó vagy hanghívást kezdeményezhetnek egy csatornán belül. A hívás WebRTC teknológiát használ, így a maximális teljesítményt biztosítja. Egy hívás kezdeményezésekor a felhasználó egy kérést küld a szervernek, amely létrehoz egy virtuális csatornát csak az adott hívásnak. Csatlakozáskor a szerver értesíti az összes felhasználót, akik a virtuális csatornának a részese (vagyis már a hívásban vannak) az új felhasználót. Ezt követően a már csatlakozott felhasználók egy WebRTC csatornát létesítenek, majd annak a csatlakozáshoz szükséges adatait visszaküldi a szervernek, amely továbbítja azt a csatlakozni kívánó felhasználónak, amely így képes csatlakozni WebRTC-n keresztül.



3.2.6 Hitelesítés

Amint a felhasználó belép a fiókjába és a hitelesítés sikeres a szerver elküldi a felhasználói tokent amivel a felhasználó hozzáférhet a funkciókhoz. A token-nek van érvényességi ideje ezért a felhasználási idő alatt a szerver új tokeneket hoz létre a felhasználónak amit a saját tárhelyén tárol kommunikáció során a szerver mindig ellenőrzi a token érvényességét és csak akkor engedi a műveletet végrehajtani ha sikeres a validálás, ezzel megakadályozva, hogy felhasználói fiók nélkül lehessen elérni a funkciókat mint például üzenetet küldeni vagy megtekinteni. A token-ek a szerver oldalon nincsenek eltárolva, helyette a kliens oldalon sütik által vannak beállítva. Ha a felhasználó kijelentkezi, de a token érvényességi ideje még tart, a szerver oldalon ez belekerül egy fekete listába, amelyet a hitelesítés során ellenőriz. Ha egy kérés érkezik, és a token hamarosan lejár, egy új tokent generál, így megakadályozva, hogy például fél óránként be kelljen jelentkeznie a felhasználónak.

STEP-BY-STEP AUTH.

3.2.7 Front-end:

A frontend a dinamikus frissítésre Handlebars-t használ. A választás azért esett erre mivel sokkal gyorsabb a felhasználó szemszögéből mint a versenytársai

4. Felhasználói dokumentáció

1. Regisztráció

A regisztrációhoz kattintson a “Don't have an account?” linkre, amely továbbviszi a regisztrációs oldalra. Másik lehetőség a “/register” elérési utat beírni a címsorba.

Ezen az oldalon be kell írnia egy azonosítót, melyet később a bejelentkezéshez fog használni. Ez az azonosító lesz az alapértelmezett felhasználóneve, amelyet az azonosítóval ellentéttel bármikor megváltoztathatja majd. Ha ezzel megvan, egy jelszót kell begépelnie a további mezőkbe. A regisztrációhoz kattintson a “Register” gombra.

2. Bejelentkezés

A bejelentkezéshez navigáljon a kezdőoldalra, vagy írja be a “/login” elérési utat a címsorba. Itt be kell gépelnie a regisztrációkor megadott azonosítót és jelszót. Ügyeljen arra, hogy ne a felhasználónevét írja be.

3. Felhasználónév és profilkép beállítása

A felhasználónév és profilkép beállításához kattintson bal felül a profilja linkjére. Ez továbbviszi a fiók beállításai oldalára. Itt megtekintheti a jelenlegi felhasználónevét és profilképét is. A felhasználónév megváltoztatásához írja át a jelenlegit a “Nickname” szövegdobozban. A profilkép megváltoztatásához kattintson az “Upload Avatar” gombra, majd válassza ki a beállítani kívánt fájlt. A véglegesítéshez kattintson a “Save” gombra.

4. Barát felvétel

Egy baráti kapcsolat két felhasználó beleegyezésével jön létre. Egyrészről el kell küldeni egy barát kérelmet, majd a másik felhasználónak el kell fogadnia azt.

Barát hozzáadására két lehetőség van, attól függően hogy van-e közös csatornája a felhasználóval.

Van közös csatorna

Ha van közös csatorna, navigáljon abba a csatornába, amelybe mindketten benne vannak. Ez után a jobb oldali felhasználók listában keresse ki a keresett felhasználót, majd kattintson a nevére. Így egy felugró ablak jelenik meg, amelyen a “Send friend request” gombbal tud egy barát kérelmet küldeni.

