**Projektmunka II.**

GKNB\_INTM005

**Adorján András Bálint – G6I6ZQ**

**Jánoki Lilla – G4O424**

**Nátz Kornél – JOO2S6**

**Pajor Gergő – BXSGQE**

**Konzulens: Kálóczi Imre**

Győr, 2022

**Tartalom**

[1. Projektszabályzat 1](#_Toc102383581)

[1.1. Tagok listája 1](#_Toc102383582)

[1.2. Kommunikáció és dokumentumkezelés 2](#_Toc102383583)

[1.3. Jogok és kötelezettségek 2](#_Toc102383584)

[1.4. Feladatkörök 6](#_Toc102383585)

[1.5. Kompetencia mátrix 6](#_Toc102383586)

[1.6. Feladatkörök bemutatása 8](#_Toc102383587)

[1.7. Tevékenység-felelősség mátrix 9](#_Toc102383588)

[1.8. Közös szótás (Glossary) 10](#_Toc102383589)

[2. A kiválasztott probléma 13](#_Toc102383590)

[2.1. Szöveges bemutatás 13](#_Toc102383592)

[2.2. A probléma folyamatábrája 14](#_Toc102383593)

[2.3. Halszálka diagram 15](#_Toc102383594)

[2.4. SWOT analízis 16](#_Toc102383595)

[3. Üzleti igény specifikáció 17](#_Toc102383596)

[3.1. Célok 17](#_Toc102383598)

[3.2. Stakeholderek 17](#_Toc102383599)

[3.3. Rendszer tervezett működése folyamatábra 19](#_Toc102383600)

[3.4. Rendszer tervezett működése - use case diagram 21](#_Toc102383601)

[4. Követelményspecifikáció 21](#_Toc102383602)

[4.1. Funkcionális követelmények 21](#_Toc102383604)

[4.1.1. Jelentkezés az útra 22](#_Toc102383610)

[4.1.2. Út visszamondása 23](#_Toc102383611)

[4.1.3. Út meghirdetése 24](#_Toc102383612)

[4.1.4. Út törlése 25](#_Toc102383613)

[4.2. Nem funkcionális követelmények 26](#_Toc102383614)

[4.2.1. Termék követelmények: 26](#_Toc102383616)

[4.2.2. Szervezeti követelmények: 27](#_Toc102383617)

[4.2.3. Külső követelmények: 27](#_Toc102383618)

[4.3. Használhatóság 28](#_Toc102383619)

[4.4. Megbízhatóság 28](#_Toc102383620)

[4.5. Rendelkezésre állás 29](#_Toc102383621)

[4.6. Teljesítmény 29](#_Toc102383622)

[4.7. Támogatottság 30](#_Toc102383623)

[5. Technikai specifikáció 31](#_Toc102383624)

[5.1. Architektúra terv 31](#_Toc102383626)

[5.2. Platform tulajdonságok és követelmények 32](#_Toc102383627)

[5.3. Fejlesztői eszközök 33](#_Toc102383628)

[6. Gazdasági számítások 34](#_Toc102383629)

[6.1. Felmerülő költségek 34](#_Toc102383631)

[6.2. TCO (Total Cost of Ownership) 34](#_Toc102383632)

[6.3. Tervezett bevételek 35](#_Toc102383633)

[6.4. Fenntarthatóság növelése, költségek csökkentése 35](#_Toc102383634)

[6.5. Jövőbeli bevételi lehetőségek 35](#_Toc102383635)

[7. Adatbázis terv 36](#_Toc102383636)

[7.1. Szöveges leírás 36](#_Toc102383638)

[7.2. Táblák 36](#_Toc102383639)

[7.3. Adatmodell 38](#_Toc102383640)

[7.4. Egyed-kapcsolat diagram 39](#_Toc102383641)

[7.5. Kulcsok és megszorítások 39](#_Toc102383642)

[7.5.1. Elsődleges kulcsok 39](#_Toc102383651)

[7.5.2. Másodlagos kulcsok 39](#_Toc102383652)

[7.5.3. Megszorítások 40](#_Toc102383653)

[7.6. Tárolt eljárások 40](#_Toc102383654)

[7.7. Továbbfejlesztési ötletek 41](#_Toc102383655)

[8. GUI terv 42](#_Toc102383656)

# Projektterv

## A probléma és a projektcél rövid felvázolása

Csapatunk célja a fotográfusok munkájának segítése a fényképek kezelésének megkönnyítésével. Csapatunk fotográfus tagja és más szakmabeliek is tapasztaltak több különböző, ehhez a témához kapcsolódó problémát. Ezek közé tartozik, hogy a fényképezőgépek nem rögzítenek megfelelő mennyiségű exif adatot; manuális objektívekből egyáltalán nem tudnak adatokat kinyerni; bizonyos információk (pl. szerzői jog és megjegyzés) jóval korlátozottabb formában rögzíthetőek a fényképezőgép segítségével, mint amit a fájlok megengednének; illetve az is, hogy különböző programok más-más exif adattípusokkal kompatibilisek és a kompatibilis adatokat is különböző prioritással kezelik. Ezen problémákat szeretnénk feldolgozni és megoldási javaslatot tenni rájuk egy általunk készített program formájában.

## A program tervezett funkcionalitásai

A programnak képesnek kell lennie az exif adatok kezelésére, tárolására és szerkesztésére. A magasszintű kompatibilitás miatt több gyártó hardveres és szoftveres termékeit és az ezek által használt exif adattípusokat is ismernie kell. A program funkcionalitásai között lesz az exif adat szerkesztés, illetve az exif adatok szerinti szűrés és rendezés.

## Szoftveres felépítés

Megoldásunk egy webes felhasználói kliensből és egy távoli elérésű adatbázis szerverből fog állni. A tervezés során fontos szempont volt a biztonság, ezért megfontoltuk a háromrétegű architektúra kialakítását is, de végül elégségesnek találtuk a Google Firebase titkosítását, így a kétrétegű architektúra mellett döntöttük.

## A projekt során használni kívánt eszközök

Mivel webes alkalmazást fejlesztünk, így a fejlesztés fő eszközei a HTML, CSS és JavaScript nyelvek lesznek. Azon feladatok elvégzésére, amelyre ezek nem alkalmasak vagy célszerűek, C#-ot fogunk használni, de szükség esetén más nyelveket is bevonunk.

Adatbázisnak a korábban is említett Google Firebase rendszert fogjuk használni.

Elsődleges fejlesztési környezetünk a Visual Studio Code lesz.

A projektkezelés támogatására a Trello és a GitHub platformokat fogjuk segítségül hívni.

A chat alapú kommunikációra elsődlegesen az erre a célra létrehozott Messenger csoportban, a megbeszélések pedig Discord-on fognak történi. Amennyiben szükséges személyes találkozókat is szervezünk és más kommunikációs platformokat (pl. Google Meet, Zoom) is bevonunk.

## Mérföldkövek

1. Projektterv dokumentáció.
2. GUI-k tervezése, megvalósítása.
3. GUI-k funkcionálissá tétele.
4. Backend megvalósítása.
5. Rétegek összekapcsolása.
6. Tesztelés, felmerülő hibák kijavítása.

Első feladatunk a projekttervezet részletes, a tantárgyi követelményeknek megfelelő dokumentációja lesz. Ez elengedhetetlen a tantárgy teljesítéséhez, de a projekt végig vezetése során is jó kiindulási alapot fog adni a fejlesztő csapatnak. Ez természetesen nem fogja kitenni a projekt teljes dokumentációját. Azt a projekt során minden hallgató bővíteni fogja és végleges verziója csak a projekt lezárásával fog elkészülni. Második lépés, a felhasználói felületek grafikus megtervezése, majd ezek megvalósítása lesz. Ezután a felhasználói felületek, azaz a frontend oldal funkcionálissá tétele következik. Természetesen a külön funkciók és programrészletek tesztelése is elengedhetetlen, erre is sort fogunk keríteni a fejlesztés során. A negyedik lépés a backend megvalósítása, az adatbázis létrehozása és testre szabása. Amikor ezek mind megvannak, akkor létre lehet hozni a már működő egységek közti kommunikációs interfészeket az együttműködés érdekében. A projektet a program teljeskörű tesztelésével és az esetlegesen felmerülő hibák kijavításával fogjuk zárni.

## Tevékenységek tervezett kiosztása

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Adorján András Bálint | Jánoki Lilla | Nátz Kornél | Pajor Gergő |
| Dokumentáció | X | X | X | X |
| Megbeszélések koordinálása | X |  |  |  |
| Kapcsolattartás | X | X |  |  |
| Folyamatábrák készítése | X | X |  | X |
| GUI-k készítése | X |  | X | X |
| Adatbáziskezelés |  |  | X | X |
| Programozás | X | X | X | X |

A dokumentáció a közös munka és a tantárgy értékelése szempontjából is rendkívül fontos része projektünknek, így mindegyik csapattag foglalkozni fog vele. Ugyanígy kitüntetett szerepet fog kapni a programozás.

A projekt során a csapattagok kommunikálni fognak egymással a korábban említett csatornákon, viszont az egyéb kommunikációval kapcsolatos feladatokat alapvetően a projektvezető fogja ellátni. Ide tartozhat például a megbeszélések koordinálása vagy kapcsolattartás a konzulenssel és esetleges harmadik felekkel.

A folyamatábrák és grafikai elemek elkészítését vegyesen osztottuk fel a csoporttagok között, az adatbázisok készítésére, kezelésére pedig az ebben tapasztaltabb csapattagokat jelöltük ki.

## Célok

Célunk, hogy elkészüljön egy olyan webalapú szoftver, ami nagyban hozzá tud járulni a fotográfusok munkájához a fényképek kezelésének megkönnyítésével.

Reméljük, hogy a projekt elkészítése hozzá fog járulni szakmai fejlődésünkhöz és a konzulensünk is elégedett lesz munkánkkal.

## Közös szótár (Glossary)

**Stakeholder**

Az üzleti igényspecifikáció a résztvevője.

**Use-Case diagram**

Magyarul használati eset diagramként is ismert, a felhasználó és a rendszer közti lehetséges interakciók grafikus ábrázolására használt módszer. A Use-Case diagram különböző használati eseteket és különböző típusú felhasználókat mutat be, gyakran más típusú diagramok is kísérik. Az egyes eseteket körök vagy ellipszisek jelölik. Az aktort (azaz a felhasználót) pálcikafigurákként ábrázolja.

**Aktor**

Use-Case diagramban egy szerepkört, a rendszer egy felhasználóját reprezentálja.

**SWOT analízis**

A SWOT analízis (vagy más néven mátrix) egy stratégiai tervezési és stratégiai irányítási elemzéshez használt technika, mely segít egy személynek vagy szervezetnek abban, hogy azonosítani tudják a versenytásakkal szembeni vagy a projekttervezéssel kapcsolatos erősségeket, gyengeségeket, lehetőségeket vagy veszélyeket.

**TCO**

A TCO (Total Cost of Ownership) egy pénzügyi becslés, melynek célja, hogy segítse a vásárlók és a tulajdonosok számára meghatározni egy termék vagy szolgáltatás közvetlen és közvetett költségeit. Ez egy vezetői számviteli eszköz, mely a teljes költségszámításhoz vagy akár az ökológiai, gazdasági számításokhoz is alkalmazható, beleértve a társadalmi költségeket is.

