**A JourneyPal dokumentációja**

Tartalomjegyzék

Tartalom

[Bevezetés 5](#_Toc194357091)

[A dolgozat témájának ismertetése 5](#_Toc194357092)

[A program funkciójának ismertetése 7](#_Toc194357093)

[A célközönség bemutatása 9](#_Toc194357094)

[Fejlesztői dokumentáció 10](#_Toc194357095)

[A program elkészítésének indoklása 10](#_Toc194357096)

[A fejlesztőkörnyezet hardveres ismertetése 12](#_Toc194357097)

[Fejlesztés során használt szoftverek ismertetése 13](#_Toc194357098)

[A választott programnyelv és fejlesztőkörnyezet indoklása 14](#_Toc194357099)

[Adatszerkezet részletes ismertetése 17](#_Toc194357100)

[1. A táblák részletes ismertetése 17](#_Toc194357101)

[2. Táblák közötti kapcsolatok 22](#_Toc194357102)

[3. Adatbázis diagram 23](#_Toc194357103)

[4. Összegzés 23](#_Toc194357104)

[A fejlesztés során használt nevezetes algoritmusok 24](#_Toc194357105)

[1. GenerateToken(User user) Metódus 24](#_Toc194357106)

[2. GenerateRefreshToken() Metódus 25](#_Toc194357107)

[Összegzés 26](#_Toc194357108)

[Unit test 27](#_Toc194357109)

[Áttekintés 27](#_Toc194357110)

[Összefoglalás 36](#_Toc194357111)

[Tesztelői hozzáférés biztosítása 36](#_Toc194357112)

[Összegzés 38](#_Toc194357113)

[Továbbfejlesztési lehetőségek 38](#_Toc194357114)

[1. Mesterséges intelligencia alapú programajánló 38](#_Toc194357115)

[2. Időjárás-előrejelző integráció 40](#_Toc194357116)

[Összegzés 41](#_Toc194357117)

[Szerveroldali API-k és vezérlők bemutatása 42](#_Toc194357118)

[AuthController metódusai 42](#_Toc194357119)

[AccountDetailsController metódusai 47](#_Toc194357120)

[AdminController metódusai 49](#_Toc194357121)

[EventController metódusai 51](#_Toc194357122)

[52](#_Toc194357123)

[TripController metódusai 54](#_Toc194357124)

[57](#_Toc194357125)

[58](#_Toc194357126)

[Adminisztrációs Felület Bemutatása 59](#_Toc194357127)

[Felhasználói dokumentáció 62](#_Toc194357128)

[A program bemutatása 62](#_Toc194357129)

[Rendszerkövetelmények 64](#_Toc194357130)

[A program használatához szükséges szoftverek és telepítésük 65](#_Toc194357131)

[A program funkcióinak ismertetése 67](#_Toc194357132)

[Regisztrációs követelmények 75](#_Toc194357133)

[Hibakezelés és hibajelzések 77](#_Toc194357134)

[Összefoglalás 79](#_Toc194357135)

[Az elkészült munka értékelése 79](#_Toc194357136)

[A JourneyPal jövője 80](#_Toc194357137)

[1. A program hasznosulása az utazók körében 80](#_Toc194357138)

[2. Lehetséges piaci bevezetés és monetizáció 81](#_Toc194357139)

[3. Továbbfejlesztési lehetőségek 81](#_Toc194357140)

[4. A JourneyPal szerepe a digitális nomádok és gyakori utazók életében 82](#_Toc194357141)

# Bevezetés

## A dolgozat témájának ismertetése

A záródolgozat témája egy utazástervező és szervező alkalmazás, a **JourneyPal**, amely segíti a felhasználókat az utazásaik megtervezésében, események hozzáadásában és a kívánt úti célok kiválasztásában. Az alkalmazás célja, hogy egy egyszerűen kezelhető, felhasználóbarát felületet biztosítson az utazásra vágyók számára, amelyben előre meghatározott desztinációk közül választhatnak, vagy saját úti célt is megadhatnak.

A JourneyPal különlegessége abban rejlik, hogy nemcsak az utazások szervezését teszi lehetővé, hanem az adott úti célhoz és időpontokhoz kapcsolódó események kezelését is. Ha egy felhasználó például lefoglal egy utazást **április 1. és április 4.** között, akkor minden egyes naphoz külön eseményeket adhat hozzá, ezzel egy részletes és strukturált utazási tervet alakíthat ki. Az események megtervezéséhez az alkalmazás egy egyszerű, átlátható kezelőfelületet biztosít, amely a „Create Event” gombbal könnyen bővíthető.

A fejlesztés során kiemelt figyelmet fordítottunk arra, hogy a JourneyPal egyszerű és intuitív felhasználói élményt nyújtson, ezért jelenleg nincs keresési vagy szűrési funkció az események között. Ehelyett az előre létrehozott úti célok – például **Tokió, Santorini, Faroe-szigetek** – közül lehet választani, illetve lehetőség van saját célállomás megadására is, bár ebben az esetben a funkciók köre korlátozottabb.

Bár a jelenlegi verzió még nem tartalmaz közösségi média megosztási lehetőségeket vagy más közösségi funkciókat, a további fejlesztések során ezeket is megvizsgáljuk, hogy a felhasználók könnyebben megoszthassák utazási terveiket barátaikkal és családtagjaikkal.

A JourneyPal célja tehát nemcsak az, hogy egy hatékony és modern utazástervező eszközt biztosítson, hanem hogy inspirációt is nyújtson azok számára, akik még csak az álmaik utazását tervezik. A felhasználók számára egy olyan alkalmazást szeretnénk nyújtani, amely könnyen kezelhető, vizuálisan vonzó és valódi segítséget nyújt a következő kalandjuk megszervezésében.

(temavalasztas, ok)

A JourneyPal alkalmazás ötlete abból a személyes élményből született, hogy mind a ketten, a csapattársammal együtt, nagyon szeretünk utazni és új helyeket felfedezni. Az utazások során gyakran szembesültünk azzal, hogy a tervezés bonyolult és időigényes lehet, különösen, ha több eseményt és programot kell összeszervezni.

A célunk az volt, hogy egy olyan felhasználóbarát megoldást hozzunk létre, amely segít az embereknek egyszerűbben és hatékonyabban megtervezni az utazásaikat. A JourneyPal lehetőséget biztosít arra, hogy minden utazással kapcsolatos információt egy helyen kezeljenek, saját utakat hozzanak létre, vagy meglévő utazások közül válasszanak. Ezzel nemcsak időt spórolhatnak, hanem élvezetesebbé is tehetik az utazás előkészítésének folyamatát.

## A program funkciójának ismertetése

**A program funkciói**

A *JourneyPal* egy utazástervező alkalmazás, amelynek célja, hogy egyszerűsítse és átláthatóvá tegye a felhasználók számára az utazásaik szervezését. Az alkalmazás főbb funkciói a következők:

* **Utazás létrehozása:** A felhasználók saját utazásokat (tripeket) hozhatnak létre, melyekhez nevet adhatnak, kiválaszthatják az utazás dátumát, valamint megadhatják az úticélt. Az úticél kiválasztásánál lehetőség van az alkalmazásban előre definiált helyek közül választani (például Tokió, Santorini, Feröer-szigetek), de egyéni helyszín is megadható, bár ebben az esetben a funkciók korlátozottabbak.
* **Események hozzáadása:** Miután egy utazás létrejött és az időpont kiválasztásra került (például április 1. – április 4.), a felhasználó minden egyes naphoz külön eseményt adhat hozzá. Ez a *Create Event* gomb segítségével történik, így az utazás minden napja részletesen megtervezhető.
* **Úticélok kezelése:** Az alkalmazás egy előre meghatározott listát kínál népszerű úti célokból, amelyeket a felhasználók kiválaszthatnak. Egyedi úti célok is megadhatók, azonban ezeknél egyelőre kevesebb funkció érhető el.

Jelenlegi verziójában az alkalmazás nem tartalmaz keresési vagy szűrési funkciókat, valamint közösségi média integrációt vagy megosztási lehetőséget. Azonban a jövőbeni fejlesztések során ezek a funkciók átgondolhatók és beépíthetők.

## A célközönség bemutatása

**A célközönség**

A *JourneyPal* célközönsége minden utazni vágyó ember, aki szeretné megtervezni a következő kalandját – legyen szó egy valós, közelgő utazásról vagy akár egy álomutazás megtervezéséről a jövőre nézve.

Az alkalmazás különösen hasznos lehet a következő felhasználói csoportok számára:

* **Hobbiutazók és világjárók:** Azok számára, akik rendszeresen útra kelnek, és szeretnék rendszerezni, megtervezni az útjaikat, eseményeiket.
* **Családok és baráti társaságok:** A közös utazások megszervezése sok kihívást jelenthet. A *JourneyPal* segíthet abban, hogy mindenki számára átlátható legyen a program.
* **Első alkalommal utazók:** Azoknak, akik még nem rutinos utazók, az alkalmazás segít lépésről lépésre megtervezni az útjukat, és csökkenteni az ezzel járó stresszt.
* **Digitális nomádok:** Azok, akik munka mellett folyamatosan úton vannak, könnyen rendszerezhetik és előre megtervezhetik következő állomásaikat.
* **Álomutazók:** Nem minden utazás történik meg azonnal. Sokan szeretnek előre tervezni, inspirációt gyűjteni a jövőre nézve. A *JourneyPal* lehetőséget ad arra, hogy bárki összeállítsa az álomutazását, még akkor is, ha azt egyelőre nem tudja megvalósítani.

Az alkalmazás célja, hogy minden típusú utazó számára egy könnyen használható, intuitív és átlátható megoldást nyújtson az utazások megtervezésére.

# Fejlesztői dokumentáció

## A program elkészítésének indoklása

**Miért volt szükség a program elkészítésére, és miben különbözik más hasonló programoktól?**

A digitális világban számos utazástervező alkalmazás létezik, amelyek különféle funkciókkal segítik az utazók életét. Azonban ezek közül sok vagy túlságosan bonyolult, tele van felesleges funkciókkal, vagy épp ellenkezőleg, annyira egyszerű, hogy nem nyújt valódi segítséget az utazás részletes megtervezéséhez. A *JourneyPal* célja az volt, hogy a felhasználóknak egy olyan intuitív és könnyen kezelhető eszközt adjon a kezébe, amely egyszerre átlátható, mégis elegendő rugalmasságot biztosít az egyéni igényekhez.

A *JourneyPal* néhány kiemelkedő előnye a hasonló alkalmazásokkal szemben:

* **Egyszerű, mégis hatékony utazástervezés:** Sok utazástervező applikáció összetett funkciókkal terheli le a felhasználókat, míg mások túlzottan minimalista megközelítést alkalmaznak. A *JourneyPal* az arany középutat képviseli: intuitív kezelőfelületet kínál, amely lehetővé teszi, hogy a felhasználók gyorsan és könnyedén létrehozzák saját utazásaikat.
* **Előre definiált úti célok és testreszabható utak:** Az alkalmazás beépített úti célokat kínál a legnépszerűbb helyszínekkel (pl. Tokió, Santorini, Feröer-szigetek), ugyanakkor lehetőséget biztosít arra is, hogy a felhasználók saját egyéni helyszíneket adjanak meg, így személyre szabhatják az utazásukat.
* **Részletes eseményszervezés:** A legtöbb utazástervező alkalmazás csak általános listák készítését engedi, azonban a *JourneyPal* lehetővé teszi, hogy az utazók minden egyes napra külön eseményeket ütemezzenek, így pontosan láthatják, hogyan fog kinézni az adott napjuk. Ez különösen hasznos lehet azok számára, akik szeretik előre megszervezni a programjaikat, és nem szeretnék, ha az utazás alatt bármi elmaradna vagy elfelejtődne.
* **Felhasználóbarát és stresszmentes tervezési élmény:** Sok alkalmazás túlzottan bonyolult navigációval és felesleges menüpontokkal nehezíti a tervezési folyamatot. A *JourneyPal* célja, hogy a felhasználók gyorsan és egyszerűen elérjék a legfontosabb funkciókat, anélkül hogy elvesznének a beállítások között.
* **Álomutazások tervezése, nem csak valós utakhoz:** Egyedülálló funkcióként a *JourneyPal* nemcsak a valós utazások tervezésére alkalmas, hanem arra is, hogy a felhasználók előre megtervezzék álomutazásaikat. Így inspirációt gyűjthetnek, elkészíthetik az ideális útvonalat és programtervet, amit később könnyedén megvalósíthatnak.

A *JourneyPal* tehát egy olyan utazástervező alkalmazás, amely hidat képez a bonyolult, professzionális rendszerek és az egyszerű, de sokszor túlzottan limitált megoldások között. Egyensúlyt teremt a funkcionalitás és a könnyű kezelhetőség között, így minden típusú utazó számára hasznos lehet – akár egy közelgő kirándulásra, akár egy távoli álomút megtervezésére készül.

