

BGSzC Pestszentlőrinci Technikum

1184 Budapest Hengersor 34.



# Záró dolgozat PlanUP

Konzulens tanár:

Szekrényes Gábor

Készítette:

Molnár Levente,  
Nagy Martin,  
Kovács-Major Márton

## Tartalom

1	Bevezetés .....	3
1.1	Feladat leírás .....	3
1.2	A felhasznált ismeretek .....	3
1.3	A felhasznált szoftverek.....	3
2	Felhasználói dokumentáció.....	5
2.1	A program általános specifikációja.....	5
2.2	Rendszerkövetelmények .....	6
2.2.1	Hardver követelmények.....	6
2.2.2	Szoftver követelmények .....	6
2.3	A program telepítése .....	7
2.4	A program használatának a részletes leírása .....	10
3	Fejlesztői dokumentáció .....	16
3.1	Az alkalmazott fejlesztői eszközök .....	16
3.2	Adatmodell leírása.....	17
3.3	Részletes feladat-specifikáció, algoritmusok.....	20
3.4	Tesztelési dokumentáció .....	22
4	Összefoglalás.....	26
4.1	Önértékelés .....	26
4.2	Továbbfejlesztési lehetőségek.....	27
5	Felhasznált irodalom .....	28
6	Ábrajegyzék.....	30

# 1 Bevezetés

## 1.1 Feladat leírás

Célunk a programmal, hogy egy olyan alkalmazást készítsünk, amely segítséget nyújt a felhasználóknak szabadidő programajánlásokban és szűrésekben, miközben egy felhasználóbarát felületen akár többedmagával is válogathatnak a lehetőségek közül.

A dokumentáció egyszerre nyújt segítséget a program telepítéséhez és használatához, és betekintést nyújt a készítésébe és tesztelési folyamatairól.

A munkafolyamat során minden eddig szerzett ismeretünket felhasználtuk, és még újakat is szereztünk.

## 1.2 A felhasznált ismeretek

Tanulmányaink során rengeteg ismeretet szereztünk amelyek hasznosnak bizonyultak a projekt készítése során. Mindenekelőtt legfontosabb volt a csapatmunka, és a feladatok hatékony felosztása egymás között. A csapatunknak voltak erősségei és gyengeségei is, melyeket figyelembe véve próbáltunk meg egy minél kiegyensúlyozottabb munkamegosztást teremteni egymás között.

Az évek során rengeteget tanultunk az adatbáziskezelésről, webszerkesztésről, keretrendszerekről és a GitHub használatáról. Ezen tanulmányok a mestermunkán is viziontláthatók.

Az iskolai tanulmányokon kívül is szert tettünk új ismeretekre, melyekre a mestermunkánk bizonyos részei miatt volt szükség. Ezek a későbbiekben részletezve lesznek.

## 1.3 A felhasznált szoftverek

A projekt során különböző programokat használtunk, volt olyan, ami feltétlen szükséges volt a koncepció megvalósításához, és volt, ami tanórai kötelezettséggel került használatra, és bizonyos szempontokból hasznosnak bizonyultak.

**Trello**: A csapatmunkához rendkívül hasznos internetes alkalmazás, mellyel nyomon tudtuk követni a haladásunkat egymás között, és szét tudtuk válogatni az ötleteket, fejlesztés alatt lévő komponenseket, és a már elkészült részeket.

**ChatGPT**: A ChatGPT-vel rengeteg problémát oldottunk meg a fejlesztés során, nagyon sok esetben tudott nekünk segíteni, a kódunkat ellenőrizni és javítani hibás kódok során. Az alkalmazásban szereplő képeket a mesterséges intelligencia készítette az esetlegesen felmerülő szerzői jogi problémák elkerülése végett.

**Visual Studio Code**: A mestermunka során a legtöbbet használt alkalmazás, ezen belül készült el a program. A bővítményekről és egyéb specifikációkról a program telepítésénél <sup>(3)</sup> lesz még szó.

**MySQL Workbench**: Az adatbázis kezeléséhez használtuk ezt az alkalmazást, itt el tudtuk készíteni az egyed-kapcsolat diagramokat, és az adatbázist szerkeszteni a Visual Studio-s alkalmazásunkkal. Legtöbbször ezen keresztül ellenőriztük az adatbevitelt és megjelenítést az oldalon.

**XAMPP**: Ez az alkalmazás biztosította a MySQL kapcsolatot. Bizonyos esetekben nélkülözhető, ez a telepítésnél <sup>(3,2)</sup> részletezve lesz.

**Github**: A mestermunka egyik központi eleme, amin keresztül történtek a fejlesztések hónapokon keresztül, különböző számítógépeken otthon és közoktatásban egyaránt. A verziókövető rendszernek köszönhetően bármikor vissza tudunk váltani előző verziókra, és nyomon tudtuk követni egymás munkáját, és hozzáférni a fájlokhoz bárhol, bármikor, bármilyen eszközről.

**Teams**: A közoktatásban is elterjedt alkalmazást használtuk a kommunikációra és bizonyos adatok tárolására annak chat felületén, illetve itt tartottuk a kapcsolatot és töltöttük fel a haladásunkat szaktanárainknak és konzulensünknek.

**Postman**: A backend részen használtuk az API-k tesztelésére és a Frontend-el való kapcsolat ellenőrzésére.

## 2 Felhasználói dokumentáció

### 2.1 A program általános specifikációja

A PlanUP lehetőséget biztosít arra, hogy a felhasználók közösen válogassanak különböző hétköznapi programok közül. Az alkalmazás leírásunk a következő: „Fedezze fel egyedül vagy akár társaságban Budapest és más városok programjait, válogasson kedvére, vagy hasonlítsa össze véleményét társaival, hogy közösen találjanak egy olyan programot, amely mindenki számára megfelel.”

Projektünkben lehetőség nyílik arra, hogy a felhasználók egy profil létrehozása után válogassanak több tucatnyi program közül, amiket később visszatekinthetnek egy összegző oldalon. Ezen az oldalon lehetőséget nyújtunk még arra, hogy a kiválasztott programhoz közvetlen elérhetőséget nyújtsunk. Mi a tökéletes összekötő kapocs vagyunk a felhasználók és a szabadidős programok között.

#### Profil létrehozása

Regisztráció szükséges az oldal használatához, hogy az egyéni vagy csoportos interakciókat elmenthesse a rendszer. A jelszavakat biztonságos hash kódolás védi, az adatbázisból nem lehet kinyerni azt.

#### LIKE/DISLIKE Rendszer

Egyszerű döntés elé kerülnek a felhasználók, vagy kedvelik az adott programot, vagy nem, és a program megjegyzi és tárolja az interakciót.

#### Szobakódok

Hozzon létre saját szobát magának és társainak, ahol egymással párhuzamosan tudnak válogatni a programok közül, amelyeket a rendszer később összevet, hogy a legoptimálisabb programokat jelenítse meg.

#### Több mint 50 program

Válogasson több mint 50 program közül akár ingyenes, fizetős, fix időpontú vagy rugalmasabb időbeosztás szerint.

#### Admin felület

Felhasználói profilokhoz lehet privilegizált ID-t rendelni az adatbázisban, amivel megjelenik a profil felületen az admin felület. Itt lehetőség nyílik új programok

hozzáadására, meglévő programok törlésére, felhasználói profilok és létrehozott szobák megtekintésére.

### *Bizza ránk a tervezést!*

A PlanUP nem csak programajánlatokat tesz fel önnek, hanem megkönnyíti még azoknak a megvalósítását is közvetlen foglalásokkal, megadott elérhetőségekkel vagy akár ajánlott tervekkel is, így önnek nincs más dolga, mint megtalálnia az ideális programot.

## **2.2 Rendszerkövetelmények**

### **2.2.1 Hardver követelmények**

Projektünket Windows 10 operációs rendszeren fejlesztettük.

Hardver követelmények:

- Minimum: 4 GB RAM, Intel i3 processzor
- Ajánlott: 8 GB RAM, Intel i5 vagy erősebb processzor

### **2.2.2 Szoftver követelmények**

Operációs Rendszer: A program a következő operációs rendszereken fut:

- Windows 10 vagy újabb
- macOS High Sierra vagy újabb
- Linux Ubuntu 20.04 vagy újabb

Szoftver Komponensek:

- .NET Framework 4.8 vagy újabb
- DirectX 12 vagy újabb (grafikus alkalmazásokhoz)
- MySQL adatbázis-szerver (verzió: 8.0 vagy újabb)

Egyéb:

- Webböngésző: A PlanUp egy webalkalmazás, ezért szükséges hozzá egy webböngésző. A fejlesztés során főleg a Microsoft Edge böngészőt használtuk, ezt is javasoljuk, de más böngészők sem okozhatnak problémát.
- Internetkapcsolat: A PlanUp egy webalkalmazás, ezért szükséges hozzá állandó internetkapcsolat.