**Egyed-kapcsolat diagram**

Az egyed-kapcsolati modell (vagy ER-modell) egy adott tudásterületen belüli, egymással összefüggő dolgokat ír le. Egy alapvető ER-modell egyed-típusokból áll (amelyek osztályozzák az érdekes dolgokat), és meghatározza az egyedek (ezen egyed-típusok példányai) között létező kapcsolatokat. A szoftverfejlesztésben az ER-modellt általában azért alkotják meg, hogy ábrázolja azokat a dolgokat, amelyekre az üzleti folyamatok végrehajtásához egy vállalkozásnak emlékeznie kell. Következésképpen az ER-modell egy absztrakt adatmodellé válik, amely egy olyan adat- vagy információs struktúrát határoz meg, amely egy adatbázisban, jellemzően egy relációs adatbázisban implementálható.

**Tárolt eljárás**

A tárolt eljárás a strukturált lekérdezési nyelv (SQL) utasítások gyűjteménye a hozzá rendelt névvel. Ezeket a tárolt eljárásokat egy relációs adatbázis-kezelő rendszerben (RDBMS) tárolják. Tehát lehetséges az eljárás többszörös meghívása, amely csökkenti a végrehajtási időt. Az adatok eljárásokkal módosíthatók, és a fő előnye az, hogy nem kapcsolódnak egy adott alkalmazáshoz.

**GUI**

A grafikus felhasználói felület vagy grafikus felhasználói interfész (angolul graphical user interface, röviden GUI) a számítástechnikában olyan, a számítógép és ember közti kapcsolatot megvalósító elemek összessége, melyek a monitor képernyőjén szöveges és grafikai elemek együtteseként jelennek meg. A grafikus felhasználói felületeken alapvető szerepe van a mutatóeszközök, például az egér használatának, amelyekkel a grafikus felület elemei intuitív módon, a fizikai világ egyfajta modelljeként kezelhetők.

**Elsődleges kulcs**

Egy relációs adatbázis használatakor minden táblában meg kell jelölni, hogy melyik mező, vagy melyik mezők együttesen az elsődleges kulcsok. Az elsődleges kulcs minden rekordban egyedi.

**Idegen (másodlagos) kulcs**

Az idegen kulcs olyan oszlop, amelyet a másik táblázatban elsődleges kulcsnak neveznek, azaz a táblázat elsődleges kulcsa idegen kulcsnak nevezhető egy másik táblázatban. Lehet, hogy az idegen kulcs duplikált & NULL értékeket tartalmaz, ha a NULL értékeket elfogadják.

# A kiválasztott probléma



## Szöveges bemutatás

A Széchenyi István Egyetem hallgatói közül sokan dolgoznak egyetemi tanulmányik mellett. Mindenki szeretne térben és időben pontosan, minél kevesebb plusz idő és energia ráfordításával eljutni munkahelyére. A pontos lokációtól függően gyalogosan, biciklivel, tömegközlekedéssel, taxival vagy saját autóval is megközelíthetik munkahelyüket a hallgatók, de ezen közlekedési módoknak számos hátránya és veszélye van.

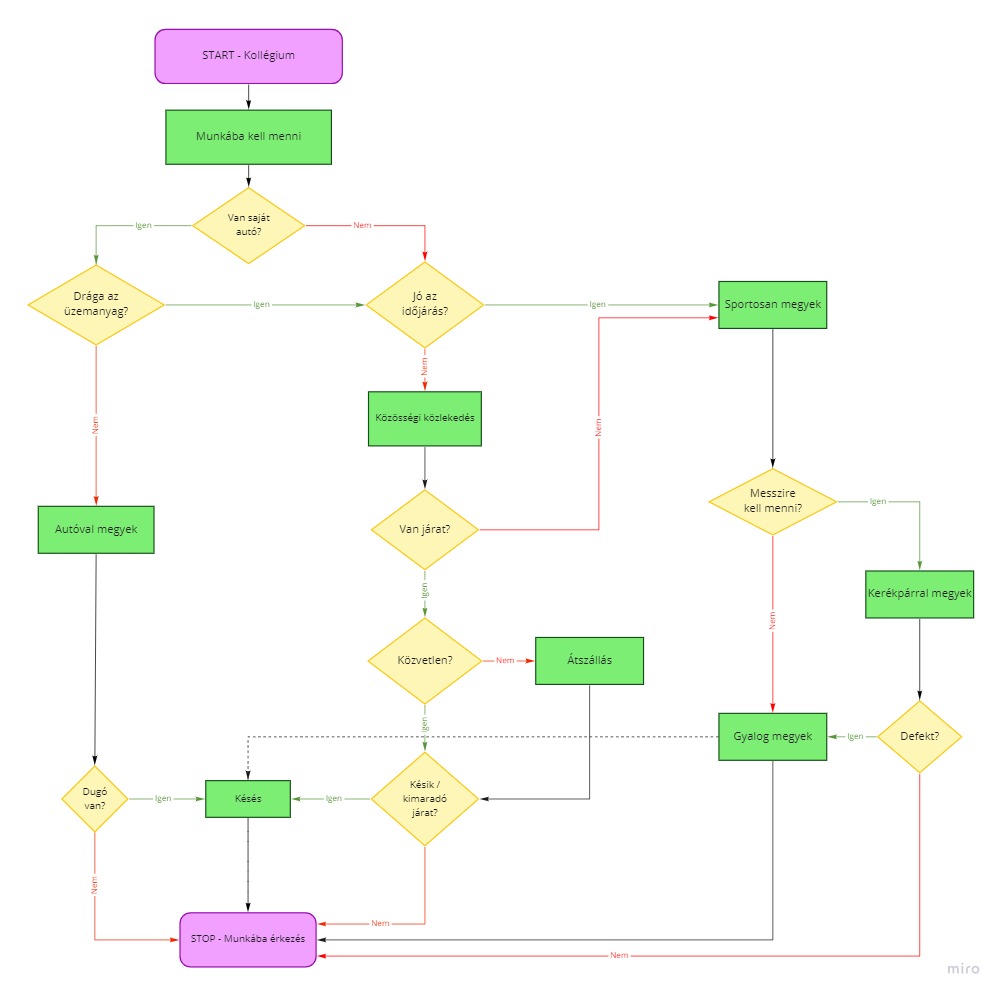
A gyaloglás és biciklizés nagyon időigényes, megterhelheti testünket és közben ki vagyunk téve az időjárási viszonyoknak, továbbá a közlekedés résztvevői és a gyárak által kibocsátott, levegőt szennyező káros anyagoknak.

A tömegközlekedés nem ér el közvetlenül Győr minden pontjára. Vannak helyek, amik egyáltalán nem lehet eljutni és vannak, amelyek az egyetemtől kiindulva csak egy vagy több átszállással érhetőek el. Sok esetben a buszok nem is közlekednek megfelelő sűrűséggel és akár órákat is várnunk kell a következő járatra. Sajnos abban sem lehetünk 100%-ig biztosak, hogy a következő járat meg fog érkezni, ugyanis sok hallgatótársunk tapasztal nap mint nap járatkimaradásokat.

A taxi árak borzasztóan magasak, főleg egy egyetemi hallgató munkalehetőségeihez és fizetéséhez képest.

A saját autó használata sok korábban említett problémára megoldást tud nyújtani, viszont legalább annyi újat is vet fel. A benzin fosszilis üzemanyag, tehát korlátozott mértékben van csal jelen a bolygón és per pillanat jóval nagyobb mértékben termelik ki, mint ahogy termelődik. Emiatt sajnos az ára is egyre magasabb. Ráadásul használat közben a környezetet szennyező anyagok jönnek létre, ezzel lokális emissziót okozva. Az autók használata nem csak a levegő, de az utak minőségét is rontja. Minél több autó közlekedik, annál nagyobb mértékben romlanak az utak és annál nagyobbak lesznek a közlekedési dugók is. A munkahelyek parkolókapacitása is korlátozott, minél több autó érkezik hozzájuk, annál hamarabb telik meg a parkoló és annál messzebb tudunk csak parkolni.

