Békéscsabai Szakképzési Centrum Trefort Ágoston Technikum, Szakképző Iskola és Kollégium

Szakképesítés neve: Szoftverfejlesztő

OKJ száma: 5421305

# Szakdolgozat

## PandaGO

Szabó Richárd
témavezető

Győrfi Krisztián
13.A

**Békéscsaba, 2024.**

Tartalom

# A program általános specifikációja

## Főbb jellemzői

A *PandaGO* egy saját úttervező weboldal amelyen a felhasználó tud önmagának és másoknak utat tervezni más országokba.

Más felhasználók által létrehozott utakhoz is lehet csatlakozni azonban az utakhoz kötött programok limitálják az utasok számát. Egyelőre még csak külföldre tervezhetőek az utak.

Az oldal kinézetre szimpla, hogy a felhasználó tájékozódását megegyszerűsítse és tudatában legyen, hogy mit csinál/csinált.

A kinézetét több oldal is inspirálta többek között a WizzAir oldala utak megjelenítése miatt és a regisztrációs és bejelentkezési oldalt pedig a Twitter régebbi kinézete inspirálta.s

## Főbb funkciói

A weboldalnak a legfontosabb funkciói közé tartozik a felhasználó kezelés, hiszen profil nélkül a felhasználó csak megtekinteni tudja a mások által létrehozott utakat.

Az utak megjelenítik, hogy ki szervezte, mennyibe kerül az út, honnan hova és, hogy mikor megy és érkezik a jármű.

Ezen kívül a felhasználó, az oldal többi funkciójához nem fér hozzá profil nélkül, így tehát a bejelentkezés és vagy a regisztrációja után veheti igénybe őket.

Miután létrehozott egy profilt amely csak pár alapadatot igényel, hozzáférhet a többi funkcióhoz, melyek közül az egyik legfontosabb, egy adott úthoz való csatlakozás, az utaknak jelenleg nincsenek megadott utasszámai így akárhányan tudnak csatlakozni az úthoz. Azonban le van kezelve, hogy egy ember többször ne tudjon ugyanarra az útra elmenni spam-et elkerülve.

Másik fontosabb funkciója maga az út létrehozása, az út létrehozásánál a felhasználó kiválaszthatja mikor tervezi indítani az utat, honnan és mivel. Ezzel együtt megtudja adni a tervezett érkezési időt és helyet. Mint mindennek a felhasználó útjainak is van ára amit meg kell adnia, azonban fizetési lehetőség nincsen az oldalon, csak látszatnak van megadva, de abba bele lesz számítva a későbbiekben hozzárendelhető programok ára is. A felhasználó miután létrehozott egy utat, vagy csatlakozott egyhez megtekintheti azokat az utakat a profilján.

A profil oldalon megtekinthető utaknál külön vannak választva a felhasználó által létrehozott és az általa csatlakozott utak. Az utaknak, amelyeket ő hozott létre megtekintheti a tervezett utasait. Amihez viszont csak csatlakozott a felhasználó onnan csak kilépni tud, viszont ha meggondolja magát vissza tud lépni az utazáshoz csak meg kell találnia a kezdőlapon.

# Rendszerkövetelmények

## Hardverkövetelmény

Minimum:

| CPU | RAM | GPU | Tárhely | OS | Internet |
|---|---|---|---|---|---|
| Intel Core i5-2300 2.8GHz vagy AMD FX-6300, 3.5GHz | 2GB | GeForce GTX 460. 1GB vagy Radeon HD 6870. 1GB | 1GB | Win10 vagy Win11 | Szükséges |

Ajánlott:

| CPU | RAM | GPU | Tárhely | OS | Internet |
|---|---|---|---|---|---|
| Intel Core i5-4570 3.2GHz vagy AMD FX-8350 4.2GHz | 4GB | GeForce GTX 660, 2GB vagy Radeon HD 7870, 2GB | 1GB | Win10 vagy Win11 | Szükséges |

## Szoftverkövetelmény

A program működéséhez szükséges szoftverek:

- XAMPP az adatbázis érdekében
- NodeJS a backend-hez szükséges
- A program bármely operációs rendszeren lefut amelyen elérhető a XAMPP

## A program telepítése

A program telepítéséhez három program igazán fontos, hogy hibamentesen indulhasson a program. Közülük a legfontosabb a NodeJS, a NodeJS felel az egész backend működéséhez, nélküle nem működne a program, az adatbázissal nem jöhet létre a kapcsolata és nem fér hozzá az adataihoz.

A NodeJS-t a weboldalukról kell letölteni, ahol a sok verzió közül szinte bármelyik megfelel jelenlegi használatára. Azonban a legfrissebb verzióját ajánlatos használni, hisz az a legbiztosabb. Az oldalon kiválaszthatja, a verzión kívül, hogy melyik operációs rendszeren szeretné használni.

A NodeJS telepítése közben nem kell semmi opción változtatni csak végig tovább amíg el nem kezdi a telepítést.

A következő fontos program a XAMPP a XAMPP felel az adatbázisért, és az adatbázis nélkül pedig nem működhetne a program. A XAMPP-ot szintén a saját oldalukról kell telepíteni. A XAMPP verziótól függetlenül végzi a dolgát, de mint a NodeJS esetében itt is ajánlatos a legfrissebb verziót telepíteni.Szerencsére a főoldalukon meg is található a legfrissebb verzió. Ahol szintén kiválasztható, hogy melyik operációs rendszerre szeretnénk a programot. De mint ahogy a követelményeknél le van írva Windows-on ajánlott.

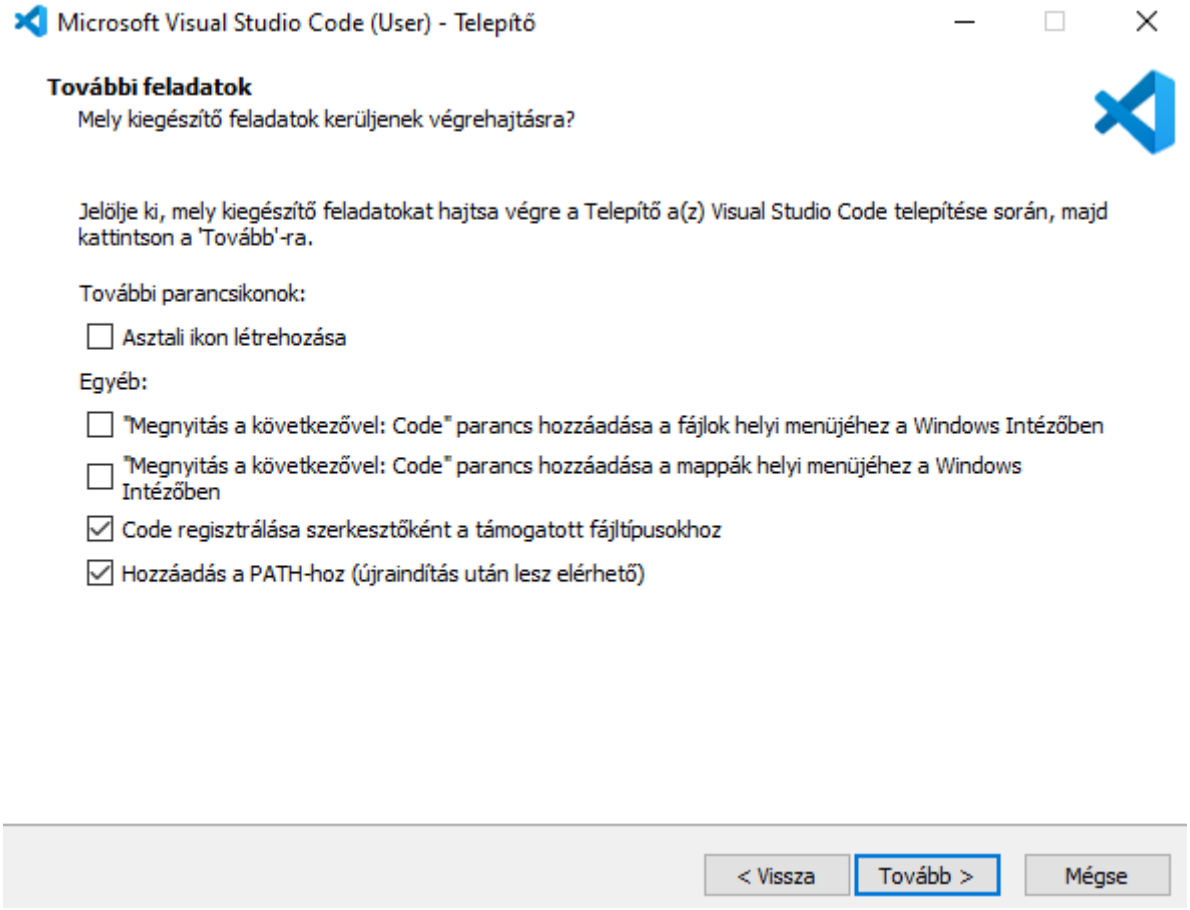
A XAMPP telepítésekor szintén csak végig kell nyomni a ’tovább’-ot és kész is a telepítése.

Az utolsó program amit ajánlatos letelepíteni az a Visual Studio Code, a Visual Studio Code a program futtatását egyszerűsíti.

A program indításához szükséges parancsokat ugyanis CMD-ben is le lehet futtatni, de a Visual Studio Code terminálja megegyszerűsíti ezt.

A Visual Studio Code szintén elérhető macOS, Linux és természetesen Windows operációs rendszereken, azonban itt az oldal a Windows-os telepítőt fogja a főoldalra helyezni.