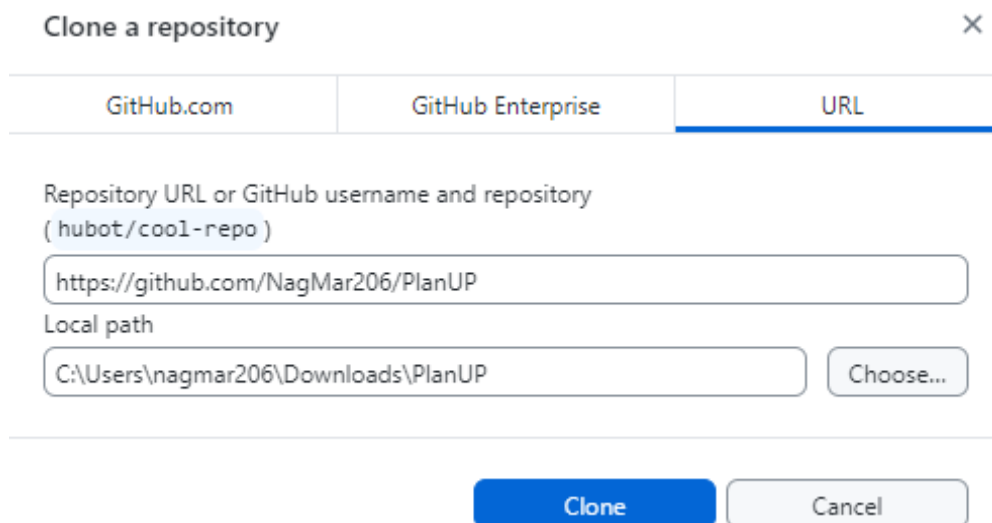
## 2.3 A program telepítése

A program telepítéséhez GitHub Desktopra, MySQL Workbench-re, XAMPP-ra és Visual Studio Code-ra van szükség. Mindenekelőtt ellenőrizzük, hogy minden szükséges alkalmazásból a legfrissebb verzió áll rendelkezésünkre.

### 1. lépés: A Program beszerzése

A Github elérhetősége a PlanUP-nak a következő linken található:

[NagMar206/PlanUP: Mestermunka](https://github.com/NagMar206/PlanUP)



Clone a repository

GitHub.com    GitHub Enterprise    URL

Repository URL or GitHub username and repository  
( hubot/cool-repo )

Local path

Itt csak annyi a dolgunk, hogy klónozzuk le a repository-ból a programot.

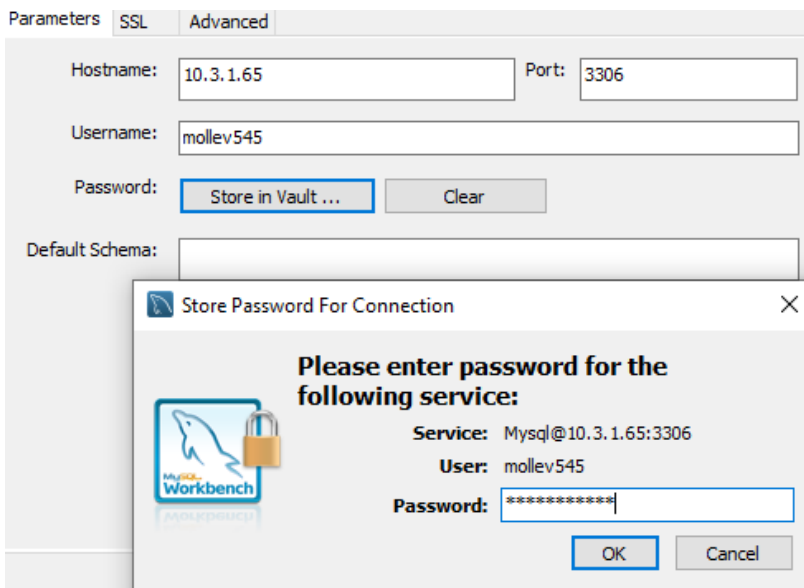
### 2. lépés: Az adatbázis telepítése

A PlanUP kétféle módszert is alkalmazhat az adatbázis kezelésére. A fejlesztés során lehetőségünk volt az iskolai szervernek a használata (ez megkönnyítette a munkafolyamatot). A program backend részében két adatbáziselérés is él, amiből az egyik, amit aktuálisan nem használunk mindig ki van kommentelve. Abban az esetben, ha az iskolai szerver a vizsga során is elérhető, akkor az iskolai szerver adatbázisa is rendelkezésre áll, és nem kell telepíteni semmit. A backend program részben ebben az esetben így kell kinéznie az adatbázis kapcsolatnak:

```
JS dbConfig.js X
planup-backend > config > JS dbConfig.js > ...
1  const mysql = require('mysql2/promise');
2  /*
3  //PlanUP Adatbázis
4  const db = mysql.createPool({
5    host: 'localhost', // XAMPP esetén ez maradjon "localhost"
6    user: 'root', // XAMPP alapértelmezett felhasználó
7    password: '', // XAMPP esetén nincs jelszó, hagyj üresen
8    database: 'planup', // Az adatbázis neve (ellenőrizd phpMyAdminban!)
9    port: 3307, // Ha a MySQL más porton fut, módosítsd
10   waitForConnections: true,
11   connectionLimit: 10,
12   queueLimit: 0
13 });
14 */
15 //Iskolai szerver
16 const db = mysql.createPool({
17   host: '10.3.1.65', // XAMPP esetén ez maradjon "localhost"
18   user: 'mollev545', // XAMPP alapértelmezett felhasználó
19   password: '72576822545', // XAMPP esetén nincs jelszó, hagyj üresen
20   database: 'mollev545', // Az adatbázis neve (ellenőrizd phpMyAdminban!)
21   port: 3306, // Ha a MySQL más porton fut, módosítsd
22   waitForConnections: true,
23   connectionLimit: 10,
24   queueLimit: 0
25 });
26
27 module.exports = db;
28
```

## MySQL Connections

Először is adjunk hozzá egy új kapcsolatot!



A következő adatok beírása szükséges a MySQL-ben:

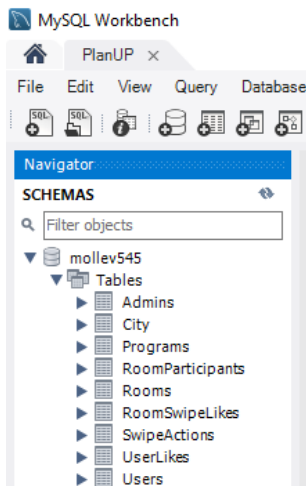
Hostname: 10.3.1.65

Port: 3306

Username: mollev545

Password (Store in Vault): 72576822545



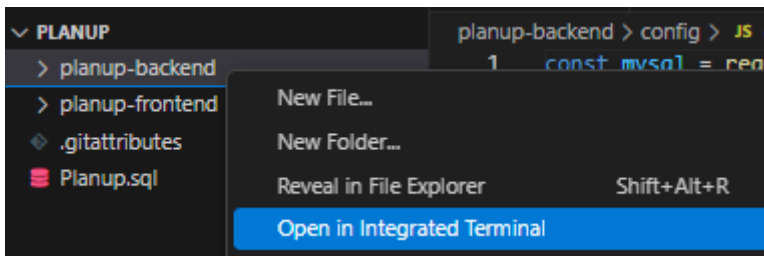


Ha kész, ezt a sémát kell látnunk.

Ha az iskolai szerver nem áll rendelkezésre, akkor a GitHub repository-ban biztosított Planup.sql fájl kell hozzáadni a kapcsolatokhoz. Ehhez már szükségünk lesz az XAMPP alkalmazásra is, amiben aktiválnunk kell az SQL kapcsolatot a 3307-es porton. Utána mindössze az “Open SQL Script” gombra kell kattintani (balról a második) aminél hozzáadjuk az Planup.sql fájlt. Miután megnyitottuk a beimportált adatbázist, nyomjuk meg a villám ikont a mentés ikon mellett, és az adatbázist hozzá is adtuk ezzel a sémákhoz. Így most már működni fog a PlanUP adatbázis, és kikommentelhetjük az iskolai szerver részt. Fontos, hogy mindig ki kell kommentelni az egyiket, mert nem lehet egyszerre kettő adatbázis kapcsolat kód.

### 3. lépés: Frontend és Backend elindítása

Jelöljük ki az egész leklónozott mappát, és nyissuk meg Visual Studio Code-ban. Ezután a “planup-backend” és “planup-frontend” mappákat nyissuk meg terminálban.