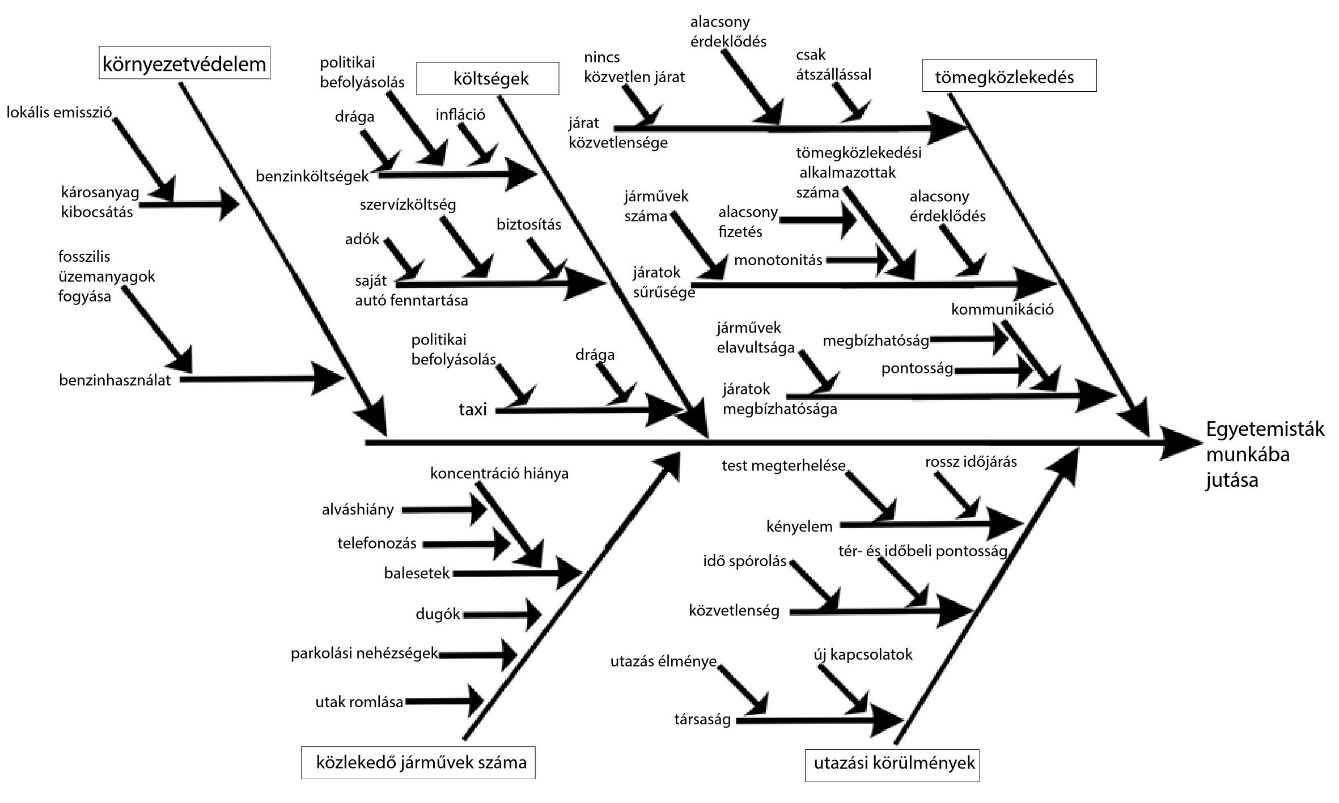
## A probléma folyamatábrája



**1. ábra - A probléma folyamatábrája**

Az 1. ábrán a kollégiumtól (START) a munkába (STOP) való eljutás lehetséges kimenetelei láthatóak. A kiindulópont minden esetben a kollégium lesz és a végcél pedig az, hogy elérjünk a munkába olyan módon, hogy az esetleges késés a lehető legminimálisabb legyen. Kezdetben megvizsgáljuk, hogy rendelkezünk-e saját autóval és amennyiben igen akkor tovább tudunk lépni arra, hogy az üzemanyag ára mennyibe kerül. Ha ezt nem találjuk soknak, akkor választhatjuk azt az opciót, hogy kocsival megyünk munkába. Azonban itt még nem ér véget a folyamat, hiszen előfordulhat az, hogy a sok autó miatt dugó keletkezik az utakon és ezért késve érkezünk meg a munkahelyünkre. Ellenkező esetben nem fogunk késni, feltéve, ha időben indultunk el otthonról. Hogyha rendelkezünk autóval, de az üzemanyag árát drágának találjuk vagy nincsen autónk, akkor érdemes lehet megvizsgálni az időjárási viszonyokat. Ha ezek kedvezőtlenek és rossz idő van, akkor rá vagyunk arra kényszerülve, hogy a tömegközlekedést használjuk. Ebben az esetben a következő kérdés az, hogy van-e járat és amennyiben van, a járat közvetlen-e, ugyanis ilyen módon nagy valószínűséggel el tudjuk kerülni a késést, viszont erre nincsen garancia, még így sem. Hogyha nincsen közvetlen járat akkor kénytelenek vagyunk átszállni, ami már jó eséllyel késést fog eredményezni, de természetesen ez esetben is előfordulhat az, hogy pontosan érkezünk meg. Amennyiben kedvezőek az időjárási feltételek vagy egyáltalán nincsen olyan járat, amivel el tudnánk jutni a munkahelyünkre, gyalogosan vagy kerékpárral kell közlekednünk annak függvényében, hogy mennyire kell messzire mennünk. Ezeknek előnye, hogy sportolunk, ami hozzájárulhat egészségünk megőrzéséhez és fel is frissülünk. Ha nem kell nagyon távolra mennünk, akkor a gyaloglás egy jó választás lehet, ha már kicsit nagyobb távot kell megtennünk, akkor érdemes lehet igénybe venni kerékpárunkat. Ekkor azonban kaphatunk defektet és ilyenkor kénytelenek vagyunk a mégis csak gyalog menni (magunk mellett tolni a biciklit), ami nagy valószínűséggel szintén késést fog eredményezni.

## Halszálka diagram



**2. ábra - A probléma halszálka diagramja**

## SWOT analízis

|  |  |
| --- | --- |
| **Strengths - Erősségek**   * függetlenség * helyettesíthetőség * jól bejáratott rendszer * offline * olcsóbb az autófenntartásnál * hallgatói kedvezmények | **Weaknesses - Gyengeségek**   * busszal kevesebb helyen lehet megállni * ki vagyunk téve az időjárásnak * alacsony járatszám * nincs mindig elég férőhely * kimaradó járatok * hétvégén kevesebb járat |
| **Opportunities - Lehetőségek**   * társalgás, kapcsolatépítés * kötött menetrend * friss levegő * mozgás | **Threats - Veszélyek**   * késés * jármű meghibásodása * utasok közti konfliktus * utasok egészségügyi állapota * testiépség veszélye |

**3. ábra - A probléma SWOT analízise**

# Üzleti igény specifikáció



## Célok

A SZEAT-APP elsődleges célja, hogy a Széchenyi István Egyetem hallgatóinak munkába jutását segítse, egyszerűbbé, gazdaságossá és környezetbaráttá tegye. A hallgató megnézheti az appon, hogy mikor és honnan indulnak sofőrök. Az appon keresztül lehetősége van jelentkezni a fuvarra. Ha az autó megtelik, akkor a sofőr benzinköltsége megtérül, az utasok pedig sokkal olcsóbban utaznak, mintha taxival mennének. Megoldásunk segítené a dugó kialakulásának elkerülését is és lokális emisszió csökkentését. Minél többen utaznak saját autó helyett közösen, annál kevesebb jármű fog közlekedni és szennyezni a környezetet. Az utasok nem lennének kitéve az időjárás viszontagságainak és a tömegközlekedés bizonytalan mivoltának sem, továbbá a hasonló korosztályú és azonos munkahelyen dolgozó utasok könnyebben tudnának ismerkedni, barátságokat kötni. Ez a közös munkavégzésre és a kollégiumi életre is pozitív hatásokkal lehet.

## Stakeholderek

Projektünk elsődleges belső érdekeltjei az applikáció fejlesztői, akik elkészítik és a továbbiakban is karban tudják majd tartani, fejleszteni az applikációt. Mellettük belső érdekeltek lehetnek még az esetleges befektetők. A külső érdekeltek körébe tartoznak a hallgatók, akik igénybe szeretnék majd venni a szolgáltatást és az appsofőrök, akik az autókat fogják vezetni. A külső érdekeltek közé tartozhatnak konkurenciaként még az ismert, személyszállítással foglalkozó sofőrök, például busz- és taxisofőrök.

**Belső érdekeltek**

A fejlesztők nélkül nem jöhetne létre és nem is működhetne az applikáció. Feladatuk megtervezni a célnak megfelelően működő alkalmazást. Későbbiekben is szükség lesz a munkájukra a program karbantartása és esetleges fejlesztések kivitelezése kapcsán.

A befektetők kulcsszerepet játszhatnak az applikáció létrehozásának, fenntartásának és továbbfejlesztésének finanszírozásában. Minél népszerűbb lesz az alkalmazás, annál több emberhez juthat el és annál több utas veheti igénybe a szolgáltatásokat. Később az alkalmazást ki is lehet bővíteni és további szolgáltatásokat biztosítani.

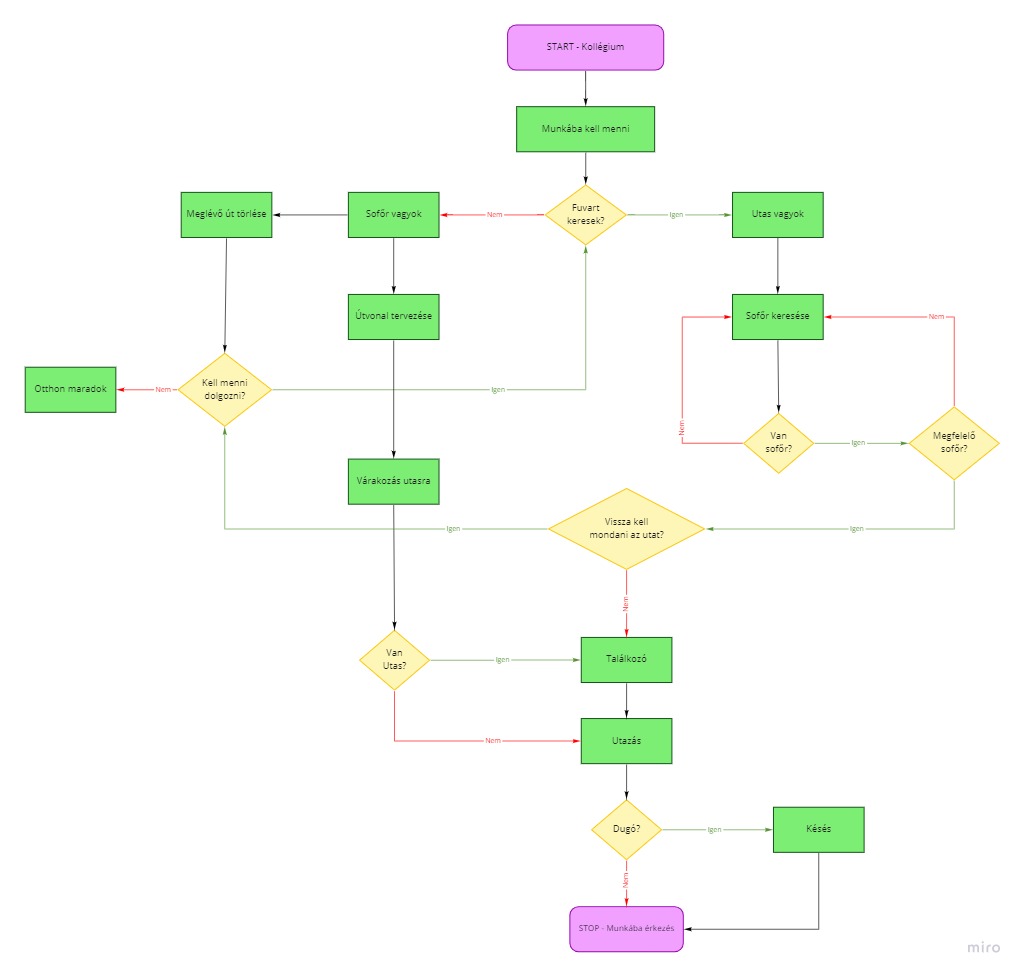
**Külső érdekeltek**

A hallgatók érdeke, hogy munkahelyükre minél gyorsabban és pontosabban érkezzenek, utazásuk pedig minél olcsóbb és kényelmesebb legyen. A SZEAT-APP használatával a hallgatóknak nem kell alkalmazkodni a kötött tömegközlekedési menetrendekhez és az időjárás viszontagságaihoz, továbbá sokkal olcsóbban is élvezheti a személyautók kényelmét, mintha taxit hívna. Az utazás közben akár új barátokra is lelhet.

Az appsofőrök érdeke, hogy minél több utast sikerüljön összegyűjteni egy útra. Cél, hogy az összegyűlt összeg minél nagyobb mértékben fedezze az autófenntartási és benzinköltségeket.

A busz- és taxisofőrök inkább vetélytársként lehetnek érdekeltek, mivel minél többen használják a SZEAT-APP-ot, annál kevesebben fognak igénybe venni más városi személyszállítási szolgáltatásokat.