## A fejlesztőkörnyezet hardveres ismertetése

A fejlesztői környezet alapját egy nagy teljesítményű asztali számítógép biztosítja, amely megfelelő erőforrásokat nyújt a szoftverfejlesztési és tesztelési feladatok hatékony végrehajtásához. A rendszer központi feldolgozóegysége egy Intel Core i7-11900KF processzor, amely 8 fizikai maggal és 16 szálas párhuzamos feldolgozási kapacitással rendelkezik. Az 5,3 GHz-ig terjedő maximális órajelfrekvencia és a 16 MB Intel Smart Cache lehetővé teszi a gyors és zökkenőmentes kódfordítást, valamint a virtuális környezetek és konténerek hatékony futtatását.

A rendszerben 32 GB DDR4 RAM található, amely megfelelő erőforrást biztosít a nagyobb memóriaigényű fejlesztői és futtatási környezetek számára. Ez különösen előnyös több példányban futó fejlesztői szerverek és konténerek, valamint virtuális gépek kezelése esetén.

A számítógép operációs rendszere Windows 10 64-bit, amely széles körű kompatibilitást biztosít a fejlesztői eszközökkel, valamint stabil és optimalizált környezetet nyújt az alkalmazás fejlesztéséhez és teszteléséhez.

## Fejlesztés során használt szoftverek ismertetése

A szoftverfejlesztési munkafolyamat során több fejlesztői környezet és segédprogram is alkalmazásra kerül a hatékony fejlesztés, hibakeresés és verziókezelés érdekében.

Az alkalmazás fejlesztésének fő környezetéül Visual Studio 2022 szolgál, amely teljes körű támogatást nyújt a .NET-alapú alkalmazások fejlesztéséhez. A Visual Studio beépített hibakeresője (Debugger), az IntelliSense kódkiegészítési funkció, valamint az integrált verziókezelési megoldások (pl. GitHub, Azure DevOps) jelentősen növelik a fejlesztési hatékonyságot.

A frontend fejlesztési feladatokhoz Visual Studio Code kerül alkalmazásra, amely különösen alkalmas React.js és egyéb webes technológiák fejlesztésére. A Visual Studio Code könnyűsúlyú, de bővíthető környezet, amely számos kiegészítővel rendelkezik, például az ESLint, Prettier és GitLens bővítményekkel, amelyek hozzájárulnak a tiszta, egységes és követhető kódstruktúra kialakításához.

A verziókezeléshez Git kerül alkalmazásra, amely biztosítja a kódváltozások nyomon követhetőségét és a csapatmunkát támogató funkciókat. A kódtárak tárolása és menedzselése GitHub segítségével történik, amely lehetővé teszi az elágazások (branch-ek) kezelését, a kódellenőrzési folyamatokat (pull request-ek), valamint a folyamatos integrációs és telepítési (CI/CD) munkafolyamatokat.

A projekt fejlesztése során a megfelelő fejlesztői és tesztkörnyezetek létrehozása érdekében esetenként konténerizációs technológiák is alkalmazásra kerülhetnek, például Docker, amely lehetővé teszi az egységes és skálázható alkalmazásfuttatási környezetek létrehozását.

Ezen eszközök együttes alkalmazása biztosítja a hatékony fejlesztési folyamatokat, az alkalmazás stabilitását és a verziókezelési rendszer gördülékeny működését.

## A választott programnyelv és fejlesztőkörnyezet indoklása

A projekt fejlesztése során a választott programozási nyelv és fejlesztőkörnyezet kiválasztásakor több szempontot is figyelembe kellett venni, beleértve a teljesítményt, a fejlesztői eszközök támogatottságát, a skálázhatóságot és a hosszú távú fenntarthatóságot.

**Programozási nyelv választása**

A projekt megvalósításához C# és JavaScript (React.js) programozási nyelvek kerültek kiválasztásra.

* **C#:**

A backend fejlesztéshez a C# nyelv mellett döntöttem, mivel egy modern, objektumorientált nyelv, amely széles körű támogatottsággal és fejlett fejlesztői eszközökkel rendelkezik. A .NET keretrendszer segítségével könnyedén kezelhetők az adatbázis-műveletek, az API-fejlesztés és a szerveroldali logika implementációja. Emellett a C# erős típusossága és magas szintű biztonsági funkciói elősegítik a hibák minimalizálását és a karbantarthatóságot.

* **JavaScript (React.js):**

A frontend fejlesztéséhez a React.js keretrendszer mellett döntöttem, mivel dinamikus, komponensalapú architektúrát biztosít, amely megkönnyíti az újrafelhasználható UI-elemek fejlesztését. A React.js egy széles körben elterjedt és aktívan fejlesztett technológia, amely nagy teljesítményű, reszponzív és interaktív felhasználói felületek létrehozását teszi lehetővé.

A választott programozási nyelvek kombinációja lehetővé teszi a hatékony full-stack fejlesztést, amelynek során a backend és a frontend technológiák jól integrálhatók egymással.

**Fejlesztőkörnyezet választása**

A fejlesztési folyamat során Visual Studio 2022 és Visual Studio Code került alkalmazásra.

* **Visual Studio 2022:**

A Visual Studio 2022 a C# és .NET alapú fejlesztéshez optimális választás, mivel beépített fejlesztői eszközökkel rendelkezik, mint például:

* + Fejlett IntelliSense és automatikus kódkiegészítés
  + Hatékony beépített hibakereső és teljesítményprofilozó eszközök
  + Entity Framework Core és egyéb modern adatbázis-kezelési megoldások integrált támogatása
  + Verziókezelési támogatás (GitHub, Azure DevOps)
* **Visual Studio Code:**

A Visual Studio Code a React.js alapú frontend fejlesztéshez került kiválasztásra. Ennek oka, hogy:

* + Könnyűsúlyú, gyors és testreszabható fejlesztői környezet
  + Széles körű bővítménytámogatás (pl. ESLint, Prettier, GitLens)
  + Fejlett Live Server és Debugging funkciók biztosítják a hatékony fejlesztést

A Visual Studio 2022 és a Visual Studio Code kombinációja lehetővé teszi a hatékony full-stack fejlesztést, amely egyaránt támogatja a szerver- és kliensoldali fejlesztési munkafolyamatokat.

## Adatszerkezet részletes ismertetése

A rendszer adatbázisának tervezése során a relációs adatmodell elveit alkalmazva négy fő táblát definiáltam: User, Trip, Event, TripNote. Ezek a táblák az alkalmazás alapvető entitásait reprezentálják, és megfelelő kapcsolatokkal biztosítják az adatok konzisztenciáját és hatékony kezelését.

### A táblák részletes ismertetése

1.1. User tábla – Felhasználók kezelése

A User tábla tárolja az alkalmazás regisztrált felhasználóinak adatait. Azonosítója egy Id mező, amely egyedi string alapú elsődleges kulcs (pk). A felhasználói adatok normalizált formában is tárolásra kerülnek, így biztosítva a gyorsabb kereshetőséget és az egyedi azonosíthatóságot.

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Típus | Leírás |
| Id | string (pk) | Egyedi felhasználói azonosító |
| UserName | string | Felhasználó neve |
| NormalizedUserName | string | Felhasználónév normalizált változata |
| Email | string | Felhasználó e-mail címe |
| NormalizedEmail | string | Normalizált e-mail cím |
| EmailConfirmed | boolean | Igaz, ha az e-mail cím megerősített |
| PasswordHash | string | A felhasználó jelszavának hash-elt verziója |
| SecurityStamp | string | Azonosító a biztonsági műveletekhez |
| ConcurrencyStamp | string | Verziókezeléshez használt egyedi azonosító |
| PhoneNumber | string | |  | | --- | | Felhasználó telefonszáma |  |  | | --- | |  | |
| PhoneNumberConfirmed | boolean | |  | | --- | | Igaz, ha a telefonszám megerősített |  |  | | --- | |  | |
| TwoFactorEnabled | boolean | |  | | --- | | Kétszintű azonosítás engedélyezve |  |  | | --- | |  | |
| LockoutEnd | datetime | |  | | --- | | Fiók zárolási lejárati időpontja |  |  | | --- | |  | |
| LockoutEnabled | boolean | |  | | --- | | Igaz, ha a fiók zárolt |  |  | | --- | |  | |
| AccessFailedCount | integer | |  | | --- | | Sikertelen bejelentkezési próbálkozások száma |  |  | | --- | |  | |
| RefreshToken | string | |  | | --- | | Frissítő token a hosszabb bejelentkezési időtartamhoz |  |  | | --- | |  | |
| RefreshTokenExpiryTime | datetime | |  | | --- | | Frissítő token lejárati időpontja |  |  | | --- | |  | |
| Provider | string | |  | | --- | | Harmadik fél általi autentikációs szolgáltató |  |  | | --- | |  | |
| ProviderUserId | string | |  | | --- | | Harmadik fél azonosítója |  |  | | --- | |  | |
| Role | |  | | --- | | string (default: 'User') |  |  | | --- | |  | | Felhasználói szerepkör (User, Admin) |

Ez a szerkezet biztosítja a **felhasználókezelés teljes körű funkcionalitását**, beleértve az **autentikációt, jelszókezelést és biztonsági műveleteket**.

**1.2. Trip tábla – Utazások nyilvántartása**

A **Trip** tábla az utazások alapvető adatait tárolja, és kapcsolatban áll a **User** táblával, azaz minden utazás egy adott felhasználóhoz tartozik.

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Típus | Leírás |
| Id | integer (pk) | |  | | --- | | Egyedi utazás azonosító |  |  | | --- | |  | |
| TripName | string | |  | | --- | | Az utazás neve |  |  | | --- | |  | |
| Destination | string | Célállomás |
| StartDate | datetime | Kezdődátum |
| EndDate | datetime | |  | | --- | | Befejező dátum |  |  | | --- | |  | |
| UserId | string | Az utazás tulajdonosának azonosítója (kapcsolódik a User.Id mezőhöz) |

**1.3. Event tábla – Utazáshoz kapcsolódó események**

Az **Event** tábla az utazásokhoz kapcsolódó eseményeket tárolja. Egy esemény mindig egy adott **Trip**-hez tartozik, így az **TripId** oszlop biztosítja a kapcsolatot.

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Típus | Leírás |
| Id | |  | | --- | | integer (pk) |  |  | | --- | |  | | |  | | --- | | Egyedi eseményazonosító |  |  | | --- | |  | |
| EventName | string | |  | | --- | | Az esemény neve |  |  | | --- | |  | |
| EventDescription | string | |  | | --- | | Az esemény leírása |  |  | | --- | |  | |
| EventLocation | string | Helyszín |
| EventLinks | json (lista) | |  | | --- | | Kapcsolódó linkek (pl. térkép, jegyfoglalás) |  |  | | --- | |  | |
| EventDate | datetime | |  | | --- | | Esemény időpontja |  |  | | --- | |  | |
| TripId | integer | Az eseményhez tartozó utazás azonosítója (kapcsolódik a Trip.Id mezőhöz) |

A JSON formátumban tárolt **EventLinks** mező lehetővé teszi, hogy az eseményekhez **dinamikusan bővíthető hivatkozásokat** rendeljünk (pl. Google Maps link, jegyvásárlási oldal).

**1.4. TripNote tábla – Utazáshoz kapcsolódó jegyzetek**

A **TripNote** tábla az utazásokhoz tartozó egyéni jegyzeteket tárolja. Minden jegyzet egy adott **Trip**-hez kapcsolódik.

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Típus | Leírás |
| Id | |  | | --- | | integer (pk) |  |  | | --- | |  | | |  | | --- | | Egyedi jegyzetazonosító |  |  | | --- | |  | |
| Content | string | |  | | --- | | Jegyzet tartalma |  |  | | --- | |  | |
| CreatedAt | |  | | --- | | datetime (default: now()) |  |  | | --- | |  | | |  | | --- | | Jegyzet létrehozásának dátuma |  |  | | --- | |  | |
| TripId | integer | Kapcsolódó utazás azonosítója (kapcsolódik a Trip.Id mezőhöz) |

Ez a szerkezet lehetővé teszi, hogy a felhasználók utazásaikhoz egyéni **jegyzeteket készítsenek és azokat időrendi sorrendben megőrizzék**.