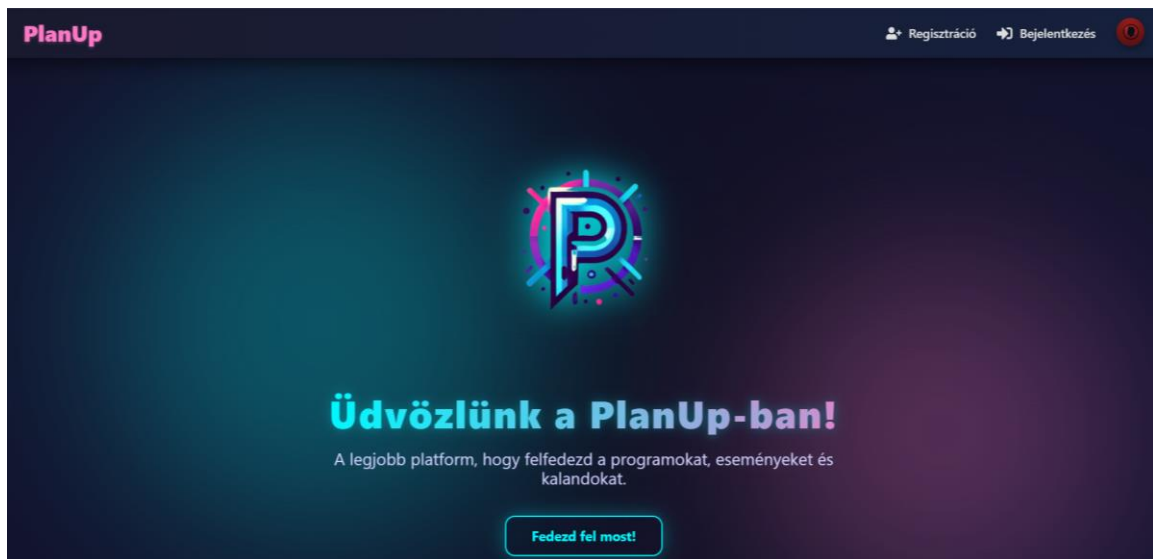


Először a backend-el érdemes kezdeni, de ugyanezen lépéseket a frontend termináljában is el kell végezni. Az “npm start” parancs használatával indíthatjuk el a részeket. Ha a frontend-et is elindítjuk a backend után, akkor maga az oldal is megnyílik, és az oldal már használatra kész.

## 2.4 A program használatának a részletes leírása

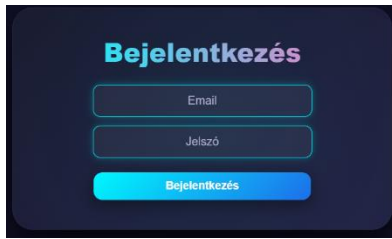
### *Főoldal* /

Megnyításkor a főoldalon találjuk magunkat, ahol üdvözlő az alkalmazás, és a menüpontok mellett a Fedezd fel most! gombbal lehet elindítani a programot. A menüsorból még elérhető a *bejelentkezés* és *regisztrációs* gomb. Ha már van felhasználói fiókunk, akkor a gombra kattintva az oldal átvizs a bejelentkező felületre, viszont mindenekelőtt szükséges a regisztráció az oldalra.



### *Regisztrációs felület* /register

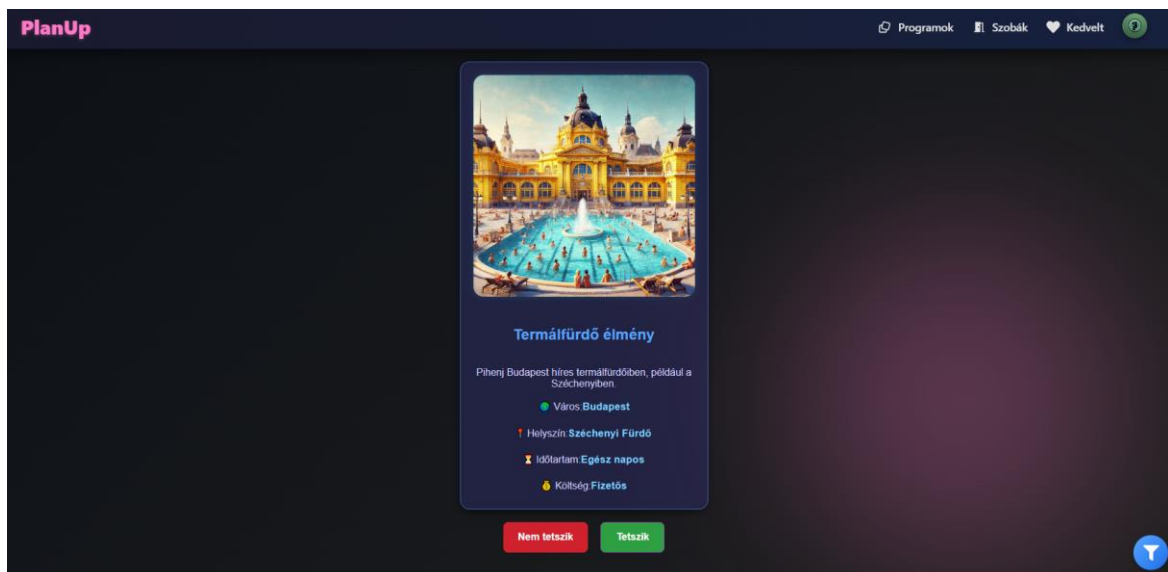
A regisztrációs oldalon lehetősége van az új felhasználóknak egy profilt létrehozniuk. Ennek a módja rendkívül leegyszerűsített, mindössze egy email, felhasználónév és jelszó megadása szükséges ehhez. Ha megtörtént a regisztráció, az oldal felviszi az adatokat az adatbázisba, és ezután a megadott adatokkal már be is léphetünk újonnan létrehozott fiókunkba!

**Bejelentkezési felület****/login**

Az imént regisztrált, vagy már azelőtt létrehozott profil adataival lesz elérhető a bejelentkezés. Csak írjuk be az e-mail címet és a jelszót, és a gomb megnyomása után az oldal kiírja, ha sikeres volt a bejelentkezés, és átirányít a válogató felületre. A felső menüsorban bejelentkezés után a profil ikon zölddé változik, ezzel indikálva a belépett felhasználó állapotot.

**Programpörgető felület****/swipe**

Megérkeztünk a fő attrakcióhoz! A PlanUP itt dobja fel a program ajánlásait a felhasználónak, kezdve egy véletlenszerű programmal. Elsőként menjünk végig a szűrő feltételeken:



Város: A PlanUP lehetőséget nyújt arra, hogy akár pontos helyszíneket megjelölve keressünk programokat, ugyanakkor nem garantált, hogy az adott helyszínre kidob programokat.

Időtartamok: A PlanUP által összegyűjtött programok sokszínűek és mértékűek, éppen ezért lehetőséget nyújtunk arra, hogy preferenciáid alapján válogathass olyan programok közül, amelyek különböző hosszúságúak.

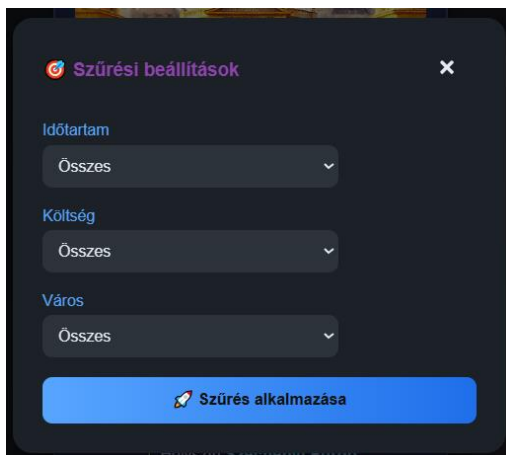
A választható időtartamok között szerepel például az "Egész hétvégés" opció, ami nem

feltétlen azt jelenti, hogy a program egy egész hétvégét vesz igénybe, inkább mint csak egy indikátorként működik, hogy a program egész hétvégén elérhető, bármikor tervez is menni a felhasználó.

A *félnapos* programok közé tartoznak olyan lehetőségek, amik nem vesznek igénybe egy teljes napot, ilyen például egy vacsora, egy hajókirándulás vagy egy operaelőadás.