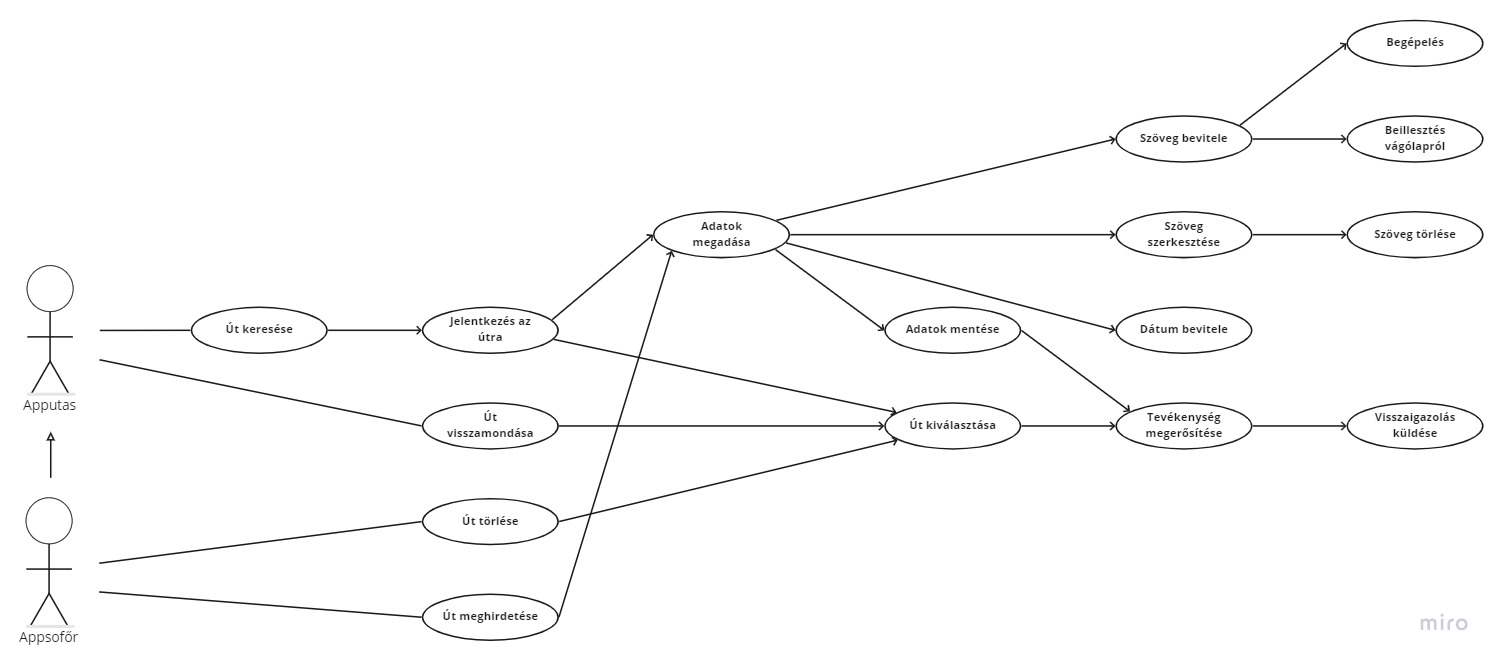
## Rendszer tervezett működése folyamatábra



**4. ábra - A megoldás folyamatábrája**

A rendszerünk tervezett működésének folyamatát mutatja be. A kiinduló pont, azaz a START a kollégium lesz és a STOP pedig az, hogy megérkezzünk a munkába. Először is a fuvar keresésének a kérdését kell megválaszolni ugyanis azért indultunk el, mert munkába kell mennünk. Ha fuvart keresünk, akkor feltételezhetően utasok vagyunk (hacsak nem adódott valami probléma az autónkkal és ezért van szükségünk fuvarra) ezért találnunk kell egy sofőrt. Addig vagyunk kénytelenek keresni ameddig nem találunk egyet és ezután azt kell átgondolnunk hogy ő lesz-e a számunkra megfelelő sofőr. Hogyha úgy döntünk valamilyen oknál fogva, hogy nem akkor a keresés újból indul addig, amíg nem találunk egy megfelelő személyt. Ha sikerült megtalálni az alkalmas sofőrt, akkor meg kell vizsgálni azt, hogy valamilyen oknál fogva vissza kell-e mondanunk az utat. Az út visszamondása történhet azért, mert nem kell mennünk mégsem dolgozni és így otthon tudunk maradni, de ha lemondtuk az utat és továbbra is kell menni dolgozni, akkor a fuvar keresés indul ismét a legelejéről. Abban az esetben, ha semmilyen zavaró körülmény nem jön közbe, akkor nem mondjuk vissza az utat és tudunk találkozni a sofőrünkkel és megtörténik az utazás. Ekkor már csak az nehezítheti meg utunkat, ha dugóba keveredünk és ez miatt a munkába, azaz a STOP-ba érkezésünk késve történik. Amennyiben megfelelő tempóban tudunk haladni akkor késés nélkül meg tudunk érkezni a munkahelyünkre feltéve, ha időben indultunk el. A másik opció lehet az, ha nem keresünk fuvart, mert sofőrök vagyunk. Ekkor meg kell terveznünk azt az útvonalat, amivel a lehető leghatékonyabb módon minden utast el tudunk vinni a munkába és mi is időben be tudunk érni dolgozni. Ha már megterveztük, hogy melyik útvonalon kívánunk menni, akkor is előfordulhat, hogy azt valamilyen oknál fogva törölnünk kell például azért, mert nem kell menni dolgozni és otthon tudunk maradni. Amennyiben ez az opció nem áll fent akkor új útvonalat kell megterveznünk és meg kell várnunk, illetve találkoznunk kell utasainkkal, feltéve, hogy vannak. Hogyha nincsenek vagy már felvettük őket, akkor elkezdődhet az utazás a munkába. Ilyenkor már csak a korábban is említett dugó problémája léphet fel, és ha ez fenn áll akkor valószínűleg késni fogunk a munkából viszont ha nem akkor pontosan meg tudunk érkezni.

## Rendszer tervezett működése - use case diagram



**5. ábra - SZEAT-APP működése - Üzleti use case diagram**

Az 5. ábra a SZEAT-APP működését szemlélteti, melyben szerepel utas és sofőr. A sofőröknek két fő szerepkörük van, az utak meghirdetése és esetleges törlése. Az utas számára az út keresés a legfontosabb, melyre utána jelentkezhet, s ezt követően, ha kívánja, le is mondhatja a fuvart. Mind az út meghirdetéshez, mind a jelentkezéshez szükség van adatok megadására, s ezen adatok a megadás és mentés után is szerkeszthetők, törölhetők, további információkkal kiegészíthetők. Az utak megtervezéséhez, törléséhez, valamint az ezekre való jelentkezéshez, vagy annak törléséhez szükséges, hogy kiválasszuk a tetszőleges útvonalat, s ezután a kívánt tevékenységet meg is erősítsük, s erről visszaigazolást kapjunk az alkalmazástól.

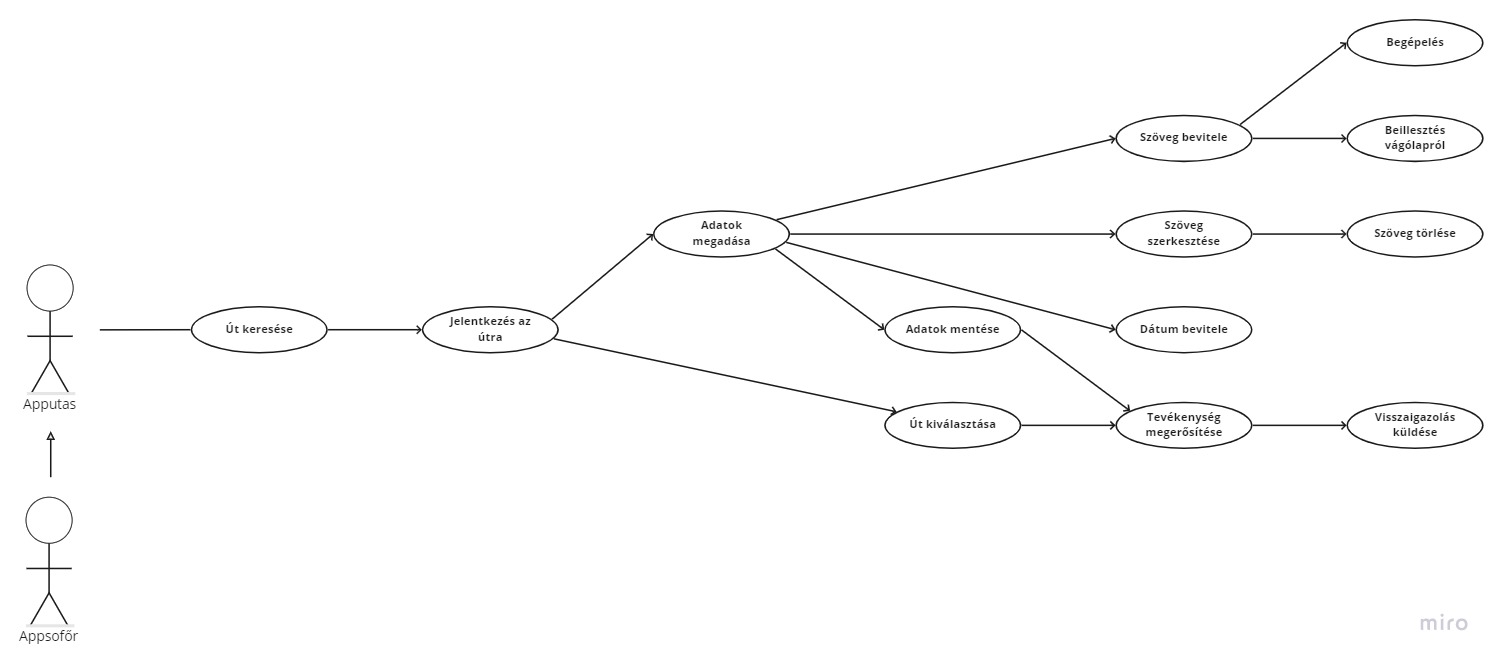
# Követelményspecifikáció



## Funkcionális követelmények



### Jelentkezés az útra



**6. ábra - SZEAT-APP - Use case - Jelentkezés az útra**

A használati esetet az Apputas indítja el!

**Előfeltétel:**

* SZEAT-APP alkalmazás, regisztrált fiók, alkalmazás ismerete a kezeléshez.
* Internet elérés.
* Appsofőr által meghirdetett út.

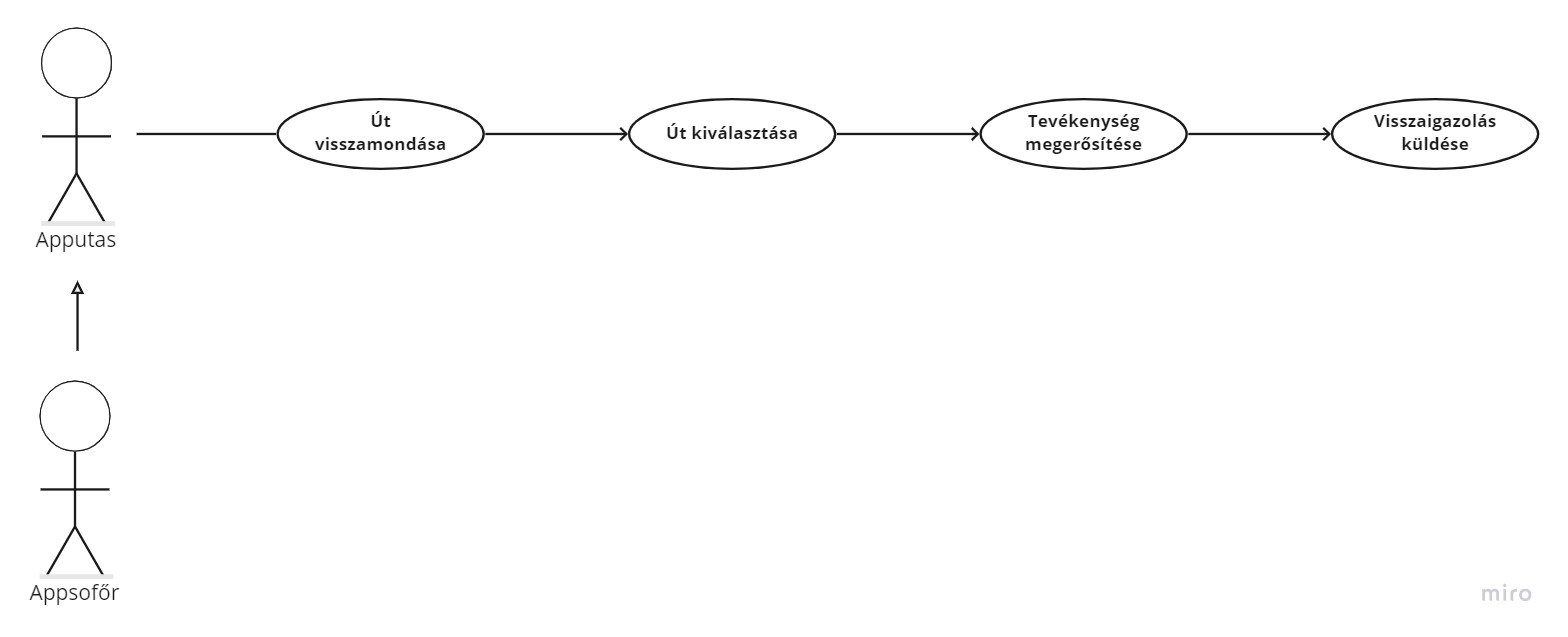
**Folyamat:**

A folyamat akkor indul el, amikor az Apputas munkába kíván menni. Megnyitja az Applikációt és a felületén kiválasztja az út keresése funkciót. Ilyenkor egy kereső felületen meglehet adni az utas által kiválasztott indulásipontot és végcélt továbbá az utazás tervezett időpontját. Ezt követően az útvonal keresése gomb megnyomásával a rendszer lekérdezi az adott paraméterekkel rendelkező Appsofőröket, majd listázza. Amennyiben ilyen útvonal létezik nincs más dolga az utasnak, mint jelentkezni az utazásra. Innentől kezdve pedig már csak az Appsofőrnek kell elfogadni az utast. Ha nem létezik ilyen útvonal, akkor az Apputasnak más alternatív utazási mód után kell néznie.

