



Eötvös Loránd Tudományegyetem

Informatikai Kar

Informatikatudományi Intézet

Programozáselmélet és Szoftvertechnológia Tanszék

Utazásfoglaló webalkalmazás

Szerző:

Kelemen Dániel István

Programtervező informatikus BSc.

Témavezető:

dr. Nikovits Tibor

mesteroktató matematikus

Budapest, 2023

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Hallgató adatai:

Név: Kelemen Dániel István

Neptun kód: EX9BLV

Képzési adatok:

Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)

Tagozat : Nappali

Belső témavezetővel rendelkezem

Témavezető neve: dr. Nikovits Tibor

munkahelyének neve, tanszéke: **ELTE IK, Információs rendszerek Tanszék**

munkahelyének címe: **1117, Budapest, Pázmány Péter sétány 1/C.**

beosztás és iskolai végzettsége: **mesteroktató matematikus**

A szakdolgozat címe: Utazásfoglaló webalkalmazás

A szakdolgozat témája:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

Manapság egyre jelentősebb azoknak a külföldön dolgozó szakembereknek a száma, akik hétvégére szeretnének hazajönni a családjaikhoz. Ennek az életmódnak az egyik nagy nehézsége a rendszeres hazautazás megszervezése. A vonat vagy busz sok esetben nem jelent jó megoldást a hosszú utazási idő miatt, saját autóval egyedül utazva pedig rendkívül drága az útiköltség és fárasztó az út. A felmerülő szükségletekre kíván megoldást nyújtani egy kisbuszokat üzemeltető, fuvarokat szervező vállalkozás. A szakdolgozatomban egy olyan utazásfoglaló webalkalmazást szeretnék elkészíteni, amely ennek a vállalkozásnak a működtetéséhez lehet nagyon hasznos.

Az alkalmazás felhasználójának lehetősége lesz előre meghatározott városokon átmenő útvonalra, valamint a visszaútra helyet foglalni. Opcionálisan lehet majd regisztrálni és bejelentkezni, vagy bejelentkezés nélkül is történhet a foglalás. A foglalás során elsősorban időpont, kiinduló- és célállomás kiválasztására lesz lehetőség. Ha kivételesen nagy csomagot szeretne valaki szállítani, ennek jelzésére is lehetőséget ad majd az alkalmazás. A programot adminisztrációs felhasználóként használva, lehetőségünk lesz a foglalások több különböző szempont szerinti lekérdezésére. Az alkalmazás a megadott időpontok és prioritási elvek alapján összeállít majd egy útvonaltervet, hogy lehetőleg mindenkit a végcéljához minél közelebbre, akár a kapuig lehessen szállítani.

A webalkalmazás Apache webserveren fog futni, a megvalósítást HTML5 és CSS3 nyelvek használatával szeretném elkészíteni, PHP 8.2.12 kóddal kiegészítve. Az adatokat tároló és kezelő adatbázis MySQL lesz.

Budapest, 2023. 11. 28.

Tartalomjegyzék

1.	Bevezetés	1
2.	Felhasználói dokumentáció.....	2
2.1.	Kompatibilitás	2
2.2.	Webalkalmazás	2
2.2.1.	Foglalás	2
2.2.2.	Felhasználói fiókok	7
2.2.3.	Adminisztrátori felület.....	10
3.	Fejlesztői dokumentáció	15
3.1.	Programozási nyelvek	15
3.2.	Fejlesztési környezet	15
3.3.	Adatbázis	15
3.3.1.	user_data	19
3.3.2.	user_custom_stops	19
3.3.3.	routes.....	19
3.3.4.	stops	20
3.3.5.	vehicles	20
3.3.6.	trips.....	20
3.3.7.	bookings.....	20
3.4.	Háttérkód.....	21
3.4.1.	Program szintű konstansok.....	21
3.4.2.	HTML függvények.....	22
3.4.3.	Database osztály	24
3.4.4.	Map osztály	27
3.4.5.	Map JavaScript	30

3.5.	HTML struktúra és sablonok	30
3.6.	HTML form-ok és beágyazott PHP	31
3.6.1.	Foglalási felület	33
3.6.2.	Általános felhasználói felület	36
3.6.3.	Adminisztrátori felület	38
3.7.	Stíluslapok	40
3.8.	Tesztelés	41
3.8.1.	Database	41
3.8.2.	Map	43
4.	Összefoglalás	44
4.1.	Konklúzió	44
4.2.	További fejlesztési lehetőségek	44
4.2.1.	Fizetési mód és árak	44
4.2.2.	Fiók kezelés	44
4.2.3.	Adatbázis	45
4.2.4.	Egyéb fejlesztési ötletek	45
5.	Irodalomjegyzék	46
6.	Ábrajegyzék	47
7.	Táblázatjegyzék	49

1. Bevezetés

Manapság egyre több szakember keres és talál munkát külföldön, mint például Ausztriában vagy Németországban, azonban, ha családjaikhoz szeretnének időnként hazatérni, nehéz lehet ezt az utat kényelmesen megtenni. A legfőbb közlekedési módok az autó és a vonat, azonban mind a kettőnek van egy nagyobb hátulütője. Ha valaki az autó mellett dönt, végig kell vezetnie az utat, ami kimerítő lehet, és nagyobb csomagjait tetőcsomagtartó nélkül nehezen tudja szállítani. Amennyiben viszont vonaton szeretne utazni, már kisebb csomagokat, szerszámokat is nehéz lehet elpakolni, nem beszélve arról, hogy ezeket az indulásnál az állomásig, majd az állomástól a célig is el kell juttatni. Ezen problémákra ad megoldást a szállító cég, mely kisbusszal szállít embereket és csomagjaikat, akár háztól-házig.

A webalkalmazás legfőbb funkciója az utak foglalásának lebonyolítása, valamint az ezekkel járó logisztika szervezése. Általános felhasználóknak lehetősége van fiók létrehozására, mely megkönnyíti a foglalás menetét, valamint megtekinthetővé teszi korábbi foglalásait. Adminisztrátori fiókoknak emellett van lehetősége más adminisztrátorok felvételére, az utazások adatainak lekérdezésére, kezelésére, és újak létrehozására.

2. Felhasználói dokumentáció

2.1. Kompatibilitás

A weblap megnyitható és használható, operációs rendszertől függetlenül, az alábbi táblázatban (1. táblázat) leírt böngészőkben, vagy azok újabb verzióiban.

PC						Mobil				
Chrome	Edge	Safari	Firefox	Opera	Internet Explorer	Chrome (Android)	Safari (iOS)	Samsung Internet	Opera Mobile	Android Browser
10	12	10.1	4	10	10	125	10.3	4	12	4.4.4

1. táblázat: Böngésző kompatibilitás

2.2. Webalkalmazás

A weblapra érkezve a legfőbb funkciókat a felső navigációs sávból (1. ábra) érhetjük el.



1. ábra: Navigációs sáv

2.2.1. Foglalás

A foglalás megkezdéséhez a „Foglalás” fülre kattintva juthatunk el, ahol első lépésnek (2. ábra) az útvonalat kell kiválasztani. A foglalási folyamat előrehaladtát egy folyamatjelző sáv jeleníti meg, ami a cím alatt található.



2. ábra: Útvonal választása

A második és harmadik lépése (3. ábra) a foglalásnak, hogy kezdő- és célpontokat adjunk az utazásunknak. Ezeket lehetséges az előre megadott megállási pontok közül választani, vagy a Google Maps térképen elhelyezett kereső mezőben konkrét helyre keresni. Keresés esetén a térkép ezt a helyet meg is jeleníti. Ha a felhasználó be van jelentkezve, és már korábban mentett el saját megállási pontokat, ezek közül is tud választani. A három helymeghatározási opció közül a címkékre kattintva tudunk választani, amik zölden jelennek meg, ha éppen aktívak, ha pedig nem, a hozzájuk tartozó beviteli mező halványabb lesz.

INDULÁSI PONT

Fix megálló ✓

Budapest - Kelenföld vasútállomás

Előzőleg használt saját pont

Budapest, Markó utca 15

Saját indulási pont keresése

Map Satellite

Google Maps

Vissza Tovább

3. ábra: Megállási pontok választása

A negyedik lépésben (4. ábra) felajánl nekünk a weblap több opciót az utazás időpontjára növekvő időrendben, amelyek közül választhatunk. Zárójelben az utazásra kijelölt járművet is láthatjuk. Amennyiben nem jelenik meg semmi, egy hibaüzenetet kapunk, mely tájékoztat, hogy nincs kihirdetett utazás, vagy azok már beteltek.

IDŐPONT

A feltüntetett időpont a [Budapest, Kelenföld vasútállomás] megállóból való indulást jelenti.

2025. 4. 25. - 19:58 (Ford Transit Pullmann)

2025. 7. 30. - 20:00 (Ford Transit Pullmann)

2025. 7. 31. - 20:00 (Ford Transit Pullmann)

2025. 8. 1. - 20:00 (Ford Transit Pullmann)

2025. 9. 17. - 00:00 (Ford Transit Pullmann)

Vissza

4. ábra: Időpont választása

Az ötödik pont (6. ábra) az utasok számának és csomagok méreteinek megadását foglalja magába. Itt a megadandó adatok mellett megjelenik az is, hogy mennyi a jármű szabad kapacitása.

UTAZÓI ADATOK

Utazók száma
 / 3 fő

Csomagok súlya
 / 1950 kg

Csomagok térfogata
 / 4930 l

A járműre szerelhető tetőcsomagtartó. Használat igényét kérjük, a megjegyzésben jelezze.

Vissza Tovább

6. ábra: Utazói adatok megadása

FOGLALÓI ADATOK

Név

E-mail

Telefonszám

Vissza Tovább

5. ábra: Foglalói adatok megadása

A hatodik lépés (5. ábra) a foglaló adatait, elérhetőségét kéri a weblap. Ez a lépés a bejelentkezett felhasználóknak nem jelenik meg, és automatikusan a fiókhoz kötött információkat használjuk a foglaláshoz.

Az utolsó lépés (7. ábra) egy áttekintő lapot foglal magába, ahol bejelentkezett felhasználóknak lehetősége van elmenteni a megjelölt utazási pontokat későbbi használatra. Itt lehet megjegyzést is fűzni a foglaláshoz, ahol például a tetőcsomagtartó használatának igényét jelezhetjük. Ha minden adat egyezik, a „Lefoglalom” gombra kattintva véglegesíthetjük a foglalásunkat. Ha sikeres volt a foglalás, az átirányítás utáni oldal fogja jelezni.

ÁTTEKINTÉS

KONTAKT INFORMÁCIÓ	
Név:	Gipsz Jakab
Email:	gipsz.jakab@inf.elte.hu
Telefonszám:	06101112222

FOGLALÁS ADATAI	
Útvonal:	Budapest - München
Utazás dátuma:	2025. 04. 25.
Kiinduló pont:	Budapest, Kelenföld vasútállomás
Indulás várható ideje:	19:58
Célpont:	Linz, Hauptbahnhof
Érkezés várható ideje:	00:12
Személyek:	1 fő
Csomag adatok:	150 kg - 100 l

MEGJEGYZÉS ÍRÁSA

Szasztok, szeretnék egy 2,5m hosszú létrát szállítani a tetőcsomagtartón.

Vissza

Lefoglalom

7. ábra: Áttekintési lap és megjegyzés

2.2.2. Felhasználói fiókok

A regisztrációhoz (8. ábra) a jobb felső sarokban lévő felhasználó ikon alatt található a link. Erre a lapra érkezve a felhasználónak egy nevet, email címet, telefonszámot és jelszót kell megadnia, majd a jelszót megismételnie. A jelszó egy 4-64 hosszú karaktersorozat. Ezután a „Regisztrálok” gombra kattintva elkészíthetjük a fiókot.