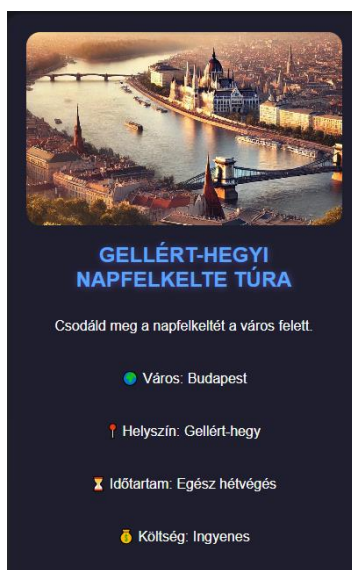
Az *egésznapos* programok, mint például egy termálfürdő élmény vagy egy kirándulás akár egy teljes napot felöllelhetnek, éppen ezért akkor érdemes ezeket az opciókat szűrni, amikor tudjuk, hogy akár a teljes nap is a program időtöltésével telhet.

**Költségopciók:** A PlanUP által összegyűjtött programok különbözhetnek még költség alapján is, rengeteg ingyenes programmal is szolgálunk, amikhez nincs szükség belépőjegy vagy részvételi díj vételére. Természetesen lehetőséget nyújtunk nagyobb költséggel járó programok böngészésére is.

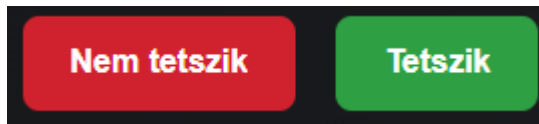


A szűrők használatához meg kell nyomni a szűrőikont, ami behozza a szűrési beállításokat. Itt lehet az előbb említett paramétereket megadni, majd a szűrő alkalmazása gombbal alkalmazni azokat.

*A programkártya leírása:*



A legelső dolog, ami szembe tűnhet a kártyán az az illusztráció, ami a programhoz készült. Ezek a mestermunkánkban mesterséges intelligencia által generált képek az esetlegesen felmerülő szerzői jogi problémák elkerülése végett. A képet követi a program cíMLEÍRÁSA, az alatt pedig a rövid, egymondatos összefoglalója. Végezetül a program helyszíne, időtartama és költsége is felsorolt, amiből az utóbbi kettő a fentebb említett lehetőségek alapján szűrhető a felhasználó által.

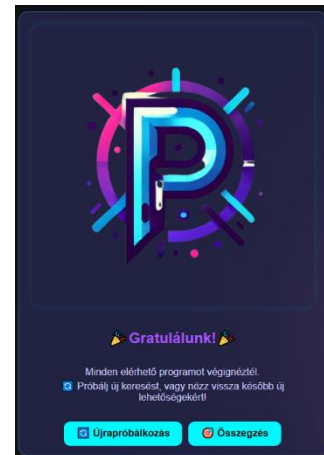


*Tetszik/Nem*

*Tetszik:*

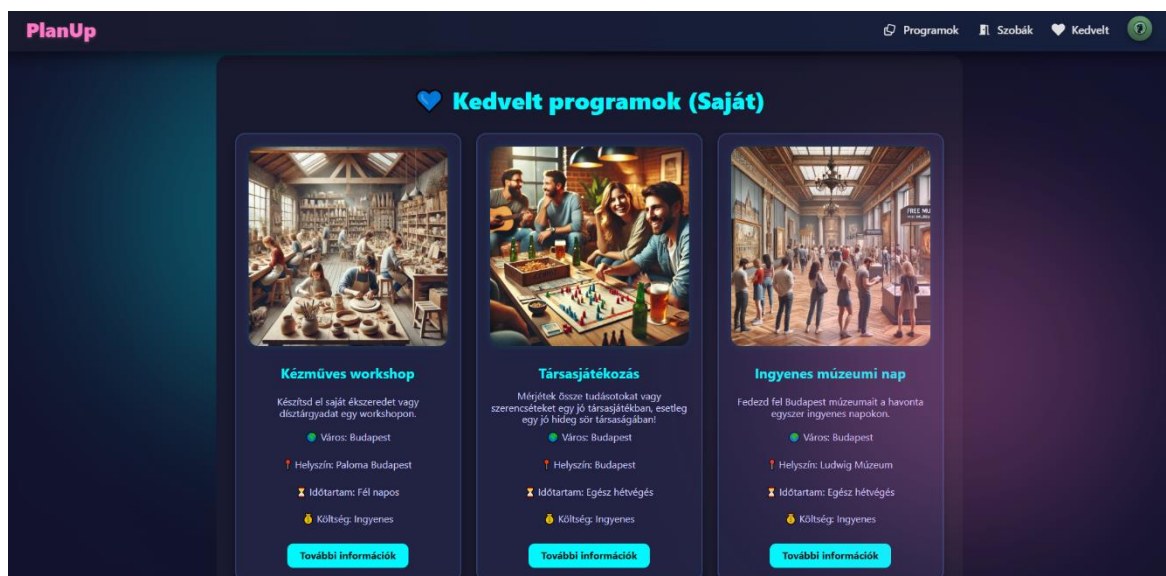
A

programkártya alatt megjelenő "Nem tetszik" és "Tetszik" gombbal dönthetünk arról, hogy a kártyán megjelent program érdekli-e a felhasználót, vagy sem. Ezt az adatot elmenti az adatbázis. Miután elfogytak a programok, megjelenik egy "Összegzés megtekintése" gomb, amivel meg lehet tekinteni a kedvelt programokat.



### **Kedvelt programok összegzése**     */liked-programs / /summary*

Helyzettől függetlenül dob tovább a program a következő oldalakra: A program a „liked-programs” részre irányít át, ha a felhasználó egyedül válogatott a programok közül. A program szobakód használata esetén a „summary” oldalra dob át minden felhasználót, akinek a szobakódja egyezik.

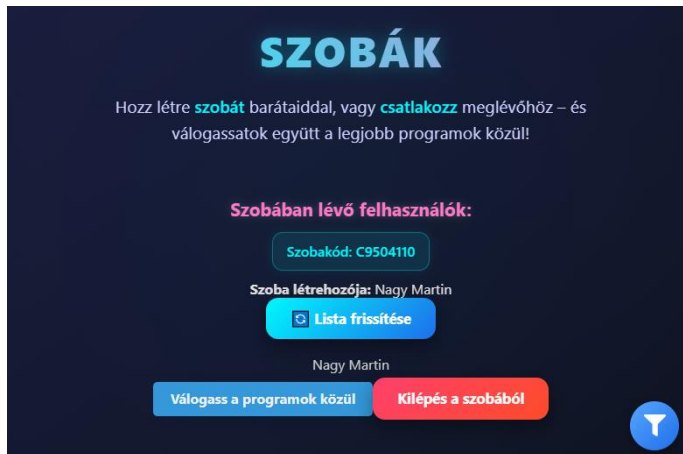


Az oldal feldob minden programot, ami kedvelve volt bármelyik felhasználó által. Minden kártya ugyanúgy jelenik meg, mint a válogatásnál annyi különbséggel, hogy a kártya alján láthatjuk, hogy a szobából hányan kedvelték az adott programot. (Ha a felhasználó egyedül pörget, csak az ő preferált programjai fognak megjelenni az oldalon).

Ezután lehetőség nyílik a választásra, kétféle módon. Ha a felhasználók a kedvelések alapján döntésre jutnak, akkor az általuk választott programról további

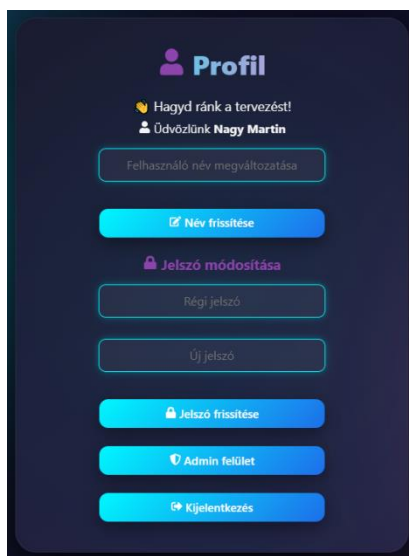
információkat kaphatnak, ha rákattintanak a kártyára. Alternatívaként, ha még a lájkok száma ellenére sem tudnak döntésre jutni - vagy döntetlen helyzet áll fent két vagy több program között - akkor az oldal alján lehetőség van egy véletlenszerű program választására. Ezen a válogatáson az összes program megjelenik, ami kedvelve volt. Itt lehetőség van pörgetni egyet, és a program véletlenszerű alapon kidob egy programot.

### *Szobák felület /rooms*



A szobák oldalon van lehetőség összekapcsolódni más felhasználókkal szobakódok alapján. Új szobát létrehozni is van lehetőségünk, és a kapott kódot kimásolva mások is becsatlakozhatnak a szobába. A felület kiírja az összes felhasználó nevét, aki a szobában tartózkodik, legfelül jelenik meg a szoba

létrehozójának a neve. A szoba létrehozójának még lehetősége van szűrőket beállítani a szobára, viszont erre csak neki van joga. Miután a létrehozó elindítja a válogatást, minden felhasználó átkerül a RoomSwipe oldalra. Az egyéni válogatás és a szobás válogatás külön van kezelve.