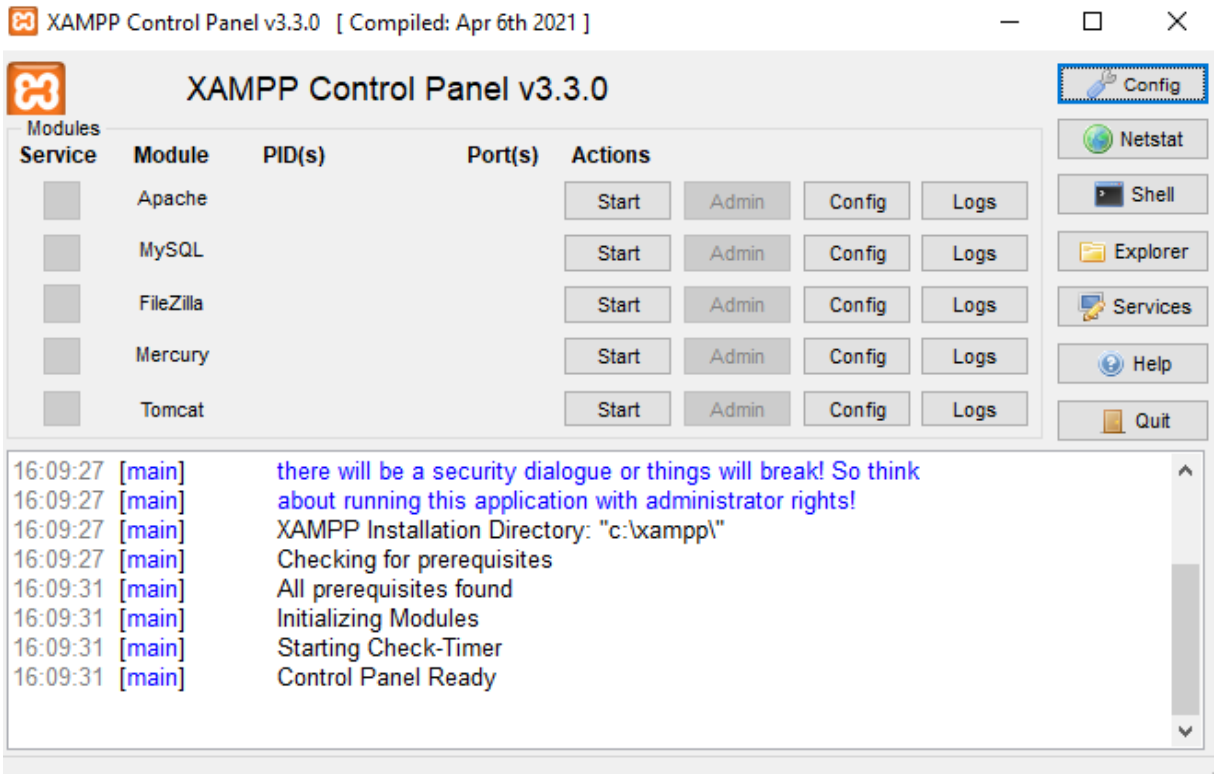
A Visual Studio Code telepítője még egyszerűbb, mint az előző programoké, itt is csak végig kell kattintgatni rajta és el is kezdi a telepítést.




*1. A Visual Studio Code telepítője*

Itt viszont ha elszeretnénk kerülni a gép újraindítását, akkor az utolsó opciót át kell választani.

A szükséges programok telepítése után célszerű megnyitni a XAMPP-ot és ott az Apache-ot és utána a MySQL-t elindítani.




*1. A XAMPP telepítője*

Miután ez mind megvan telepítse le a CD-n található GitHub repository-ról a kódot, utána csomagolja ki és indítsa el a VSC-t (Visual Studio Code).

VSC-ben a program tetején található sávból a ’File’ menüpontból válassza ki a mappa megnyitást és keresse ki hova töltötte le a projektet.

Ezután szintén a legfelső sávból a ’Terminal’ menüpontból az új terminált válassza ki. Célszerű kettőt indítani egy a frontend-nek egy a backend-nek.


Az backend-es terminálba írja, hogy ’cd backend’ a frontend-es terminálba pedig ’cd project’, majd mindkettőbe a következő parancsot kell írni ’npm install’ vagy ’npm i’ rövidítve. Ha befejezte mindkét terminálon a telepítést a backend-es terminálon a következő parancs kell ’npx prisma migrate dev’ ha ez is készen van jöhet az utolsó lépés.

Az utolsó lépés maga a program elindítása, ha minden korábbi lépést hiba nélkül sikerült megcsinálnia akkor csak egy-egy sor kell még a két terminálba.

A backend-es terminálba ’npm start’ a frontend-es terminálba pedig ’npm run dev’. És kész is a program telepítésével, az utolsó dolog, hogy a frontend-es terminálban megjelenő linket megnyitja és használhatja is a programot.

## A program használatának a részletes leírása

A program telepítés után használhatóvá válik, mint ahogy a rövid összefoglalóban említettem, a projekt PandaGO egy saját úttervező program ahol saját magunknak és másoknak utakat tervezhetünk.

A program elindítása után a linket követve a projekt kezdőlapjára kerülünk ahol megtalálni mások által készített utak vagy éppen saját is lehet, ha kijelentkezve maradt.

A kezdőlap és az egész projekt stílus szempontjából szimpla, mert szerintem a weboldalaknak nem kell túl csicsásnak lennie, de minden lényeges információt megtalál amit szeretne tudni.

Hogyha kijelentkezve van márpedig ha először használja így lesz akkor csak a bejelentkezés és regisztráció funkció lesz elérhető és a kezdőlapon az utak megtekintése.


Az utazáshoz való csatlakozás a program backend-jén le van kezelve, hogy ne tudjon profil nélkül jelentkezni rá és, hogy ne tudjon többször ugyanarra az útra jelentkezni. kivéve ha már korábban kilépett belőle és meggondolta magát.

De mivel először használja a programot ezért nincsen profilja, de hogyha létre szeretne hozni egyet, akkor a kezdőlapon található menübáron megtalálhatja a regisztrációt és a bejelentkezést, a bejelentkezés természetesen csak azok számára hasznos akiknek már van profiljuk.

A regisztrációs oldal és a bejelentkezési oldal hasonlóan néznek ki azonban eltérnek a többi oldaltól, mivel ehhez a két oldalhoz más oldal adta az inspirációt.

A regisztrációs oldalon meg kell adnia az email címét, azt is helyes formátumban tehát tartalmaznia kell @-ot és pontot.

Azt nem vizsgálja meg, hogy létezik-e az email cím azonban ha helytelen formátumban adja meg az oldal nem fogja tovább engedni.

Az email címet ellenőrzi, hogy nem használja már más felhasználó, ha talál ugyanazzal az email címmel egy felhasználót akkor szintén nem fogja tovább engedni, viszont ha helyesen adja meg az email címét és más felhasználó nem használja még akkor tovább léphet a többi adat megadására.

A következő adat amit szintén kötelező megadni az a teljes neve, ez az oldal nem felhasználó neveket használ hiszen más hivatalosabb oldalon se név nélkül vagy felhasználó néven elérhető.

Ez a mező nincsen lekezelve semmivel, hiszen bárki bármilyen névvel születhetett.

A következő mező és adat az a telefonszám a telefonszámot 06-os formátumban kell megadni, ahogy az oldal is jelzi, hiszen nagy valószínűséggel hazai telefonszáma van.

Ha viszont nem akkor tovább fejlesztéssel inkluzívabb lesz és elfogadható lesz minden külföldi telefonszám is.

Ezen kívül a telefonszámnak tizenegy karakteresnek kell lennie, kamu telefonszámok elkerülése végett.