The image shows a web form titled "REGISZTRÁCIÓ". It contains the following fields and buttons:

- Név: Text input field containing "Gipsz Jakab".
- E-mail: Text input field containing "gipsz.jakab@inf.elte.hu".
- Telefonszám: Text input field containing "06101112222".
- Jelszó: Password input field with masked characters ".....".
- Jelszó ismét: Password input field with masked characters ".....".
- Mégse: Light blue button.
- Regisztrálok: Dark blue button.

8. ábra: Regisztrációs lap

Amennyiben a felhasználó rendelkezik fiókkal, a regisztrációs fül mellett található „Bejelentkezés” menüpontra kattintva tud bejelentkezni. Itt az email cím és jelszó megadása után az oldal hiba, valamint sikeres bejelentkezés esetén is jelez (9. ábra).



9. ábra: Sikeres bejelentkezés

Bejelentkezett felhasználóként a felhasználó ikon alatti menüsor átváltozik, és 4 menüpont jelenik meg (10. ábra).



10. ábra: Felhasználói menü

A „Jelszó módosítás” menüpontban (11. ábra) a felhasználóhoz tartozó jelszót lehet módosítani, a jelenlegi, és egy új jelszó kétszeri megadásával. Mind hiba, mind sikeres jelszóváltás esetén visszajelez az alkalmazás.

The form has a dark blue header with the title "JELSZÓ MÓDOSÍTÁS" in white. Below the header, there are three input fields with labels: "Jelenlegi jelszó", "Új jelszó", and "Új jelszó ismét". Each field contains a series of dots representing masked text. At the bottom, there are two buttons: a light blue "Mégse" button on the left and a dark blue "Jelszó megváltoztatása" button on the right.

11. ábra: Jelszó módosítás

A „Fiók váltása” menüpont visszavisz minket a bejelentkezésre, ahol más fiókba át tudunk jelentkezni.

A „Kijelentkezés” menüpontra kattintva kiléptet a webalkalmazás a fiókból.

A „Felhasználó” menüpontra kattintva a fiók korábbi foglalásai és elmentett utazási pontjai jelennek meg. Az első táblázat az utazási pontokat tartalmazza, melyeket lehet törölni, vagy újat felvenni közéjük. Új utazási pont megadásához a táblázat felső sorában (13. ábra) kell egy útvonalat választani, majd egy helyet megadni, mely az adott útvonal alapértelmezett megállóitól bizonyos távolságon belül van, végül az „Új felvétele” gombra kell kattintani.

SAJÁT UTAZÁSI PONTOK			
ID	Útvonal	Cím	Műveletek
Siker: Cím sikeresen elmentve!!			
#	Budapest - München		Új felvétele
29	Budapest - München	Pázmány Péter sétány 1/c	Törölés

13. ábra: Új utazási pont felvétele

AKTÍV FOGLALÁSOK								
ID	Útvonal	Utazás dátuma	Kiinduló pont	Célpont	Személyek	Csomag adatok	Megjegyzés	Műveletek
59	Budapest - München	2024. 6. 21. - 17:06	Budapest - Kelenföld vasútállomás	Győr - vasútállomás	1 fő	20 kg - 50 l	Sziasztok, szeretném használni a tetőcsomagtartót. Egy 2,5m hosszú létrát szállítanék.	Törölés

LEJÁRT FOGLALÁSOK							
ID	Útvonal	Utazás dátuma	Kiinduló pont	Célpont	Személyek	Csomag adatok	Megjegyzés
Nincsen korábbi foglalásod							

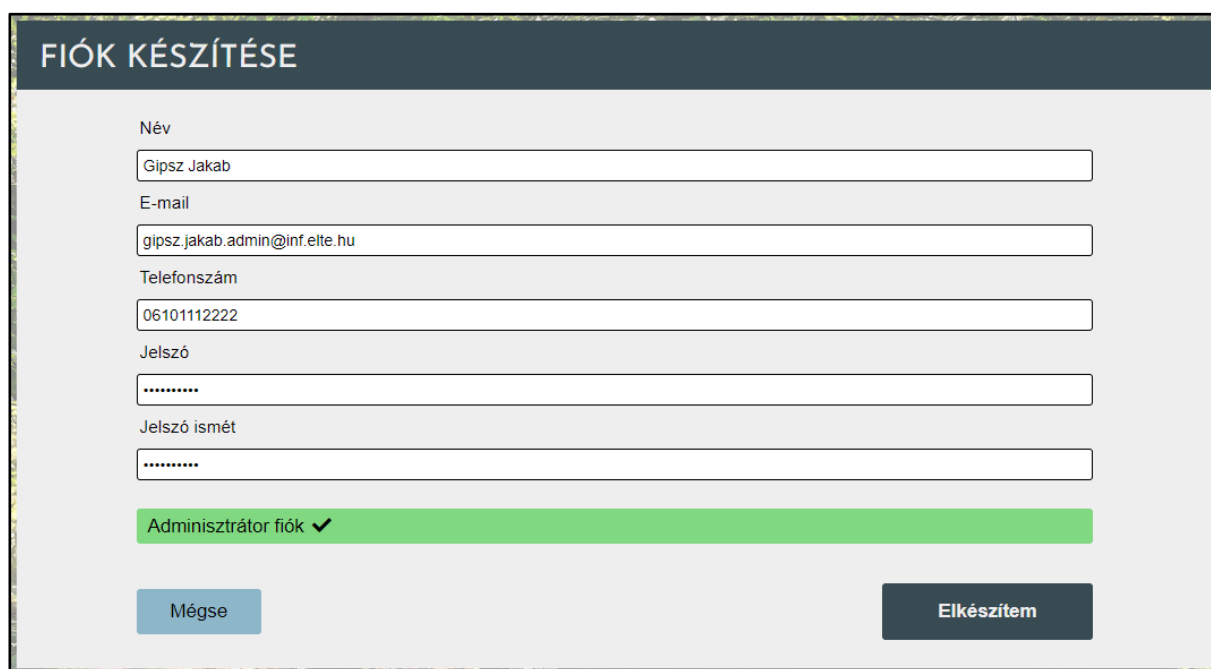
12. ábra: Saját foglalási adatok

A második táblázat (12. ábra) a fiókhoz kötött aktív foglalási adatokat tartalmazza. Amely esetben a felhasználó úgy dönt, hogy törölni kívánja a foglalását, a „Műveletek” oszlopban

lévő „Törlés” gombbal teheti ezt meg adott időhatáron belül. A harmadik táblázatban (12. ábra) a fiókhoz kötött, már lejárt foglalások tekinthetők meg.

2.2.3. Adminisztrátori felület

Adminisztrátori fiókok az alap funkciókon kívül 3 újabb menüt érhetnek el. A „Fiók regisztrálása” menüpontban egy regisztrációs lapot (14. ábra) lehet kitölteni, a „Regisztráció” oldaltól, azzal a különbséggel, hogy itt lehet más fiókot adminisztrátorként létrehozni.



FIÓK KÉSZÍTÉSE

Név
Gipsz Jakab

E-mail
gipsz.jakab.admin@inf.elte.hu

Telefonszám
06101112222

Jelszó
.....

Jelszó ismét
.....

Adminisztrátor fiók ✓

Mégse Elkészítem

14. ábra: Adminisztrátori fiók regisztrálása

A második adminisztrátori menü az „Útadatok” pont alatt található. Itt az első táblázatban (15. ábra) a kihirdetett utak vannak kilistázva, valamint a táblázat első sorában lehet új utat regisztrálni az útvonal, indulási idő és a jármű megadásával. A már életben lévő utak információit a „Részletek” gombra kattintva lehet megtekinteni, vagy amennyiben megadott időkereten belül vagyunk, és még nincs rá jelentkező utas, lehet őket törölni is. A második táblázatban (15. ábra) a régebbi utak információi szerepelnek, melyek részletei szintűgy elérhetőek.

UTAZÁSI ADATOK					
ID	Útvonal	Indulás ideje	Jármű	Jelentkezések	Műveletek
#	Budapest - München ▾	mm/dd/yyyy --:-- --	Ford Transit Pullmann (XYZ123) ▾	-	Új felvétele
48	Budapest - München	2024. 6. 21. - 17:06	Ford Transit Pullmann (XYZ123)	5	Részletek Törlés
53	Budapest - München	2024. 6. 5. - 13:06	Ford Transit Pullmann (XYZ123)	0	Részletek
2	Budapest - München	2025. 3. 21. - 13:02	Ford Transit Pullmann (XYZ123)	5	Részletek Törlés
43	München - Budapest	2025. 6. 20. - 09:00	Ford Transit Pullmann (XYZ123)	2	Részletek Törlés
3	Budapest - München	2026. 3. 26. - 13:32	Ford Transit Pullmann (XYZ123)	5	Részletek Törlés

LEJÁRT UTAZÁSI ADATOK					
ID	Útvonal	Indulás ideje	Jármű	Jelentkezések	Műveletek
54	Budapest - München	2024. 6. 4. - 13:17	Ford Transit Pullmann (XYZ123)	0	Részletek
52	Budapest - München	2024. 6. 3. - 13:06	Ford Transit Pullmann (XYZ123)	0	Részletek
56	Budapest - München	2024. 5. 17. - 13:18	Ford Transit Pullmann (XYZ123)	0	Részletek
31	Budapest - München	2024. 4. 26. - 18:02	Ford Transit Pullmann (XYZ123)	4	Részletek
42	Budapest - München	2024. 4. 26. - 07:00	Ford Transit Pullmann (XYZ123)	1	Részletek
9	Budapest - München	2024. 4. 18. - 10:30	Ford Transit Pullmann (XYZ123)	0	Részletek
5	München - Budapest	2024. 3. 26. - 12:27	Ford Transit Pullmann (XYZ123)	1	Részletek
1	Budapest - München	2024. 3. 1. - 09:33	Ford Transit Pullmann (XYZ123)	0	Részletek

15. ábra: Utazási adatok táblázata

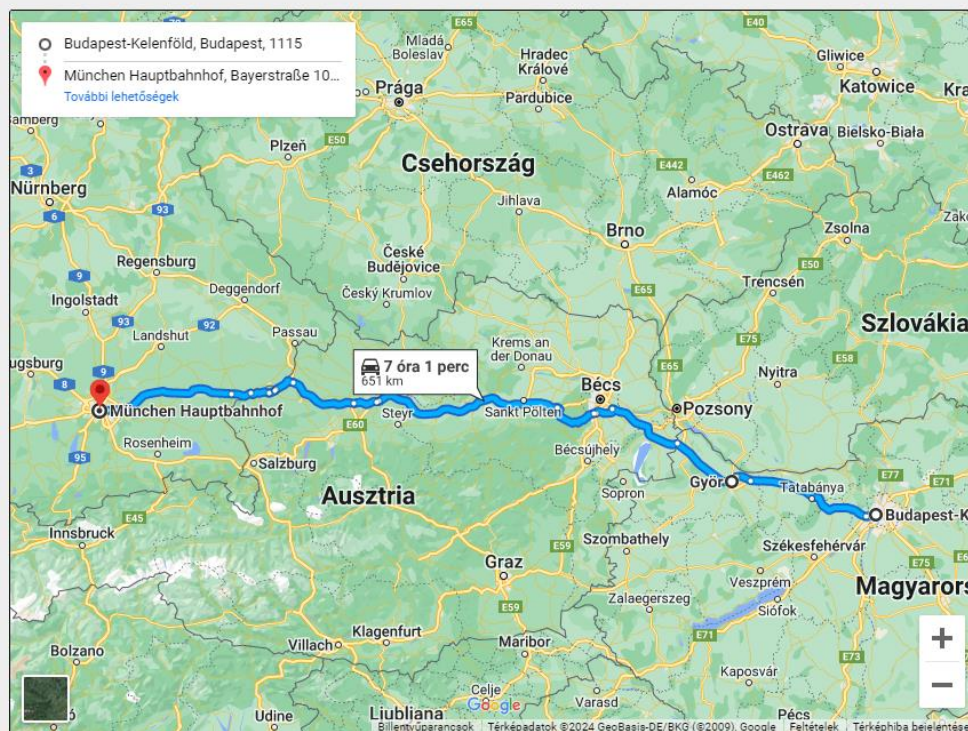
Az utak részleteire kattintva meg lehet tekinteni egy táblázatban (17. ábra) a jelentkezők adatait és a rendelésükhöz fűzött megjegyzéseket. Amennyiben van/vannak már foglalások az útra, egy útitervet (16. ábra) is készít az alkalmazás, mely a kiinduló pontból minden megállót érintve juttat el a legmesszebbi kijelölt pontra. Innen a „Vissza” gomb megnyomásával juthatunk vissza az Útszervezési adatokhoz.