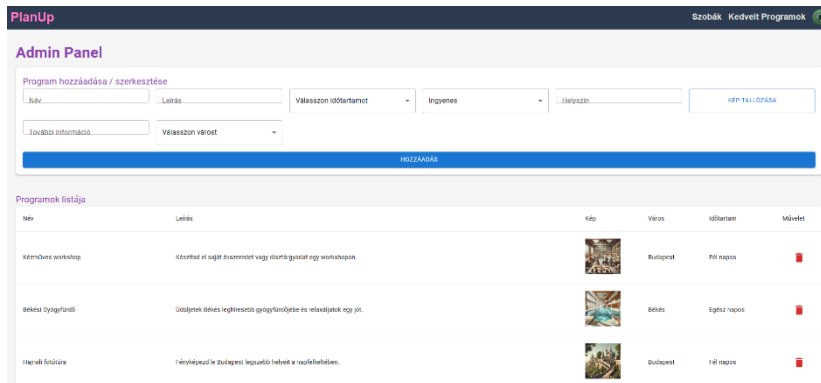
### *Felhasználói profil /profile*

A profil oldalon lehetősége van a felhasználónak megváltoztatni a nevét, jelszavát, illetve kijelentkezhet az oldalról. Privilegizált profilok esetén itt jelenik meg a gomb az admin felület eléréséhez, ez a gomb általános felhasználók számára nem jelenik meg.



### Admin felület/admin

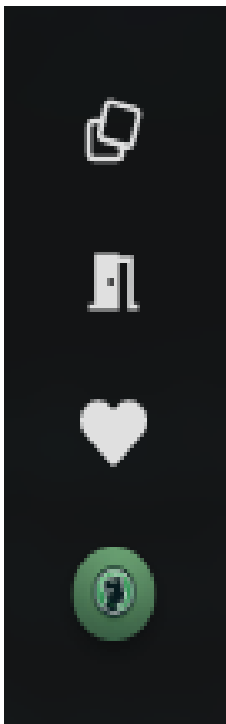
Ez a felület fejlesztői szempontból releváns, átlagos felhasználók számára nem elérhető a PlanUP ezen része. Jelenleg 3 admin jogosultsággal rendelkező profil létezik, ezek mind a PlanUP fejlesztőihez tartoznak. Az admin felület egy felhasználóbarát környezetet biztosít az új programok feltöltéséhez, frissítéséhez vagy törléséhez, ugyanígy a felhasználók profiljának kezeléséhez is. Ha arra kerülne a sor, hogy a programunkat



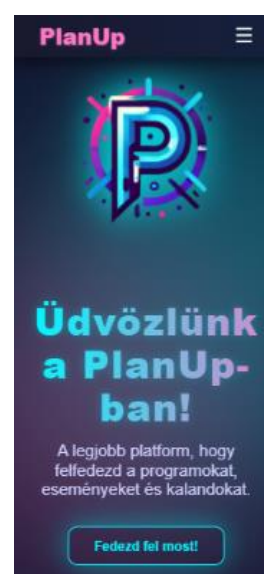
eladnánk, az admin felülettel biztosíthatjuk azt is, hogy a következő tulajdonosok mélyebb programozói tudás nélkül is tudják kezelni a PlanUP dinamikus elemeit.

### Reszponzív verziók mobilon

A fejlesztés során ügyeltünk arra, hogy az oldal teljesen reszponzív legyen, és a mobilos felhasználói élmény is értékelhető legyen.



A főoldalon, és ezzel együtt minden oldalon a Navbar a hamburgermenü megoldást kapta a kisebb felbontással rendelkező kijelzők megjelenítésénél. A három ikon (felülről) a Programok, Szobák, Kedvelt Programok és a Profil felülethez vezetnek.



## 3 Fejlesztői dokumentáció

Ebben a részben a program logikáját, kódjának megértését és annak továbbfejlesztését részletezzük.

### 3.1 Az alkalmazott fejlesztői eszközök

A PlanUP alkalmazás fejlesztéséhez és dokumentációjának elkészítéséhez az alábbi fejlesztői eszközöket használtam:

- **Programozási nyelvek:**
  - **JavaScript (JS):** A frontend (React) és a backend (Node.js) fejlesztéséhez használt elsődleges nyelv.)
  - **JSX:** A React komponensek szerkezeti és vizuális felépítésének megvalósításához.
  - **CSS / Tailwind CSS:** A felhasználói felület modern és reszponzív stílusának kialakításához.
  - **SQL:** Az adatok strukturált lekérdezéséhez és módosításához a MySQL adatbázisban.
  - **JSON:** Adatok cseréjéhez a frontend és a backend között (pl. API válaszok formátumaként).
- **Fejlesztői környezet:**
  - **Visual Studio Code:** A projekt teljes körű fejlesztéséhez használt kódszerkesztő, kiegészítve különféle bővítményekkel (pl. Prettier, ESLint, React Snippets.).
  - **Vite:** A React frontend fejlesztésének gyors buildelését és fejlesztői kiszolgálását biztosító eszköz.
  - **Node.js:** A backend kiszolgáló logika és API végpontok futtatásához használt környezet.
- **Adatbázis-kezelő rendszer:**
  - **MySQL (XAMPP):** A felhasználók, programok, szobák és kapcsolódó adatok tárolására szolgáló relációs adatbázis-kezelő, a 3307-es porton konfigurálva.
  - **MySQL Workbench:** Az adatbázisok vizuális modellezéséhez, lekérdezések írásához, teszteléshez és karbantartáshoz.
- **Verziókezelő rendszer:**
  - Git és GitHub (verziókövetés, projektmenedzsment és közös elérés)
- **Modulok, csomagok, könyvtárak:**
  - React (<https://react.dev/>) – webes felhasználói felület kialakítására, MIT licenc alapján
  - Express.js (<https://expressjs.com/>) – backend API-kezeléshez, MIT licenc alapján
  - Axios (<https://axios-http.com/>) – HTTP kérések kezelésére, MIT licenc alapján
  - JSON Web Token (JWT) (<https://jwt.io/>) – autentikációhoz, MIT licenc alapján
  - Cookie-parser (<https://github.com/expressjs/cookie-parser>) – HTTP süti kezeléséhez, MIT licenc alapján



- **Képszerkesztő program:**
  - Adobe Photoshop CC (képek szerkesztése, grafikai elemek előkészítése)
- **Dokumentációkészítő eszközök:**
  - Microsoft Word (szöveges dokumentáció összeállítása)
  - Microsoft PowerPoint (bemutatók, prezentációk készítése)

A felhasznált külső modulok mindegyike szabadon felhasználható, nyílt forráskódú licenc alatt érhető el (MIT licenc). Ezek felhasználása és integrálása megfelel a licencfeltételeknek.

## 3.2 Adatmodell leírása

A PlanUP adatbázisunk a következő táblákat tartalmazza oldalanként:

**Admins tábla:** Ebben a táblában csak az admin privilégium ID-jét tároljuk. Akihez hozzárendeljük ezt az ID-t, annak elérhetővé válik az admin felület.

- id (int): Az adminisztrátor egyedi azonosítója, elsődleges kulcs.
- username (varchar): Az adminisztrátor felhasználóneve, egyedi.
- password (varchar): Az adminisztrátor jelszava (hash formátumban).

**City tábla:** Az elérhető városokat tartalmazza, amelyek szűrési feltételként szolgálnak a programok keresésénél. A CityID-t hívja meg a program, és a Name-t írja ki az oldalra.

- CityID (int): A város egyedi azonosítója, elsődleges kulcs.
- Name (varchar): A város neve.
- Programs: Programok adatai.

**Programs tábla:** Az összes elérhető programot tárolja, amelyből a felhasználók válogathatnak. A ProgramID alapján menti az interakciókat, a Price, Cost, Duration szűrőknek van.

- ProgramID (int): A program egyedi azonosítója, elsődleges kulcs.
- Name (varchar): A program neve.
- Description (text): A program leírása.
- Price (decimal): A program ára.
- Cost (tinyint): A program költsége (1 -> van, 0 -> nincs).
- Duration (tinyint): A program időtartama.
- CreatedAt (timestamp): A program létrehozásának időpontja.
- Location (varchar): A program helyszíne.
- Image (varchar): A programhoz kapcsolódó kép.
- MoreInfoLink (varchar): További információk linkje.
- CityID (int): Hozzárendelt város az idegen kulcs.

**RoomParticipants tábla:** Kapcsolótábla a felhasználók és a szobák között. Nyilvántartja, ki melyik szobában van benne. A szobában jelenlévő felhasználók mentése ID-k alapján. Ez ahhoz szükséges, hogy a program tudja, hogy a felhasználók egy szobában vannak.