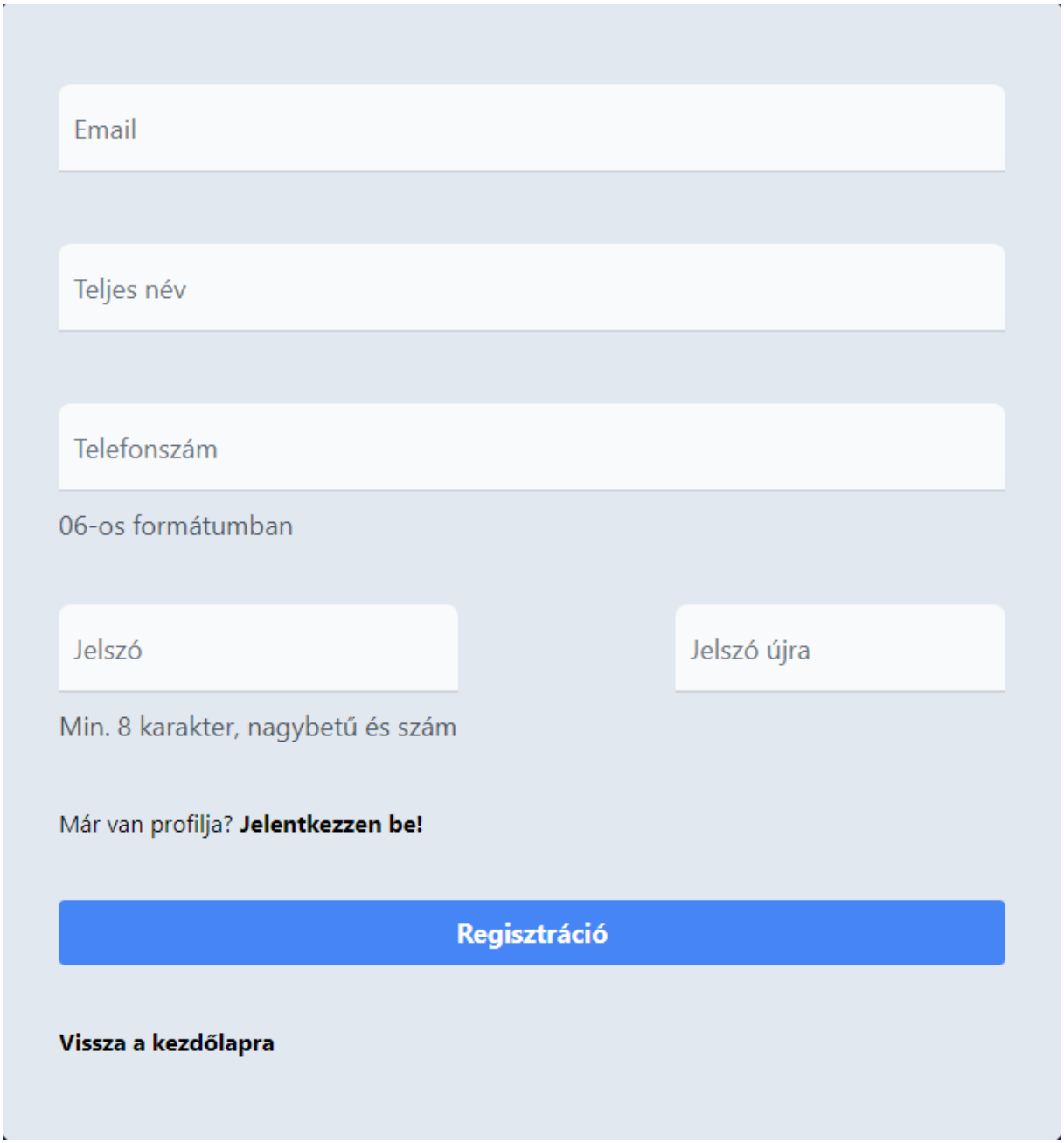
Ha az email címét, teljes nevét és telefonszámát is mind helyes megadta utána következik a jelszó, amit természetesen ki kell tölteni.

Biztonság kedvéért van egy ’Jelszó újra’ mező, hogy a felhasználó ne felejtse el a jelszavát.

A jelszó se mindegy milyen formátumban van megadva, legalább egy nagy betűt és egy számot kell tartalmaznia, és a hossza is megszabott, legalább 8 karakteresnek kell lennie biztonságosabb jelszavak létrejötte érdekében.

Ha helyesen megadta jelszavát, ne felejtse el ugyanazt megadni még egyszer, mert ha nem egyeznek a jelszavak, akkor nem fog tudni regisztrálni. Miután sikeresen megadta a szükséges információkat, sikeresen regisztrálhat.

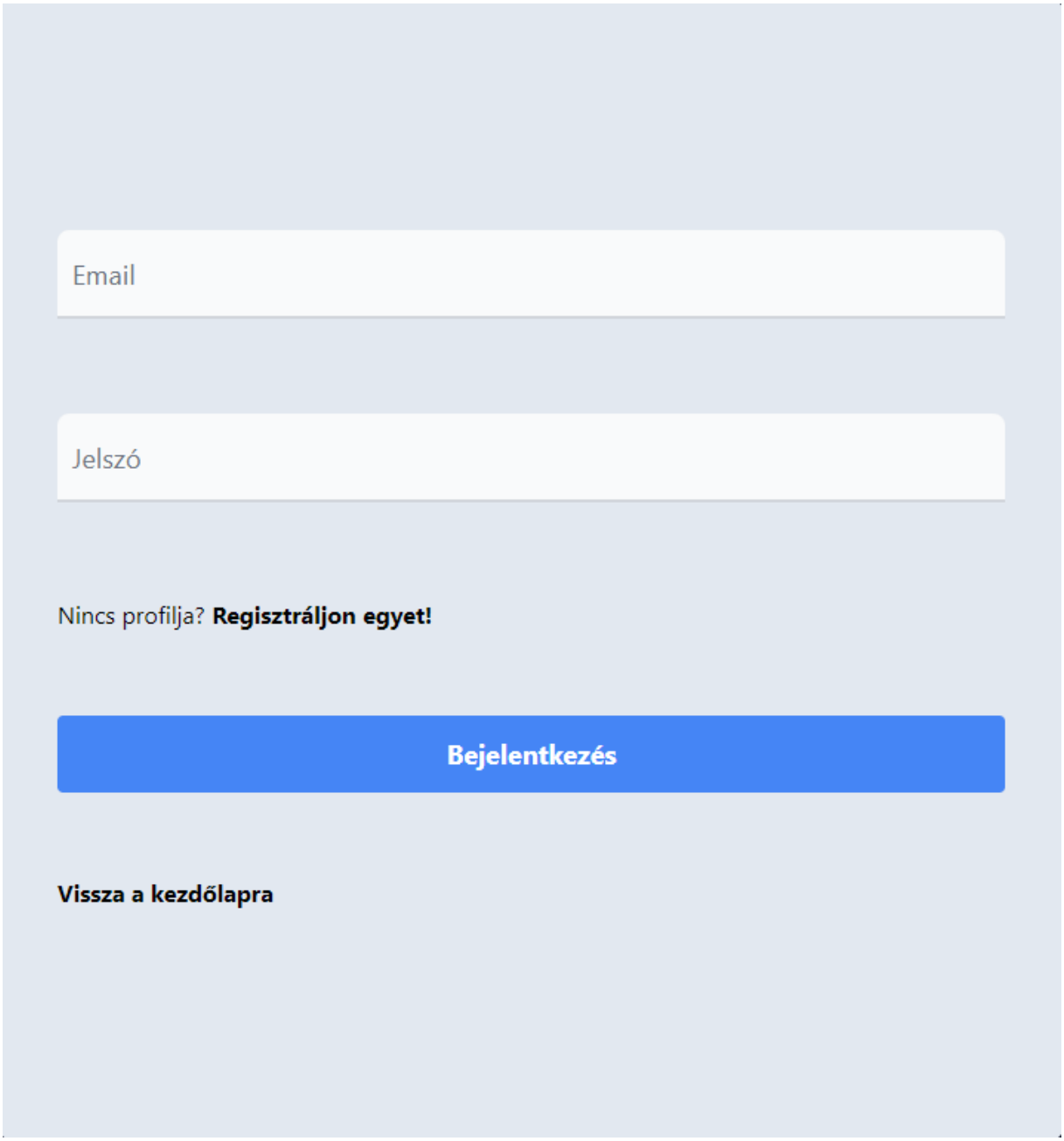
Az oldal alján találhat egy linket a bejelentkezési oldalra, ha mégis van profilja, és egy linket a kezdőlapra, ha meggondolta magát és csak meg szeretné tekinteni az utakat.




A bejelentkezési oldal hasonló kinézetre de kevesebb adatot kér a bejelentkezéshez.

A bejelentkezéshez szükséges adatok közé tartozik az email megadása, itt nincs lekezelve.

Az email-en kívül a helyes jelszó megadása kötelező a sikeres bejelentkezéshez. Ezen az oldalon is elérhető a regisztrációs oldalhoz link és a kezdőlaphoz is.




Bejelentkezés vagy regisztráció után megváltozik az oldal navigációs sáv. A regisztráció és bejelentkezés menüpontok helyére négy menüpont kerül.

Egy a kezdőlapra dobja a felhasználót ha visszaszeretne térni a kezdőlapra bármilyen oknál fogva.

Az utána lévő menüpont az ’Út létrehozása’ ami elég egyértelmű, hogy mit takar ez a menüpont vezet az út létrehozó oldalra, ahol meglehet adni, hogy mi a tervezett indulási időpont, dátummal és idővel együtt.

Az indulási idő utáni mező amit köteles kitölteni egy sikeres út létrehozásához, az a tervezett indulás helye ahol egyelőre csak fővárosokat tud megadni, azonban tud országokkal keresni fővárost, vagy éppen fordítva, de a fővárost fogja felvenni indulási helynek.

Ha ezt a két mezőt kitöltötte helyesen, a következő kitöltendő mező, az a tervezett utazás járműje lesz, ami kicsit korlátozott, tehát csak négy fajta közül lehet választani, repülőgép, hajó, vonat és busz.

A következő két megadandó adat ismerős lehet, mivel hasonlókat kellett megadnia az első kettőnél, csak jelenleg a tervezett út érkezési dátumát és idejét kell megadni, ugyanolyan formátumban mint az indulásinál.

A másik pedig a tervezett érkezés helye, vagyis az úti cél, itt is ugyanolyan formátumban adhatja meg mint az indulási helyet, tehát keresheti főváros szerint országot, vagy országot főváros szerint, viszont mindkét esetben angolul vannak megadva a városok és az országok is.

Ha minden más adatot megadott akkor az utolsó lépése, hogy az út árát megadja, forintban kéri.

Jelenleg még nincsen jelentősége az út árának, mivel nincsen fizetési lehetőség az oldalon, de már lesz jelentősége, utána pedig a programokból lesz kiszámítható az út ára, mivel az utakhoz külön-külön programokat lehet majd rendelni.

Ha ezekkel készen van az oldal alján lévő ’Létrehozás’ gombbal létre tudja hozni a tervezett utazását, és meg is találhatja az oldal kezdőlapján.