ÚT FOGLALÁSI ADATAI

FOGLALÁSI ADATOK							
ID	Foglalási név	Elérhetőségek	Foglalás ideje	Kiinduló pont	Célpont	Személyek	Csomag adatok
51	Kelemen Dániel István	kelemen.daniel@knm.hu; 06303613777	2024. 5. 12. - 15:45	Budapest - Kelenföld vasútállomás	München - Hbf.	1 fő	0 kg - 0 l
55	Kelemen Dániel István	kelemen.daniel@knm.hu; 06303613777	2024. 5. 21. - 16:28	Budapest - Kelenföld vasútállomás	München - Hbf.	1 fő	30 kg - 30 l
56	Kelemen Dániel István	kelemen.daniel@knm.hu; 06303613777	2024. 5. 21. - 16:30	Győr - vasútállomás	München - Hbf.	3 fő	60 kg - 30 l
59	Gipsz Jakab	gipsz.jakab@inf.elte.hu; 06101112222	2024. 6. 4. - 13:21	Budapest - Kelenföld vasútállomás	Győr - vasútállomás	1 fő	20 kg - 50 l

17. ábra: Utazás foglalási adatai

ÚTITERV

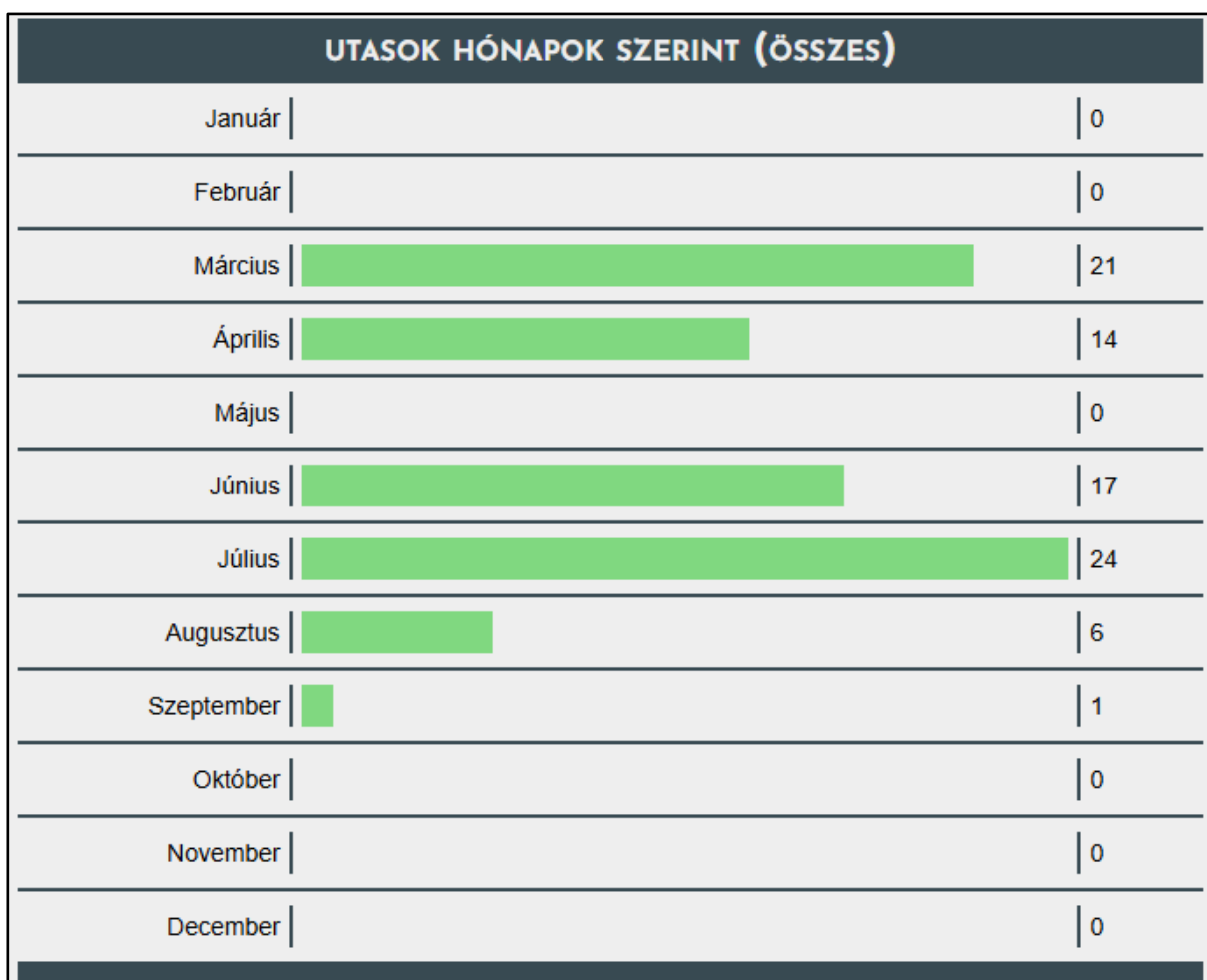


Vissza

16. ábra: Útiterv

A harmadik adminisztrátori menüpont a „Statistika”, ahol az utazásokról alkotott különböző statisztikai adatokat lehet megnézni diagramokon. A diagramok bal oldalán a kigyűjtési szempont van, középen egy arányosan méretezett csík vizualizálja az értékeket, amik a jobb szélén meg is vannak jelenítve. Az adatokat 3 nagy pont szerint gyűjti ki a webalkalmazás számunkra:

- Hónapok (18. ábra): hónapokra bontva meg lehet nézni, hogy mikor szoktak az utasok foglalni.



18. ábra: Hónap szerinti diagram

- Útvonalak (20. ábra): útvonalak szerinti felbontásban lehet megnézni merre utaznak az emberek.

UTASOK ÚTVONALAK SZERINT (ÖSSZES)		
Budapest - München	<div></div>	55
München - Budapest	<div></div>	28
Budapest - Krakkó		0
Krakkó - Budapest		0

20. ábra: Útvonalak szerinti diagram

- Megállók (19. ábra): az egyes útvonalakon itt lehet látni, hogy mely megállók a legforgalmasabbak, mind fel-, mind leszállás szempontjából.

MEGÁLLÓ FORGALOM (BUDAPEST - MÜNCHEN)		
Budapest, Kelenföld vasútállomás	<div></div>	44
Győr, vasútállomás	<div></div>	35
Wien, Hauptbahnhof	<div></div>	28
Linz, Hauptbahnhof	<div></div>	12
Salzburg, Hauptbahnhof	<div></div>	17
München, Hauptbahnhof	<div></div>	30

19. ábra: Megállók szerinti diagram

3. Fejlesztői dokumentáció

3.1. Programozási nyelvek

A webalkalmazás háttérkódja PHP 8.2.12 verzióban íródott, az adatok tárolásáért MySQL adatbázis felel. A megjelenítés egy HTML5 nyelven íródott webalkalmazáson keresztül történt, mely CSS3 kóddal lett kiegészítve.

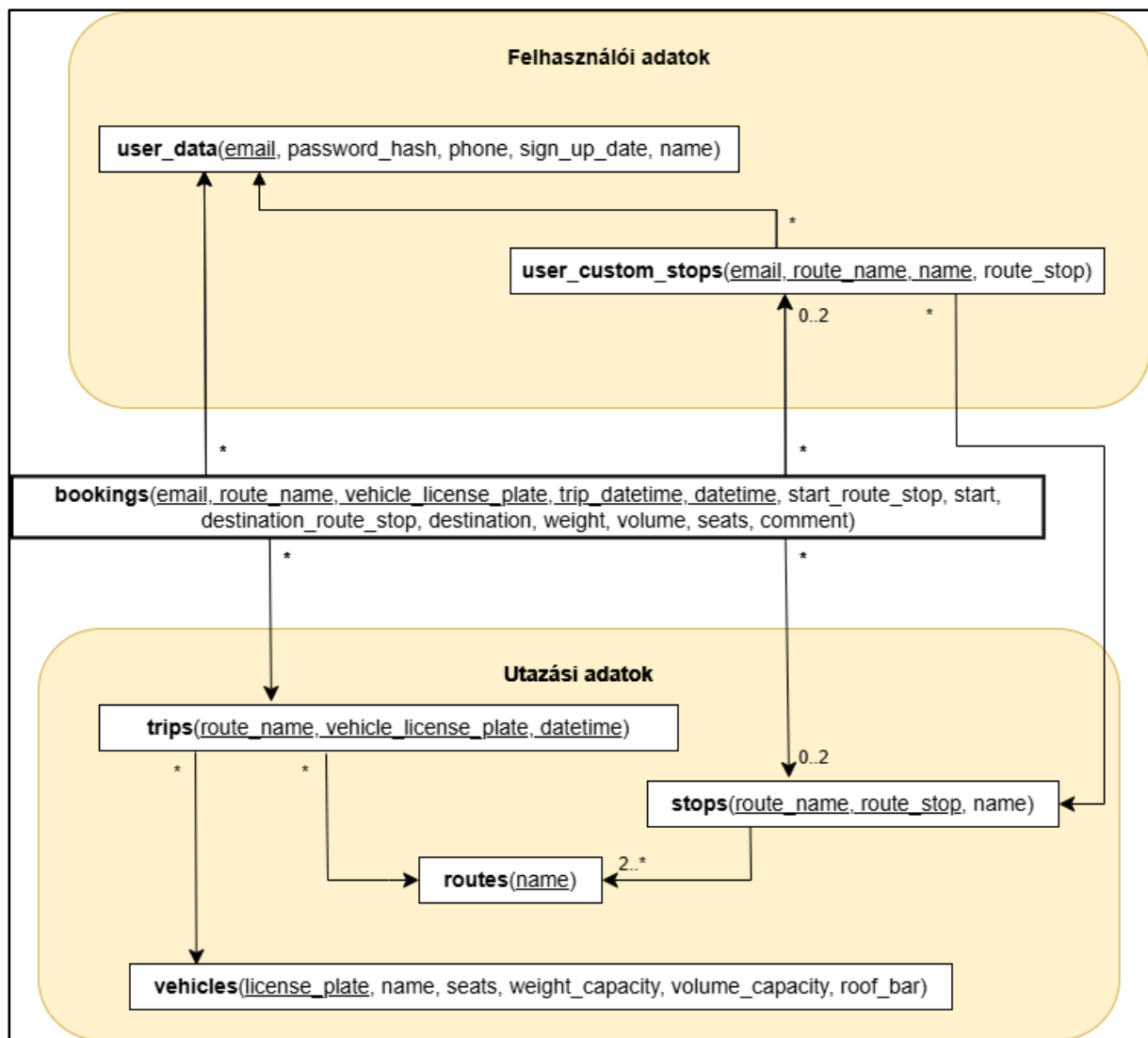
3.2. Fejlesztési környezet

A program Windows 10 operációs rendszeren, egy lokálisan futtatott szerveren került fejlesztésre. A szerver a XAMPP 8.2.12 verziója biztosította.