- ParticipantID (int): A résztvevő egyedi azonosítója, elsődleges kulcs.
- RoomID (int): A szoba azonosítója (idegen kulcs).
- UserID (int): A felhasználó azonosítója (idegen kulcs).
- isReady (tinyint): A résztvevő állapota (készen áll-e).

**Rooms tábla:** Ez a tábla kezeli az úgynevezett „szobákat”, ahol több felhasználó közösen tud programokat válogatni. A RoomID jelenik meg például az admin felületnél, amivel azonosítja a szobákat. A RoomCode a szobakód amivel tudnak csatlakozni a felhasználók, a CreatedByUserID a host, a szobának a létrehozójának a UserID-je, így jelenítjük meg csak neki a filtereket.

- RoomID (int): A szoba egyedi azonosítója, elsődleges kulcs.
- RoomCode (varchar): A szoba kódja, egyedi.
- CreatedByUserID (int): A szoba létrehozó felhasználó azonosítója (idegen kulcs).
- CreatedAt (timestamp): A szoba létrehozásának időpontja.
- Filters (json): Szűrők megadása.
- IsStarted (tinyint): A szoba állapota (elkezdődött-e).

**SwipeActions:** Ezen táblákba mentjük a felhasználó interakciókat, az Actions mező menti az interakciót, ami alapján eldönti a program, hogy megjelenítse-e az összegzésnél, vagy sem.

- SwipeID (int): A kedvelés azonosítója, elsődleges kulcs.
- UserID (int): A felhasználó azonosítója (idegen kulcs).
- ProgramID (int): A program azonosítója (idegen kulcs).
- Action (enum): A végrehajtott művelet (kedvelés / nem kedvelés).
- Timestamp (timestamp): Az akció időpontja.

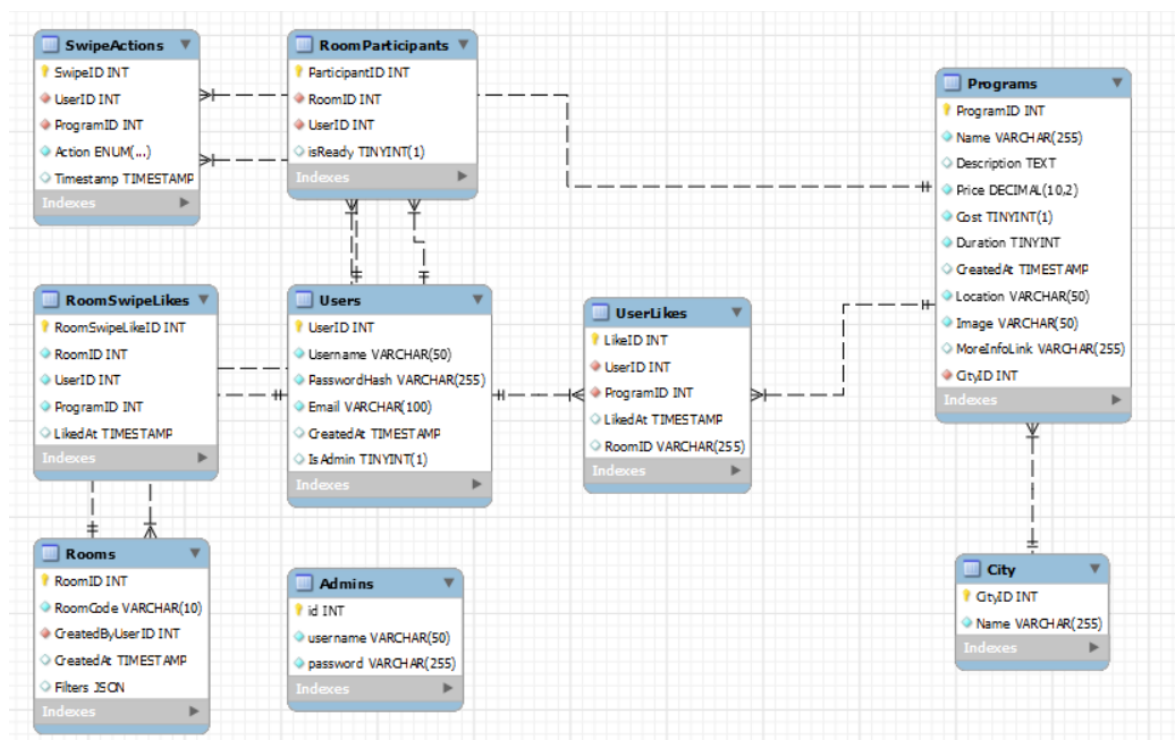
**UserLikes:** A program ide menti a lájkolt programok ID-jét, UserId-t ProgramID-t és RoomID-t egyaránt.

- LikeID (int): A kedvelés azonosítója, elsődleges kulcs.
- UserID (int): A felhasználó azonosítója (idegen kulcs).

- ProgramID (int): A program azonosítója (idegen kulcs).
- LikedAt (timestamp): A kedvelés időpontja.
- RoomID (varchar): Melyik szobához tartozik a kedvelés.

**Users:** Ide mentjük a felhasználókat ID-vel, a felhasználónevük, jelszavuk (hashelve) emailjük és admin jogosultságuk itt kerül tárolásra.

- UserID (int): A felhasználó egyedi azonosítója, elsődleges kulcs.
- Username (varchar): A felhasználóneve.
- PasswordHash (varchar): A felhasználó jelszó hash formátumban.
- Email (varchar): A felhasználó email címe.
- CreatedAt (timestamp): A felhasználó létrehozásának időpontja.
- IsAdmin (tinyint): Admin státusz (1 -> admin, 0 -> nem admin).



### 3.3 Részletes feladat-specifikáció, algoritmusok

#### Véletlenszerű Program Válogató (Swipe feature)

```

javascript
app.get("/programs/random", async (req, res) => {
  try {
    const { userId } = req.query;
    if (!userId) {
      console.error("Hiányzó userId paraméter!");
      return res.status(400).json({ error: "Hiányzó userId paraméter." });
    }

    let likedPrograms = [];
    const [likedProgramsRows] = await db.execute(
      "SELECT ProgramID FROM UserLikes WHERE UserID = ?",
      [userId]
    );
    likedPrograms = likedProgramsRows.map(p => p.ProgramID);

    let sqlQuery, queryParams;
    if (likedPrograms.length > 0) {
      sqlQuery = "SELECT * FROM Programs WHERE ProgramID NOT IN (?) ORDER BY RAND() LIMIT 1";
      queryParams = [likedPrograms];
    } else {
      sqlQuery = "SELECT * FROM Programs ORDER BY RAND() LIMIT 1";
      queryParams = [];
    }

    const [randomProgramRows] = await db.execute(sqlQuery, queryParams);
    const randomProgram = randomProgramRows.length > 0 ? randomProgramRows[0] : null;

    if (!randomProgram) {
      console.log("Nincs több elérhető program.");
      return res.json(null);
    }

    console.log("Visszaküldött program:", randomProgram);
    res.json(randomProgram);
  } catch (error) {
    console.error("Általános hiba történt a random program lekérésekor:", error);
    res.status(500).json({ error: "Szerverhiba a program betöltésekor.", details: error.message });
  }
});

```

Működés:

- A funkció véletlenszerűen kiválaszt egy programot az adatbázisból.
- Ha a felhasználó már kedvelt bizonyos programokat (*UserLikes* tábla), ezeket kizárja a keresésből.
- Az SQL lekérdezés véletlenszerű sorrendben választ egy programot.
- Ha nincs több elérhető program, üres választ ad vissza.

```

const handleEndSwipe = () => {
  if (roomId) {
    console.log(`Szobás válogatás vége, átirányítás a Summary oldalra. RoomID: ${roomId}`);
    navigate(`/summary?room=${roomId}`);
  } else {
    console.log("Egyéni válogatás vége, átirányítás a LikedPrograms oldalra.");
    navigate(`/liked-programs`);
  }
};

```

A válogatás vége a ProgramSwipe.js-ben a következő módon van kezelve:

- Ha a

szobaválogatásnak vége van (nincs több program) és van szobaID (tehát többen válogatnak) akkor kiírás után átvizet egy Summary oldalra a szoba ID-jével ellátva.

- Ha nincs szobaID, akkor az egyéni válogatásnak vége, és átirányít a LikedPrograms oldalra az egyéni felhasználói kedvelések megtekintéséhez.