*1. Példa adatok*

Miután létrehozott egy utat, az oldal visszafogja irányítani a kezdőlapra, ahol megtekintheti mások és saját útját.

A kezdőlap, bejelentkezés óta annyit változott, hogy most már tud is jelentkezni az utakra, amiket korábban csak láthatott.


*1. Példa egy jelentkezett útra*

A menüsorban a másik két elem, a profil megtekintése és a kijelentkezés, először a kijelentkezésre térnék ki röviden.

A kijelentkezés, mint a többi korábbi menüpont, csak akkor jelenik meg ha a felhasználó be van jelentkezve.

A működése elég egyértelmű, ha be van jelentkezve a felhasználó és ki szeretne jelentkezni, csak rákattint és ki is van jelentkezve, a menüsor is frissülni fog.

Az utolsó és egyben az egyik legfontosabb menüpont a saját profil megtekintése.

A saját profil megtekintése egy új oldalra fog átirányítani, ahol megtekintheti a saját adatait, az általa tervezett utazásokat és azokat az utazásokat amelyekre tervezett menni.


*1. Bejelentkezés út áni menüsor*

A profil oldalán, az oldal bal szélén megtekintheti a saját adatait, amiken egyelőre még nem tud változtatni, és egy üres profilképet, amit szintén még nem tud változtatni.

Az oldal jobb oldalán pedig megtekintheti a tervezett utazásait, tehát az utazásokat, amelyre tervezett jelentkezni.

A jobb oldal tetején található gombokkal változtathatja, hogy melyik utazás csoportot szeretné megtekinteni, alapból a tervezett utazásokat fogja látni.

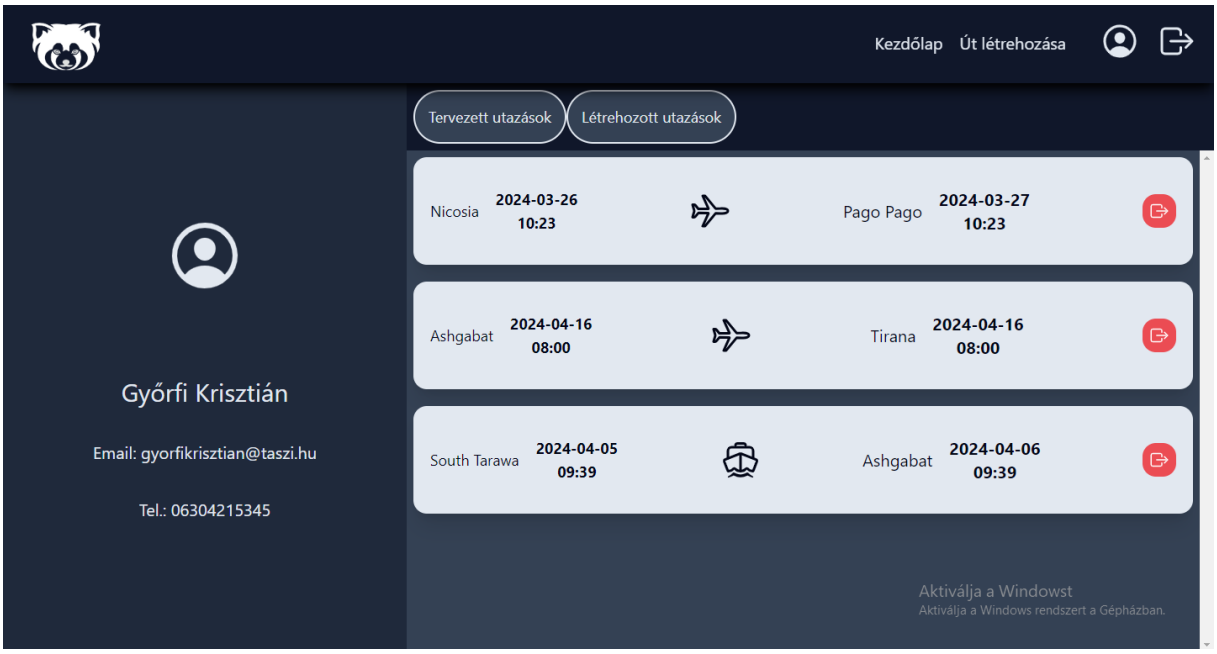
A tervezett utazásoknál, az információkon kívül lesz lehetősége kilépni az adott utazásról.

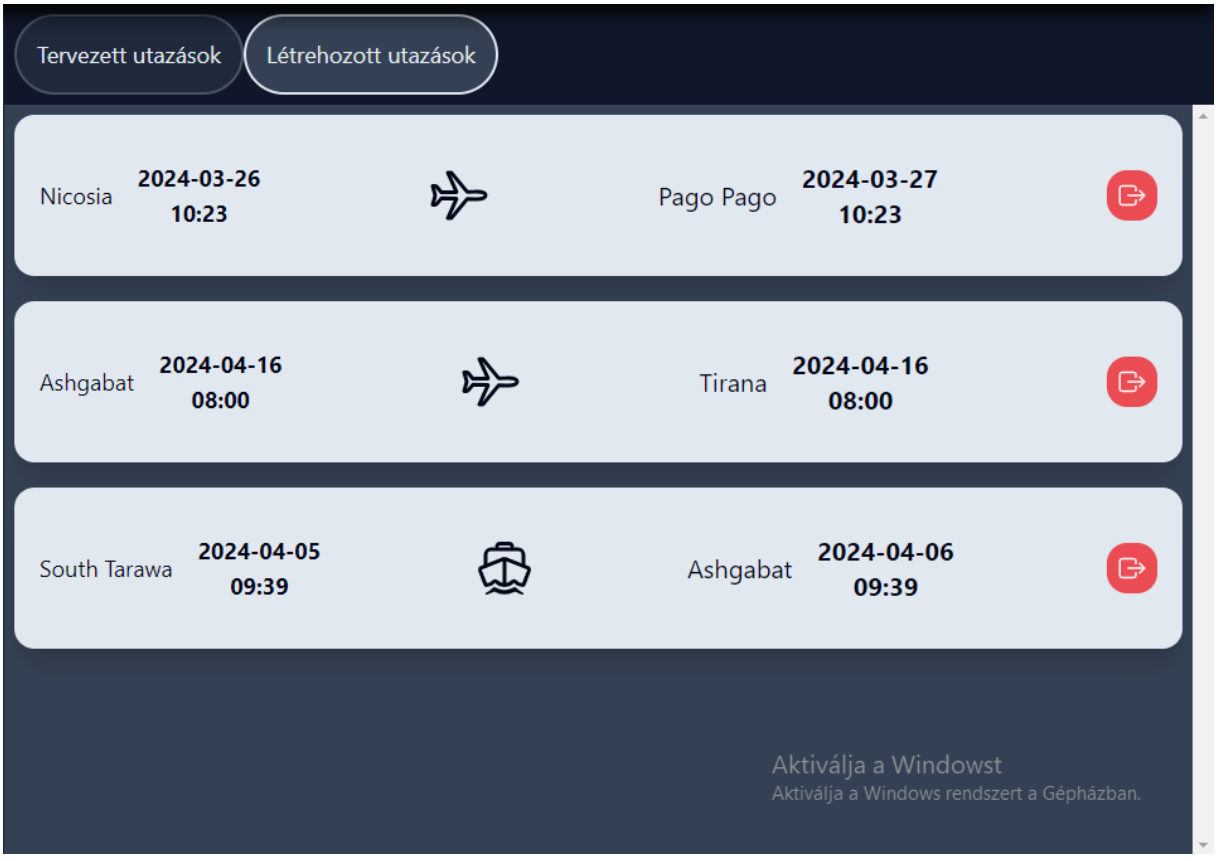
Ha rákattint az ikonra, egy felugró ablak megjelenik, hogy ’Biztosan el szeretné hagyni ezt az utat?’, ha igennel válaszol kilép az útról és törlődik a listából, viszont ha meggondolta magát akkor kattintson a ’Nem’ gombra és akkor eltünteti a felugró ablakot.