A fejlesztéshez és futtatáshoz elengedhetetlen az internetes kapcsolat is, ugyanis ezen keresztül tud a webalkalmazás a Google Maps API-al kommunikálni. A kommunikációhoz elengedhetetlen egy, a Google Maps API által generált, kulcs is, amely használata IP címhez van kötve.

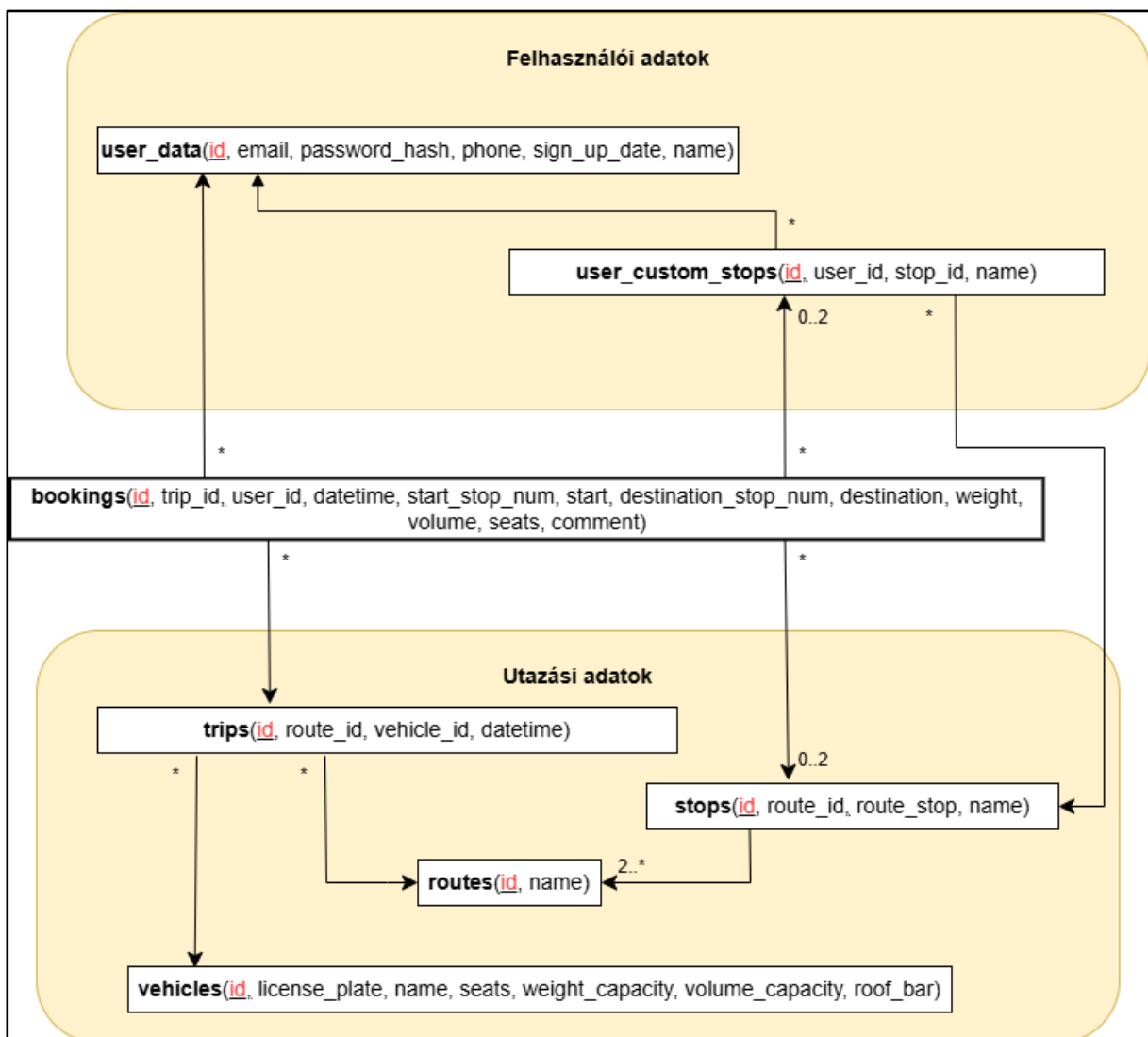
3.3. Adatbázis

A project háttérében egy MySQL alapú, MariaDB-ben megvalósított adatbázis felel az adatok tárolásáért. Az adatbázisban használt karakter mezők utf8mb4 karakterkódolásúak. A táblák 2 fő kategóriába sorolhatók: a felhasználói adatok tárolásáért, valamint az utazási adatok tárolásáért felelős csoportok.



21. ábra: Adatbázis terv (eredeti)

A megvalósítási tervek (22. ábra) szerint a táblák mindegyike rendelkezik egy-egy unsigned int típusú *id* mezővel, mely mindenhol elsődleges kulcsként funkcionál, és létezik rá index. Az eredeti adatbázis terveiben (21. ábra) ugyan nem így volt, a végleges verzióban több okból is ez a megvalósítási döntés maradt meg. A legfőbb indok a kulcsok kezelésének egyszerűsítése volt. Az eredeti tervekben ugyanis néhány helyen több oszlopból képeződött az elsődleges kulcs és ezzel az *id* mező nem volt szükséges, azonban ez az adatbázis esetleges bővítését, és az idegen kulcsok képzését is nagyban megnehezíti.



22. ábra: Adatbázis terv (megvalósítás)

2. táblázat: Rekordok méretei az egyes táblákban

Tábla neve	Maximális hosszúságú rekord mérete (byte)		
	Eredeti	Megvalósítás	Különbség
user_data	540	544	+4 (<1%)
user_custom_stops	286	101	-175 (65%)
routes	65	67	+2 (3%)
stops	131	70	-61 (47%)
vehicles	82	84	+2 (2%)
trips	80	12	-68 (85%)
bookings	915	719	-196 (21%)

Egy másik, kevésbé intuitív indok a tárolási hely spórolása. Az egyes táblák sorainak méretéből (2. táblázat) látszik, hogy ahol „pazarlás” volt az *id* mező bevezetése, annak kicsi mérete miatt nem nőtt lényegesen a teljes rekord mérete, azonban megengedte, hogy a többi táblában nagy mértékben csökkenjen. Ez a táblázat nyilvánvalóan torzítja a valóságot az igazi méretcsökkenéshez képest több okból is:

- Sok adat tárolásához varchar adattípust használunk, ahol a maximális byte méretet lehet megadni, amit nem minden rekord fog kihasználni, így nagyban eltérhet a nyert tárhely egyes rekordok között is.
- A táblákban nem azonos számú rekord kerül tárolásra. Például a legtöbb rekordot tartalmazó táblák a *bookings*, a *user_data*, a *user_custom_stops*, és a *trips*, amik összességében elég jelentős méretbeli csökkentést kaptak. A többi tábla előreláthatólag ritkán lenne írva, így ott elhanyagolhatóbb bármi féle változás.
- A táblák és rekordok mérete nagyrészt így is az adatbázistól és annak megvalósításától függ, így csak a fejlesztési szempontból egyszerűbben befolyásolható értékek szerinti optimalizáció van kigyűjtve a MariaDB hivatalos oldaláról^[1].
- Az elsődleges- és idegen kulcsokra készült indexek mérete is változik, hiszen a megvalósításra került verzióban ezeknek csak unsigned int-ekkel kell dolgozniuk, így a méretük csökken.

3.3.1. user_data

A felhasználók tárolásáért egy *user_data* nevű tábla felel. Az adatbázis első verziójában az *email* volt az elsődleges kulcs, azonban az *email* mező használata a túl nagy adatsűrűség miatt nem lett volna célszerű az idegen kulcsok használatára, így bevezetésre került az *id*, ami egyszerűsíti ezt. A felhasználói fiók típusát is az *id* határozza meg egy konstans szerint. A konstans értéknél kisebb *id*-vel rendelkező felhasználók adminisztrátori jogokkal rendelkeznek. Alapvető esetekben regisztrációkor nem kaphat fiók ilyen számú *id*-t.

A jelszót közvetlenül, biztonsági okok miatt, nem tárolja a tábla, csak egy titkosított változatát a *password_hash* mezőben, amit a PHP webalkalmazás készít el. Ez a mező csak akkor lehet *NULL* értékű, ha a „felhasználó” nem készített fiókot, hanem automatikusan lett létrehozva. Az ideiglenes fiók szerepe a későbbiekben *bookings* táblánál lesz kifejtve. Ezen fiókok beazonosíthatóak e-mail alapján, de bejelentkezésre nincs lehetőségük. Az email mezőre létezik index hatékonysági szempontból, mivel az *id* mellett ez a legtöbbet keresett mező.

3.3.2. user_custom_stops

A másik tábla a felhasználói csoportban a *user_custom_stops*, aminek az elsődleges kulcsát, mely a felhasználóra hivatkozó idegen kulcsból és a *name* mezőből állt, kiváltotta egy *id*. Amikor a *user_data id* mezőt kapott, az idegen kulcs a *user_id* mezőre változott, aminek segítségével a felhasználóhoz rendeli az egyedi megállási pontokat, amiket a foglalás során már használt és a továbbiakban használhat. A *name* mező tárolja a pontos hely nevét. A *stop_id* egy idegen kulcs, ami a *stops* táblára hivatkozik. Funkciója, hogy eltárolja, melyik megálló közelében van az adott hely, ami a programban nagyban könnyíti ezek használatát.

3.3.3. routes

A *routes* tábla egy *name* mezőből áll, és az útvonalakat tárolja. A páros számú *id*-hez tartozó *name* mező, mindig az 1-gyel kisebb *id*-jú út nevének megfordítottja, vagyis a retúrja. Ez akkor lesz biztosított, ha az adatbázisban található *add_new_route* nevű procedúrát használjuk új rekordok felvételére, ami 2 bemeneti helynévből helyes formátumú útvonalakat készít.

3.3.4. stops

A *stops* táblában az útvonalakhoz tartozó fix megállókat tároljuk. Itt található egy, a *route* táblára vonatkozó idegen kulcs, a *route_id*, ami mindig páratlan, azaz az útpárok között mindig az elsőre vonatkozik. Ezt a tulajdonságot a *STOPS_ROUTE_ID_ODD* nevű constraint biztosítja. A *route_stop* mező a *route_id*-vel hivatkozott út mentén a megálló sorszámát jelenti, azaz ez adja meg a megállók közötti haladási sorrendet.

3.3.5. vehicles

A *vehicles* táblában a rendelkezésre álló járművek adatait mentjük el. A *seats*, *weight_capacity* és *volume_capacity* a maximális teherbírást írják le, amit a foglaláskor és az útiterv készítésénél kell figyelembe venni. A *roof_bar* boolean arra van, hogy a tetőcsomagtartót jelezze. A *license_plate*-hez, mint egyedi mezőhöz, tartozik index.

3.3.6. trips

A *trips* tábla egy összekötő az utak (*routes*) és járművek (*vehicles*) közé, melyet 2 idegen kulccsal old meg (*route_id* és *vehicle_id*), valamint eltárolja azt is, hogy mikor indul az utazás a *datetime* mezőben.

3.3.7. bookings

A két nagy táblacsoportot összekötő tábla a *bookings*, ami a foglalások tárolásáért felelős. Ez a tábla egy utazáshoz a *trip_id*-hoz rendel egy felhasználói fiókot (*user_id*), vagy pedig egy ideiglenes felhasználót, egyéb adatok eltárolása mellett. Amennyiben bejelentkezett a foglalás megkezdése előtt a felhasználó, automatikusan a fiókjához lesz csatolva, ha pedig nem volt, a fiókadatok tárolását ugyan úgy a *user_data* táblában oldjuk meg ideiglenesen letárolva.