**Hibák:**

* Nincsen Appsofőr a szükséges útvonalra.
* Nem működik az Applikáció által használt szerver/adatbázis.

### Út visszamondása



**7. ábra - SZEAT-APP - Use case - Út visszamondása**

A használati esetet az Apputas vagy az Appsofőr indítja el!

**Előfeltétel:**

* SZEAT-APP alkalmazás, regisztrált fiók, alkalmazás ismerete a kezeléshez.
* Internet elérés.
* Appsofőr részéről: utassal való rendelkezés egy útvonalra.
* Apputas részéről: sofőrrel való rendelkezés egy útvonalra.

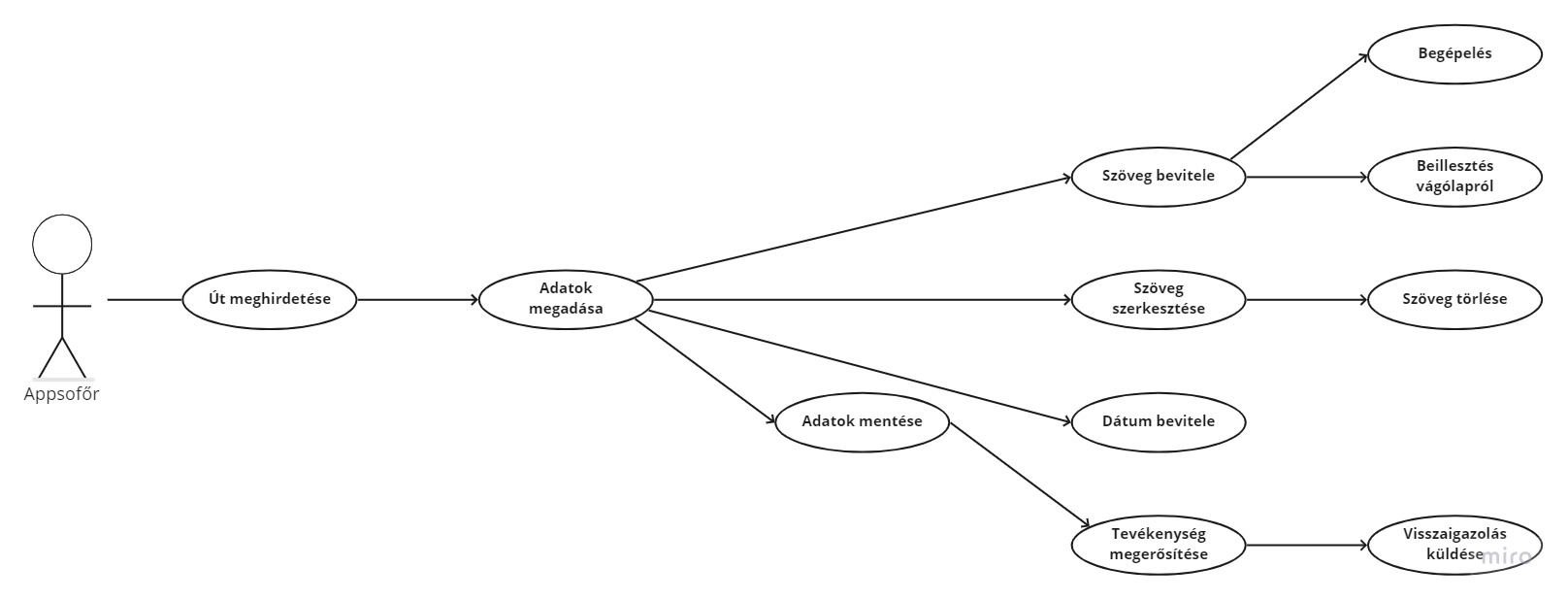
**Folyamat:**

A folyamat akkor indul el, amikor az Appsofőrnek vagy az Apputasnak bármely nyomós indoknál fogva le kell mondania a már előre elfogadott útját. Ebben az esetben mindkét felhasználónak a felületén a menüsorban a következő funkciót kell kiválasztani: saját útjaim, majd ezt követően a rendszer kilistázza a jövőben tervezett útjait a felhasználónak. Ebből a listából kell kiválasztani azt az útvonalat melyet a felhasználó törölni kíván. A törlés gomb megnyomásával egyidőben a rendszer figyelmezteti az aktuális másik felet (Apputast, ha az Appsofőr mondja le vagy fordítva), hogy az előzőleg már elfogadott útvonal törlésre került.

**Hibák:**

* Nem működik az Applikáció által használt szerver/adatbázis.
* Nem kapja meg az értesítést a másik fél.

### Út meghirdetése



**8. ábra - SZEAT-APP - Use case - Út meghirdetése**

A használati esetet az Appsofőr indítja el!

**Előfeltétel:**

* SZEAT-APP alkalmazás, regisztrált fiók, alkalmazás ismerete a kezeléshez.
* Internet elérés.
* Igazolt Appsofőri státusz.

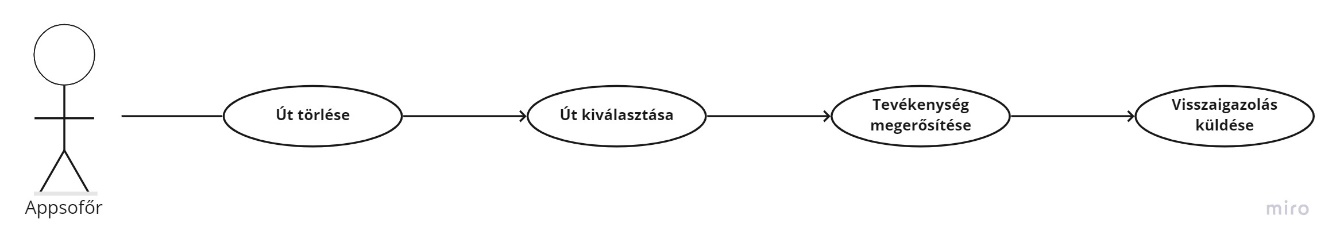
**Folyamat:**

A folyamat akkor indul el, amikor az Appsofőr egy útvonalat kíván létrehozni. Megnyitja az Applikációt és a felületén kiválasztja az út meghirdetése funkciót. Ilyenkor egy tervező felületen a sofőr meg tud adni egy útvonalat, indulási időpontot, helyszínt, az utazás várható időtartamát, és a szabad helyek számát is. Ezt követően az út meghirdetése gomb megnyomásával a rendszer nyilvánosságra hozza az adott paraméterekkel az útvonalat az applikáción belül. Ezek után a sofőrnek az utasok jelentkezéseit kell elfogadnia. Amennyiben legalább egy utas jelentkezik, a sofőr végre tudja hajtani a szolgáltatást. Ha nem jelentkezik egy utas sem, akkor a sofőr egyedül utazik.

**Hibák:**

* Nincsen Apputas az útvonalra.
* Nem működik az Applikáció által használt szerver/adatbázis.

### Út törlése



**9. ábra - SZEAT-APP - Use case - Út törlése**

A használati esetet az Appsofőr indítja el!

**Előfeltétel:**

* SZEAT-APP alkalmazás, regisztrált fiók, alkalmazás ismerete a kezeléshez.
* Internet elérés.
* Sofőr által meghirdetett út.
* Igazolt Appsofőri státusz.

**Folyamat:**

A folyamat akkor indul el, amikor az Appsofőrnek bármely nyomós indoknál fogva le kell mondania a már előre elfogadott útját. Ebben az esetben az Appsofőrnek a felületén a menüsorban a következő funkciót kell kiválasztani: saját útjaim, majd ezt követően a rendszer kilistázza a jövőben tervezett útjait a sofőrnek. Ebből a listából kell kiválasztani azt az útvonalat melyet a sofőr törölni kíván. A törlés gomb megnyomásával egyidőben a rendszer figyelmezteti az utast, hogy az előzőleg már elfogadott útvonal törlésre került.

**Hibák:**

* Nem működik az Applikáció által használt szerver/adatbázis.

## Nem funkcionális követelmények



### Termék követelmények:

Hatékonysági követelmények:

* Karbantartott autó.
* Letisztult átlátható applikációs felület

Hordozhatósági követelmények:

* Utas nem hozhat 20kg-nál nagyobb csomagot.

Méret követelmények:

* Utasok számával arányos autó méret.
* Elegendő tárhellyel rendelkező eszköz (10 MB).

Teljesítmény követelmények:

* Applikációt futtatni képes eszköz (mobiltelefon).
* Megfelelően optimalizált alkalmazás

Használhatósági követelmények:

* Internetkapcsolat.
* Applikáció előre feltelepítése.
* 18 éves kor betöltése.
* Érvényes műszakival rendelkező jármű
* Évszaknak megfelelő gumiszett.

### Szervezeti követelmények:

Szabványügyi követelmények:

* Megfelelő teljesítményű motor.

Implementációs követelmények:

* Applikáció univerzális használhatósága.
* Jelenlegi operációsrendszerekkel kompatibilis applikáció tervezése.
* Elfogadható válaszidő.
* Alkalmazásban a sofőrök átlátható véleményezése.

### Külső követelmények:

Etikai követelmények:

* Utasok és sofőr közötti közös megbecsülés, kölcsönös tisztelet.
* Sofőr ne legyen priuszos.

Együttműködési követelmények:

* Pontos érkezés az utas és sofőr részéről is.

Jogi követelmények:

* Személyes adatok megfelelő kezelése.

Biztonsági követelmények:

* Jogosítvány megléte.
* Utazás közben biztonsági öv kötelező használata.
* Megfelelően funkcionáló légzsákok.

Titokvédelmi követelmények:

* Útvonal bizalmas kezelése (Ne tudjanak róla illetéktelen személyek).

## Használhatóság

Egy gyakorlott felhasználó könnyedén felfedezheti az alkalmazást, figyelembe véve a hasonló szoftverek felhasználói felületét. Vannak felhasználók, akik a menüket és vannak, akik az ikonokat részesítik előnyben. A két grafikai kialakítást egybeolvasztva lenne megvalósítva az app.