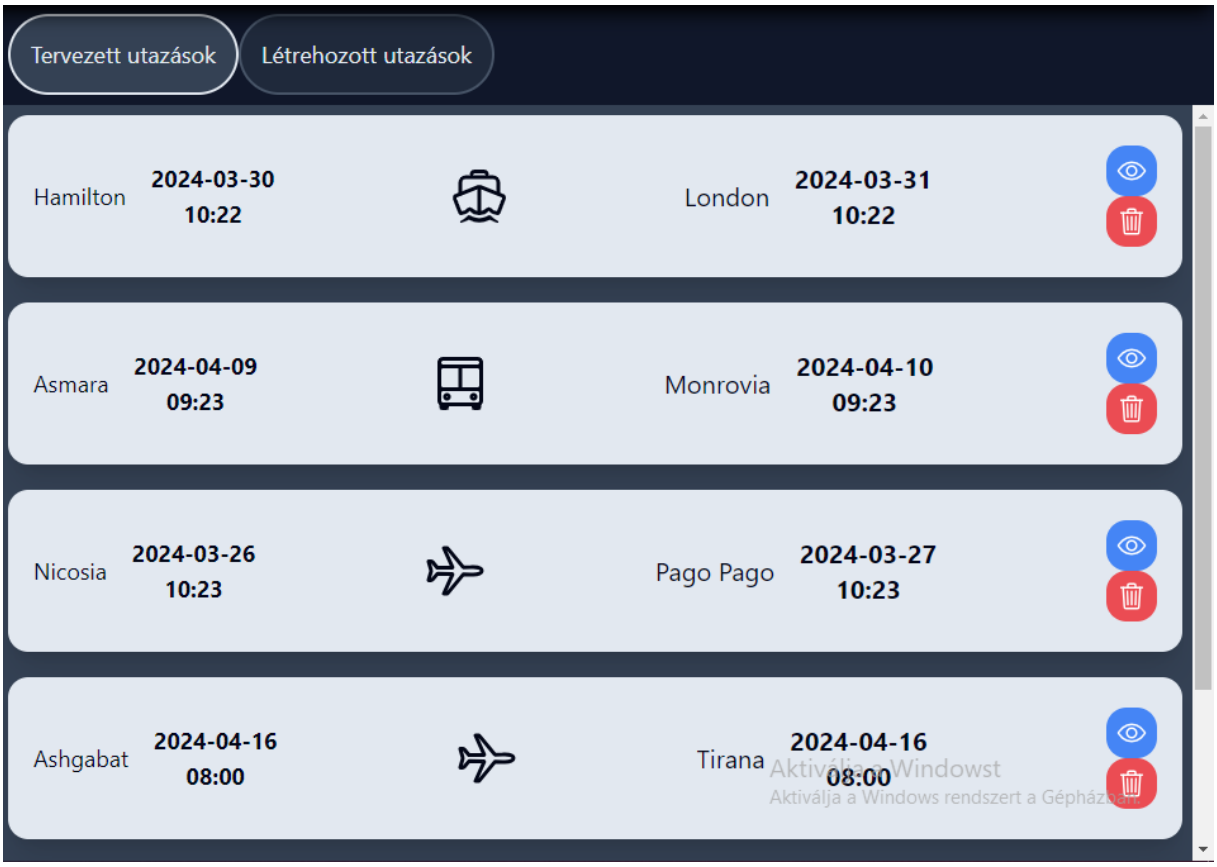
A létrehozott utakat szintén ott tekintheti meg ha másik gombra kattint, akkor megjelennek az új utak.

Létrehozott utaknál két opciója van, megtudja tekinteni az adott útra jelentkezett embereket, és alapadataikat, természetesen a jelszót nem.

A másik opció pedig az út törlése, ha szeretné törölni az utat csak szimplán rákattint a kuka ikonra és ki is törölte az utat.


Kezdőlap
Út létrehozása
Tervezett utazások
Létrehozott utazások
Győrfi Krisztián
Email: gyorfikrisztian@taszi.hu
Tel.: 06304215345
Nicosia 2024-03-26 10:23
Pago Pago 2024-03-27 10:23
Ashgabat 2024-04-16 08:00
Tirana 2024-04-16 08:00
South Tarawa 2024-04-05 09:39
Ashgabat 2024-04-06 09:39
Aktiválja a Windowst
Aktiválja a Windows rendszert a Gépházban.


Tervezett utazások
Létrehozott utazások
Nicosia 2024-03-26 10:23
Pago Pago 2024-03-27 10:23
Ashgabat 2024-04-16 08:00
Tirana 2024-04-16 08:00
South Tarawa 2024-04-05 09:39
Ashgabat 2024-04-06 09:39
Aktiválja a Windowst
Aktiválja a Windows rendszert a Gépházban.


Tervezett utazások
Létrehozott utazások
Hamilton
2024-03-30
10:22
London
2024-03-31
10:22
Asmara
2024-04-09
09:23
Monrovia
2024-04-10
09:23
Nicosia
2024-03-26
10:23
Pago Pago
2024-03-27
10:23
Ashgabat
2024-04-16
08:00
Tirana
2024-04-16
08:00
Aktiválja a Windowst
Aktiválja a Windows rendszert a Gépházban.

## Témaválasztás indoklása

Nekem mindig is közelállt szívemhez az utazás, kiskorom óta, bármikor utaztunk valahova nagyon vártam.

De mivel az utazás sokszor hosszú és unalmas lehet, és kevés olyan oldal található ahol több fajta utazást is lehet szervezni, ezért úgy döntöttem felvállalom ezt, saját felhasználásra is sok haszna lenne, de így, hogy több felhasználós így mások is használhatják saját vagy mások útjainak megszervezésére.

A weboldalak készítését az utóbbi években szintén megkedveltem ezért kifejezetten jó ötletnek tűnt és az is volt, hogy megterveztem és elkészítettem több-kevesebb sikerrel.

Az utóbbi évben a React keretrendszer nagyon minimális elsajátítása után, és a Prisma ORM használata után a backend használata is egyszerűbb.

Véleményem szerint az utazás tervező alkalmazások és oldalak nem elég ambiciózusok, és ezért nehéz olyat találni amit bárki szívesen használna.

Igaz, hogy még lehet rajta csiszolni, de az utazás szeretete miatt szerintem megérte elkészíteni ezt a projektet.

## Az alkalmazott fejlesztői eszközök

Az alkalmazás fejlesztéséhez backend-en ExpressJS-t használtam Prisma ORM segítségével. Az ExpressJS-t azért választottam mert egyszerű a használata és sokkal érthetőbb a szintaxisa mint a PHP-nak számomra. Az Express, könnyen tanulható, és a használata se sokkal nehezebb mint megtanulni magát a keretrendszert, hiszen az Express az a NodeJS-re egy keretrendszer. Könnyen formázható, bárhogyan szeretnénk használni, az Express segít benne, hogy akadálymentesen formázza saját képére a programozó. Továbbá az Express nagyon hatékonyan támogatja a HTTP metódusokat, mint például a GET, DELETE és a POST. Az Expressen kívül használtam még a backenden az Argon2-őt jelszavak titkosítására, egyszerűen használható és biztonságos jelszó hash-elést biztosít.

Ezeken felül használtam még a CORS-t, a CORS teszi lehetővé, hogy a backend-es API-kat amiket használok, elérhetővé váljon a frontend számára. A CORS tehát lényegében engedélyezi az erőforrások megosztását.

Szintén backend részen használtam a JSON Web Token-t a JWT segítségével hozhattam létre a védett backend végpontjaimat. A JWT token-eket hoz létre amiket vissza olvasva tudja a programozó, hogy ki próbált hozzáférni és ezzel engedélyezheti is adott embereknek.

A Nodemon-t is használtam. megegyszerűsítette a hiba javítást. és nem kellet folytonosan újra indítani a backend-et csak azért mert egy kis elírás vagy hiba történt. A Nodemon figyeli a projekt fájljait és hogyha talál változást akkor újraindítja a backend-et, hogy egyszerűsítse a fejlesztő dolgát.

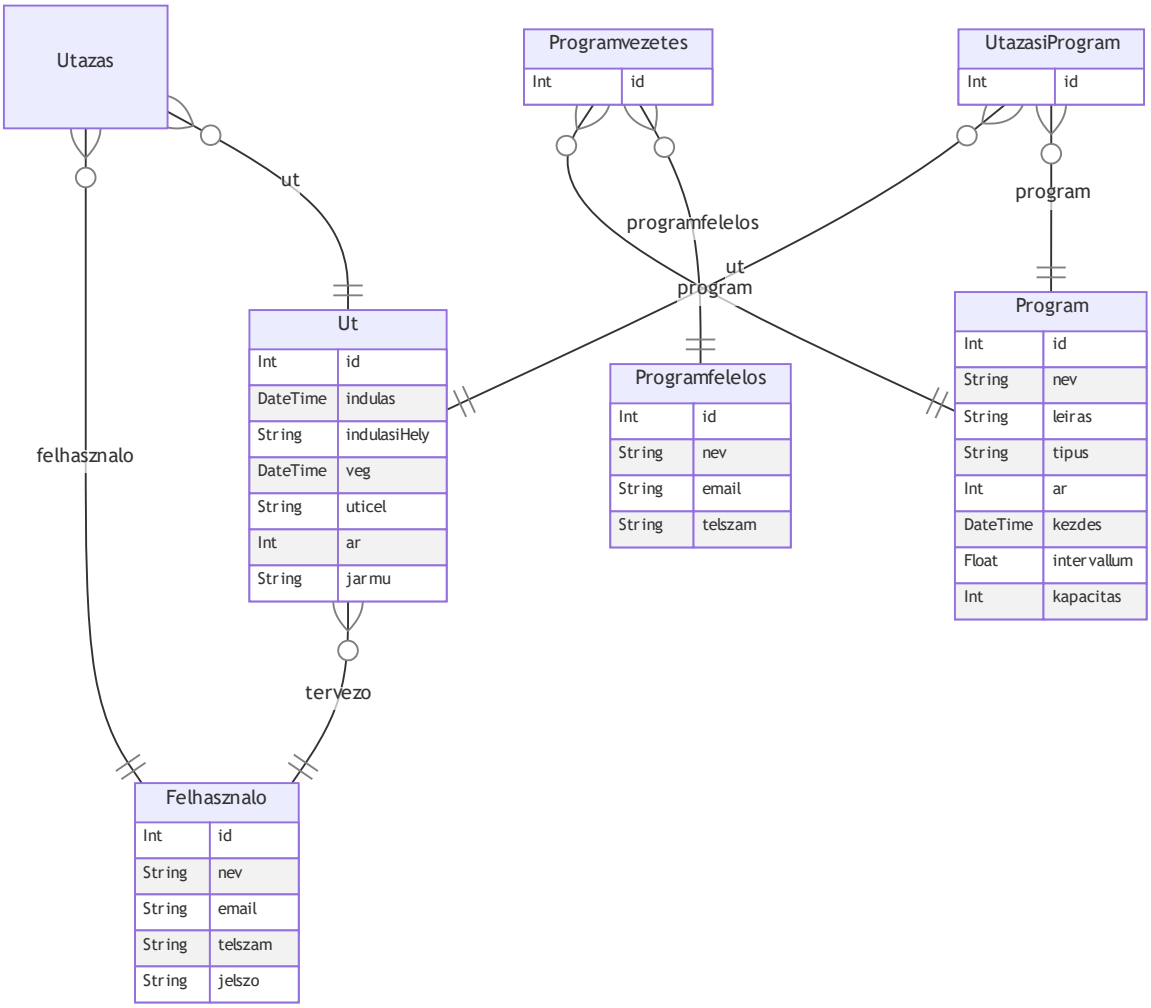
Mint korábban említettem használtam a projektemben a Prisma nevű ORM-et. Az ORM-ekről általánosságban, az ORM összeköti a projektet és az adatbázist, az ORM-ek objektumokként kezelik az adatbázis tábláit és sorait, egyszerűsítve a fejlesztő dolgát hiszen objektumokkal foglalkozik. NodeJS-re például a leghíresebb ORM-ek a Sequelize, TypeORM és amit én is használtam a Prisma.