El kell tárolni még a foglaláshoz tartozó indulási és megállási információkat, amik egy megállóra vonatkozó idegen kulcsból (*start_stop_num*, *destination_stop_num*), és a tényleges megállási hely nevéből (*start*, *destination*) állnak. A *weight*, *volume* és *seats* a járműben lefoglalt helyeket jelentik, valamint a *comment* mezőben jelenik meg a megjegyzés, amit a foglaláshoz csatoltak.

3.4. Háttérkód

A háttérkód legfontosabb fájljai a *szallito_html/code* mappában találhatók.

3.4.1. Program szintű konstansok

```
1  <?php
2      namespace Consts
3      {
4          // path értékek
5          const CODE_PATH = __DIR__;
6          const DB_PATH = CODE_PATH . "\database.php";
7          const FUNC_PATH = CODE_PATH . "\html_functions.php";
8          const MAP_PATH = CODE_PATH . "\map.php";
9
10         // const értékek
11         const ADMIN_ID_UPPER_LIMIT = 999;
12         const MAX_DISTANCE_FOR_UNIQUE_LOC = 30_000;
13         const DAYS_TO_BOOKING_MODIFICATIONS = 1;
14         const DAYS_TO_TRIP_MODIFICATIONS = 5;
15         const MONTHS = [
16             "Január",
17             "Február",
18             "Március",
19             "Április",
20             "Május",
21             "Június",
22             "Július",
23             "Augusztus",
24             "Szeptember",
25             "Október",
26             "November",
27             "December"
28         ];
29     }
```

23. ábra: Consts névtér

A *consts.php* mappa tartalmazza a *Consts* névtérrel, ahol a webalkalmazás háttérkódjának legtöbb univerzális konstans értéke megtalálható (23. ábra).

A path konstansokban található a kód forrásfájljainak teljes útvonala, melyekkel be lehet őket *include*-olni. A const értékek alatt találhatóak a lényeges konstansok:

- **ADMIN_ID_UPPER_LIMIT:** egy pozitív integer, mely azt határozza meg, mi az a legnagyobb *id*, amivel egy felhasználó még adminisztratív jogokkal rendelkezik.
- **MAX_DISTANCE_FOR_UNIQUE_LOC:** egy méterben megadott érték, aminél távolabb egy megadott saját megállási pont nem lehet a tőle számított legközelebbi fix megállótól.

- DAYS_TO_BOOKING_MODIFICATION: napokban megadott érték. Ha ezen az adott időn belül indul el egy adott utazás, már nem lehet az arra vonatkozó foglalásokon módosítani (felvenni újat, törölni meglévőt).
- DAYS_TO_TRIP_MODIFICATION: napokban megadott érték. Ezen konstans az utazások módosítására vonatkozik. Egy új kihirdetésénél legalább ennyi időnek kell lennie a hirdetési és indulási idő között, valamint, ha már ennél kevesebb lenne, nem lehet az utazást törölni.
- MONTHS: a hónapok magyar neveit sorrendben tároló tömb. Ebben való indexeléssel lehet egy hónap sorszámát a nevére konvertálni.

3.4.2. HTML függvények

A FUNC_PATH konstanst követve eljutunk egy *html_functions.php* fájlba, amiben egyszerűbb kiszervezett függvények találhatók (24. ábra).

```
szallito_html > code > html_functions.php > ...
1  k?php
2  function get_if_set($array, $key, $return_if_not_found = ""):string
3  > { ...
10 }
11
12 function datetime_local_to_sql_datetime($date):string
13 > { ...
15 }
16
17 function sql_datetime_to_date($datetime):string
18 > { ...
23 }
24
25 function test_input($data = NULL):string
26 > { ...
37 }
38
39 function formerror($error):string
40 > { ...
42 }
43
44 function formsuccess($success):string
45 > { ...
47 }
```

24. ábra: HTML függvények

A *get_if_set* függvény 2 kötelező paramétert vár, amelyből az első egy, lehetőleg asszociatív, tömb, a második pedig egy kulcs. Amennyiben a tömbben megtalálható ez a kulcs, a mögötte lévő érték string-gé alakított változatát, ha pedig nem, a harmadik, opcionális paraméter értékét adja vissza.

A *datetime_local_to_sql_datetime* függvény egy HTML *datetime-local* típusú input elemének a kimenetét alakítja át SQL *datetime* formátumra.

Az *sql_datetime_to_date* függvény egy SQL *datetime* formátumú dátumot alakít át a webalkalmazásban megjelenítésre kerülő dátum formátummá.

A *test_input* függvény a HTML input mezőkbe írt értékek helyességéért és biztonságáért felel. Egy beadott string-ről először eltávolítja a fölösleges whitespace karaktereket, majd eltünteti a szövegből a „\” karaktereket, végül pedig a HTML nyelv speciális karaktereit alakítja át, így egy biztonságossá téve az inputot a későbbi használatra. A validálciós függvény a W3Schools ^[2] weboldaláról származik.

A *formerror* és *formsuccess* függvények HTML tárolókat hoznak létre dinamikusán, amikbe a paraméterül kapott szöveget írják ki. Ezek a függvények a hibák és sikeres akciók kiírására vannak.

3.4.3. Database osztály

A *Database* osztály (25. ábra) a database.php fájlban található, és a webalkalmazás és adatbázis közötti kommunikációért felel, ami a PHP által biztosított MySQL API [3] használatával működik.

Database
<ul style="list-style-type: none">- <u>HOSTNAME</u>: string {const}- <u>USERNAME</u>: string {const}- <u>PASSWORD</u>: string {const}- <u>DATABASE</u>: string {const}- <u>MAX_TRIES</u>: int {const}- <u>DATE_FORMAT</u>: string {const}- <u>db</u>: Database[]- <u>db_conn</u>: mysqli
<ul style="list-style-type: none">+ <u>instance()</u>: Database- Database()+ ~Database()+ <u>get_data_by_table_id</u>(\$table_name, \$id): array+ <u>signup</u>(string, string, string, string, bool): array+ <u>make_temporary_user</u>(string, string, string): array+ <u>get_user</u>(string): array+ <u>login</u>(string, string): array+ <u>change_password</u>(int, string, string): array+ <u>get_routes</u>(): mysqli_result+ <u>save_trip</u>(int, int, string): array+ <u>get_future_trip_info</u>(): mysqli_result+ <u>get_old_trip_info</u>(): mysqli_result+ <u>get_available_trips</u>(int, int, int): mysqli_result+ <u>delete_trip</u>(int): void+ <u>get_stops</u>(int): mysqli_result+ <u>get_stops_from</u>(int, int): mysqli_result+ <u>get_stop_by_custom_stop_id</u>(int): array+ <u>save_custom_stop</u>(int, int, string): array+ <u>get_custom_stop_info</u>(int): mysqli_result+ <u>get_custom_stops</u>(int, int): mysqli_result+ <u>delete_custom_stop</u>(int): void+ <u>get_vehicles</u>(): mysqli_result+ <u>get_remaining_seats</u>(int, int, int): int+ <u>get_remaining_weight</u>(int, int, int): int+ <u>get_remaining_volume</u>(int, int, int): int+ <u>get_has_roof_bar</u>(int): bool+ <u>book</u>(int, int, int, string, int, string, int, int, int, string): array+ <u>get_future_booking_info_by_user</u>(int): mysqli_result+ <u>get_old_booking_info_by_user</u>(int): mysqli_result+ <u>get_booking_info_by_trip</u>(int): mysqli_result+ <u>delete_booking</u>(int): void+ <u>get_stats_by_month</u>(date, date): mysqli_result+ <u>get_stats_by_routes</u>(date, date): mysqli_result+ <u>get_stats_by_route_stops</u>(date, date, int): mysqli_result- <u>get_next_admin_id</u>(): int- <u>get_new_user_id</u>(): int- <u>get_max_capacity</u>(int): array- <u>get_booked_capacity_from</u>(int, int, int): array

25. ábra: Database osztálydiagram

3.4.3.1. Konstansok

Az osztályban elhelyezett konstansok közül a `HOSTNAME`, `USERNAME`, `PASSWORD`, és `DATABASE` értékek az adatbázissal való kapcsolat kiépítéséhez kellenek. A `MAX_TRIES` értéke azt határozza meg, hányszor próbálkozhat újra a konstruktor az adatbázishoz való csatlakozáskor. A `DATE_FORMAT` a webalkalmazásban használt dátum formátumot tárolja, amivel az SQL átalakít saját dátumokat.

3.4.3.2. Konstruálás

Mivel az osztály kommunikációt folytat és nem tárol lényeges adatokat, a hatékonyság miatt egykeként lett implementálva, amelyért a statikus *db* nevezetű tömb és az *instance* függvény felel. A *db_conn* adattag a tényleges adatbázis kapcsolat, amelyen keresztül elérjük azt. Ez a konstruktorban kerül beállításra, ha nem történik a csatlakozás közben hiba.

3.4.3.3. Metódusok

Csak a bonyolultabb logikájú és jelentősebb metódusok vannak itt részletezve.

3.4.3.3.1 `get_data_by_table_id`

A *get_data_by_table_id* egy általános használatú metódus, amely egy tábla nevét és egy *id*-t kér. Ha létezik a tábla és benne egy ilyen *id*-vel rendelkező rekord, ezt visszaadja, ha nem, egy hibaüzenetet küld.

3.4.3.3.2 `signup`

A felhasználók adatainak kezelésére használatos függvények közül a *signup* a leglényegesebb. Először megnézi az email cím alapján, hogy létezik-e hozzárendelt fiók. Ha igen, és csak ideiglenes, mindössze frissíti az adatokat és rendel hozzá egy jelszót. Ha nem, az *admin* bemeneti paramétertől függően, aminek igaz értéke azt jelenti, hogy adminisztrátori felhasználót regisztrálunk, készít egy új *id*-t a felhasználónak, amihez utána az adatait rendeli és új rekordot készít. Ezután ellenőrzésnek lekéri az emailhez tartozó fiókot, ami, ha nem létezik vagy nem rendelkezik jelszóval, hibát dob, különben pedig visszaadja.

3.4.3.3.3 `get_available_trips`

Az utazások kezelését biztosító függvények közül a *get_available_trips* az egyetlen bonyolultabb logikával rendelkező függvény. Itt az *intersects_trip* változóban egy olyan SQL

feltétel szerepel, ami azt nézi meg, hogy a bemeneti paraméterekben megadott megállók között melyik foglalással fog az ember közösen utazni. Az útvonaltól függő variáció amiatt keletkezik, hogy a páros számúak retúr utak, tehát a megállók sorrendje fordított. Ezt a változót a belső SELECT (26. ábra) feltételben használja, ahol az általuk összesített, már lefoglalt helyek száma, nem érheti el a jármű kapacitását, mivel ekkor tele van és már nem lehet rá helyet foglalni.

```
>query("SELECT t.id id, date_format(t.datetime, "" . Database::DATE_FORMAT . ""') datetime
FROM (trips t) join (vehicles v)
ON t.vehicle_id = v.id
WHERE route_id = {route} AND datetime > ADDDATE(current_timestamp(), INTERVAL "" . Consts\DAYS_TO_BOOKING_MODIFICATIONS . "" DAY)
AND v.seats > (SELECT nvl(sum(b.seats), 0) FROM (bookings b)
WHERE b.trip_id = t.id
AND {intersects_trip})
ORDER BY datetime asc;");
```

26. ábra: *get_available_trips* lekérdezés

3.4.3.3.4 save_custom_stops

Egy felhasználó saját megállójának mentését a *save_custom_stops* metódus implementálja. Első lépésnek lekéri a beadott megállónévhez legközelebbi fix megálló adatait a *stop* változóba, és ellenőrzi, hogy nem-e ugyan az a neve, vagy túl közeli-e a megálló. Ezután a *custom_stops* változóba lekéri az adott felhasználó ugyan ezen fix megálló közelében mentett saját pontjait, amiket ugyan úgy ellenőriz, mint a *stop*-ot. Ha valóban nem redundáns a megállási pont, új rekordként elmenti azt.