Nincs közös csatorna

Ha nincs közös csatorna, tudnia kell a másik felhasználó felhasználónevét. Kattintson a bal oldalon lévő “Friends” gombra. Ez a barátok oldalra navigálja. A “Search Users” szekcióban tud felhasználókat keresni. Ha a keresett felhasználó nem létezik, akkor a beírthoz legközelebb hasonlító felhasználók jelennek meg. A barátkérelem elküldéséhez kattintson az “Add friend”.

5. Barátkérelem elfogadása

A barátkérelmek megtekintéséhez és elfogadásához kattintson a bal oldalt lévő “Friends” gombra. Ez a barátok oldalra navigálja majd. Itt a “Friend requests” szekcióban tekintheti meg a bejövő barátkérelmeket. Az elfogadáshoz kattintson az “Accept” gombra.

6. Üzenetelés

Az üzenetelésre szükség van egy csatornára, amelyen keresztül folyni fog a beszélgetés. Ha egy baráttal kíván csevegni, kattintson a bal oldalt lévő “Friends” gombra. Ez a barátok oldalra navigálja majd. Itt a jobb oldalt lévő barát listában keresheti ki a barátot, akinek üzenetet akar küldeni. Ha megvan, kattintson a barát nevére. Ez átirányít egy csatornába, amelyben mindketten benne vannak.

Ha egy általános csatornába kíván üzenetet küldeni, a bal oldalon lévő csatornák listában keresheti ki az adott csatornát. Ha ez megvan, kattintson a csatorna nevére.

Ha a csatorna egy csatorna-csomagon keresztül érhető el, akkor a csomag nevére való kattintással tekintheti meg a csomagban lévő csatornákat. Majd ezután átirányít a csatorna oldalára.

Az oldal közepén jelennek meg az új üzenetek. Középen lent van a beviteli mező. Az üzenet szöveges tartalmát írja be a szövegdobozba. A küldéshez nyomjon enter-t, vagy kattintson a papírrepülő ikont tartalmazó gombra.

7. Csatornába lépés

   Egy csatornába egy meghívó segítségével tud belépni. Egy meghívó egy link, amelyre navigálva belép a csatornába, ezzel használva a meghívót. A link naggyából így néz ki: “http://host/invitations/4647a854-9329-403a-826e-cdf0404f6696/use”.

8. Csatorna létrehozása

   Csatorna létrehozásához kattintson a bal oldali lista legalján lévő “New Channel” gombra. Ez megjelenít egy felugró ablakot, ahol a csatorna nevét tudja megadni a “Name” beviteli mezőben. Ezt követően a “Create” gombra kattintva hozhatja létre a csatornát.

9. Meghívó létrehozása

Meghívó létrehozásához lépjen az adott csatorna oldalára, majd kattintson a jobb fent lévő “három összekötött pont” ikont tartalmazó gombra. Ez megjelenít egy felugró ablakot, ahol megtekintheti az eddigi meghívókat. Új meghívó létrehozásához kattintson a “Create” gombra. A meghívó linkjét a “Copy to Clipboard” gombra kattintva másolhatja a vágólapra. Ez a funkció csak biztonságos kapcsolaton keresztül működik. Ha ez a helyzet áll fenn, a meghívó linkjét a “négyzetből kijövő nyíl” ikonon keresztül érheti el. Erre az ikonra kattintva átnavigál egy oldalra, ahol a címsorból másolhatja ki a linket, vagy használhatja a böngészője által kínált “link másolása” menüpontot a jobb egérgomb lenyomásával.

Ezt a linket elküldheti egy barát-barát között létrejött csatornán, vagy leírhatja egy papírra és postázhatja a felhasználónak. Az utóbbi csak akkor lehetséges, ha tudja a felhasználó levelezési címét.

5. Összegzés

Úgy gondoljuk a webalkalmazás sikeresen elérte a célját mivel a projekt megkezdésekor kitűzött célokat teljesítette.

6. Irodalom és hivatkozásjegyzék

- Hivatkozások:
  - https://dbdiagram.io/
  - https://app.diagrams.net/
- További információ:
  - https://nodejs.org/en
  - https://webrtc.org/
  - https://handlebarsjs.com/
  - https://github.com/panva/jose
  - https://expressjs.com/