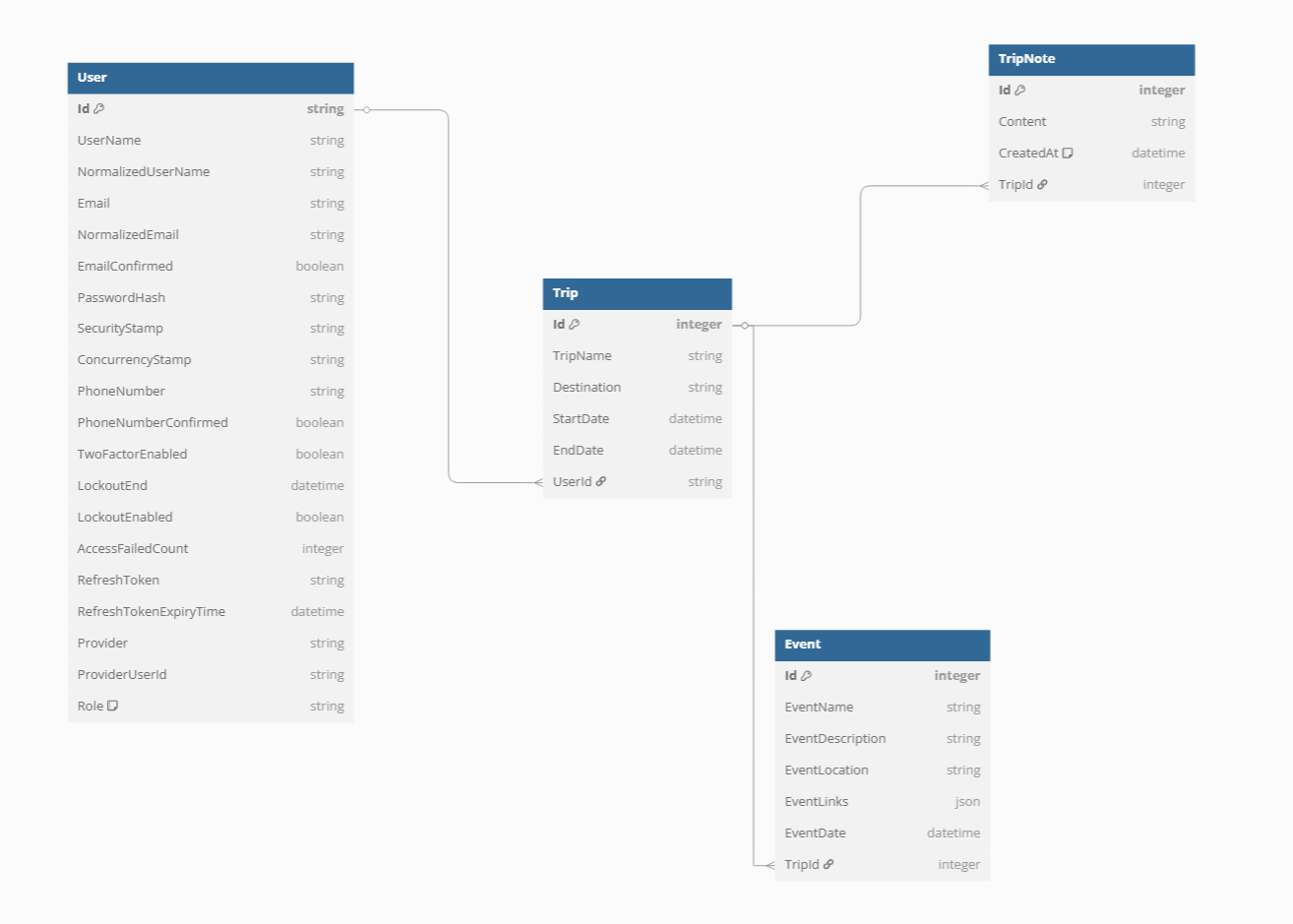
### 2. Táblák közötti kapcsolatok

Az adatbázis **relációs szerkezete** az alábbi kapcsolatokat tartalmazza:

* **User - Trip (1:N kapcsolat)**: Egy felhasználó több utazást is létrehozhat, de egy utazás csak egy felhasználóhoz tartozhat.
* **Trip - Event (1:N kapcsolat)**: Egy utazáshoz több esemény tartozhat, de egy esemény csak egy adott utazás része lehet.
* **Trip - TripNote (1:N kapcsolat)**: Egy utazáshoz több jegyzet is rögzíthető, de egy jegyzet csak egy adott utazásra vonatkozhat.

Ezek a kapcsolatok biztosítják az **adatok integritását** és lehetővé teszik a **hatékony lekérdezéseket** az egyes entitások között.

### 3. Adatbázis diagram

Az adatbázis vizuális szemléltetéséhez az alábbi diagram készült, amely ábrázolja a táblák közötti kapcsolatokat:

### 4. Összegzés

A kialakított adatmodell egy **strukturált, normalizált és jól skálázható** adatbázist biztosít az alkalmazás számára. A táblák közötti kapcsolatok biztosítják az **adatok konzisztenciáját**, miközben a JSON típusú mezők lehetőséget adnak a **rugalmasabb adattárolásra**.

Ez a szerkezet lehetővé teszi a hatékony **felhasználókezelést, utazásszervezést, eseménykezelést és egyéni jegyzetek mentését** egyetlen átlátható adatbázison belül.

## A fejlesztés során használt nevezetes algoritmusok

### GenerateToken(User user) Metódus

Ez a metódus egy JSON Web Tokent (JWT) generál, amely a felhasználó hitelesítésére és engedélyezésére szolgál. A JWT-t a JwtSecurityTokenHandler osztály segítségével hozza létre. Az alábbi lépéseken keresztül történik a token előállítása:

* T**oken Handler Inicializálása**: A JwtSecurityTokenHandler osztály használatával történik a token kezelésének előkészítése. Ez felelős a JWT létrehozásáért és a benne lévő adatok aláírásáért.
* **Titkos Kulcs Generálása**: A metódus először egy titkos kulcsot hoz létre, amelyet a SymmetricSecurityKey osztály segítségével inicializálunk. Ez a kulcs biztosítja a titkosítást és a token érvényességét. A kulcsot egy UTF8 karakterkódolással rendelt titkos kulcsként adja meg, amely alapértelmezetten elég hosszú a biztonságos titkosításhoz.
* **Token Leíró (Token Descriptor)**: A tokenhez tartozó SecurityTokenDescriptor objektum határozza meg a létrehozandó token paramétereit. Ezek közé tartoznak a felhasználó azonosítói (felhasználónév, ID és szerep), a lejárati idő (Expires), valamint a hitelesítéshez szükséges aláírási hitelesítő adatok (SigningCredentials), amely a HMAC SHA-256 titkosítást alkalmazza. A tokenben szereplő Issuer és Audience paraméterek biztosítják, hogy a token csak az adott forrásoktól (localhost) érvényes.
* **Token Generálása és Visszaadása**: A generált JWT token az WriteToken metódussal kerül visszaadásra, amely a token szöveges reprezentációját adja meg, amelyet később a kliens számára továbbíthatunk a felhasználói hitelesítéshez.

**Főbb Paraméterek és Funkciók:**

* **Claims**: Az adatok, amiket a token tartalmaz, például a felhasználó azonosítói (név, ID, szerep), amelyek az authentikáció és az engedélyezés során használhatók.
* **Expires**: A token lejárati ideje, amely meghatározza, hogy mikor válik érvénytelenné. Ez az érték a biztonságot szolgálja, mivel a tokenek idővel lejárnak.
* **Issuer és Audience**: A forrás és a célcsoport, akik számára a token érvényes.

### GenerateRefreshToken() Metódus

A GenerateRefreshToken metódus egy új frissítő tokent generál, amelyet a felhasználói hitelesítési rendszer hosszabb ideig tartó autentikálási mechanizmusainak részeként alkalmaznak. A frissítő token célja, hogy új hozzáférési tokeneket generáljon, amikor az előző token lejár, ezzel lehetővé téve a felhasználói session folytatását.

* **Random Byte Generálás**: A metódus először egy véletlenszerű bájtsorozatot generál 32 bájt hosszúságban a RandomNumberGenerator osztály segítségével, amely biztosítja a titkos kulcsok és az egyedi azonosítók megfelelő biztonságát.
* **Base64 Kódolás**: Az így előállított véletlenszerű bájtsorozatot Base64 formátumra kódolja, hogy biztonságosan továbbítható és tárolható legyen. A Base64 kódolás segít a nyers bináris adatok biztonságos szöveges reprezentációjának kialakításában.
* **Frissítő Token Visszaadása**: A kódolt frissítő token a metódus visszatérési értékeként jelenik meg, amely a kliens számára elérhetővé válik, hogy új hozzáférési tokenek generálására használhassa.

**Főbb Paraméterek és Funkciók:**

* **Véletlenszerűség**: A véletlenszerűség biztosítása kulcsfontosságú a frissítő token biztonsága érdekében, mivel ez megakadályozza a támadók számára a token előre kitalálását.
* **Base64**: Az így generált token könnyen továbbítható a hálózaton, mivel szöveges formában kerül elküldésre.

### Összegzés

Mindkét metódus alapvető szerepet játszik a modern hitelesítési rendszerekben, ahol a felhasználó számára egy JWT token biztosítja a hozzáférést a rendszerhez, míg a frissítő token biztosítja, hogy a felhasználói session hosszú távon fenntartható maradjon anélkül, hogy folyamatosan újra be kellene jelentkezniük. Az ilyen típusú megoldások növelik a rendszer biztonságát és kényelmét, mivel a felhasználók nem kénytelenek minden egyes alkalommal újra megadni a hitelesítő adatokat.

## Unit test

### Áttekintés

Ez a dokumentum a JourneyPalAdmin alkalmazás InputValidator és UserHelper segédosztályainak unit tesztjeit írja le. A tesztek ellenőrzik, hogy a bemenet validáló és felhasználó összehasonlító logika megfelelően működik különböző feltételek mellett.

**Teszt Kategóriák**

**InputValidator tesztek**

* Az InputValidator statikus osztály tesztei, amely sztringek, email címek, telefonszámok és jelszavak validálását végzi.

**IsStringValid tesztek**

[Theory]

[InlineData(null, false, false)]

[InlineData("", false, false)]

[InlineData("valid", true, true)]

[InlineData("short", false, false, 6, 10)]

[InlineData("exactlength", true, true, 5, 11)]

public void IsStringValid\_ShouldValidateCorrectly(string input, bool allowNullOrEmpty, bool expected, int minLength = 0, int maxLength = int.MaxValue)

{

bool result = InputValidator.IsStringValid(input, allowNullOrEmpty, minLength, maxLength);

Assert.Equal(expected, result);

}

* **Cél:** A sztring bemenet validálása a null/üres engedélyezés, valamint minimális és maximális hossz alapján.
* **Tesztesetek:**
  + Null vagy üres sztringek false értéket adnak vissza, kivéve ha explicit engedélyezett.
  + A megfelelő hosszúságú sztringek true értéket adnak vissza.
  + A minLength-nél rövidebb vagy maxLength-nél hosszabb sztringek false értéket adnak vissza.

**IsEmailValid tesztek**

[Fact]

public void IsEmailValid\_ValidEmail\_ReturnsTrue()

{

string email = "test@example.com";

bool result = InputValidator.IsEmailValid(email);

Assert.True(result);

}

[Fact]

public void IsEmailValid\_InvalidEmail\_ReturnsFalse()

{

string email = "invalid-email";

bool result = InputValidator.IsEmailValid(email);

Assert.False(result);

}

* **Cél:** Email cím formátum ellenőrzése a .NET MailAddress osztályával.
* **Tesztesetek:**
  + Helyes formátumú email címek (pl. test@example.com) true értéket adnak vissza.
  + Helytelen formátumú email címek (pl. invalid-email) false értéket adnak vissza.

**IsPhoneNumberValid tesztek**

[Fact]

public void IsPhoneNumberValid\_ValidPhoneNumber\_ReturnsTrue()

{

string phoneNumber = "+1234567890";

bool result = InputValidator.IsPhoneNumberValid(phoneNumber);

Assert.True(result);

}

[Fact]

public void IsPhoneNumberValid\_InvalidPhoneNumber\_ReturnsFalse()

{

string phoneNumber = "abc123";

bool result = InputValidator.IsPhoneNumberValid(phoneNumber);

Assert.False(result);

}

* **Cél:** Telefonszám formátum ellenőrzése regex segítségével.
* **Tesztesetek:**
  + Helyes formátumú telefonszámok (pl. +1234567890) true értéket adnak vissza.
  + Helytelen formátumú telefonszámok (pl. abc123) false értéket adnak vissza.

**IsPasswordValid tesztek**

[Fact]

public void IsPasswordValid\_ValidPassword\_ReturnsTrue()

{

string password = "StrongPass123!";

bool result = InputValidator.IsPasswordValid(password);

Assert.True(result);

}

[Fact]

public void IsPasswordValid\_TooShortPassword\_ReturnsFalse()

{

string password = "short";

bool result = InputValidator.IsPasswordValid(password, minLength: 8);

Assert.False(result);

}

* **Cél:** Jelszó hosszának ellenőrzése.
* **Tesztesetek:**
  + A minLength követelménynek megfelelő jelszavak true értéket adnak vissza.
  + A minLength-nél rövidebb jelszavak false értéket adnak vissza.

**UserHelper tesztek**

* Az UserHelper statikus osztály tesztei, amely felhasználó validálást és összehasonlítást végez.