3.4.3.3.5 book

A *book* metódus a megadott adatokkal létrehoz egy foglalást, azonban előtte ellenőrzi, hogy van-e még megfelelő mennyiségű hely az adott járaton.

3.4.3.3.6 get_next_admin_id

A privát metódusok közül a *get_next_admin_id* adminisztrátori fiók készítéséhez biztosít *id*-t, amit egy automatikus inkrementálással kap meg.

3.4.3.3.7 get_new_user_id

A *get_new_user_id* általános fiókokhoz készít *id*-t két random szám összeadásával, majd annak ellenőrzésével. A két random szám az adatbázis által tárolt érték bitszáma miatt kell, ami $[0, 2^{32})$ nagyságban tárol számokat. Az *mt_rand(min, max)* ^[4] függvény rendeltetett használata miatt a $max - min \leq mt_getrandmax$ egyenletnek fent kell állnia, viszont az *mt_getrandmax* értéke operációs rendszertől függetlenül $2^{31} - 1$. Ekkor tehát ha vesszük a

$mt_rand(ADMIN_ID_UPPER_LIMIT, mt_getrandmax) + mt_rand(0, mt_getrandmax) + 1 = [ADMIN_ID_UPPER_LIMIT, 2^{31}) + [0, 2^{31}) + 1 = [ADMIN_ID_UPPER_LIMIT + 1, 2^{32})$

számításokat, a végén helyes tartományban kapjuk az id értéket. Ezt utána leellenőrzi, hogy már használt-e, és addig próbálkozik, amíg nem talál egy szabad számot.

3.4.3.3.8 get_max_capacity

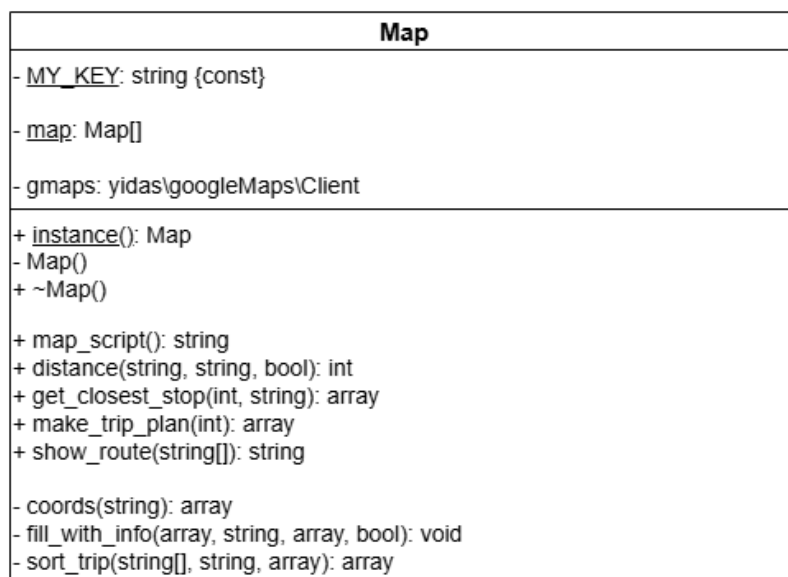
A *get_max_capacity* függvény egy utazáshoz rendelt jármű teljes kapacitását kérdezi le, ezt pedig egy asszociatív tömbbe letárolva adja vissza.

3.4.3.3.9 get_booked_capacity_from

A *get_booked_capacity_from* az utazás 2 adott pontja közötti úton már lefoglalt kapacitást adja vissza, a *get_available_trips* függvénnyel egyező módon.

3.4.4. Map osztály

A *map.php* fájlban definiált *Map* osztály (27. ábra) a Google Maps-re épülő függvények megvalósításával foglalkozik, amit a Google Maps Services for PHP^[5] (innenről csak Service) GitHubon elérhető, API-al való kommunikációért felelős kód segítségével tesz meg. A kód letöltéséhez és integrálásához Composer lett használva, ami a *vendor* mappában tárolja a Service forrásfájljait. A kódban, egy API-beli változás miatt, valamennyi változtatást végre kellett hajtani, így nem teljesen felel meg az eredeti változatnak.



27. ábra: Map osztálydiagram

3.4.4.1. Konstansok

A Google Maps API használatához kellő saját kulcsot az osztályon belül egy privát konstans adja meg, aminek `MY_KEY` a neve.

3.4.4.2. Konstruálás

Mivel az osztály nem tárol adatokat a Service kliensen (*gmaps*) kívül, a hatékonyabb kód érdekében egyke tervezési minta használatával van megvalósítva, amiért itt is a statikus *map* adattag és *instance* metódus felel.

3.4.4.3. Metódusok

A *map_script* metódus egy HTML script elemet szúr be a weboldalra, ami a helyválasztó térképek működése miatt szükséges.

3.4.4.3.1 distance

A *distance* függvény 2 kötelező paramétere a kezdő- és célpontok, valamint a *return_time_val*-ban jelezhető, hogy a távot nem méterben, hanem percekben mérje. Itt amennyiben nem felismerhető helynevet adtunk meg valahova, a *dist* tömb *status* kulcsa mögötti érték fogja jelezni, egyébként a kért formátumban adja vissza a két pont közötti távolságot.

3.4.4.3.2 get_closest_stop

A *get_closest_stop* függvény 2 bemeneti paramétere az útvonal azonosítója és egy hely neve, a visszatérési értéke a legközelebbi fix pontot meghatározó *id*, ha van ilyen, és a kettő távolsága. Először egy *stops* változóba lekérjük az összes útmenti fix megállót, majd egy minimum kiválasztást végzünk el a *distance* függvény segítségével, ha a hely felismerhető. Ha nem, egy *error*-t dobunk vissza.

3.4.4.3.3 make_trip_plan

A *make_trip_plan* a háttérkód leghosszabb és legkomplikáltabb függvénye, ami egy utazásból (*trip_id*) készít egy viszonylag optimális útitervet a kapacitás és egyéb tényezők figyelembevételével.

Előkészületként kigyűjti a *bookings*-ba a releváns foglalásokat, majd, ha van foglalás, a *base_stops*-ba a fix megállót, és a *vehicle*-be a használandó jármű adatait. Ezután egy *stops* asszociatív tömböt készít el, aminek a hossza a fix megállók számával egyezik, és azok

sorszámait benne a kulcsok. Az értékek újabb asszociatív tömbök, amikben a kulcsok olyan helyek nevei, ahol szállnak le vagy fel utasok, értékei pedig az azon megállókhöz tartozó összesített kapacitási adatok. Ezek az adatok újabb asszociatív tömbben vannak tárolva, 3 kulccsal (*seats*, *volume*, *weight*), amik értékei a megállóban le- és felszálló utasok által felszabadított autókapacitás összegei, amit a *fill_with_info* metódus segítségével készít el és számol ki. Ezek után a *first* változóban eltárolja az első fix megállót, a *from*-ban pedig azt, hogy mi annak a helynek a neve, ahonnan a következő optimális megállót kell keresni, ami kezdetben az első fix megálló. A *remaining_capacity*-ban az autó fennmaradó kapacitását összegezzük a továbbiakban.

A feldolgozás első lépésében megnézzük, hogy száll-e fel valaki az első fix megálló közelében. Ha nem, minden rendben, és innen indítja a járatot menetrend szerint, viszont, ha igen, megint kell egy esetet vizsgálnunk: Ha nem szállnak fel az első fix megállónál, fölösleges onnan indulni, tehát meg kell keresni az optimális kiinduló pontot, ami a legmesszebbi pont a második fix ponttól, hiszen, ha a legmesszebbi pontról indulunk, nem kell fölöslegesen kétszer végig menni a városon. Ha a *from*-ban eltároltuk már a végleges kiindulási pontot, akkor le kell vonni a megmaradó kapacitásból azt a helyet, amit az ott felszállók elfoglalnak, majd a *stops*-ból törölni ezt a megállót. Ezután létrehozuk az *ordered_stops* tömböt, amibe már sorrendbe rakjuk be a megállókat, és a függvény végén ebben adjuk vissza az eredményt is. Kezdetben csak a kiinduló pontot mentjük ide.

A feldolgozás további lépéseit egy ciklus végzi, ami minden fix megálló közelében rendezi az ottani megállási pontokat a *sort_trip* függvény segítségével, aztán pedig konkatenálja azok neveit. Ezután új kezdőpontot határoz meg, ami az eddig rendezett megállók legutolsója lesz.

3.4.4.3.4 *fill_with_info*

A *fill_with_info* egy privát függvény, ami az *array*-ben átadott tömb *key* kulcsát tölti fel az *info*-ból. Ehhez ellenőrzi, hogy van-e már definiálva ilyen kulcs, és ha nincs újat csinál, ha van akkor megfelelően összegez.

3.4.4.3.5 *sort_trip*

A *sort_trip* egy privát függvény, ami rekurzívan működik. Ha az átadott rendezendő tömb üres, visszaad egy üres tömböt. Ha nem üres, egy minimum keresést végez, a megállók és a *from*-ban átadott indulópont közti időbeli távolság alapján, figyelembe véve azt, hogy legyen

elég helye egy embernek felszállni a csomagjaival. Ennek mindig lesz helyes eredménye a jelentkezési procedúra miatt, ami nem enged olyan útra helyet foglalni, ahol nem fér el valaki a kocsiban. Ha kiválasztotta a legoptimálisabb megállót és belerakta a *min* változóba, a *remaining_capacity*-ba változtatja az értékeket, majd kiveszi az átadott tömbből azt. Ezután rekurzívan innen indít keresést a többi megállóval és kapacitással. Ha kész a maradék megálló rendezése, az elejére beszúrja a *min*-be mentett megállót és visszaadja a tömböt. Ez egy rekurzív megvalósítása a legközelebbi szomszéd algoritmusnak (nearest neighbour algorithm), viszont mivel a kiindulópont adott és nem véletlenszerű, egyszerűbben és hatékonyabban talál optimálisabb megoldást a megálló környékén belül.

3.4.4.3.6 show_route

A *show_route* egy kiegészítő metódusa a *make_trip_plan*-nek, mivel annak kimenetéből, de akár más módszerrel rendezett helyek tömbjéből is tud egy HTML *iframe*-ben útitervet megjeleníteni. Először egy ellenőrzést végez, mivel, egyértelműen nem lehet 2 pontnál kevesebből útvonalat készíteni. Ezután a Google Maps API-nak megfelelő formátumúra változtatja a helyneveket, majd az első és utolsó pont kivételével sorban összerakja őket egy *waypoint* string-be. A térkép közepét a kiinduló pontra állítja, a megadott *zoom* szintet beállítja, végül visszaadja a megjelenítendő elemet.

3.4.5. Map JavaScript

A *szallito_html/script* mappában található *map_script.js* az interaktív helykeresőért felel, amit a *foglalas_indulo_pont.php* és a *foglalas_celpont.php* használ. A script-ek a hivatalos dokumentációkból^{[6][7]} lettek összerakva, kisebb változtatásokkal.