## LikedPrograms Lekérés (Fetch):

```
javascript
const fetchLikedPrograms = async () => {
  try {
    const endpoint = `${apiUrl}/programs/liked?userId=${validUserId}`; // Egyéni like-ok lekérése
    const response = await axios.get(endpoint, { withCredentials: true });
    setLikedPrograms(response.data);
  } catch (err) {
    console.error("Hiba a kedvelt programok lekérésekor:", err);
    setError("Nem sikerült betölteni a kedvelt programokat.");
  }
};
```

Működés:

- Feladata: Lekéri az adott felhasználó által kedvelt programokat az API-ból.
- API hívás: Az

útvonal `/programs/liked` paraméterként kapja a felhasználói azonosítót (`userId`).

- Állapotfrissítés:
  - A válaszban kapott adatokat beállítja a `likedPrograms` állapotba.
  - Ha hiba történik, beállít egy hibaüzenetet az `error` állapotba.

## Admin felület megjelenése a Profile oldalon:

```
javascript
useEffect(() => {
  if (!user) return;
  axios.get(`http://localhost:3001/profile/${user}`, { withCredentials: true })
    .then((response) => {
      setUsername(response.data.username);
      setIsAdmin(response.data.isAdmin || false); // Ellenőrizzük, hogy
    })
    .catch((error) => {
      console.error("Hiba a profil lekérésekor:", error);
      setLoading(false);
    });
}, [user]);

// Ha a felhasználó admin, jelenjen meg az Admin felület gomb
{isAdmin && (
  <button className="admin-button" onClick={navigateToAdmin}>
    Admin Felület
  </button>
)}
```

### 1. Felhasználói adatok lekérése:

- A `useEffect` hook minden alkalommal lefut, amikor a `user` állapot változik.
- Az API hívás (`GET /profile/:user`) lekéri a felhasználó profilját, amely tartalmazza az `isAdmin` attribútumot.

- Az `isAdmin` értékét beállítja az állapotba (`setIsAdmin`).

### 2. Admin jogosultság ellenőrzése:

- Ha az API válaszban az `isAdmin` attribútum igaz (`true`), akkor a felhasználó admin.
- Az admin státusz alapján feltételesen megjelenik egy gomb (`<button>`), amely az admin felületre navigál.

Ez a logika biztosítja, hogy csak az admin jogosultságú felhasználók férjenek hozzá az adminisztrációs funkciókhoz. Ez kulcsfontosságú a rendszer biztonsága és szerepkör-alapú hozzáférés-kezelése szempontjából. Szimplán URL beírásával sem érhető el az admin felület az átlag felhasználók számára.

```
javascript
useEffect(() => {
  axios.get('http://localhost:3001/api/auth/status', { withCredentials: true })
    .then(res => {
      if (res.data.isAdmin) {
        setAuthorized(true);
      } else {
        alert('Nincs jogosultságod az admin felülethez!');
        window.location.href = '/';
      }
    })
    .catch(() => {
      alert('Nem vagy bejelentkezve!');
      window.location.href = '/';
    })
    .finally(() => setLoading(false));
}, []);
```

Az AdminPanel kódjában ez a kód felelős azért, hogy ellenőrizze a felhasználó jogosultságát a megjelenítéshez. Ha a felhasználó admin, engedélyezi a hozzáférést, ha nem, akkor figyelmeztetés után visszairányítja a főoldalra a felhasználót.

### 3.4 Tesztelési dokumentáció

#### Alkalmazott módszertan:

- **Fekete doboz tesztelés** – a program viselkedését vizsgáljuk bemeneti adatok és válaszüzenetek alapján, a belső működés ismerete nélkül.
- **Fehér doboz tesztelés** – a kód logikai útvonalainak lefedésére, például adatbázisműveletek tesztelése közvetlen végpontokon keresztül.

#### *Tesztelési Szituáció #1*

##### Normál használat – Program like-olása szobán belül

- **Teszt típusa:** Fekete doboz
- **Lépések:**
  1. Felhasználó belép egy meglévő szobába (/rooms)
  2. Rákattint a „Válogass a programok közül” gombra
  3. Egy programra kattint a „Tetszik” gombbal
- **Várt válasz:** 200 OK, Program sikeresen like-olva.

- **Siker esetén teendő:** program bekerül a RoomSwipeLikes és UserLikes táblába
- **Hiba esetén:** ha már like-olta → 400 Bad Request, üzenet: "A program már like-olva van."
  - **Teendő:** új program kérése fetchFilteredProgram() függvénnyel automatikusan megtörténik

### *Tesztelési Szituáció #2*

#### **Extrém eset – Üres jelszóval történő jelszóváltoztatás**

- **Teszt típusa:** Fekete doboz
- **Lépések:**
  1. *Bejelentkezett felhasználó üresen hagyja a régi vagy az új jelszó mezőt (/profile)*
  2. *Rákattint a „Jelszó frissítése” gombra*
- **Várt válasz:** hibaüzenet jelenik meg: "Mindkét mező kitöltése kötelező!"
- **Teendő:** felhasználónak ki kell töltenie a mezőket → újrapróbálkozás
- **Hiba lehetőség:** ha backendhez eljut a kérés, de null értékkel → adatbázis hibát dobhat → backend validáció szükséges

### *Tesztelési Szituáció #3*

#### **Hibakezelés – Nem létező szobakód megadása**

- **Teszt típusa:** Fekete doboz
- **Lépések:**
  1. *A felhasználó egy nem létező szobakódot ír be (/rooms)*
  2. *Rákattint a „Csatlakozás” gombra*
- **Várt válasz:** hibaüzenet jelenik meg: "Nem sikerült csatlakozni a szobához."
- **Backend válasz:** 404, "A szoba nem található"
- **Teendő:** helyes szobakód újbóli megadása

- *Javaslat: jobb UX érdekében legyen külön „A megadott szobakód érvénytelen” üzenet*

#### *Tesztelési Szituáció #4*

#### **Végpont teszt – Likes funkció**

##### *Teszt célja:*

##### *Ellenőrizni, hogy a rendszer megfelelően:*

- *felismeri, ha a program már like-olva van,*
- *beszúrja az új adatot a SwipeActions és UserLikes táblákba, ha még nincs ott,*
- *megfelelő hibakódot ad vissza sikertelen esetben.*

```
const [existingLike] = await req.db.execute(
  "SELECT * FROM UserLikes WHERE UserID = ? AND ProgramID = ?",
  [userId, programId]
);
if (existingLike.length > 0)
  return res.status(400).json({ error: "A program már like-olva van." });

await req.db.execute(
  "INSERT INTO SwipeActions (UserID, ProgramID, Action) VALUES (?, ?, 'like')",
  [userId, programId]
);

await req.db.execute(
  "INSERT INTO UserLikes (UserID, ProgramID, RoomID) VALUES (?, ?, ?)",
  [userId, programId, roomId]
);
```

##### *Teszteset: Már létező like*

*Előfeltétel: UserLikes táblában már szerepel a (UserID = 5, ProgramID = 10) sor*

*Tesztadat: userId: 5, programId: 10, roomCode: nem számít*

*Várt eredmény: HTTP 400, { error: "A program már like-olva van." }*



***Leírás: Ezzel a teszttel elérjük az existingLike.length > 0 ágot***

***Teszteset: Új like beszúrása***

***Előfeltétel: UserLikes táblában nincs UserID = 6, ProgramID = 15 sor***

***Tesztadat: userId: 6, programId: 15, roomCode: ROOM123***

***Várt eredmény: HTTP 200, { success: true, message: "Program sikeresen like-olva." }***

***Leírás: Lefedi a existingLike.length === 0 ágot + mindkét INSERT műveletet***

***Teszteset: Adatbázishiba (kapcsolat megszakadt)***

***Előfeltétel: Adatbázis leállítva***

***Tesztadat: userId: 5, programId: 10***

***Várt eredmény: HTTP 500, { error: "Szerverhiba a like mentésekor." }***

***Leírás: Eléri a catch ágot – ez egy negatív útvonal lefedése***

## 4 Összefoglalás

### 4.1 Önértékelés

Molnár Levente: Nagyon jó élmény volt a közös munka, még ha eleinte kicsit nehézkesen is haladtunk a kezdeti tudáshiány miatt.

Azonban miután pontosabban megbeszéltük, hogy mit szeretnénk, hogyan nézzen ki és hogyan működjön, valamint utánaolvastunk a megvalósítás lehetőségeinek, már sokkal gördülékenyebben tudtunk haladni.

Az órák során egyre jobban átláttuk a projekt működését, és fokozatosan fejlődött a technikai tudásunk is. Megtanultunk hatékonyabban együttműködni, hibákat keresni és kijavítani, valamint gyorsabban megvalósítani ötleteket.

Összességében nemcsak egy működő funkciót készítettünk el, hanem rengeteget tanultunk a gyakorlatban is – és ez az, ami igazán értékes volt számunkra.