**IsUserValid tesztek**

[Theory]

[InlineData("validUser", "test@example.com", "+1234567890", true)]

[InlineData("", "test@example.com", "+1234567890", false)] // Üres felhasználónév

[InlineData("validUser", "invalid-email", "+1234567890", false)] // Helytelen email

[InlineData("validUser", "test@example.com", "abc123", false)] // Helytelen telefonszám

public void IsUserValid\_ShouldValidateCorrectly(string username, string email, string phone, bool expected)

{

var user = new User { UserName = username, Email = email, PhoneNumber = phone };

bool result = UserHelper.IsUserValid(user);

Assert.Equal(expected, result);

}

[Fact]

public void IsUserValid\_NullUser\_ReturnsFalse()

{

bool result = UserHelper.IsUserValid(null);

Assert.False(result);

}

* **Cél:** Egy User objektum validálása a UserName, Email és PhoneNumber tulajdonságok alapján.
* **Tesztesetek:**
  + Érvényes felhasználók (minden mező helyes) true értéket adnak vissza.
  + Érvénytelen felhasználók (bármely mező helytelen) false értéket adnak vissza.
  + Null felhasználók false értéket adnak vissza.

**AreUsersEqual tesztek**

[Fact]

public void AreUsersEqual\_SameUsers\_ReturnsTrue()

{

var user1 = new User { UserName = "user1", Email = "user1@example.com", PhoneNumber = "+1234567890" };

var user2 = new User { UserName = "user1", Email = "user1@example.com", PhoneNumber = "+1234567890" };

bool result = UserHelper.AreUsersEqual(user1, user2);

Assert.True(result);

}

[Fact]

public void AreUsersEqual\_DifferentUsers\_ReturnsFalse()

{

var user1 = new User { UserName = "user1", Email = "user1@example.com", PhoneNumber = "+1234567890" };

var user2 = new User { UserName = "user2", Email = "user2@example.com", PhoneNumber = "+9876543210" };

bool result = UserHelper.AreUsersEqual(user1, user2);

Assert.False(result);

}

* **Cél:** Két User objektum összehasonlítása a tulajdonságaik alapján.
* **Tesztesetek:**
  + Azonos tulajdonságokkal rendelkező felhasználók true értéket adnak vissza.
  + Különböző tulajdonságokkal rendelkező felhasználók false értéket adnak vissza.

### Összefoglalás

Ezek a unit tesztek lefedik az InputValidator és UserHelper osztályok alapvető funkcionalitását, biztosítva a megbízható validálást és összehasonlítást. A tesztek tartalmaznak szélsőséges eseteket is (pl. null bemenet, helytelen formátumok), hogy a valós felhasználási esetekben is megbízhatóan működjenek.

## Tesztelői hozzáférés biztosítása

A fejlesztés során egy előre meghatározott tesztfelhasználói fiókot hoztam létre, amely lehetővé teszi a rendszer funkcióinak kipróbálását anélkül, hogy manuálisan új felhasználót kellene regisztrálni minden tesztelési ciklushoz.

**Tesztfelhasználó adatai:**

* Felhasználónév: admin
* Jelszó: asdASD123#

**Tesztelési jogosultságok:**

* Az admin felhasználó a rendszerben adminisztrátori jogosultságokkal rendelkezik, így teljes hozzáférést kap a rendszer minden funkciójához.
* A felhasználói adatok és jogosultságok az adatbázisban kerülnek tárolásra, és az autentikáció során ellenőrzésre kerülnek.
* A bejelentkezés után az adminisztrátori felhasználó elérheti a kezelőfelület minden elemét, például az események, utak és jegyzetek menedzselését.

**Tesztfelhasználó létrehozása és kezelése:**

A tesztfelhasználó regisztrálása és hitelesítése az adatbázisban történik. Amennyiben a tesztelői környezetben szükség van a felhasználó automatikus létrehozására (például az első indításkor), az alábbi lépéseket követtem:

1. **Admin felhasználó létrehozása az adatbázisban:**
   * A fejlesztési és tesztelési fázisban a rendszer inicializálása után kell az **api/auth/admin-register** végponttal regisztrálni admin felhasználót
2. **Jelszó biztonságos tárolása:**
   * A jelszót **nem tárolom egyszerű szövegként**, hanem erős hash algoritmussal (pl. **bcrypt**) titkosítom a biztonság érdekében.
   * A hitelesítés során a rendszer a bevitt jelszót a tárolt hash értékkel veti össze.
3. **Tesztelési környezet beállítása:**
   * A tesztelési célokra biztosított belépési adatok kizárólag **lokális vagy tesztkörnyezetben** érhetők el.
   * Éles környezetben ez a fiók **nem kerül automatikusan létrehozásra**, vagy további biztonsági intézkedésekkel védett.

### Összegzés

A tesztelők számára elérhetővé tett adminisztrátori fiók lehetőséget biztosít az alkalmazás funkcionalitásának teljes körű kipróbálására. A hitelesítési rendszer megfelelően biztosítja, hogy a belépési adatok biztonságos módon kezelődjenek, és a jogosultságkezelés megfelelően szabályozza a hozzáféréseket.

## Továbbfejlesztési lehetőségek

### 1. Mesterséges intelligencia alapú programajánló

Cél:

A felhasználók utazási célpontjai és preferenciái alapján egy intelligens ajánlórendszer segítségével automatikusan javasoljon programokat, látnivalókat és eseményeket.

**Megvalósítás:**

* Az AI algoritmus a felhasználó által megadott utazási célpont, az utazás időpontja, valamint a korábbi preferenciák alapján generál személyre szabott javaslatokat.
* A rendszer adatokat gyűjt különböző forrásokból, például turisztikai adatbázisokból, helyi eseménynaptárakból, vagy akár korábbi felhasználói értékelésekből.
* A gépi tanulás segítségével a rendszer képes lenne felismerni a felhasználó érdeklődési körét (pl. történelmi helyszínek, múzeumok, természetjárás, gasztronómia) és ennek megfelelően ajánlásokat tenni.
* Az ajánlások dynamikusan frissülnének, így a rendszer figyelembe venné az aktuális eseményeket és szezonális programokat.

**Technológiai megvalósítás:**

* Adatgyűjtés: Külső API-k (pl. Google Places API, TripAdvisor API, Eventbrite API) használata
* Gépi tanulás modell: Python alapú Natural Language Processing (NLP) és Recommender System (pl. TensorFlow, Scikit-learn)
* Backend integráció: Az ajánlások beépítése a meglévő adatbázisba és API-ba

**Felhasználói előnyök:**

* Automatizált programajánlás személyre szabott élményekkel
* Naprakész információk az elérhető eseményekről és helyszínekről
* Intelligens javaslatok az érdeklődési kör alapján

### 2. Időjárás-előrejelző integráció

Cél:

Az alkalmazásba beépített időjárásfigyelő rendszer biztosítaná, hogy a felhasználók előre láthassák az időjárási körülményeket az utazásuk során.

**Megvalósítás:**

* Az alkalmazás az utazás időpontja és helyszíne alapján valós idejű időjárás-adatokat jelenítene meg.
* Az előrejelzést dinamikusan frissítené, így a felhasználó naponta látná, hogy milyen idő várható.
* Lehetőség lenne figyelmeztetések beállítására (pl. viharjelzés, extrém hőmérséklet).
* Az időjárási adatokhoz kapcsolódóan az alkalmazás javasolhatna megfelelő ruházati tippeket vagy alternatív programokat rossz idő esetére.

**Technológiai megvalósítás:**

* Időjárási adatok forrása: OpenWeatherMap API, WeatherAPI, vagy AccuWeather API
* Adatok feldolgozása: JSON alapú adatlekérés és a UI-ra való megjelenítés
* Értesítések és figyelmeztetések: Push notification implementációja mobil és webes felületekre

**Felhasználói előnyök:**

* Pontos időjárás-előrejelzés az utazás minden napjára
* Figyelmeztetések szélsőséges időjárási viszonyokra
* Alternatív tervek készítése az aktuális időjárás alapján

### Összegzés

Ez a két fejlesztés jelentős mértékben növelné az alkalmazás **felhasználóbarátságát és funkcionalitását**. Az **AI alapú ajánlórendszer** személyre szabott programokat kínálna, míg az **időjárásfigyelő rendszer** segítene az utazás pontosabb megtervezésében.

## Szerveroldali API-k és vezérlők bemutatása

A rendszer szerveroldali architektúrája RESTful API-k segítségével biztosítja a kliensoldali alkalmazások számára az adatokhoz való hozzáférést és azok kezelését. Az API végpontok (endpoints) a Web API Controller-ekben kerültek implementálásra, amelyek felelősek az üzleti logika végrehajtásáért, az adatbázissal történő kommunikációért és a HTTP-kérések feldolgozásáért.

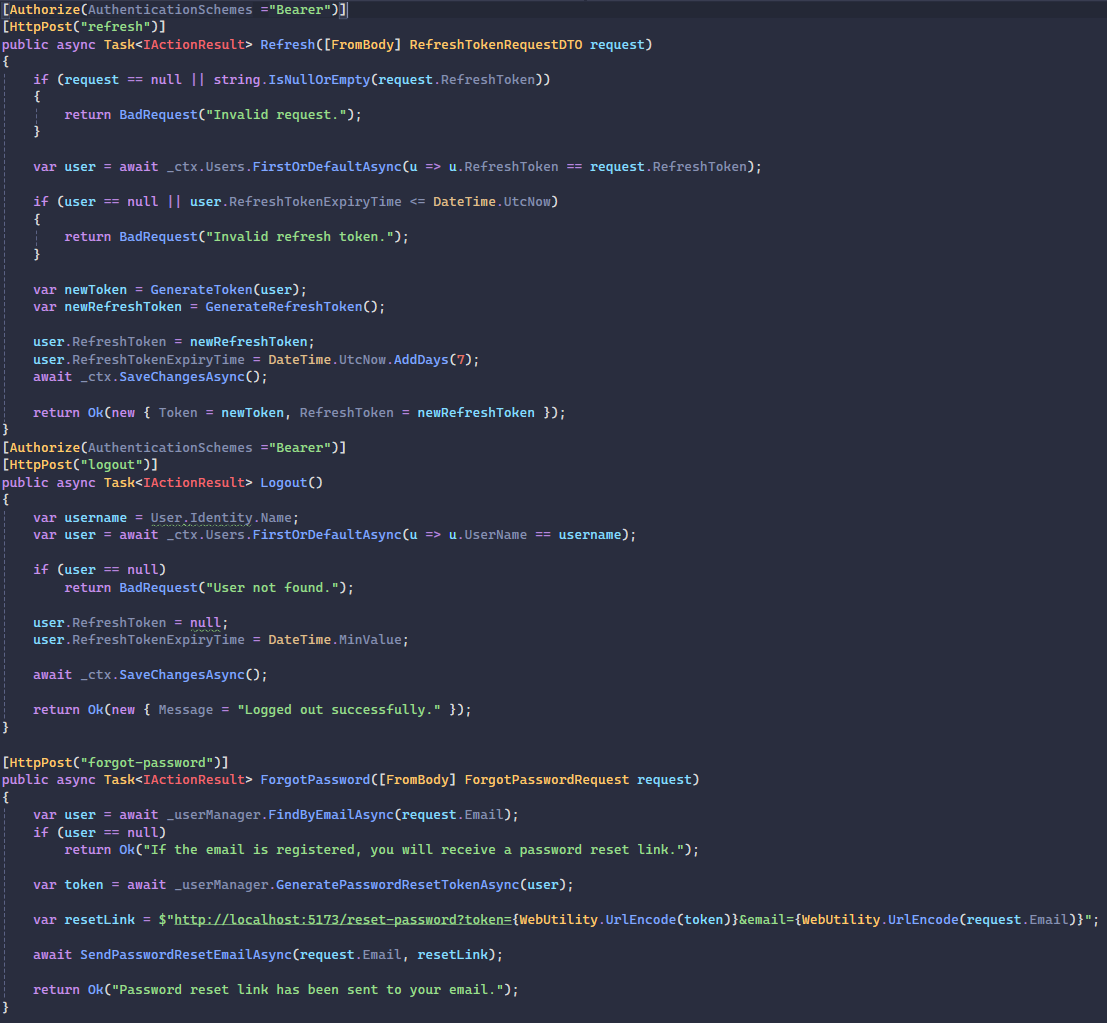
A következő fejezetekben részletesen bemutatom az egyes API vezérlőket, azok feladatait, valamint a hozzájuk tartozó végpontokat, a támogatott HTTP-metódusokat és a válaszstruktúrákat.