3.5. HTML struktúra és sablonok

A weblapok HTML5 nyelvben íródtak, aminek az alap elemei az összefogó *html* tag-en belül a *body* és a *head*. A *head* általános metaadatokat tartalmaz, mint például az egyedi, azaz laptól függő részek: a *title* elem és esetleges stíluslapok; vagy az univerzális elemei, amik a *metadata.html* fájlban találhatóak meg (28. ábra). A karakterek kódolása, egyéb metaadatok és az általános stíluslapok ebben a fájlban vannak deklarálva, majd az egyes oldalakra beszúrva PHP kódon keresztül.

```

1 <meta charset="UTF-8">
2 <meta name="viewport" content="width=device-width, initial-scale=1.0">
3 <style>
4   @import url('https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@0,100..700;1,100..700&display=swap');
5   @import url("https://use.fontawesome.com/releases/v5.7.2/css/all.css");
6   @import url("css/style.css");
7   @import url("css/navbar.css");
8 </style>

```

28. ábra: metadata.html

A *body* tartalmazza a másik univerzálisan *include*-olt fájlt, a *navbar.php*-t. Ez a fájl egy *nav* elemet tartalmaz, amiben a felső navigációs sáv kerül megvalósításra. A statikus linkeken kívül vannak változó elemek is, amiket egy PHP kódrészlet (29. ábra) dönt el, hogy megjelenésre kerüljenek-e adott feltételek szerint:

- Ha nincs bejelentkezett felhasználó, csak bejelentkezni és regisztrálni lehet.
- Ha van bejelentkezett felhasználó, meg lehet tekinteni a saját felhasználói felületet, fiókot váltani (bejelentkezni) és kijelentkezni.
- Ha a bejelentkezett felhasználó adminisztrátor, az adminisztrátori felületeket is elérheti.

```

9      <?php if (!isset($_SESSION["usr"])): ?>
10         <a href="belepes.php">Bejelentkezés</a>
11         <a href="regisztracio.php">Regisztráció</a>
12      <?php else: ?>
13         <p><strong><?= $_SESSION["name"] ?></strong><br><i><?= $_SESSION["email"] ?></i></p>
14         <a href="felhasznalo.php">Felhasználó</a>
15         <?php if ($_SESSION["admin"]): ?>
16             <a href="admin_adatok.php">Útadatok</a>
17             <a href="admin_regisztracio.php">Fiók regisztrálása</a>
18         <?php endif; ?>
19         <a href="belepes.php">Fiók váltása</a>
20         <a href="kilepes.php">Kijelentkezés</a>
21      <?php endif; ?>

```

29. ábra: navbar.php kódrészlet

3.6. HTML form-ok és beágyazott PHP

A weblapok sok esetben egy kitöltendő *form* elembe kérik be a feldolgozásra és tárolásra váró adatokat. A *form* egyik kötelező paramétere a *method* megadása, ami legtöbb esetben *POST*-ra van állítva. Ez azt jelenti, hogy nem jelenik meg a küldött adat a weboldal URL-jében, ami abból a szempontból előnyös, hogy a felhasználó véletlenül sem tud átírni fontos keresési paramétereket. A *form* másik kötelező paramétere az *action*, ami a kitöltött adatok küldési célpontját adja meg. Ez alapesetben a saját oldal lesz, amit a `$_SERVER["PHP_SELF"]` szuperglobális változóval lehet megadni, kiegészítve a *htmlspecialchars* biztonságot biztosító függvénnyel, ami esetleges támadások ellen véd. A bekért adatok feldolgozása a lapon a

„Form processing” kommenttel ellátott PHP kódrészlet alatt történik. A kódhoz való belépési feltétel az esetek nagy többségében az, hogy a lap lekérési módszere *POST* legyen, és a küldés gomb megnyomási értékét is ellenőrizzük általában.

A lapok közötti adatmegosztást a PHP alapértelmezett, *\$_SESSION* nevű szuperglobális változójával van megoldva, ami egy asszociatív tömb. Ennek használata miatt minden lap rendelkezik az elején egy *session_start* függvényhívással, ami lehetővé teszi a változó elérését. A tömb specialitása, hogy felhasználói oldalanként egyedi, azaz alkalmas szinkronizációra sok felhasználó kiszolgálása esetén, és nem fognak köztük adatok összekeveredni.

Sok oldalon hiba lehet a bekért adatokban, és nem mindent fogad el a feldolgozás, valamint néhol vissza is lehet lépni előző oldalakra. Ennek megkönnyítése érdekében általában létrehozunk változókat, amik eltárolják a *form* által adott értékeket (31. ábra), majd visszaírják azokat a beviteli mezőkbe (30. ábra). Ezeket a változókat a *test_input* függvénnyel kérjük le a *\$_POST* vagy *\$_GET* tömbből, amiben alapértelmezetten érkeznek az input mezők értékei, így biztonságossá téve a bekérés folyamatát.

```
<?php // Form processing
    if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["login"]))
    {
        $email = test_input($_POST["email"]);
        $psw = test_input($_POST["psw"]);
```

31. ábra: Form adatok mentése

```
<label for="email">Email</label>
<input type="email" name="email" id="email" maxlength="128" autofocus required value="<?= $email ?>"

<label for="psw">Password</label>
<input type="password" name="psw" id="psw" minlength="4" maxlength="64" required value="<?= $psw ?>">
```

30. ábra: Mentett adatok betöltése

3.6.1. Foglalási felület

A foglalási felületen megjelenített adatok nagy része nem statikus, hanem az adatbázisból lekért értékeket jeleníti meg az egyszerű bővíthetőség szempontjából. Ezek úgy lettek elérve, hogy PHP kóddal, dinamikusan jelenítünk meg HTML elemeket az adatbázisban tárolt adatok segítségével, általában ciklusba zárva.

A foglalási procedúra alatt a `$_SESSION` tömb *booking* kulcsú elemébe tároljuk a kinyert adatokat, ami maga is egy asszociatív tömb. A folyamat során minden lap első ellenőrző feltétele a *booking* tömb és az előző lapon megadott adatok megléte. Amennyiben valami nincs rendben, a program visszaugrik az előző lapra a hibák elkerülése érdekében.

A foglalásnál lehetősége van a felhasználónak visszalépni is, a `$_SESSION[„booking”]` tömbből tölti vissza az elemeket az input mezőkbe.

3.6.1.1. foglalas_utvonal.php

A *foglalas_utvonal.php* a *Database* osztály *get_routes* metódusát használva kilistázza az összes mentett útvonalat, amiket egy-egy *button* elembe helyez, melyek típusa *submit*. Ezeket az *id* kulccsal különbözteti meg és a rájuk írt szöveg a *name* mezőből származik.

A *form* feldolgozása a kiválasztott *id route_id* kulcsba való eltárolását foglalja magába, majd a felhasználót a következő lapra lépteti. Ellenőrzésekre nincsen szükség, hiszen nem lehet rossz adatot megadni.

3.6.1.2. foglalas_indulo_pont.php és foglalas_celpont.php

A következő 2 lépés, a *foglalas_indulo_pont.php* és *foglalas_celpont.php* kódja nagy részben egyezik egymással. Előkészületi munkaként beállítja a pont típusát, ami *station* lesz, ha még nem volt megadva semmi. Ezek után a *stops* változóba kigyűjti a lehetséges megállási alappontokat az adatbázisból:

- Az induló pontnál ez bármi lehet az útvonal mentén, kivéve az utolsó megállót, amit le is vesz a *last_stop* változóba.
- A célpontnál csak olyanok lehetnek, amik az előzőleg megadott induló pont és a végállomás között helyezkedik el.

Végző előkészületként a *search* változónak ad alapértéket.

A HTML *form* felépítése a két lapnak egyezik. 3 darab, egymást kizáró, *radio* osztállyal ellátott div elemből és a hozzájuk csatlakozó választási metódusokból áll:

- A *station* az alapértelmezett megállási pontokat szimbolizálja és egy *select* elembe felsorolja azokat, ahol választani lehet közülük.
- A *precustom* csak akkor jelenik meg, ha van bejelentkezett felhasználó és van hozzárendelve a fiókhhoz olyan elmentett megállási pont, ami az adott útvonalhoz kötődik. Ha ezek teljesülnek, itt is egy *select* elemben lehet közülük választani.
- A *custom* opcióhoz tartozik egy text input elem, ami egy kereső, valamint a *map* osztályú *div*, ami a *map_script* segítségével egy interaktív Google Maps helykeresőt hoz létre.

A feldolgozás menete a megadott típus mentésével kezdődik, majd a kiválasztott típushoz megadott ellenőrzéseket kell végezni:

- *Station* esetén nincs dolgunk, feltételezhetjük, hogy az adatbázisban hiba nélkül szerepelnek az adatok.
- *Precustom* esetén meg kell vizsgálni, hogy az egyedi állomáshoz tartozó *route_stop* mező benne van-e az elvárt tartományban, azaz induló pont esetén nem az utolsó, és célpont esetén valóban az induló pont és a végállomás között helyezkedik el.
- *Custom* esetén először futtatja a *test_input* függvényt, majd megnézi, hogy volt-e kiválasztott hely, ha igen, akkor azt is, hogy a Google Maps felismeri-e ezt a helyet. Ha ezek teljesülnek, kiszámolja, hogy a legközelebbi fix megállópont hol van, és ellenőrzi, hogy az elég közel van-e. Végző ellenőrzésként itt is meg kell vizsgálnia, hogy megfelelő tartományba esik-e, mint a *precustom* opció esetében.

Az ellenőrzések elbukásának esetén egy hibaüzenetet küld, míg jó adatok bevitele esetén azokat a megfelelő kulcsokba tárolja és a felhasználót a következő lapra lépteti a program.

3.6.1.3. foglalas_idopont.php

A *foglalas_idopont.php* fájl lekérdezi azokat az időpontokat, amikben van még hely a kijelölt járműben a kijelölt utazási pontok között. Ezeket az útvonalakhoz hasonlóan *button* elemekbe rakja, majd feldolgozásnál elmenti az *id*-t. Amennyiben nincsen már szabad hely egy járaton sem, vagy nincs az adott útvonalra meghirdetve járat, hibaüzenetet jelenít meg.

3.6.1.4. foglalas_utazo.php

A foglalas_utazo.php lekéri és megjeleníti az input mezők mögött az adott időpontban történő utazáshoz kapcsolódó megmaradó kapacitási adatokat a járműben, valamint, hogy van-e opcionális tetőcsomagtartó. Ezek a számok kizárólag nemnegatívak és az ülések száma nagyobb vagy egyenlő 1-gyel, amit az foglalas_idopont.php által használt *get_available_trips* metódus garantál. Az értékeket ezután egy-egy input elem *max* tulajdonságához rendeli, hogy a felhasználó mindenképpen csak megfelelő értékeket tudjon megadni.

A *form* feldolgozása az adatok eltárolásából és a továbblépésből áll, további ellenőrzés nem szükséges.

3.6.1.5. foglalas_foglalo.php

A foglalas_foglalo.php egy opcionális lépés, amit csak akkor kell kitöltenie a felhasználónak, ha nincs fiókkal bejelentkezve. Ha ez így lenne, a program azonnal tovább léptet.

Előkészületi munkaként beállítjuk az *name*, *email* és *phone* változók alapértékeit.

A feldolgozás első lépése a *test_input* futtatása a biztonság miatt, majd lekéri, hogy van-e ilyen létező felhasználó az e-mail cím alapján. Ha van teljes körű felhasználói fiók hozzárendelve, megnézi, hogy egyeznek-e az adatok, és ha nem hibát dob. Ha csak ideiglenes fiók van hozzárendelve, vagy egyáltalán nincs, akkor csinál az adatokból egy új ideiglenes fiókot és annak az adatait tárolja el, majd lép tovább.

3.6.1.6. foglalas_attekintes.php

Az utolsó foglalási pontban, a foglalas_attekintes.php lapon a program a *booking* tömbbe mentett adatokat kiírja egy táblázatba, és ha be van jelentkezve a felhasználó és voltak *custom* pontok, megjeleníti egy *checkbox* input elemet, ami az elmentésükre szolgál. Kapnak a felhasználók egy opcionálisan kitölthető megjegyzés *textbox*-ot is.

