Békéscsabai SZC Nemes Tihamér Technikum és Kollégium

Szakképesítés megnevezése: Szoftverfejlesztő és -tesztelő

Azonosító száma: 5 0613 12 03

**VIZSGAREMEK**

Message-App

Készítették:

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

Békéscsaba, 2024/2025

# 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. Csoportok aminek több felhasználó is tagja tud lenni és csevegést folytatni. Megjelenési szempontból pedig a személyre szabhatóság.

Miért ezt választottuk?

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

# Indoklás

A mai világban szinte mindenkinek kulcsfontosságú az online kapcsolattartás. Lehet ez kapcsolattartás ismerősökkel, szakmai kommunikáció vagy új kapcsolatok szerzése az online térben. A Message-App alkalmazás ezt az igényt hivatott kiszolgálni.

# Fejlesztői dokumentáció

* 1. Követelmények:

A Message-App 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. Kinézeti személyre szabhatóságot nyújtani a felhasználónak.
4. Gyors és könnyen kezelhető felhasználói felület(UI).
   1. Rendszerterv:
      1. Backend:

Backend framework-nek a Node.js-t választottuk. Azért esett erre a választás mivel nagyon elterjedt a használata szoftver rendszerek körében. Emellet rengeteg modul áll rendelkezésre az npm csomagkezelőn keresztűl ezzel segítve a fejlesztési folyamatot.

Az API úgy épül fel, hogy a felhasználó által kezdeményezett esemény minden esetben egy HTTP kéréssel valósuljon meg. Ez a következő pontban magyarázódik meg.

\*(API DOCS HERE)

* + 1. Adatbázis:

Az adatbázis SQLite-ot használ. Az SQLite lehetővé teszi, hogy az adatbázis az alkalmazás backendjén fusson, ezáltal növelve az átláthatóságot és csökkentve a komplexitást. Viszont az alkalmazásban megvalósított absztraciónak köszönhetően könnyedén át migrálható MySQL-ra. Tárolja a felhasználókat, üzeneteket és a csatornákat. A felhasználók regisztrációja hitelesítve van és az érzékeny adatok titkosított formában kerülnek tárolásra az adatbázisban.

\*(DB ER DIAGRAM HERE)

* + 1. 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)

* + 1. 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.

\*(MAYBE ANOTHER FLOWCHART)

* + 1. Hitelesítés:

Amint a felhasználó belép a fiókjába és a hitelesítés sikeres a szerver elküldi a felhasználói tokent amivel a felhasználó hozzáférhet a limitált 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 siker a validálás ezzel megakadályozva, hogy felhasználói fiók nélkül lehessen elérni a funkciókat mint pl. üzenet küldés.

STEP-BY-STEP AUTH.

* + 1. Frontend:

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

# Felhasználói dokumentáció

1. Regisztráció
2. Bejelentkezés
3. Profilkép beállítása
4. Barát felvétel
5. Üzenetelés
6. Csoportba lépés
7. Csoport létrehozása

# Összegzés

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

# Irodalom és hivatkozásjegyzék