### AuthController metódusai

Konstruktor és a register API végpont:

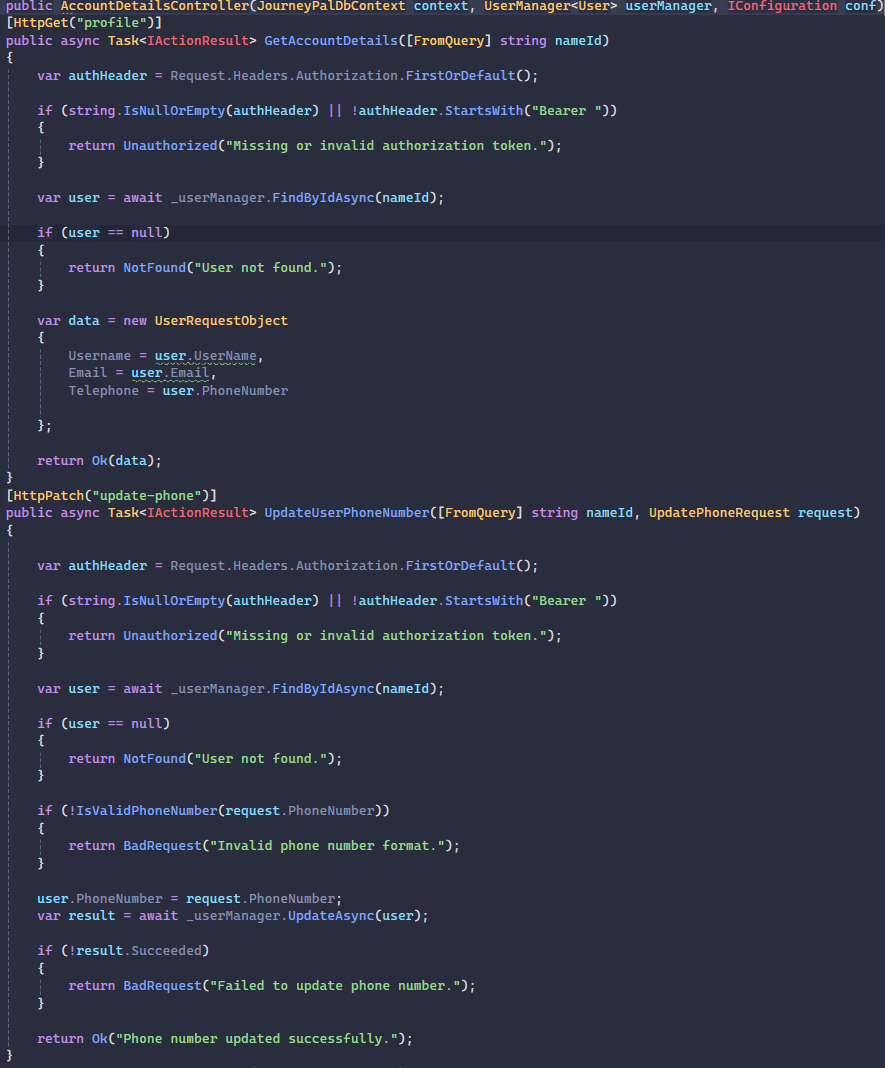


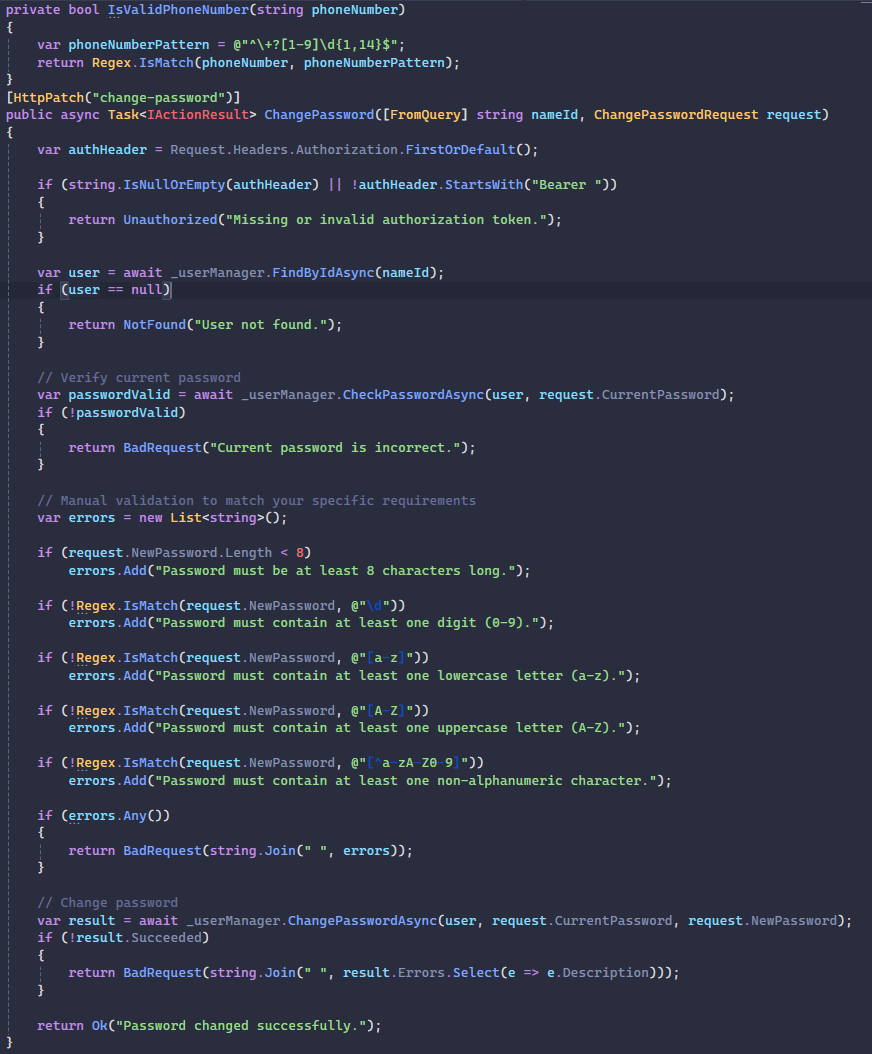
Login:

Refresh, Logout és ForgotPassword végpontok:

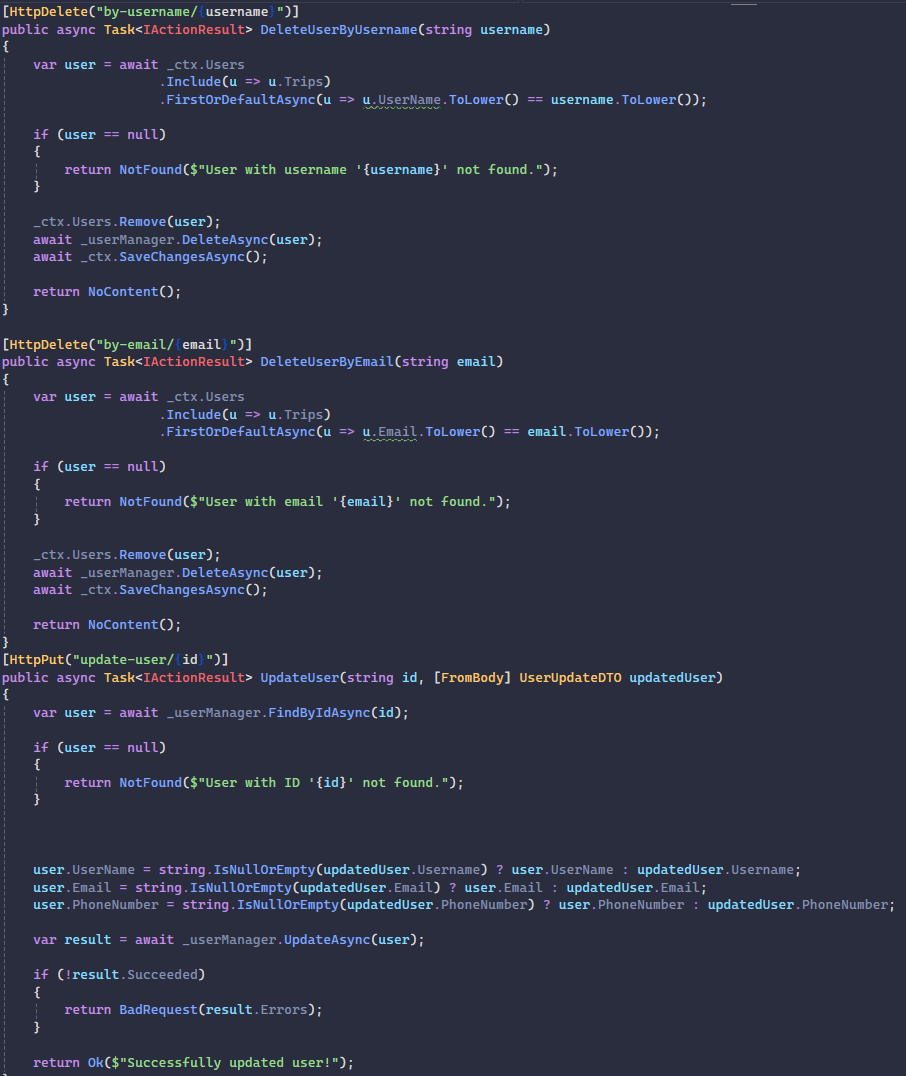
ResetPassword végpont:

### AccountDetailsController metódusai

Profil és UpdatePhoneNumber végpont:

UpdatePassword végpont:

### AdminController metódusai



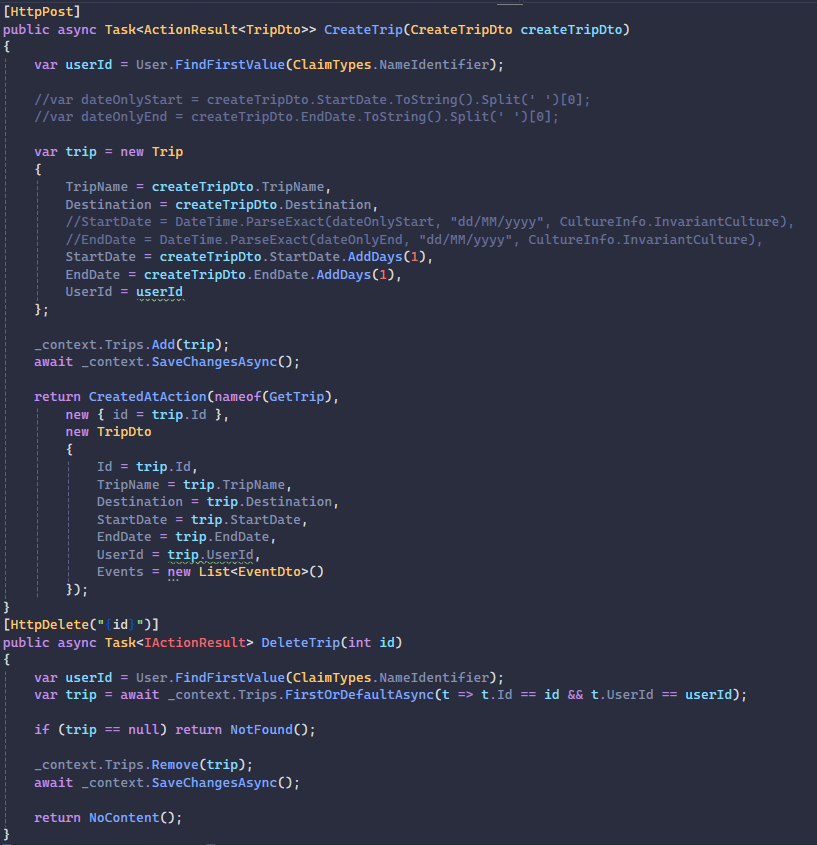


### EventController metódusai

# 



### TripController metódusai



# 

# 

# Adminisztrációs Felület Bemutatása

Az alkalmazás **adminisztrációs felülete** egy dedikált kezelőfelületet biztosít az adminisztrátorok számára, amely lehetővé teszi a regisztrált felhasználók adatainak kezelését. Az adminisztrátor kizárólag olyan műveleteket hajthat végre, amelyek a rendszer működésének fenntartásához szükségesek, és nem fér hozzá az egyéni utazási adatokhoz, mivel ezek a felhasználók személyes adatai közé tartoznak.

**Adminisztrátor jogosultságai**

Az adminisztrációs felületen keresztül a rendszergazda az alábbi műveleteket végezheti el:

* **Felhasználók listázása**:  
  Az adminisztrátor áttekintheti a rendszerben regisztrált felhasználók listáját, amely tartalmazza az alapvető adatokat, például a felhasználónév, e-mail cím és a fiók státusza.
* **Felhasználói adatok módosítása**:  
  A regisztrált felhasználók adatai szerkeszthetők az adminisztrációs felületen keresztül. A módosítható adatok a következők:
  + Telefonszám
  + E-mail cím
  + Felhasználónév
  + Jelszó (biztonságos módon, titkosított formában)
* **Felhasználói fiókok törlése**:  
  Ha egy felhasználó megszegi a felhasználási feltételeket vagy más okból kifolyólag szükségessé válik a fiókja törlése, az adminisztrátor jogosult a fiók végleges eltávolítására a rendszerből. A törlés előtt egy megerősítési folyamat zajlik le annak érdekében, hogy elkerülhetők legyenek a véletlen műveletek.