A projekt során fullstack fejlesztőként minden részéhez hozzáadtam valamit – az adatbázis kezeléstől kezdve a backend működésén át egészen a felhasználói felületig. Ugyanakkor leginkább a program alapjait, a frontend felépítését és a dizájn kialakítását készítettem el, így nemcsak technikailag, hanem vizuálisan is meghatároztam az alkalmazás végső megjelenését és működését.

Nagy Martin: Nagyon tetszett számomra hogy egy projekten több hónapon keresztül dolgozhattunk, szívesen láttam volna még ilyet az elmúlt években is. Nagyjából januárban kezdődtek el a munkálatok és ahogy belelendültünk úgy épült fel egyre szebben és jobban az oldal. Az én munkám a legelején még a programok gyűjtése és képek generálása volt, de ezt később leadtam, hogy a projekt swipe funkcióját fejlesszem, az interakciók mentését az adatbázisba, és legvégül a szobarendszert. A swipe funkciót közösen írtuk Leventével, a LikedPrograms és a Rooms részekkel én küzdöttem meg. Eleinte még úgy nézett ki, hogy a ProgramSwipe kezelte a szobából indult programválogatást is, végül ezt különválasztottuk a RoomSwipe-ra, hogy ne legyen összeakadás se a kódban, se a működésben. Lényegesen sok időt töltöttem még a projekt dokumentációjával is. Összességében nagyon jó volt a közös munka, a Github rendszeres használata is hasznos volt, habár voltak érdekes elakadások, de semmi olyan amivel ne tudtunk volna megbirkózni. A PlanUP fejlesztése egy általános témává vált a mindennapokban, és úgy érzem, sikerült valami olyat alkotnunk, amibe megérte ennyi munkát fektetni.

Kovács-Major Márton: A közös munka az elején nehézkesen indult de aztán bele jöttünk mindannyian. A feladatok elosztásába is belejöttünk miután rájöttünk hogy kinek mi az erőssége inkább és ki mihez ért. Könnyebben kezdtünk el kommunikálni és ötletelni hogy hogyan akarjuk megvalósítani a feladatot amit csinálni szeretnénk. Levente és Martin csinálja a nehézemelést mert én nagyon alacsony szintű hozzáértéssel inkább csak a szimplább feladatokban mint a dizájnban való segítség és az elképzelés megötletelésében meg a gyűjtő munka és az adatbázis készítésében tudtam segíteni.

## 4.2 Továbbfejlesztési lehetőségek

A fejlesztés során rengeteg ötlet merült fel, amit bizonyos okok miatt sajnos kénytelenek voltunk elvetni. Voltak nagyraágyó terveink is, amikhez a tananyagban sajnos nem találtunk segítséget, és emiatt nem akartunk felkészületlenül belevágni.

Az egyik ilyen lehetőség például a tényleges érintés alapú 'swipe' funkció. Viszont mivel a PlanUP jelenleg egy webalkalmazás, ezért az elsődleges felhasználói élmény a számítógépes környezet, és a legtöbb gépen nem megoldható az érintőképernyős kezelés. Emiatt jelenleg úgy tartottuk, hogy a tényleges 'swipe' lehetőség egy későbbi prototípusban kapna szerepet.

Szeretnénk még a szobák funkciót továbbfejlesztani, lehetőséget adni a 'host'-nak előre beállítható szűrőket, és hogy azonnal elindítsa mindenki számára a válogatást. Jelen állásban nem tudtuk ezt megoldani, mivel az efféle valós idejű frissítésekkel csak a mestermunkák készítése során találkoztunk.

Szeretnénk még továbbá lehetőséget biztosítani a PlanUP felhasználóknak, hogy visszajelzéseket küldjenek az oldal üzemeltetőinek, ha esetleg egy program valamilyen oknál fogva nem elérhető jelenleg. Természetesen ezt az adminok is tudják maguktól ellenőrizni, illetve az adott programokhoz tartozó weboldalak üzemeltetőitől is függ, hogy mennyire aktuális információkat tesznek elérhetővé.

Az eredeti elképzelésünk egy Budapestre központosított programajánló volt, viszont a fejlesztés során kitekintettünk a nagyobb képre is, és elkezdtünk más városok programjai között is kutatni. Szeretnénk ezt a programbázist növelni a jövőben, hogy minél több lehetőség közül válogathassanak a felhasználók.

A korai fejlesztéseknél ötletként felmerült még egy dinamikus helymeghatározás alapú válogatási lehetőség is, ami a felhasználó lokációját használva ajánlana programokat a közelében. Ez még kibővíthetett volna akár azzal is, hogy a PlanUp egy útvonaltervet is készített volna az adott programhoz.

Az előbbi bekezdésekből is lesűrhető, hogy a PlanUP egy ambíciózus projekt volt részünkről, és nagy terveink vannak vele. Jelenlegi állapotában (amit nevezzünk most 1.0-nak) a program teljesíti az előre felvázolt terveinket.

## 5 Felhasznált irodalom

### React dokumentáció

- **Link:** <https://react.dev>
- **Cím:** React – A JavaScript library for building user interfaces
- **Megtekintés ideje:** 2025.04.16.
- **Használat:** React frontend fejlesztéshez (Vite, komponensek, useState, useEffect, props stb.)

### Express.js dokumentáció

- **Link:** <https://expressjs.com>
- **Cím:** Express – Node.js web application framework
- **Megtekintés ideje:** 2025.04.16.
- **Használat:** Backend API fejlesztés (index.js, route-ok, middleware-ek)

### Socket.IO

- **Link:** <https://socket.io>
- **Cím:** Socket.IO – Bidirectional and low-latency communication
- **Megtekintés ideje:** 2025.04.16.
- **Használat:** Szobákon belüli élő frissítéshez (válogatás elindítása stb.)

### MySQL dokumentáció

- **Link:** <https://www.mysql.com>
- **Cím:** MySQL – The world's most popular open source database
- **Megtekintés ideje:** 2025.04.16.
- **Használat:** Adatbázis-struktúra megtervezéséhez (Users, Rooms, Programs, RoomParticipants, UserLikes stb.)

### Vite dokumentáció

- **Link:** <https://vitejs.dev>
- **Cím:** Vite – Next Generation Frontend Tooling

- **Megtekintés ideje:** 2025.04.16.
- **Használat:** A frontend projekt létrehozásához (npm install vite@latest, Vite + React)

### JWT.io

- **Link:** <https://jwt.io>
- **Cím:** JSON Web Tokens – Introduction
- **Megtekintés ideje:** 2025.04.16.
- **Használat:** Bejelentkezés, autentikáció JWT alapon

### MDN Web Docs – HTTP Cookies

- **Link:** <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>
- **Cím:** HTTP Cookies
- **Megtekintés ideje:** 2025.04.16.
- **Használat:** HttpOnly cookie-k használata a bejelentkezés után

## 6 Ábrajegyzék

1. ábra: Projekt logó .....	1
2. ábra: Github klónozás .....	7
3. ábra: Adatbázisok elérhetőségei .....	8
4. ábra: MySQL Workbench - Kapcsolat létrehozása .....	8
5. ábra: MySQL Workbench - Kapcsolat részletei .....	8
6. ábra: MySQL Workbench - Sémák.....	9
7. ábra: Visual Studio Code - Terminál létrehozása .....	9
8. ábra: PlanUP - Főoldal.....	10
9. ábra: PlanUP - Regisztráció .....	10
10. ábra: PlanUP - Bejelentkezés.....	11
11. ábra: PlanUP - Válogatás .....	11
12. ábra: PlanUP - Szűrők ikon.....	12
13. ábra: PlanUP - Szűrők ablak .....	12
14. ábra: PlanUP - Programkártya .....	12
15. ábra: PlanUP - Válogatás vége kártya .....	13
16. ábra: PlanUP - Interakció gombok.....	13
17. ábra: PlanUP - Kedvelt programok listázása .....	13
18. ábra: PlanUP - Szobák .....	14
<b>19. ábra: PlanUP - Profil.....</b>	<b>14</b>
20. ábra: PlanUP - Admin felület.....	15
21. ábra: PlanUP - Kedvelt programok reszponzív .....	15
22. ábra: PlanUP - Főoldal reszponzív .....	15
23. ábra: PlanUP - Navbar ikonok reszponzív.....	15
24. ábra: PlanUP - Adatbázis EER diagram .....	19
<b>25. ábra: Kód - Véletlenszerű válogatás .....</b>	<b>20</b>
26. ábra: Kód - Válogatás vége.....	20
27. ábra: Kód - Kedvelt programok lekérése .....	21
<b>28. ábra: Kód - Admin felület gomb elérése.....</b>	<b>21</b>
29. ábra: Kód - Admin jogosultságok ellenőrzése .....	22
30. ábra: Kód - Kedvelt program esemény kezelése.....	24