A Prisma könnyű és hatékony adatbáziskezelést biztosít a projektekhez. A Prisma egyszerűsíti az adatbázis sémájának gyors leírását. Adatbázis független, a Prisma több SQL alapú adatbázist is támogat, többek között a PostgreSQL-t az SQLite-ot és a MySQL-t amit használtam projektemhez.

Frontenden React-et használtam ami egy JavaScript keretrendszer. A React komponensekből épül fel, ez lehetővé teszi a fejlesztők számára, hogy a projekteket több kisebb újra felhasználható kódrészletekké szedje.

A React Virtual DOM-ot használ ami egy hatékony és gyors módszer az alkalmazás felületének frissítésére. A Virtual DOM segít minimalizálni a valós DOM manipulációk számát, ami jelentősen meggyorsítja a projektet. A React támogatja az egyirányú adatáramlást, amely segít abban, hogy az alkalmazás állapota egyértelmű és előrelátható legyen. Ez a megközelítés megkönnyíti az adatok nyomon követését a projekten belül, és segíti a hibakeresést. A React támogatja a moduláris fejlesztést, ami lehetővé teszi, a fejlesztők számára, hogy különféle funkciókat különálló modulokba szervezzenek, és azokat könnyen összeszereljék a projektben. Ezen kívül a React támogatja a JSX-et ami egy olyan kiterjesztése a JavaScriptnek amely lehetővé teszi a HTML szerű kódszerkezet használatát a JavaScript kódban. Ez lehetővé teszi a fejlesztők számára, hogy deklaratív módon definiálják a felhasználói felületeket.

# Adatmodell leírása




*1. Az adatbázis modellje*

Az adatbázis Prisma sémája:

```
model Felhasznalo {
  id       Int      @id @default(autoincrement())
  nev      String
  email    String   @unique
  telszam String   @unique
  jelszo   String
  utazas   Utazas[]
  ut       Ut[]

  @@map("felhasznalok")
}

model Utazas {
  felhasznalo Felhasznalo @relation(fields: [floId], references: [id],
onDelete: Cascade)
```

```
  floId       Int
  ut          Ut          @relation(fields: [utId], references: [id],
onDelete: Cascade)
  utId        Int

  @@id(name: "id", [floId, utId])
  @@map("utazasok")
}

model Ut {
  id             Int             @id @default(autoincrement())
  indulas        DateTime
  indulasiHely   String
  veg            DateTime
  uticel         String
  ar             Int
  jarmu          String
  tervezoId      Int
  tervezo        Felhasznalo     @relation(fields: [tervezoId], references:
[id], onDelete: Cascade)
  utazas         Utazas[]
  utazasiProgram UtazasiProgram[]

  @@map("utak")
}

model UtazasiProgram {
  id      Int     @id @default(autoincrement())
  ut      Ut      @relation(fields: [utId], references: [id])
  utId    Int
  program Program @relation(fields: [prmId], references: [id])
  prmId   Int

  @@map("utazasiprogramok")
}

model Program {
  id             Int             @id @default(autoincrement())
  nev            String
  leiras         String
  tipus          String
  ar             Int
  kezdes         DateTime
  intervallum    Float
  kapacitas      Int
  programvezetes Programvezetes[]
  utazasiProgram UtazasiProgram[]

  @@map("programok")
```

```
}

model Programvezetes {
  id             Int            @id @default(autoincrement())
  program        Program        @relation(fields: [prmId], references: [id])
  prmId          Int
  programfelelos Programfelelos @relation(fields: [pfsId], references: [id])
  pfsId          Int

  @@map("programvezetesek")
}

model Programfelelos {
  id             Int              @id @default(autoincrement())
  nev            String
  email          String           @unique
  telszam        String           @unique
  programvezetes Programvezetes[]

  @@map("programfelelosok")
}
```

A táblák és mezői részletes leírása:

- Felhasználó, a felhasználó tábla a felhasználó adatait tartalmazza:
  - nev: a felhasználó teljes neve
  - email: a felhasználó email címe, nem lehet jelen többször ugyanaz az email cím
  - telszam: a felhasználó telefonszáma, nem jelenhet meg többször
  - jelszo: a felhasználó jelszava
- Út, a tervezett út adatait tartalmazza
  - indulas: az út tervezett indulási dátuma és ideje
  - indulasiHely: az út tervezett indulási helye
  - veg: az út tervezett érkezési dátuma és ideje
  - uticel: az út tervezett úticélja
  - ar: az út költsége
  - jarmu: a jármű amivel az utazás történik
  - tervezoId: az utat tervező felhasználó id-je
- Utazás, az utat és az utasokat köti össze, több-többös kapcsolatot elkerülve
  - floId: az utas id-je
  - utId: az út id-je
- Program, az utakhoz kapcsolható programok
  - nev: a program neve

    - leiras: a programról rövid leírás
    - tipus: a program típusa pl.:múzeum látogatás
    - ar: a program költsége
    - kezdes: a program kezdési ideje
    - intervallum: a program hossza
    - kapacitas: a program kapacitása
- Utazási program: az utat és a hozzá köthető programokat kapcsolja össze
    - utId: az út id-je
    - prmId: a program id-je
- Programfelelős, a programot vezető személy adatait tárolja
    - nev: a programvezető neve
    - email: a programvezető email címe
    - telszam: a programvezető telefonszáma
- Program vezetés, a programfelelőst és a programot köti össze (mivel egy programot vezethet több ember és egy ember vezethet több programot)
    - prmId: a program id-je
    - pfsId: a programfelelős id-je

## Részletes feladatspecifikáció, algoritmusok

A projekt legfontosabb függvényei tartoznak az olyan függvények amelyek a felhasználó adataival foglalkozna. Ebből a két legfontosabb a regisztráció ami bekéri a felhasználó teljes nevét, email címét, telefonszámát és jelszavát, ezeket a backend le ellenőrzi, hogy megfelelnek-e, az elvárásoknak. Például az email cím tartalmaz-e pontot és kukacot, vagy a telefonszám helyes formátumban van azaz 06-os és 11 karakterből áll.

A jelszó tartalmaz-e legalább egy nagybetűt, legalább egy számot és legalább 8 karakteres-e.

A bejelentkezésnél pedig, megnézi, hogy van-e a megadott email címmel az adatbázisban felhasználó, ha nincs természetesen nem engedi tovább a felhasználót. Ha az email címmel talált felhasználót utána csak a jelszó helyességét ellenőrzi. Egy másik függvény ahol a felhasználó adataival végzünk műveleteket, az a felhasználó adatainak visszaadása. Bejelentkezés vagy regisztráció után az oldal lokális tárolójában (local storage) letárol egy token-t ami alapján vissza keresi a felhasználót. Frontenden meghív egy backendes API-t ami token alapján vissza adja a saját adatainkat.

A felhasználó kezelésen kívül a legtöbb függvényt az utakkal való műveletek teszik ki.

Többek között egy új út létrehozása, út törlése, utazásra jelentkezés, utazásból való kilépés és két lekérés ami vissza adja az általunk létrehozott és az utak amelyekhez csatlakoztunk.

Az út létrehozásához, több adat kell mint egy szimpla profil létrehozáshoz. Az út létrehozásához szükséges adatokat nem kezeli a backend hiszen a frontendes megoldás biztosítja a helyes formátum megadását.

```
const createJourney = async (req, res) => {
    try{
        var {indulas, indulasiHely, veg, uticel, ar, jarmu} = req.body;

        if(!indulas || !indulasiHely || !veg || !uticel || !ar || !jarmu){
            res.json({message: "Hiányzó adatok!", journey: {
                indulas,
                indulasiHely,
                veg,
                uticel,
                ar,
                jarmu
            }});
            return;
        }

        indulas = new Date(indulas);
        veg = new Date(veg);

        const journey = await prisma.ut.create({
            data:{
                indulas: indulas,
                indulasiHely: indulasiHely,
                veg: veg,
                uticel: uticel,
                ar: ar,
                jarmu: jarmu,
                tervezoId: req.user.id
            }
        })

        res.json({message: "Út létrehozva!"})

    } catch(error) {
        res.json({message: error.message})
    }
}
```

A bekért adatok le ellenőrzi, hogy megadta-e egyáltalán az összes adatot, ha nem akkor egy ’Hiányos adatok’ hibaüzenettel tér vissza. Ha viszont minden adatot megadott, akkor az indulás időpontját és az érkezés időpontját átalakítja helyes formátumba, amit elfogad az SQL.