**Továbbfejlesztési lehetőségek**

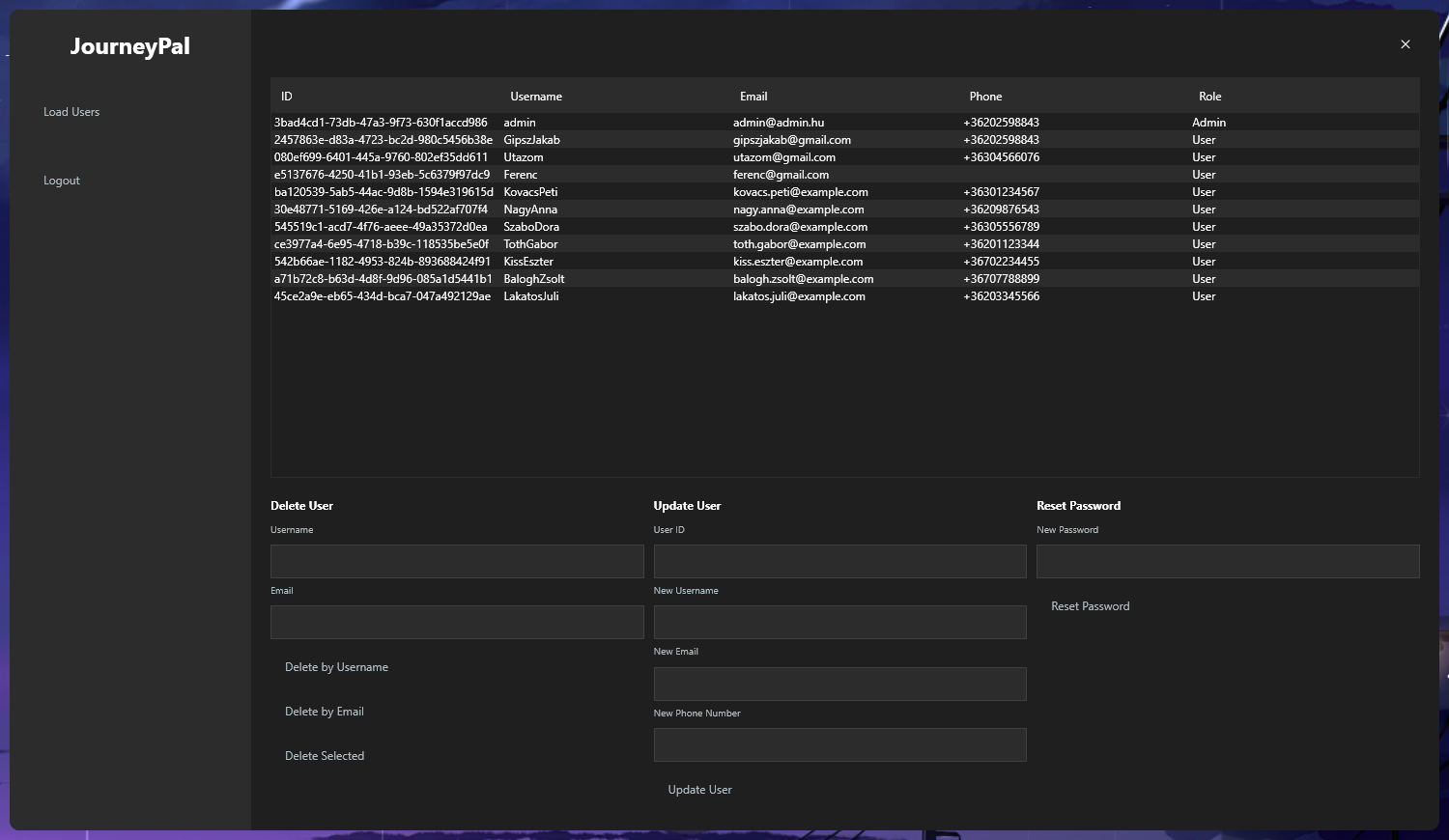
A jövőbeli fejlesztések során az adminisztrációs felület tovább bővülhet olyan funkciókkal, amelyek segítik az alkalmazás teljesítményének és használatának monitorozását:

* **Forgalmi statisztikák és analitika**:  
  Az adminisztrátor számára egy vizuálisan megjelenített statisztikai felület biztosítható, amelyen napi, heti és havi bontásban követheti a rendszer aktivitását. A grafikonok és diagramok segítenek átlátni a felhasználói aktivitás alakulását, például:
  + Hány új regisztráció történt adott időszakban
  + Milyen gyakran lépnek be a felhasználók az alkalmazásba
  + Milyen eszközökről történik a bejelentkezés

Fontos megjegyezni, hogy az adminisztrátor **nem fér hozzá a felhasználók privát utazási adataihoz**, mivel ezek kizárólag a felhasználó saját tulajdonát képezik, és az alkalmazás szigorúan betartja az adatvédelmi irányelveket.

Ezzel az adminisztrációs felület átlátható, hatékony és biztonságos módon teszi lehetővé a rendszer működésének ellenőrzését és fenntartását.

Képek az adminisztrációs felületről:



# Felhasználói dokumentáció

## A program bemutatása

**A JourneyPal bemutatása a felhasználó számára(16)**

A **JourneyPal** egy modern és felhasználóbarát utazástervező alkalmazás, amely segít az utazni vágyóknak megszervezni álomútjaikat. Az alkalmazás lehetővé teszi, hogy a felhasználók **új utazásokat hozzanak létre, előre megadott úti célok közül válasszanak**, vagy akár saját úti célokat is megadjanak. Az utazásokhoz események adhatók hozzá, amelyekkel részletes programtervet alakíthatnak ki, így minden napjuk előre megtervezett és szervezett lehet.

A JourneyPal célja, hogy egyszerű, könnyen kezelhető megoldást nyújtson az utazások megtervezéséhez, legyen szó rövid hétvégi kiruccanásról vagy hosszabb világjárásról. A program **átlátható felülettel és intuitív funkciókkal** segíti a felhasználókat, így akár kezdők, akár tapasztalt utazók is gyorsan és hatékonyan használhatják.

## Rendszerkövetelmények

**A program használatához szükséges hardvereszközök**

A **JourneyPal** egy modern, böngészőalapú alkalmazás, amelyet úgy terveztünk, hogy **bármilyen eszközről könnyedén elérhető és használható legyen**. A fejlesztés során kiemelt figyelmet fordítottunk arra, hogy az oldal **mobilbarát** legyen, így a felhasználók **okostelefonon, tableten és asztali számítógépen egyaránt** kényelmesen böngészhessék és használhassák a programot.

Mivel a JourneyPal **webalapú alkalmazás**, nincs szükség semmilyen speciális hardveres követelményre vagy telepítésre. Elég egy **modern webböngészővel rendelkező eszköz**, amely támogatja a legújabb technológiákat, így a felhasználók gyorsan és egyszerűen tervezhetik meg utazásaikat.

A programot kifejezetten **reszponzív felépítéssel** készítettük, így a különböző képernyőméreteken egyaránt átlátható és felhasználóbarát marad. Az asztali verzió lehetőséget biztosít az alaposabb tervezésre, míg a mobilnézet ideális a gyors módosításokra és az utazás közbeni információk elérésére.

Összességében a JourneyPal bárhol, bármikor elérhető, és minden olyan felhasználónak szól, aki szeretné **könnyedén és hatékonyan megszervezni az utazásait** – legyen szó egy rövid hétvégi kiruccanásról vagy egy hosszabb, összetettebb utazásról.

## A program használatához szükséges szoftverek és telepítésük

A **JourneyPal** egy webalapú alkalmazás, amelynek futtatásához néhány előzetes szoftveres beállítás szükséges. A következő lépések részletesen ismertetik, hogyan lehet a programot elindítani egy olyan felhasználó számára is, aki nem rendelkezik fejlesztői tapasztalattal.

**PaperCut SMTP szerver telepítése és futtatása**

A **PaperCut SMTP szerver** egy egyszerű eszköz, amely segít a helyi e-mailek küldésében és fogadásában. A JourneyPal működéséhez szükséges, hogy ez a szerver futásban legyen.

**Lépések a PaperCut SMTP szerver telepítéséhez:**

1. **Töltse le a PaperCut SMTP szervert** az alábbi hivatalos weboldalról:  
   <https://github.com/ChangemakerStudios/Papercut-SMTP/releases>
2. **Válassza ki a megfelelő verziót**, amely kompatibilis az operációs rendszerével (Windows, macOS vagy Linux).
3. **Csomagolja ki a letöltött fájlt** egy tetszőleges helyre a számítógépén.
4. **Indítsa el a szervert** az alábbi módon:
   * Windows esetén: **Nyissa meg a kicsomagolt mappát, és kattintson duplán a Papercut.SMTP.exe fájlra.**
   * macOS vagy Linux esetén: **Nyisson egy terminált, navigáljon a mappába, majd futtassa a ./Papercut-SMTP parancsot.**
5. A szerver **alapértelmezett beállításokkal** fog futni, ami azt jelenti, hogy **nincs szükség konfigurálásra**. Az eszköz egy egyszerű kezelőfelületen keresztül megjeleníti a beérkező e-maileket, amelyeket a rendszer küld ki.

**Visual Studio Code és Node.js telepítése**

A JourneyPal fejlesztéséhez és futtatásához szükség van a **Visual Studio Code (VS Code)** fejlesztői környezetre és a **Node.js** keretrendszerre.

**Node.js telepítése.**

A Visual Studio Code-on belül npm install majd npm run dev parancsokkal érhetjük el az offline környezetben való tesztelést, és néhány perc alatt teljes mértékben működésbe hozhatja az alkalmazást.

## A program funkcióinak ismertetése

**A JourneyPal program használatának részletes ismertetése**

A **JourneyPal** egy felhasználóbarát, webalapú alkalmazás, amely segít az utazások könnyed megtervezésében. A program célja, hogy segítséget nyújtson azoknak, akik szervezni szeretnék a következő nagy kalandjukat, és megkönnyítse az utazások napi eseményeinek kezelését. Az alábbiakban részletesen bemutatom a program használatát, lépésről lépésre.

**1. Regisztráció és bejelentkezés**

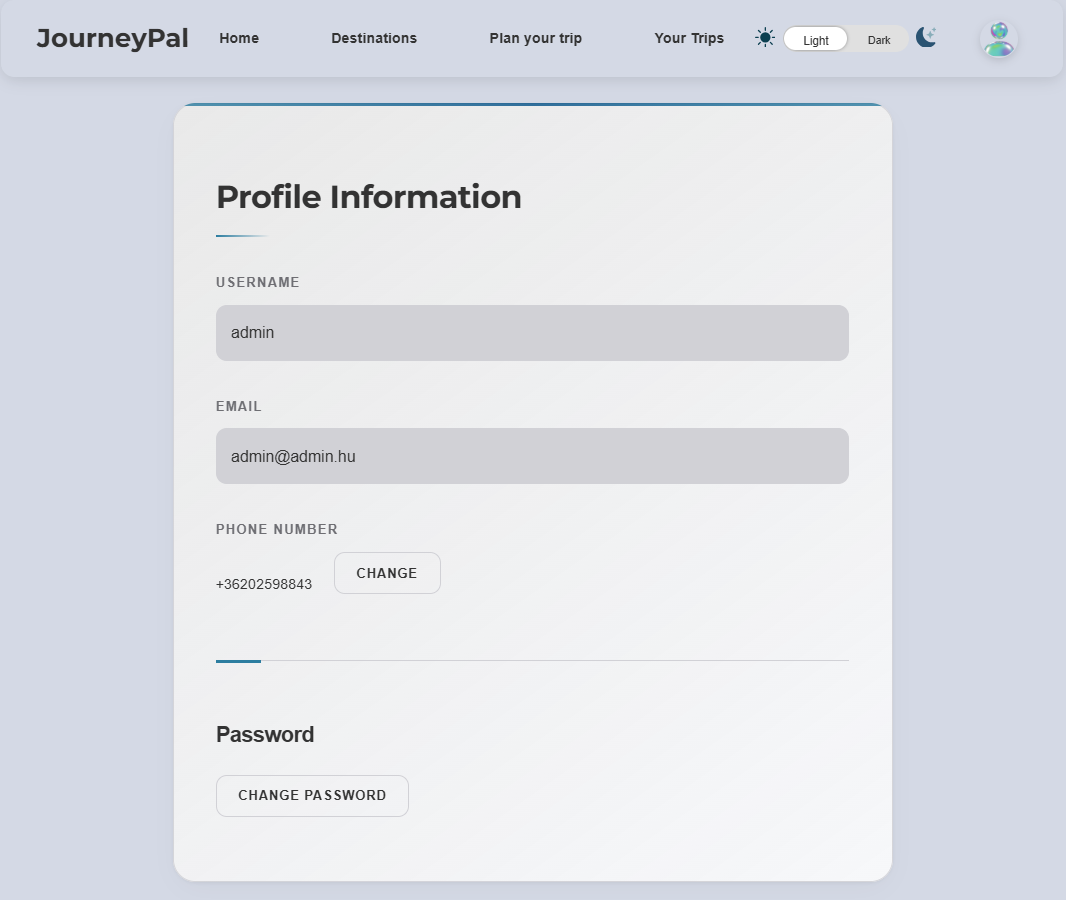
Első alkalommal a felhasználónak regisztrálnia kell a programba, hogy személyre szabott élményt kapjon és saját utazásokat hozhasson létre.