Egy kezdő felhasználó számára az eszközrendszere több helyről elérhető és segítségükre van a súgó rendszer is, ami a megfelelő ikonra (egy kérdőjelre) kattintva megtalálható.

Egy, a felhasználók kérdései alapján felépített online súgó és egy fórum, amely megteremtheti a folyamatos kapcsolatot a készítőkkel, nem csak a használhatóságot, hanem a későbbi továbbfejlesztést is megkönnyítheti.

A felület letisztult, könnyen használható és átlátható legyen. A rendszer legyen látványos és figyelemfelkeltő a kellő minimalizmussal.

Az alkalmazás jól optimalizált, nem foglal a kelleténél több helyet, és nem igényel a kelleténél több nyersanyagforrást (CPU, RAM, GPU).

Felesleges adatokat ne tároljunk.

## Megbízhatóság

**Biztonság**: Fontos, hogy minden felhasználó csak ahhoz a részhez férhessen hozzá, amihez jogosultsága van. Hiszen nem engedhetjük meg, hogy egy egyszerű felhasználó járatokat töröljön, vagy egy alkalmazás karbantartó hozzáférjen az utasok adataihoz. Ezért kulcskérdés a felhasználók megfelelő szerepkörbe való osztása mely az adminisztrátor feladata. Valamint egy olyan azonosítási procedúrával kell rendelkeznie a rendszernek mely megkerülhetetlen, és egyértelműen azonosítja a felhasználót.

**Karbantarthatóság**: Az esetleges hibák fellépésekor, akár a tesztelés, akár a felhasználói használat során (amennyiben ezt jelzik a fejlesztők felé) a hiba detektálása és javítása azonnal megkezdődik. A hiba súlyosságától függően elképzelhető, hogy a rendszert le kell kapcsolni a hibajavítás idejére. Kisebb hibák esetén elég egy csak a hibás fájlok kicserélésének idejére leállítani az alkalmazást futtató webszervert.

**Hordozhatóság**: A szoftvert (kliens oldali) alkalmazóknak lehetősége van operációs rendszert, vagy platformot váltani, ezt ugyanis a platformfüggetlen fejlesztés lehetővé teszi.

**Hibakezelés**: Az alkalmazás szabályos használata esetén nem fordul elő hibajelenség, nem jelenik meg hibaüzenet, minden szakszerűen működik. Amennyiben a felhasználó hibás adatot ad meg, a program azt megfelelően kezeli, és biztosítja annak javítását.

**Jogszabályok**: Jogszabályok betartása. A felhasználó adatainak biztonságának megőrzése

## Rendelkezésre állás

A rendszer alapvetően a munkában elterjedt idősávokban vannak a legmagasabb kihasználtsági csúcsok. Office hour vagy akár a 2 vagy 3 műszakos idősávok előtt 1,5-2 órával és utána is szintén. Ezen időszakokban lehet a legnagyobb rendszer terhelésre számítani.

A rendszer felügyeletét reggel 5 órától este 11ig biztosítani szükséges, az ezen kívül eső intervallumon csak a következő napra lehet a problémákat egy automatikus hangrögzítő segítségével leadni.

MTBF: A rendszer kritikus követelménye, hogy az esetleges hibák/leállások között a lehető legtöbb idő teljen el, azaz a lehetőleg minimálisnak kell lennie a felléphető hibák számának, melyek a működést befolyásolják. Statisztikák alapján egy leállás átlag ideje 32 perc. És ilyen mértékű leállás fél évente egyszer történik, tehát az MTBF együtthatónk egy évre számolva 16-ra számítható.

MTTR: Ebben az esetben is azt kell figyelembe venni, hogy a rendszer mennyi ideig van használhatatlan állapotban, de ebben az esetben fontos szempont a hogy a leállások alatt mennyi javítás volt szükség elvégezni annak érdekében, hogy a rendszer újra működőképes legyen. Ebben az esetben is , mint az MTBF esetében a rendszer érdeke, hogy a lehető legalacsonyabb hibával működjön. A rendszer karban tarthatósága egyszerű, mert az applikáción kívül a webszervert kell az adatbázissal karban tartani. A webszerverről a biztonság kedvéért naponta egyszer hajnali 3-kor biztonsági mentés készül, továbbá havonta egyszer újraindítják.

## Teljesítmény

A rendszer biztonságának megőrzése érdekében folyamatosan figyeli biztonsági problémák szempontjából és állapotjelentést ad, amely megjelenik egy oldalon. Az állapotjelentés figyelmeztet a négy fontos területtel kapcsolatos általános problémákra és javaslatokat ad a kijavításukra. A rendszer különböző módon reagál alacsony és magas terhelésre.

A teljesítmény akkor optimális, amikor a felhasználó zökkenőmentesen és bármiféle lassúlás nélkül tudja használni a rendszert.

Teljesítményvesztés oka lehet hardveres vagy szoftveres.

A rendszer folyamatosan számon tartja a háttérben futó feladatok erőforrástól elvett mennyiségét, ha valamelyik túl sokat használ, akkor elmenti a rendszer aktuális állapotát, majd megpróbálja a folyamatot leállítani, ezzel elérve az optimális teljesítményt. Erről a folyamatról a felhasználó értesítést kap.

A teljesítményvizsgálatot fontos elvégezni az adatbázisra és az operációs rendszerre egyaránt. A rendszert hatalmas mennyiségű munka terheli, annyira, hogy a rendszer már nem működik egy adott pillanatban. Ez lehetővé teszi számunkra a rendszer szűk keresztmetszeteinek azonosítását. Az adatbázis-tesztek során a leggyakoribb problémák a következők: A rendszerre gyakorolt nagy nyomás szerepet játszik az adatbázis-tranzakciók állapotának meghatározásában. Ha az adatbázis szoftver hibás elképzeléssel rendelkezik a tranzakciók állapotáról, akkor azonnal következetlenek lesznek. A vizsgálat elvégzése után a régi teszt adatokat meg kell tisztítani, és teret kell adni az új tesztadatokhoz. Az operációs rendszernél hasonlóan kell eljárni.

## Támogatottság

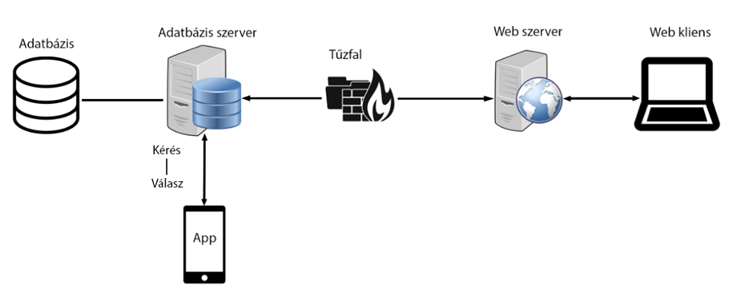
**Felhasználói támogatottság**: Biztosítjuk a feladatok ellátása közben kialakult hardveres és szoftveres hibák kivizsgálását és azonnali elhárítását.

Az **információk** különböző módon jelennek meg. Súlyos hiba esetén egy felkiáltó jel mellett piros, nagy betűkkel írja ki, ha probléma merült fel a rendszerrel. Kisebb hiba esetén narancssárga felkiáltó jel mellett narancssárga, közepes méretű betűkkel írja ki a hiba okát. További információkat fekete színnel írat ki a képernyőre.

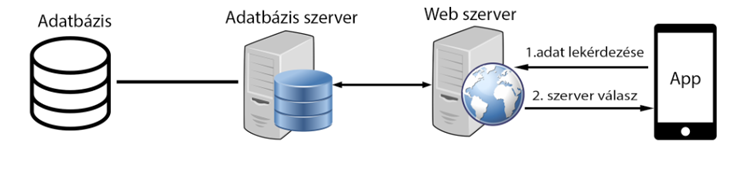
# Technikai specifikáció



## Architektúra terv



**10. ábra - Fejlesztői architektúra terv**



**11. ábra - Felhasználói architektúra terv**

A tervezett rendszer a következő összetevőkből áll:

* Adatbázis
* Adatbázis szerver
* Tűzfal
* Web szerver
* Web kliens
* Android/IOS készülék

Az architektúrában az adatforgalom szempontjából a következőképpen zajlik:

Az applikáció a web szerverrel kapcsolatba lép és lekéri a szükséges adatokat, innen kétféleképpen alakulhat. Az első eset, hogy a kérés eredményének meghatározására nem szükséges az adatbázis. Ebben az esetben a web szerver azonnal tud válaszolni. A másik eset, ha szükséges az adatbázis, ilyenkor a web szerver továbbítja a kérést az adatbázis szerver felé. Miután megkapta az adatbázisból a szükséges adatot, elküldi a web szervernek, végül pedig a szerver elküldi a választ az web kliensnek.

Gondolkodás után az Androidot és az IOS-t választottuk ki a megfelelő platformoknak. Azért erre esett a választás, mert az emberek így kényelmesen, bárhol, bármikor hozzáfér a szükséges információkhoz, csak egy telefonra vagy egy tabletre van szükségük.

A teljesítménytesztelés segít a rendszerek megfelelő karbantartásában és a hibák kijavításában, mielőtt a problémák elérnék a rendszerfelhasználókat. Segít fenntartani az alkalmazások hatékonyságát, válaszképességét, méretezhetőségét és sebességét az üzleti követelményekhez képest. Ha hatékonyan végez, a teljesítménytesztelésnek meg kell adni a szűk keresztmetszetek megszüntetéséhez szükséges diagnosztikai adatokat, amelyek gyenge teljesítményt eredményeznek. Szűk keresztmetszet akkor fordul elő, ha az adatfolyam megszakad vagy leáll, mert nincs elegendő kapacitás a számítási feladat kezeléséhez. A terheléstesztelés a rendszer teljesítményét méri a számítási feladatok növekedésével. Azonosítja, hogy az alkalmazás hol és mikor szakad meg, így az éles környezetbe való szállítás előtt kijavíthatja a problémát.

## Platform tulajdonságok és követelmények

**A rendszer futtatásához ajánlott hardverkörnyezet:**

Az alkalmazás használatához szükség van 200.0 MB szabad tárhelyre a készüléken.

**A rendszer futtatásához szükséges szoftverkörnyezet:**

A szoftver Android rendszer esetén az Android 12.0 Lollipop vagy annál újabb verzióval képes működni, míg Apple készülékek esetében az iOS 15.4 a követelmény. A szoftver egyidejűleg több felhasználó kiszolgálására is alkalmas.

Egy platform független **böngészőből is elérhető változata** is lesz az applikációnak, amely további szabadságot nyújt a felhasználóknak.

## Fejlesztői eszközök

Az alapvető platformunk a chat alapú kommunikációhoz az erre a célra létrehozott **Messenger** csoport. Itt folyik a mindennapos kommunikáció, és itt szoktuk összegyűjteni a kérdéseket és a megoldandó feladatokat.

A gyakorlati órák után tartott személyes megbeszéléseken, kívül minden vasárnap este tartunk online megbeszélést, amihez a **Discord** programot használjuk. Általában itt kerülnek elfogadásra az adott heti és kiosztásra a következő heti feladatok, továbbá itt vitatjuk meg a felmerülő kérdéseket is.

A megbeszélésekről szöveges dokumentáció készül a **MS Word** programmal. Ebben a programban történik a projekt dokumentum vezetése, szerkesztése is.