Miután ezzel végzett létrehozza magát az utat, indulási időponttal, indulási hely-el, érkezési idővel és úticéllal, az árát és a járművet is hozzárendeli. Az utolsó adatot, amit felvisz, az adatbázisba nem a felhasználó adja meg, hanem amikor megpróbál hozzáférni ehhez végponthoz, mivel nem lehet profil nélkül hozzáférni ezért kell hozzá token, a token alapján

pedig tudni, hogy ki próbál hozzáférni, és ezért az út létrehozásánál hozzárendeli a szervező id-jét amivel később vissza lehet adni hogy ki készített az adott, utat ami kifejezetten hasznos több lekérdezésnél is. Például az összes létrehozott út megjelenítésénél, hogy látható legyen a szervező neve. Vagy amikor meg szeretnénk tekinteni a saját útjainkat akkor csak azokat, adja vissza. Ennek az ellentéte az út törlése, az út törlésénél, az út id-jét kell megadni és a felhasználó id-jét, hogy csak a saját útjait törölhesse.

```
const removeJourney = async (req, res) => {
    try {
        var {utId} = req.params;
        const {tervezoId} = req.user.id;

        utId = Number(utId);

        const journey = await prisma.ut.delete(
            {
                where:{
                    id: utId,
                    tervezoId: tervezoId
                }
            }
        )

        res.json({message: "Út sikeresen törölve!"})
    } catch (error) {
        res.json({message: error.message})
    }
}
```

Maguknak az utazásoknak is vannak függvényeik, nem csak az utaknak. Az utazások függvényei közé tartozik, az úthoz csatlakozás, egy lekérdezés, amely visszaadja az összes utat, amelyre csatlakozott a felhasználó, ezt meg is jeleníti frontenden, a profil oldalon. Továbbá, van még az utazás elhagyása függvény és egy függvény, amely vissza adja egy adott út összes tervezett utasát.

Az úthoz való csatlakozás függvénye elég egyszerű, bekéri az út id-jét, amihez csatlakozni szeretnénk. Megnézi, hogy megadta-e az adatot, ha nem hibával tér vissza, másképp megkeresi azt az utat, amelyikhez csatlakozni szeretne. Ha nem találja szintén hibával tér vissza, mivel nem létezik az az utazás. Viszont abban az esetben, ha megtalálja az utat, létrehozza az utazás táblában a felhasználó id-jével és a talált út id-jével

A következő utazási függvény, a lekérdező felhasználó által jelentkezett utakat adja vissza.

Szintén fontos, hogyha elszeretnénk hagyni egy utazást, akkor azt megvalósíthassuk. Ebben segít az utazás elhagyása függvény. Ez a függvény bekéri a felhasználó id-jét, aki elszeretné hagyni az utazást, és annak az útnak az id-jét, amit elszeretne hagyni. Miután megtalálta azt az utat, amihez az id tartozik, megkeresi az utazások között azt az id párost, ahol az út id-je is stimmel és a felhasználóé is egyezik.

Az adott utazás utasait lekérő függvény, kicsit bonyolultabb, mint az előzőek. Bekéri az út id-jét, majd létrehoz egy lekérdezést amely, az adott utazásnak az összes utasát lekéri és visszaadja az alapadatait, tehát a teljes nevüket, az email címüket és a telefonszámukat, de annyiban biztosítva van, hogy ne férjen bárki, bárki adataihoz, ezért ezt a függvényt csak saját utaknál hívhatjuk meg.

```
const getPassengers = async(req, res) => {
    try {
        var {utazasId} = req.params;
        utazasId = Number(utazasId);

        let utasok = await prisma.utazas.findMany({
            where: {
                utId: utazasId
            },
            select: {
                felhasznalo: {
                    select: {
                        id: true,
                        nev: true,
                        email: true,
                        telszam: true,
                    }
                }
            }
        });
        res.json(utasok);
    } catch (error) {
        res.json({message: error.message})
    }
}
```

## Forráskód

Navigáció

A navigációt frontenden a React Router bővítményével tettem lehetővé, így mindenhol mindegyik oldal amihez linkel, elérhető lesz a projekten belül.

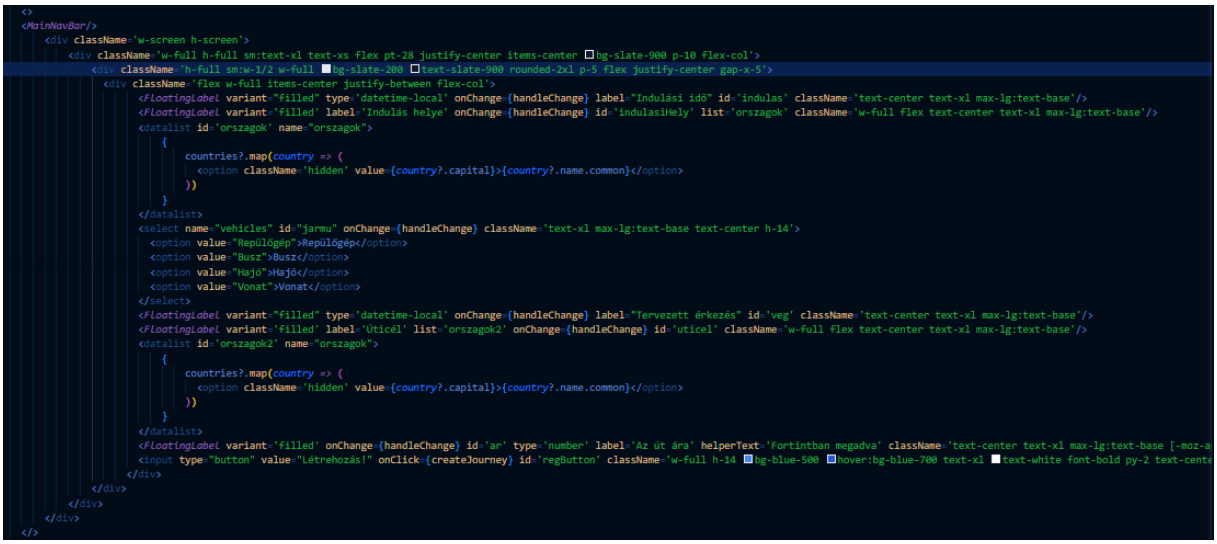
```
function App() {

  return (
    <>
    <UserProvider>
      <Router>
        <Toaster/>
      <Routes>
        <Route index element={<Kezdolap />} />
        <Route path="*" element={<Ures />} />
        <Route path='/home' element={<Kezdolap />} />
        <Route path='/login' element={<Bejelentkezes />} />
        <Route path='/register' element={<Reg/>} />
        <Route path='/me' element={<Profile />} />
        <Route path='/journey' element={<Journey />} />
      </Routes>
    </Router>
    </UserProvider>
    </>
  )
}
```

A menüsort egy külön komponensbe emeltem, hogy több oldalon is elérhető legyen, hiszen nem csak egy oldalon akartam és nem is csak egy oldalon használtam. A menüsor változik tartalomra, ha bejelentkezve van a felhasználó, és ha nincsen. Ha nincsen bejelentkezve akkor csak az ikon-t tartalmazza és egy regisztrációs oldalhoz, és bejelentkezési oldalhoz a linkeket. Azonban ha be van jelentkezve akkor a menüsor tartalma átvált, az ikon ugyanúgy megmarad, de a jobb szélen található linkek megváltoznak, lesz egy ’Kezdőlap’ nevű link amely elég egyértelműen visszairányít a kezdőlapra, utána az ’Út létrehozása’ link, amely az út létrehozó oldalra irányítja át a felhasználót. Az utána következő pedig egy profilkép helyettesítő ikon, amire rákattintva a saját profilját tekintheti meg, benne az adataival.

Oldalak

A kezdőlap, amit azonnal láthat a felhasználó, ott található az összes utazás, független attól, hogy be van-e jelentkezve. Annyi korlátozás van, hogy aki nincsen bejelentkezve nem jelentkezhet az utakra és egy hibával tér vissza az oldal. A kezdőlapon található utakat egy külön komponens adja vissza, egyszerűsítve és szépítve a kódot.

Az út készítő lap szintén egy külön oldal, amelyben nincsen másik komponens felhasználva a menüsoron kívül.




A saját profil oldal magában elég üres, két komponens tölti ki az üres teret, a két komponens vissza adja az utakat, amelyekre tervezett menni és azokat az utakat, amelyeket létrehozott. Mindegyik komponens tartalmazza az utak indulási idejét és indulási helyét, az árát, utasszállítóját, tervezett érkezési időpontot és tervezett úticélt. Az egy nagy különbség a kettő között a gombok amiket kaptak, a saját tervezésű utaknak egy törlő gombja van, a tervezett utaknak pedig egy utasok megtekintése gomb és egy út elhagyása gomb.

## Tesztelési dokumentáció

### Különböző felhasználói tevékenységekre való reakciója

Ha a felhasználó üresen hagy bármilyen mezőt a regisztrációs oldalon hiba üzenettel tér vissza az oldal. Abban az esetben is ha a két jelszó nem egyezik.

Ha a felhasználó megpróbál profil nélkül jelentkezni egy útra, az oldal elutasítja, mivel nincsen kit hozzáadnia az adatbázishoz, és üres mezőt pedig nem enged az adatbázis létrehozni.

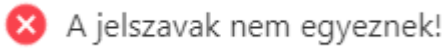
Ha a felhasználó az URL-ben szeretné átírni a linket egy olyan oldalra ahol nem lehet profil nélkül, visszairányítja a kezdőlapra. Például a saját profil oldalt nem tekintheti meg profil nélkül, és utat sem hozhat létre profil nélkül hasonló okok miatt, mint az útra jelentkezésnél.