**Regisztráció:**

1. Nyissa meg a JourneyPal weboldalt a böngészőjében.
2. Kattintson a profil ikonra majd a**"Sign Up"** gombra a kezdőoldalon.
3. Adja meg nevét, e-mail címét, és válasszon egy erős jelszót majd ismételje meg.

**Bejelentkezés:**

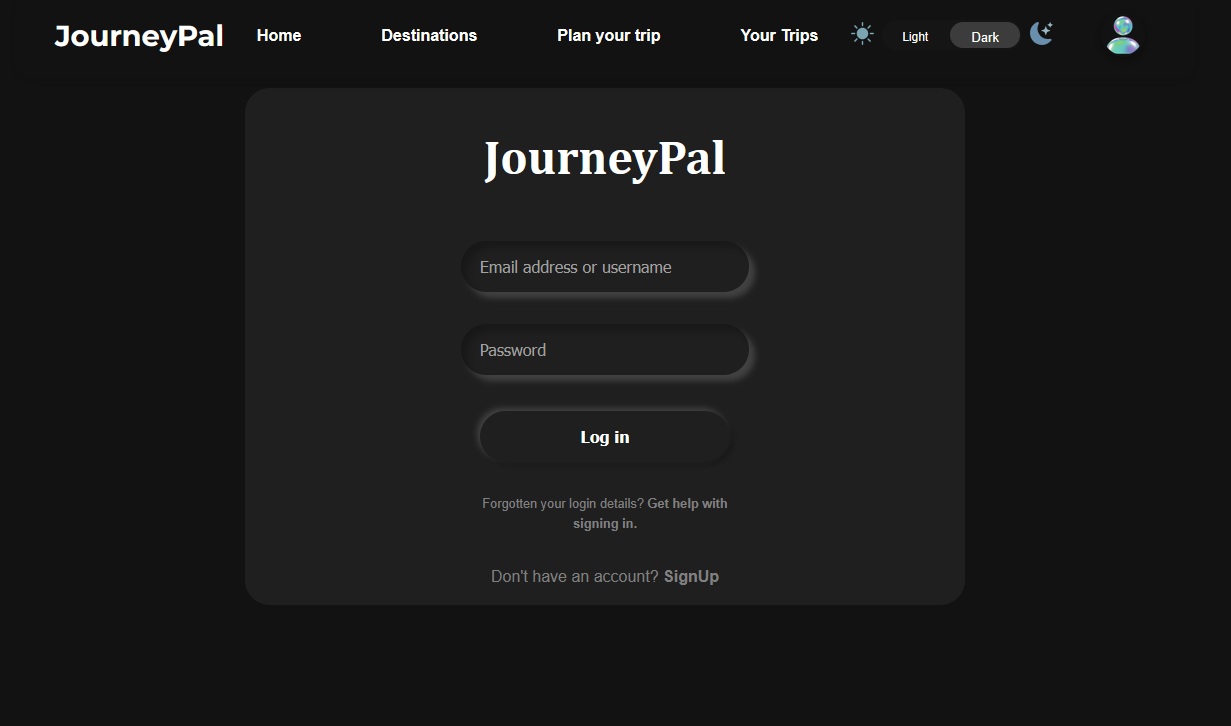
1. Miután sikeresen regisztrált, lépjen vissza a kezdőoldalra és kattintson a profil ikonra majd a **"Log In"** gombra.
2. Adja meg regisztrált e-mail címét vagy felhasználónevét és jelszavát.
3. A sikeres bejelentkezés után az oldal a felhasználó profilját fogja megjeleníteni.



1. Kattintson a **"Your Trips"** gombra, hogy hozzáférjen az utazásaihoz.

**Sikertelen regisztráció vagy bejelentkezés, majd többszöri újra próbálás esetén a következő email címen érheti el a fejlesztőket:**

**support.journeypal@proton.me**

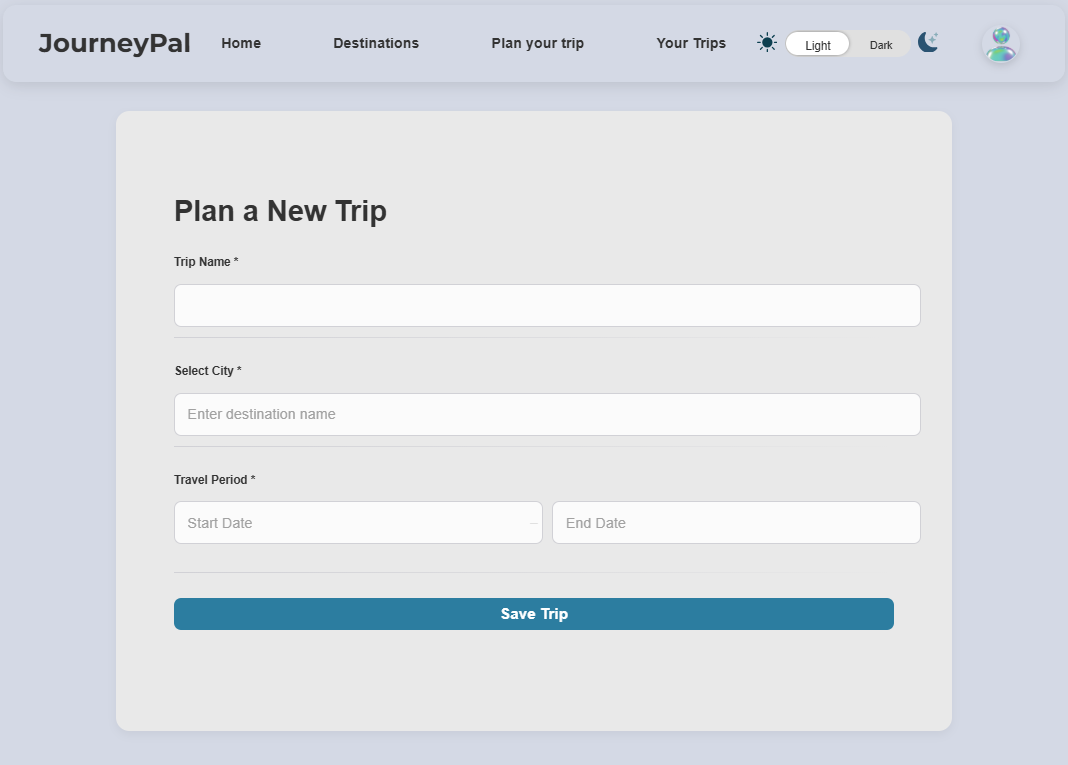
1. 

**2. Utazás létrehozása**

Miután bejelentkezett, elkezdheti saját utazásainak tervezését. Az alábbi lépések segítenek az utazás létrehozásában.

**Lépések:**

1. A kezdőoldalon található **"Plan your trip"** gombra kattintva kezdje el a folyamatot.
2. Töltse ki a szükséges adatokat:
   * **Trip Name**: Adjon egy könnyen megjegyezhető nevet az utazásának.
   * **Úti cél**: Válasszon egy előre definiált úti célt, például **Tokió**, **Santorini** vagy **Faroe-szigetek**, vagy írjon be egy saját helyszínt, ha a kívánt hely nem szerepel a listán.
   * **Travel Period**: Válassza ki a kívánt kezdő- és befejező dátumot.
3. Kattintson a **"Save Trip"** gombra.

****A rendszer automatikusan elmenti az új utazást, amelyet később szerkeszthet és kiegészíthet eseményekkel.

**3. Események hozzáadása az utazáshoz**

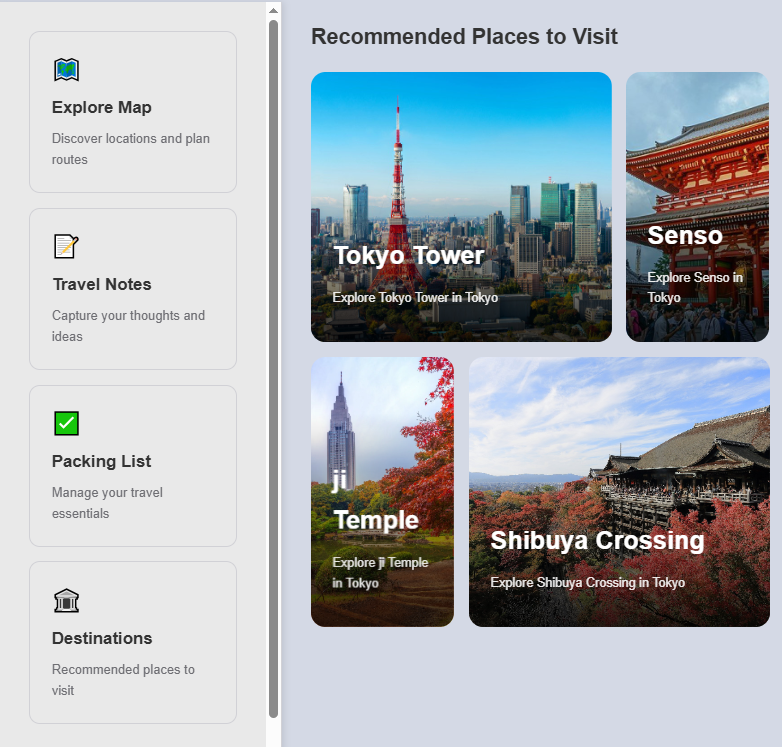
Miután létrehozta az utazását, további eseményeket adhat hozzá, hogy részletesebben megtervezze a napjait.

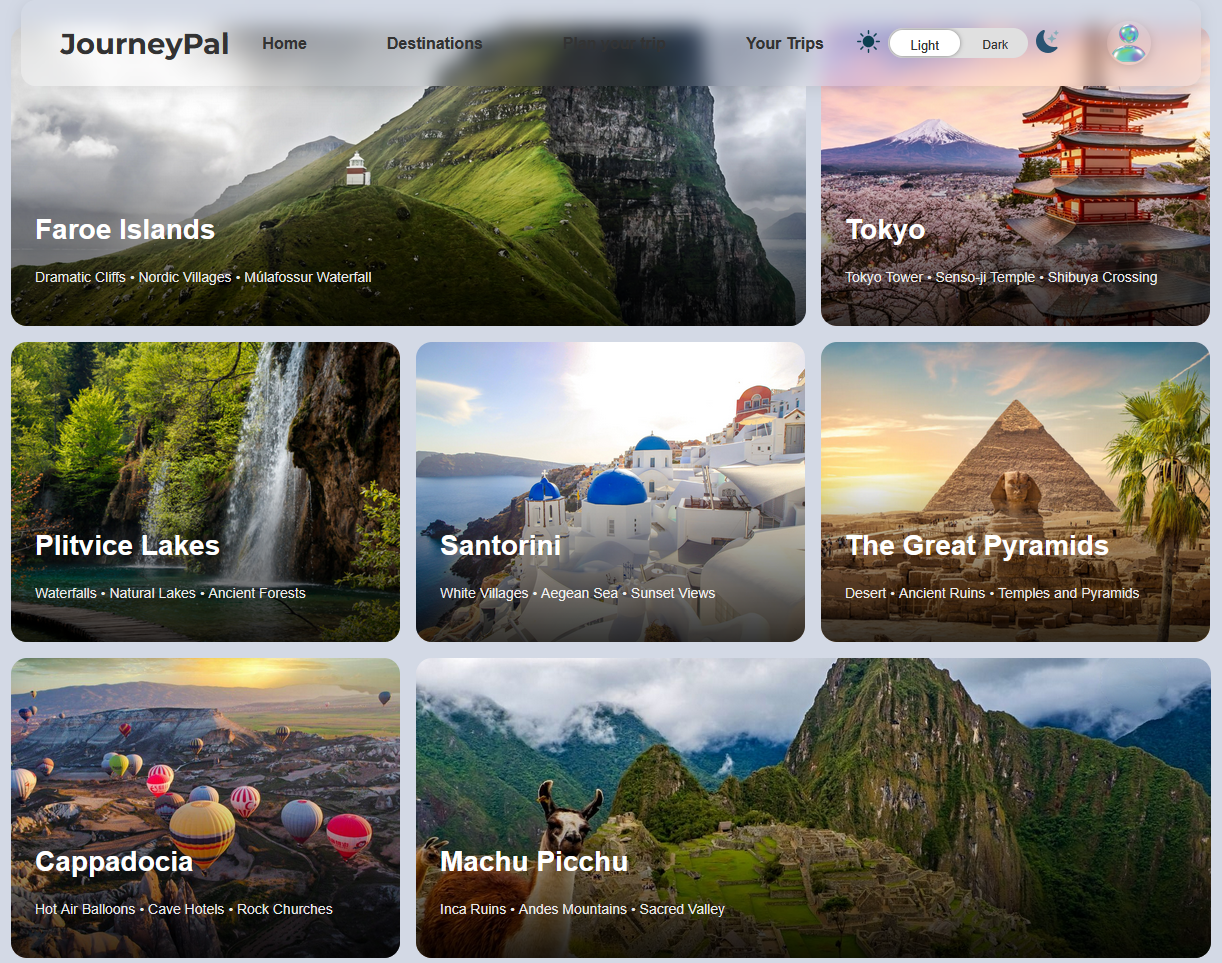
**Lépések:**

1. A **"My Trips"** (Saját utazások) menüpont alatt válassza ki az előzőleg létrehozott utazását.
2. A részletes nézetben válassza ki a **"Create Event"** (Esemény hozzáadása) gombot.
3. Töltse ki az esemény részleteit:
   * **Event name**: Nevezze el az eseményt, például "Látogatás a Tokiói Toronyhoz".
   * **Start time**: Válassza ki, melyik időpontra szól az esemény (meghatározott időpont).
   * **Description**: Adjon meg egy rövid leírást az eseményről, például: "Megtekintjük a Tokiói Torony panorámáját."
   * **Location**: Adja meg a pontos helyszínt (pl. Tokió, Shiba Park).
   * **Links**: Kedv szerint csatoljon egy létező oldal linkjét.
4. Kattintson a **"Save Event"** gombra.