A dokumentumok kezelésére, tárolására az erre a célra létrehozott **GitHub** repository-t használjuk. Ide felkerülnek minden megbeszélés után a jegyzőkönyvek, illetve itt található a projekt dokumentum legfrissebb verziója is. A bevált gyakorlat szerint a projekttagok elküldik a munkáikat a projektvezetőnek, aki ezeket elfogadja és beilleszti a megfelelő formátumban a közös projekt dokumentumba.

A folyamatábrák és use case diagrammok a **Miro** program ingyenes, online verziójával készültek, a halszálka diagrammhoz **Photoshop**-ot használtunk.

Szoftveres háttérnek a **Visual Studio Code**-ot és a C++ programozási nyelvet választottuk.

Adatbázis tervezéséhez pedig egy nyílt forráskódú, webes adatbázis kezelő felületet, a **MSSQL Server Management Studio** és a **MSSQL**-t fogjuk alkalmazni.

# Gazdasági számítások



## Felmerülő költségek

* Fejlesztési költségek
* Karbantartási költségek
* Üzemeltetési költségek
* Technikai követelmények
* Jogi költségek
* Domain név bérlése
* Amortizáció

## TCO (Total Cost of Ownership)

A TCO-t magyarul egy termék valódi árának is nevezik. Röviden összefoglalva ez a szám egy rendszer teljes élettartamára vetített költségek összege, mely sok tényezőből épül fel.

**Munkabér:**

6 fő van a csapatban, átlagosan heti 10 órát foglalkozik egy csapattag a feladattal, valamint a projekt 1 félévig, azaz tételezzük fel, hogy 6 hónapig tart. A hónapokat átlagosan 4 hétnek vettük. A bruttó órabér legyen kb. 10.000 Ft. Ezen adatok alapján a teljes projektben a bérek költsége itt látható:

Bruttó munkabér = 6 \* 10 \* 4 \* 10.000 Ft/hó = 2.400.000 Ft/hó 🡪 6 hónapra: **14.400.000 Ft** bruttó költség

**Technológiai háttér:**

Mivel mind a VS Code mind a MSSQL Server Management Studio ingyenes szoftver, így ezek költsége **0 Ft**.

Amennyiben a vállalkozás növekedik szükség lehet új szerver vásárlásra, valamint egyéb váratlan költségek is felmerülhetnek, erre szánjunk br. **6.000.000 Ft**-ot a félév során.

**Egyéb költségek:**

Vannak bizonyos alapköltségek is, melyek a villamos energia, fűtés, víz, valamint iroda bérlés. Egy kisebb lakást, melyet irodaként be lehet rendezni kb. Győrben 150.000 Ft/hó + rezsiért lehet bérelni. Utóbbira számoljunk átlag 15.000 Ft-ot emberenként, összesen 90.000 Ft-ot. Így ez félévre vetítve: (150.000 + 90.000) \* 6 br. Ft = bruttó **1.440.000 Ft**

A fenti költségeket összesítve a CTO: 14.400.000 + 6.000.000 + 1.440.000 = **21.840.000 Ft**

## Tervezett bevételek

* Szponzorok szerzése, befektetők bevonása
* Reklámok megjelenítése az appon belül
* Terjeszkedés más településekre, más egyetemekre

## Fenntarthatóság növelése, költségek csökkentése

* Saját szerverpark üzemeltetése
* Saját domain hostolása

## Jövőbeli bevételi lehetőségek

* Prémium előfizetés lehetősége
* Szolgáltatások fizetőssé tétele
* Egyszeri és havidíjas szolgáltatások bevezetése
* Program bérbeadása

# Adatbázis terv



## Szöveges leírás

A féléves feladatunkban egy utasszállító applikációt készítettünk. Az adatbázisunkban eltároljuk, a sofőrök, utasok, és az útvonalak adatait. A fuvart követően az adatokat eltároljuk fél éven keresztül, ezek után pedig automatikusan törlésre kerülnek. Az adatok eltárolásába regisztrációnál a felhasználóknak bele kellett egyeznie. A törölt járatok esetén az adatokat el sem tároljuk az adatbázisba.

Alapvetően 5 tábla van: **User**, **Driver**, **Car**, **Passenger**, **Route**. A User tábla tárolni fogja a felhasználó ID-ját, a vezeték és keresztnevét külön-külön, a születési dátumát és a címét. A Driver tábla tárolni fogja a sofőr, az autó, és a felhasználó ID-ját külön. Továbbá a táblában a jogosítvány megléte és a sofőr esetleges büntetett előélete is el lesz tárolva. Emellett a sofőr jó, rossz, és összesített értékelése is külön el lesznek tárolva. A Car tábla tárolni fogja a kocsi ID-ját, márkáját, ezen belül a pontos modellt, a színét, a rendszámát és a rendelkezésre álló szabad helyek számát. A Route tábla tárolni fogja az útvonal és az autó ID-ját, az indulást és az érkezést, az út pontos dátumát és árát. A Passenger tábla tárolni fogja az utas, a felhasználó és az útvonal ID-jét.

Egyelőre ennyi adatot tárol az adatbázis, viszont az applikáció fejlődése során lehetőség lesz újabb típusú adatokat rögzíteni.

## Táblák

|  |
| --- |
| **Relációs séma** |
| USER(UserID, FirstName, FamilyName, DateOfBirth, Address) |
| DRIVER(DriverID, UserID, CarID, HaveDrivingLicense, CriminalRecord, PlusRate, MinusRate, Rate) |
| CAR(CarID, Brand, Model, Colour, LicensePlateNumber, FreeSeats) |
| ROUTE(RouteID, CarID, Departure, Destination, DateOfRoute, Price) |
| PASSENGER(PassengerID, UserID, RouteID) |

|  |
| --- |
| **USER tábla** |
| UserID [int] NOT NULL |
| FirstName [nvarchar](25) NOT NULL |
| FamilyName [nvarchar](25) NOT NULL |
| DateOfBirth [datetime] NOT NULL |
| Address [nvarchar](50) NOT NULL |

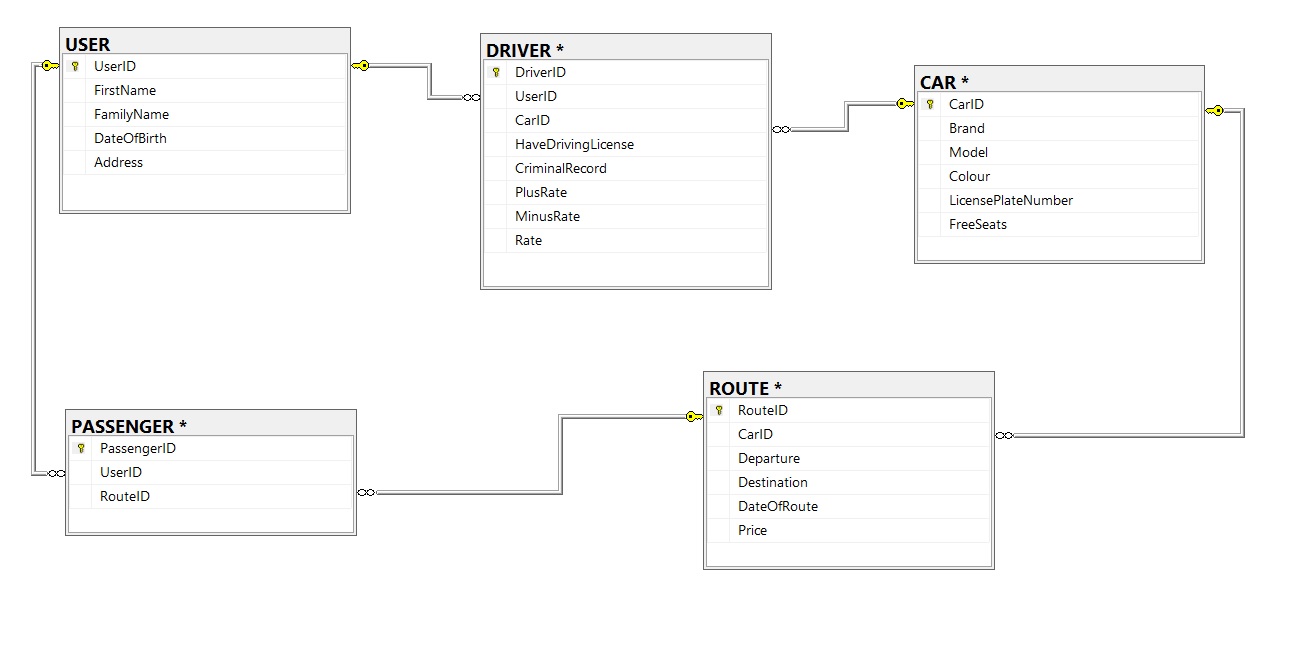
|  |
| --- |
| **PASSENGER tábla** |
| PassengerID [int] NOT NULL |
| UserID [int] NOT NULL |
| RouteID [int] NOT NULL |

|  |
| --- |
| **DRIVER tábla** |
| DriverID [int] NOT NULL |
| UserID [int] NOT NULL |
| CarID [int] NOT NULL |
| HaveDrivingLicense [bit] NOT NULL |
| CriminalRecord [bit] NOT NULL |
| PlusRate [int] NOT NULL |
| MinusRate [int] NOT NULL |
| Rate [float] NOT NULL |

|  |
| --- |
| **CAR tábla** |
| CarID [int] NOT NULL |
| Brand [nvarchar](15) NOT NULL |
| Model [nvarchar](15) NOT NULL |
| Colour [nvarchar](15) NOT NULL |
| LicensePlateNumber [nchar](6) NOT NULL |
| FreeSeats [nchar](1) NOT NULL |

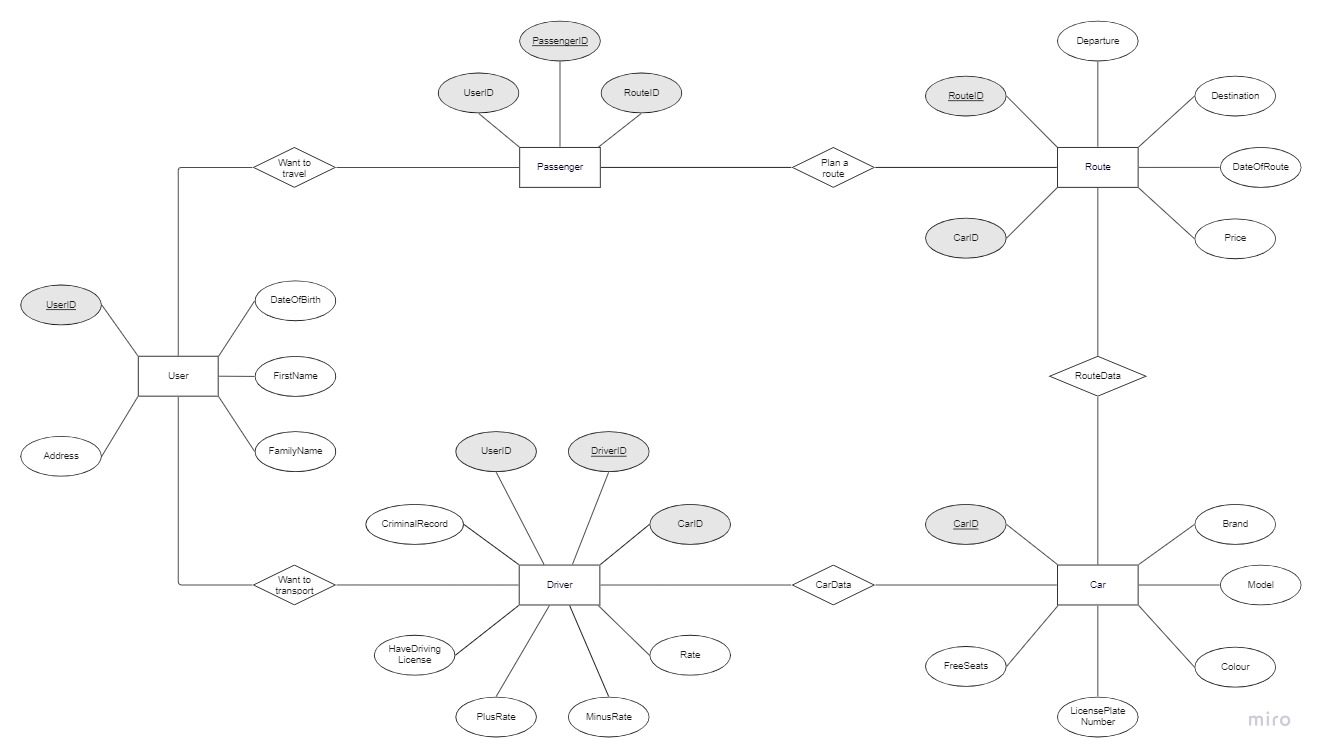
|  |
| --- |
| **ROUTE tábla** |
| RouteID [int] NOT NULL |
| CarID [int] NOT NULL |
| Departure [nvarchar](50) NOT NULL |
| Destination [nvarchar](50) NOT NULL |
| DateOfRoute [datetime] NOT NULL |
| Price [money] NOT NULL |