Abban az esetben, ha a felhasználó megpróbálna többször jelentkezni ugyanarra az útra, el lesz utasítva, mivel egy ember egyszer mehet el ugyanarra az útra.

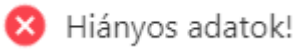
Ha a felhasználó bármilyen oknál fogva bejelentkezés után, hiányzó adatokkal próbál létrehozni egy utat, hibaüzenetet fog kapni.

### Az oldal által biztosított üzenetek és miket tegyünk az esetükben

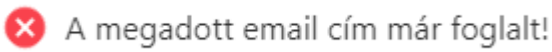

Ha profil nélkül próbál jelentkezni egy útra ezt az üzenetet fogja kapni, a megoldás rá, hogy vagy regisztrál egy profil abban az esetben ha először van az oldalon, ha pedig már van profilja, akkor csak jelentkezzen be és már tud is jelentkezni a kívánt útra.




Ha regisztrálni próbál és minden adatot megadott a helyes formátumban és mégis ezt az üzenetet kapta, akkor nem egyezik a maga által megadott jelszó és az újra megadott jelszó, javítsa ki az újra megadott jelszavát és elérhető lesz a regisztráció.

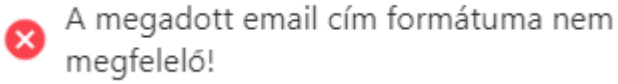



Ezt a hibaüzenetet több helyen is megkaphatja, például regisztrációnál, ha elfelejtette kitölteni valamelyik mezőt, vagy bejelentkezésnél és út létrehozásánál is megkaphatja ezt az üzenet, a megoldás rá nagyon egyszerű, töltse ki az összes mezőt az adott oldalon.



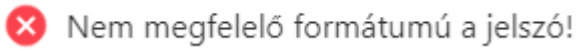
Ezt a hibaüzenetet abban az esetben kapja meg, hogyha regisztrációnál egy használatba lévő email címmel próbál regisztrálni. Erre a megoldás, hogy vagy egy másik email címmel regisztráljon, vagy ha ön használja azt az email címet, akkor jelentkezzen be.


Ha ezzel a hibaüzenettel találkozik, akkor a telefonszáma már használatban van egy másik profilon. A megoldása, vagy regisztráljon egy másik telefonszámmal vagy jelentkezzen be abba amelyikben használja.

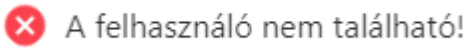



Ezt a hibaüzenetet csak akkor kapja meg hogyha hibásan adta meg az email címét, tehát vagy hiányzik belőle a ’@’ vagy a ’.’. Erre a megoldás, hogy javítsa ki az email címét.


Ebben az esetben, hasonlóan az email címeshez, akkor kapja, hogyha hibásan adta meg a telefonszámát. Két hiba lehet a megadott telefonszám formátumában vagy nem 06-os telefonszámot adott meg, vagy nem tizenegy karakteres a telefonszáma. Erre a megoldás, hogyha nem 06-os a telefonszáma, akkor egyelőre nem használható, viszont ha csak nem elég hosszú, akkor elírta a telefonszámát, javítsa ki és hibamentesen regisztrálhat.




A korábbi hibaüzenetekhez hasonlóan, ugyanaz a hiba nem megfelelő jelszót adott meg, vagy hiányzik belőle legalább egy nagybetű, vagy egy szám, vagy lehet, hogy nincs meg nyolc karakter hosszú a jelszó. A megoldás erre, hogy legalább egy nyolc karakterből álló, jelszót kell megadnia, ami legalább egy nagybetűt és egy számot tartalmaz.




Ezt a hibaüzenetet a bejelentkezési oldalon kaphatja meg, ha ezt az üzenetet kapta, vagy elírta az email címét vagy még nem regisztrált vele. Egyszerű megoldása van, ha nincsen profilja, regisztráljon egyet, abban az esetben, viszont ha csak elírta, akkor javítsa ki.

Ezzel az üzenettel szintén a bejelentkező oldalon találkozhat, ha ezt az üzenetet kapta, akkor elírta az email címhez tartozó jelszót. A megoldása egyszerű, ha elírta a jelszót javítsa ki, ha viszont elfelejtette, akkor egyelőre még nincs elfelejtett jelszó opció, tehát regisztrálnia kell egy újat.

## Továbbfejlesztési lehetőségek

Korábban többször is említettem, hogy több helyen is hiányosnak érzem a projektet, de ezeket a későbbiekben lehet tovább fejleszteni. Az egyik számomra legfontosabb tovább fejlesztési rész az a reszponzivitás lenne, a projektem asztali számítógépek vagy laptopokra van elsősorban készítve, ezért a telefonos kinézete néhány helyen nem a legszebb, van néhány megoldás ami ki segíti, tehát nem teljesen veszett eset. Azonban több idő ráfordításával még lehetne szépíteni a telefonos kinézetén.

A másik nagyobb hiányossága a projektemnek, amivel lehetne bővíteni a tartalmát az a programos táblák implementálása a projektbe, mivel létre vannak hozva az adatbázisban, de még nem volt rá helyzetem, hogy rendesen implementálhassam a projektembe. Például út létrehozásnál, nem csak az alap adatokat lehetne megadni, hanem hozzá lehetne rendelni a programok, ezekkel együtt megjeleníteni a program vezetőjét vagy éppen vezetőit. Lehetne egy külön oldal, ahol létre lehet hozni új programokat, de csak akkor, ha a felhasználó egy program vezető, amit regisztrációnál lehetne kérvényezni.

Szintén egy olyan funkció, ami bővíthetné a hasznosságát a projektemnek az az admin jog létrehozása. Az admin jog létrehozása magával vonná az admin panel létrehozását, ami bővíteni a projektem terjedelmét. Ha lenne admin jog, nem lehetne bármilyen felhasználónak utakat létrehoznia, mert így hogy bárki hozzáférhet, létrehozhat nem létező időpontokban nem létező helyekre rendezett utakat. Az admin joggal ki lehetne szűrni az ilyeneket.

Egy olyan funkció, amit hiányosnak tartok az az út létrehozásánál az időpont megadása, mivel bármikorra lehet utat tervezni, még ha meg is történt már.

Szintén az utazások bővítéséhez köthető, minden ország, ha nem is mindegyik városába, de több városába lehessen utazni.

Telefonszámokat lehessen minden országból hozzáadni, ha már internacionális utazó oldal akkor ne legyen lekorlátozva csak magyarországi telefonszámokra.

Továbbá ha már van pénzről szó az oldalon, akkor legyen fizetési mód. Arra is legalább egy vagy két oldalt lehetne fordítani, viszont akkor meg nem lehet szimplán kilépni az utazásból.

Egy nagyon fontos lépés még hiányzik a projektből, nem lehet módosítani az adatokon, például ha valaki létrehozza a profilját, megad minden adatot helyesen, de két hónapra rá bármilyen oknál fogva, vagy telefonszámot kellett cserélnie, vagy email címet ezt nem tudja jelenleg megtenni. Profilt sem lehet törölni. Szintén adatmódosítás, az utaknak is lehessen módosítani az adatait, például ha lemondák teljesen az utat legyen feltüntetve, vagy ha tolódik

akár az érkezés, akár az indulás, átlehessen írni. Meg hogyha a felhasználó elfelejti a jelszavát, legyen lehetősége visszaállítani és ne kelljen új profilt létrehoznia. Még nagyon a projekt elején meg szerettem volna valósítani az email címes hitelesítést, addig limitált hozzáférése lett volna a felhasználónak amíg az email címén nem erősítette meg, hogy ő az, ezzel elkerülhető a nem létező email címek megadása. Az elfelejtett jelszóhoz is lehetett volna email címen keresztül megoldása, hogy ne lehessen olyan egyszerűen más adatain változtatni, hogyha ismered az email címét akkor már tudod is módosítani.

# Irodalomjegyzék


- korábbi (iskolai) projektjeimből sok inspirációt merítettem, például a middleware-em egy blog oldalból alakítottam át
- osztálytársak segítségét is felhasználtam (kifejezetten Fekete Károly Richárd segítségét)
- https://www.w3schools.com
- https://react-hot-toast.com
- https://twitter.com/?lang=hu (régi bejelentkezési oldal kinézete)
- https://wizzair.com (az utak megjelenítésére)

# Plágium-nyilatkozat

Alulírott Győrfi Krisztián (név)
Békéscsaba (szül.hely)
2004.11.26 (szül.idő)
Dombi Éva (anyja neve)
jelen nyilatkozat aláírásával kijelentem, hogy a
PandaGO című záródolgozat (a továbbiakban: dolgozat) önálló munkám, a dolgozat készítése során betartottam a szerzői jogról szóló 1999. évi LXXVI. tv. szabályait, valamint a Békéscsabai Szakképzési Centrum által előírt, a dolgozat készítésére vonatkozó szabályokat.

Tudomásul veszem, hogy a dolgozat esetén plágiumnak/szerzői jogsértésnek számít:

- szó szerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más szerző publikált gondolatainak saját gondolatként való feltüntetése

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst, illetve a feladatot kiadó oktatót nem tévesztettem meg.

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot nem magam készítettem vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Békéscsabai Szakképzési Centrum a dolgozatot elégtelennek minősíti.

Békéscsaba, 2024. április 4

Tanuló aláírása