Az esemény ekkor automatikusan hozzáadódik az utazásához és az adott naphoz.

**4. Előre definiált úti célok és egyéni helyszínek**

A JourneyPal rendelkezik néhány **előre definiált úti céllal**, mint például **Tokió, Santorini és Faroe-szigetek**, amelyekből választhat, ha nem biztos abban, hogy milyen helyszínt válasszon.



Ha olyan helyet szeretne hozzáadni, amely nem szerepel az előre definiált listán, egyszerűen **beírhatja** a kívánt úti célt a helyszín mezőbe. A rendszer automatikusan menti az új helyszínt, és hozzáadja a triphez.

**5. Adminisztrátori lehetőségek**

Amennyiben adminisztrátori hozzáféréssel rendelkezik, lehetősége van az alábbi funkciók használatára:

* **Felhasználói fiókok kezelése**: Lehetősége van új felhasználók hozzáadására, jogosultságok módosítására, valamint inaktiválásukra.
* **Események módosítása**: Az adminisztrátorok bárkinek az eseményeit módosíthatják vagy törölhetik.
* **Események statisztikái**: Megtekintheti a felhasználók által hozzáadott események statisztikai adatait.

Ez a részletes ismertető segít abban, hogy minden felhasználó könnyedén elkezdhesse a JourneyPal használatát, és kihasználja a program minden funkcióját az utazásai megtervezésére.

## Regisztrációs követelmények

A JourneyPal használatához nem szükséges azonnal regisztrálni. Az alábbi funkciókat bárki, regisztráció nélkül, szabadon elérheti:

1. **Kezdőoldal megtekintése**
   * A felhasználók bármilyen eszközről, regisztráció nélkül is hozzáférhetnek a JourneyPal főoldalához, amely tartalmazza az alapvető információkat a programról, a legnépszerűbb úticélokat, valamint a program által kínált szolgáltatásokat.
   * Az oldal áttekinthető és informatív, a felhasználóknak lehetőségük van bepillantani a program fő funkcióiba, anélkül hogy regisztrálniuk kellene.
2. **Úti célok böngészése**
   * A főoldalon a felhasználók böngészhetnek az előre definiált népszerű úti célok között, mint például Tokió, Santorini vagy Faroe-szigetek.
   * A részletes információk, mint például az adott helyszín nevezetességei és a legjobb időszakok, szintén elérhetőek regisztráció nélkül.

**Funkciók, amelyek regisztrációhoz kötöttek**

A felhasználói élmény teljes kihasználásához regisztráció szükséges. Az alábbi funkciók eléréséhez regisztrációra van szükség:

1. **Saját utazás létrehozása**
   * A felhasználók, miután regisztráltak, lehetőséget kapnak arra, hogy saját utazásaikat hozzák létre, megadva a helyszínt, dátumokat és eseményeket.
2. **Események hozzáadása az utazáshoz**
   * A regisztrált felhasználók saját utazásaikhoz egyedi eseményeket adhatnak hozzá, így könnyedén nyomon követhetik az utazásuk részleteit.
3. **Egyéni úti célok hozzáadása**
   * Ha egy felhasználó egyedi helyszínt szeretne hozzáadni, ezt csak regisztrált felhasználóként teheti meg.

A regisztráció lehetővé teszi, hogy a felhasználók személyre szabott élményt kapjanak, és könnyedén kezelhessék az utazásaikat.

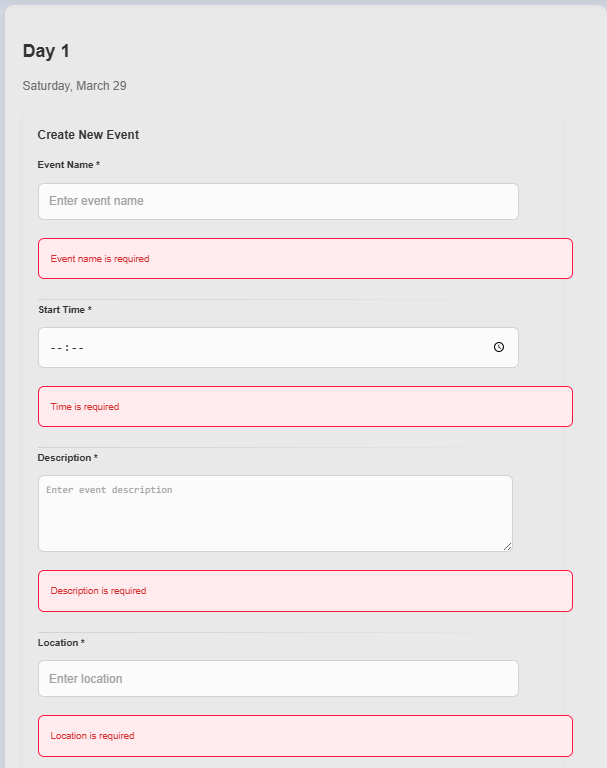
## Hibakezelés és hibajelzések

A JourneyPal a regisztrációnál több dologra is szűrést végez:

* Legalább 1 számnak szerepelni kell a jelszóban.
* Legalább 8 karakternek kell lennie a jelszónak.
* Kell szerepelni kisbetűnek illetve nagybetűnek is.
* Kell hogy szerepeljen legalább 1 különleges karakter.(/\*-.)
* Egyedi email címnek kell szerepelnie, amivel még nem történt regisztráció

A JourneyPal event létrehozásakor minden űrlapi elem kitöltése kötelező (a link megadásán kívül).

Amennyiben a felhasználó ezt a feltételt nem telejsíti, a rendszer jól látható üzenettel jelzi ezt a felhasználónak.



# Összefoglalás

## Az elkészült munka értékelése

A JourneyPal fejlesztése során az egyik legfontosabb célkitűzés az volt, hogy egy felhasználóbarát, átlátható és hatékony utazástervező alkalmazást hozzunk létre. A projekt megvalósítása során számos kihívással kellett szembenézni, de ezek sikeres leküzdése hozzájárult ahhoz, hogy a végeredmény egy jól működő és hasznos alkalmazás legyen.

Az alkalmazás alapvető funkcióit sikeresen implementáltuk: a felhasználók egyszerűen létrehozhatják saját utazásaikat, kiválaszthatják az előre meghatározott úti célokat vagy megadhatnak saját helyszíneket, valamint eseményeket rendelhetnek hozzá az egyes napokhoz. A fejlesztés során kiemelt figyelmet fordítottunk a felhasználói élményre, ezért az oldal könnyen kezelhető és átlátható felületet kapott.

A program jelenlegi állapotában egy stabil és használható verzió, amely megfelelő alapot biztosít a jövőbeli fejlesztésekhez. Természetesen további bővítési lehetőségek is adódnak, például a közösségi funkciók beépítése, keresési és szűrési lehetőségek hozzáadása, vagy akár a mobilalkalmazás verzió kidolgozása.

Összességében elégedett vagyok az elkészült munkával, mivel a kezdeti célkitűzéseket sikerült teljesíteni, és a JourneyPal egy működő, jól használható eszközzé vált azok számára, akik szeretnék megtervezni álomutazásukat.

## A JourneyPal jövője

A JourneyPal nem csupán egy vizsgafeladatként készült projekt, hanem egy olyan alkalmazás, amelynek valós felhasználói értéke lehet a jövőben. Az utazás szervezése sokak számára időigényes és bonyolult folyamat, ezért a program célja, hogy ezt leegyszerűsítse és egy átlátható, könnyen használható platformot biztosítson.

### **1. A program hasznosulása az utazók körében**

A JourneyPal lehetőséget biztosít arra, hogy a felhasználók előre megtervezzék utazásaikat, legyen szó egy hétvégi kiruccanásról vagy egy hosszabb nyaralásról. A jelenlegi funkciók segítségével:

* **Könnyedén összeállítható egy utazási terv** előre meghatározott vagy egyéni úti célokkal.
* Az egyes napokra események rögzíthetők, így a felhasználók **átlátható programot készíthetnek** az utazásukhoz.
* A későbbi frissítések során akár olyan funkciók is beépíthetők, amelyek **automatizált ajánlásokat tesznek** a népszerűbb úti célokhoz.

Mivel az emberek egyre nagyobb hangsúlyt fektetnek az utazási élmények tudatos megtervezésére, a JourneyPal segíthet abban, hogy minden utazás gördülékenyen és szervezetten történjen.

### **2. Lehetséges piaci bevezetés és monetizáció**

Bár a JourneyPal egyelőre egy ingyenesen használható alkalmazás, hosszabb távon különböző üzleti modellek mentén lehetne továbbfejleszteni:

* **Prémium verzió bevezetése**, amely extra funkciókat kínál, például személyre szabott utazási ajánlásokat vagy offline elérhetőséget.
* **Partnerségek kiépítése** utazási irodákkal, légitársaságokkal és szállásfoglaló platformokkal, amelyek integrálhatók az alkalmazásba.
* **Hirdetési lehetőségek** bevezetése, például szponzorált úti célok vagy ajánlott programok megjelenítése.

A felhasználói bázis növekedésével a JourneyPal akár egy jelentős utazástervező platformmá válhat a piacon.

### **3. Továbbfejlesztési lehetőségek**

A jelenlegi verzió egy stabil alapot biztosít, de rengeteg potenciál rejlik benne a további funkciók beépítésére:

* **Közösségi funkciók**: A felhasználók megoszthatnák egymással utazási terveiket, élményeiket, és akár mások programjait is lemásolhatnák vagy inspirálódhatnának belőlük.
* **Interaktív térkép és navigáció**: Egy vizuális felület, amely segít az utazóknak jobban megérteni a célállomásuk földrajzi elhelyezkedését.
* **Automatikus ajánlórendszer**: Az alkalmazás elemzhetné a felhasználók utazási szokásait, és személyre szabott úti célokat javasolhatna.

Ezekkel a funkciókkal a JourneyPal hosszú távon egy mindenre kiterjedő utazástervező platformmá fejlődhet.

### **4. A JourneyPal szerepe a digitális nomádok és gyakori utazók életében**

A program különösen hasznos lehet a **digitális nomádok**, üzleti utazók és olyan emberek számára, akik rendszeresen utaznak. Egyre több ember dolgozik távolról, és igényli azokat az eszközöket, amelyek segítenek **egyszerűsíteni az utazási terveiket**.

A jövőbeli frissítések során akár olyan funkciók is bevezethetők lennének, amelyek segítenek:

* **Szállásfoglalásban és repülőjegyek keresésében**,
* **Helyi coworking irodák vagy internetkapcsolatot biztosító helyszínek megtalálásában**,
* **Étel- és programajánlások személyre szabott listázásában**.

Ezek a lehetőségek jelentősen növelnék az alkalmazás értékét azok számára, akik rendszeresen új helyekre utaznak.

#### **Összegzés**

A JourneyPal nem csupán egy záródolgozathoz készült projekt, hanem egy olyan alkalmazás, amely valódi értéket teremthet az utazók számára. A jövőbeli fejlesztésekkel és megfelelő marketingstratégiával akár egy széles körben használt utazástervező platformmá is válhat. Az utazás mindig is az emberi élet fontos része volt, és egy modern, digitális eszköz, mint a JourneyPal, jelentősen megkönnyítheti ezt az élményt mindenki számára.