## Adatmodell



**12. ábra - Adatmodell**

## Egyed-kapcsolat diagram



**13. ábra - Egyed-kapcsolat diagram**

## Kulcsok és megszorítások



### Elsődleges kulcsok

**USER tábla:** UserID

**Driver tábla:** DriverID

**CAR tábla:** CarID

**ROUTE tábla:** RouteID

**PASSENGER tábla:** PassengerID

### Másodlagos kulcsok

**Driver tábla:** UserID, CarID

**ROUTE tábla:** CarID

**PASSENGER tábla:** UserID, RouteID

### Megszorítások

**USER tábla:**

* Az alkalmazás használójának minimum 18 évesnek kell lennie (jelenlegi dátum - születési dátum >=18)

**DRIVER tábla:**

* A jogosítvány megléte nem lehet régebbi, mint a születési dátum
* A bűnözési pontok száma nem lehet 10-nél nagyobb

**CAR tábla:**

* A szabad helyek száma nem lehet nagyobb, mint a férőhelyek száma (ha csak autók lesznek az adatbázisunkban akkor nem lehet pl. 5-nél nagyobb)
* A rendszámtáblán a rendszámnak megfelelő formátumban kell szerepelnie, azaz 3 betű kötőjel és 3 szám

**ROUTE tábla:**

* A célállomás nem lehet országon kívüli
* A fuvar ára nem lehet 6 számjegynél nagyobb jegyű
* Az indulási dátum nem lehet az aktuális dátumnál nagyobb

## Tárolt eljárások

Az adatbázisban tárolt eljárások a program üzemeltetését segítik.

* CreateUser: Az applikációban megadott adatok alapján automatikusan hoz létre egy új felhasználót. Az ID kiosztás is automatikusan történik.
* AddDriver: Első használatkor rákérdez az app, hogy utazni vagy utaztatni szeretnénk. A sofőr lehetőség választása esetén az applikáció felveszi a szükséges további adatokat és átadja az adatbázisnak, ami ezeket az adatokat automatikusan elmenti és ezek alapján létrehoz egy új sofőrt. Az ID kiosztás automatikusan történik.
* AddPassenger: Első használatkor rákérdez az app, hogy utazni vagy utaztatni szeretnénk. Az utas lehetőség választása esetén az applikáció felveszi a szükséges további adatokat és átadja az adatbázisnak, ami ezeket az adatokat automatikusan elmenti és ezek alapján létrehoz egy új utast. Az ID kiosztás automatikusan történik.
* AddCar: Amennyiben sofőrök vagyunk, akkor megadhatjuk az autónk adatait, a szín és a márka kötelező mezők, hogy beazonosítható legyen a jármű az utas számára találkozáskor, a többi adat megadása opcionális. Az applikációban megadott adatokat az adatbázis lementi és létrehoz egy új autót. Az ID kiosztás automatikusan történik.

## Továbbfejlesztési ötletek

Továbbfejlesztési ötleteknek olyan lehetséges alkalmazás funkciókat találtunk ki, amik megkönnyíthetik a SZEAT-APP későbbi használatát.

* **SelectCollege:** Mivel az app egy egyetemisták számára fejlesztett megoldás, így beépített funkcióként ki lehetne választani az indulási helyként, hogy melyik kollégiumi portától induljon a fuvar. Természetesen más opció is megadható, ez csak kényelmi megoldás.
* **CreateRoute:** Nem alapvető szükséglet, de beépíthető olyan funkció, mely grafikusan megjeleníti a start és a cél közti útvonalat a könnyebb átláthatóság kedvéért, becsült utazási időt is lehet vele számolni. A funkció megvalósításához lehetséges más térkép rendszerek bevonása.
* **CreateSchedule:** Hogyha valaki már előre tudja, hogy minden héten egy bizonyos időpontban utazna vagy tudna utasokat szállítani, akkor előre is felviheti a menetrendet és eszerint tud tervezni a többi sofőr vagy utas is.
* **RepeatRoute:** Mikor új útvonalat akarunk létrehozni, felajánlja a korábbi már felvitt utakat és akár azokból is választhatunk. Az adatokat automatikusan kitölti a rendszer, ezzel gyorsítva a folyamatot.

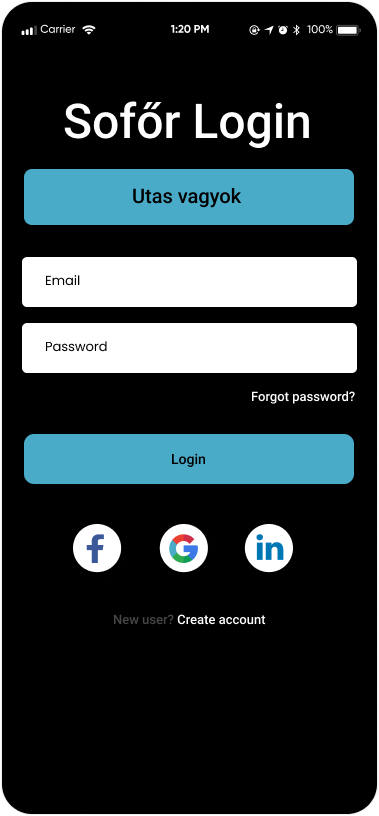
# GUI terv

Csapatunk a GUI elkészítéséhez a **Figma** szoftvert használta, GUI tervünk ezen a [linken](https://www.figma.com/proto/9dh5zvuCKOzIUUhD9acIUq/szeatapp?node-id=2%3A110&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=2%3A110) elérhető.

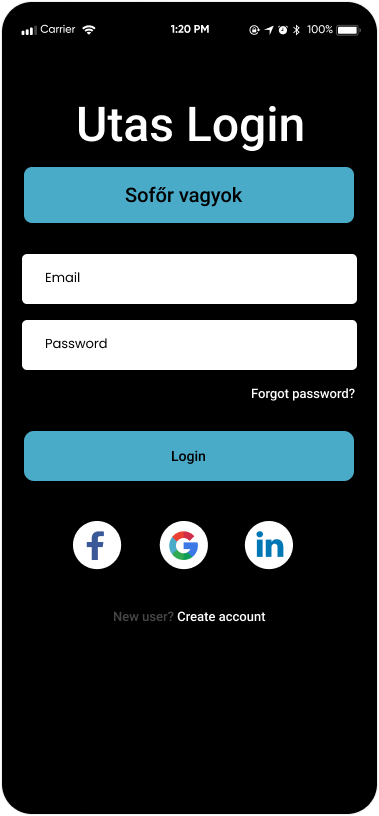
A képen szöveg látható

Automatikusan generált leírás

**14. ábra - Az applikáció nyitóképe, melyet belépéskor lát a felhasználó**



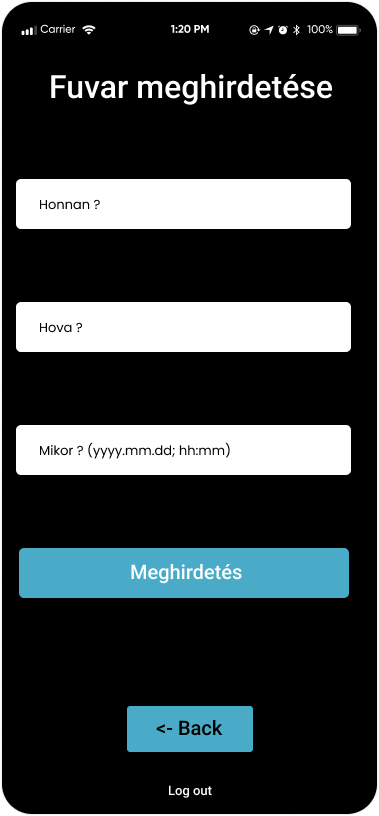
**15. ábra - Az applikáció sofőrök számára fenntartott belépési felülete**



**16. ábra - Az applikáció utasok számára fenntartott belépési felülete**



**17. ábra - A sofőrök számára fenntartott menü**

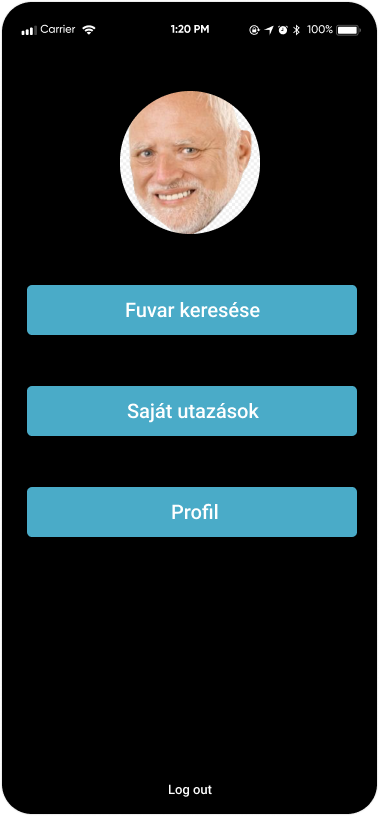


**18. ábra - Fuvarok meghirdetésének felülete**

A képen szöveg látható

Automatikusan generált leírás

**19. ábra - A sofőrök ezen a felületen áttekinthetik a saját, meghirdetett fuvarjaikat**



**20. ábra - Az utasok számára fenntartott menü**

A képen szöveg, kültéri, képernyőkép látható

Automatikusan generált leírás

**21. ábra - Az applikáció utasok számára fenntartott keresési felülete**

A képen szöveg látható

Automatikusan generált leírás

**22. ábra - Az utasok ezen a felületen láthatják az aktuális utazásaikat**

A képen szöveg látható

Automatikusan generált leírás

**23. ábra - Az utas ezen a felületen tud jelentkezni az adott fuvarra**

A képen szöveg látható

Automatikusan generált leírás

**24. ábra - Az utas ezen a felületen látja az aktuális utazásával kapcsolatos információkat**

A képen térkép látható

Automatikusan generált leírás

**25. ábra - Ezen a felületen tekinthető meg térképen az útvonal**