A feldolgozásnál a kommentet a *test_input* függvényen átfuttatja, majd, ha ki volt pipálva a *checkbox*, elmenti az utazási pontokat és megpróbálja a foglalást véglegesíteni. Mivel az adatok bevitele és a véglegesítés között nem elhanyagolható idő telik el, és a weboldalnak szinkronban kell lennie a felhasználók között, a *book* függvény is ellenőrzi, hogy van-e még megfelelő mennyiségű hely a kijelölt járműben, és hibaüzenettel tér vissza, ha a foglalási kísérlet megghiúsult. Az eredménytől függetlenül azonnal kidob az oldal az index.php-ra.

3.6.2. Általános felhasználói felület

3.6.2.1. index.php

Az oldalon egy rövid bemutatkozás mellett az útvonalakat egy ciklus listázza ki az adatbázisból, és azokat a *show_route* metódussal jeleníti meg. Itt jelenik meg üzenet, ha a belépés sikeresen végződött, és a foglalási folyamat eredménye is.

3.6.2.2. regisztracio.php

A regisztráció során a név, az email, a telefonszám, valamint a jelszó és annak megismétlése a kötelezően kitöltendő input mező. Az oldal az adatok elküldése után ellenőrzi azok meglétét és helyességét. Ha a két jelszó nem egyezik, az ismételt jelszót törli. Ha mindent rendben talál, a Database osztály *signup* metódusával megpróbál létrehozni egy felhasználói fiókot az adott email cím alapján. Ha az email már használatban volt hibát dob, ha pedig ideiglenes felhasználóként volt mentve, felülírja annak adatait. Sikeres regisztráció esetén azonnal be is jelentkeztet a fiókba, és egy üzenetet is kiír az index.php-n, ahova átírányítja a felhasználót.

3.6.2.3. belepes.php

A regisztrációnál megadott email és jelszó alapján megpróbál belépni az email alapján azonosított felhasználói fiókba. A jelszó hashelt változatainak egyezése esetén sikeresen belép a felhasználó, amiről az index.php-n megjelenő üzenet tájékoztat, ahova automatikusan át lesz irányítva. A belépési adatokat a *\$_SESSION* szuperglobális tömbbe tárolja, a *name*, a *usr*, az *admin*, és az email kulcsokba. Amennyiben a felhasználó átjelentkezett egyik fiókból egy másikba, az adatok felül lesznek írva. A legtöbb esetben innentől a webalkalmazás a *\$_SESSION[„usr”]* elem meglétének ellenőrzésével fogja megállapítani, hogy van-e bejelentkezett felhasználó.

3.6.2.4. jelszo_modositas.php

A *psw*, *new_psw* és *new_psw_r* inputok kötelezően kitöltendőek.

Amennyiben a két új jelszó egyezik, meghívja a *Database change_password* metódusát. Hibát és sikert is megjelenít az oldal.

3.6.2.5. kilepes.php

Egy nem „látogatható” oldal, azaz nincs HTML kód benne, és csak arra szolgál, hogy kijelentkeztesse a felhasználót a fiókjából. Ezt a *session_destroy* függvényhívással teszi, ami az eddigi *\$_SESSION* szuperglobális tömbbe helyezett adatokat törli. Ez az operáció megszünteti az eddigi foglalási folyamatot is. Ezek után visszacob a kezdőoldalra.

3.6.2.6. felhasznalo.php

A HTML kódban mind a 3 itt megjelenő tábla *foreach* ciklussal és a *Database* osztály egyes függvényeinek használatával listázza ki a releváns adatokat. Amennyiben nincs még a táblához adat, azaz a lekérdezés 0 sort adott vissza, a *foreach* ciklusba nem lép be a kód, csak egy előre definiált üzenetet jelenít meg.

A 3 tábla egy form elemben van benne, aminek különböző beadási módjai vannak, aminek értéke a megnyomott gombon múlik. A mód mögötti érték mindig egy *id*, ami a gombhoz kapcsolódó adatnak az adatbázisbeli elsődleges kulcsa.

A feldolgozó kód eldönti az mód alapján, hogy mit akart a felhasználó, és aszerint fog ellenőrzéseket tenni. 3 félélt különböztetünk meg:

- *delete_booking*: Az aktív foglalások táblában az adott sor melletti „Törlés” gombbal lehet törölni a foglalást, azonban megerősítést is kér az oldal az operáció elvégzése előtt a beépített *confirm* függvény hívásával (32. ábra).
- *delete_stop*: A saját utazási pontok közül lehet ezzel a móddal törölni, azonban itt is megerősítést kér az oldal a törlés elvégzése előtt.
- *new_stop*: A felhasználónak lehet saját utazási pontokjaihoz – a foglalási folyamaton kívül is – ezzel felvenni új megállót. Az *input* mezőkkel ellátott táblázati sor melletti „Új felvétele” gomb hívja meg. A *stop_name input* mező kitöltöttségét ellenőrzi a program először, majd beállítja az útvonal hivatkozását megfelelő formátumra, azaz a páros *id* (visszaút) megadása esetén, levon egyet és a páratlanra állítja. Ezután a *Map* osztály *get_closest_stop* függvényével kikeresi a korrigált útvonalról a legközelebbi alapértelmezett megállási pontot, és ellenőrzi, hogy elég közel van-e hozzá a *stop_name*-ben megadott hely. Ha minden feltételnek megfelelt a megadott utazási pont, elmenti a *Database save_custom_stop* metódusával, majd a sikerről

üzenetet ír ki a *formsuccess* függvénnyel. A mentés megghiúsulása esetén a *formerror* függvény jelenít meg hibaüzenetet.

```
<button class="clickable" name="delete_booking" value="<?= $booking["id"] ?>"
onclick="return confirm('Biztosan törölöd a foglalásod?')"> Törlés </button>
```

32. ábra: *confirm* függvény

3.6.3. Adminisztrátori felület

Adminisztrátori felületnek azokat az oldalakat nevezzük, amelyek eléréséhez be kell legyen jelentkezve felhasználó, és a fióknak adminisztrátori jogokkal kell rendelkeznie.

Ezt minden oldalon a *\$_SERVER* szuperglobális tömbben mentett felhasználói adatokból olvassuk ki (33. ábra). Ha az oldalra lépést nem megfelelő jogosultsággal próbálják megtenni, automatikusan át lesznek irányítva a kezdőlapra.

```
if (!isset($_SESSION["usr"]) || !$_SESSION["admin"])
{
    header("Location: index.php");
    return;
}
```

33. ábra: Adminisztrátori jog ellenőrzése

3.6.3.1. admin_regisztracio.php

Az *admin_regisztracio.php* oldal működésilég nagyon hasonlít a *regisztracio.php* oldalra, az általános jogosultságvizsgálat eltéréseivel. Alapelveként itt adminisztrátorok hozhatnak létre új fiókokat, amik maguk is lehetnek adminisztrátorok. Ezt egy *checkbox* elemmel lehet beállítani.

3.6.3.2. admin_adatok.php

Az utazásokról 2 táblázatban vannak kilistázva az adatok, *foreach* ciklusokon belül.

A *felhasznalo.php*-hoz hasonlóan a feldolgozás módokon alapul, amiket a beadásra használt gomb határoz meg. A mód mögötti érték mindig egy *id*, ami az adatbázisban lévő utazás elsődleges kulcsa.

- *delete_trip*: Jövőbeli, már meghirdetett utat, és az ahhoz kapcsolódó összes foglalást lehet ezzel törölni. A törlés előtt megerősítést kér az oldal egy *confirm* függvény (32. ábra) használatával.

- *new_trip*: Ezzel lehet meghirdetni új utazási időpontot. Az *input* és *select* mezők kitöltése kötelező az „Új felvétele” gomb melletti sorban, amit a feldolgozás során ellenőriz is a program. Ha minden ki van töltve, megpróbálja elmenteni a *Database* osztály *save_trip* metódusával, és annak sikeréről, valamint hibáiról üzenetet ír ki.
- *view_trip*: Ez a mód nem szerepel a feldolgozásban, mivel ez egy speciális mód. A feldolgozás hívását a „Részletek” feliratú *button* elem nem az *admin_adatok.php*, hanem az *utazas_adatok.php* felé intézi *GET* metódussal, ami alább van részletezve. Ezt a *button* elemben a *formmethod* és *formaction* tulajdonságok kitöltésével lehet elérni, ami felülírja az alapértelmezett, azaz a *form* által meghatározott tulajdonságokat.

3.6.3.3. utazas_adatok.php

Erre a lapra 2 féle módszerrel lehet eljutni: az *admin_adatok.php* oldalon a *view_trip* mód alkalmazásával, vagy pedig, a böngészőben az URL átírásával. Az URL-ből való elérést az teszi lehetővé, hogy a *GET* metódust várjuk el az oldal lekérdezésére.

Mivel az oldalon tetszőlegesen átírható a *query* paraméterek között a *view_trip*, azaz az utazás *id*-je, meg lehet adni nem létező utazás kulcsát is. Ha nem létezik ilyen utazás, a *get_booking_info_by_trip* metódus nem fog semmit visszaadni, és egy üres utazás adatait fogjuk megjeleníteni.

Amennyiben van foglalás, elkészítünk a *sorted_stops* változóba egy útvonaltervet a *Map* osztály *make_trip_plan* metódusával. Ezután a tervet a *Map* osztály *show_route* metódusával közvetlenül meg tudjuk jeleníteni egy térképen.

Egy táblázatban *foreach* ciklussal ki vannak listázva az utazáshoz tartozó foglalások is. Ha nincs még foglalás, egy előre definiált üzenetet jelenít meg helyette az oldal.

3.6.3.4. admin_statisztika.php

Az oldalon az általános ellenőrzés után – melyben megbizonyosodunk, hogy adminisztrátori fiókkal szeretné a felhasználó az adatokat megtekinteni – egyből inicializáljuk a diagramok megjelenítéséhez szükséges adatokat.

Az adatokat a *stats_by* kezdetű tömbökbe tároljuk, majd a *stats_by_max* változóba kiválasztjuk a maximális tömbértéket a megjelenítéshez. Van, ahol a *stats_by* tömbben az értékek asszociatív tömbök, amikben a diagramra kerülő feliratok adatai is el vannak tárolva.

Ezután a HTML részben egy-egy *foreach* ciklussal jelenítjük meg a diagram sorait.

3.7. Stíluslapok

A stíluslapok, amik az oldal kinézetéért felelnek, a *szallito_html/css* mappában találhatóak és a CSS3 szabvány szerint lettek fejlesztve.

A *style.css* a legáltalánosabb elveket, valamint a CSS konstansokat tartalmazza. Ezt a fájlt a *metadata.html*-ben applikáljuk az oldalra, így biztosítva, hogy mindenhol érvényes legyen. A CSS konstansok a *:root* elemben (34. ábra) találhatóak meg, és ezeket '*--<konstans-név>*':

```
:root {
  --large-screen: 55vw;
  --medium-screen: 70vw;
  --small-screen: 80vw;
  --base-width: 80%;
  --bg: #eee;
  --bg-title: #384A52;
  --title: #eee;
  --button: #8db7c9;
  --button-hover: #384A52;
  --button-hover-font: #eee;
  --progress-green: rgb(128, 216, 128);
}
```

34. ábra: *:root* elem

<érték>' formátumban vannak felvéve. A konstansokat a '*var(--<konstans-név>)*' kifejezéssel tudjuk használni. Ugyan *var* – azaz variable – a kifejezést hívó függvény neve, ezek az értékek deklarálás után nem változnak, így konstansok. Itt a legfontosabb és legtöbbet használt értékeket lehet eltárolni, ezzel átláthatóbbá téve a kódot. Ezek az értékek minden más stíluslapon is elérhetőek, amennyiben a *style.css* után lettek a head elembe felvéve, ami a *metadata.html* és a PHP oldalak struktúrájából implicit adódik.

A *navbar.css* a *style.css*-hez hasonlóan egy általános stíluslap. Ez a fájl a *navbar.php*-ban lévő *nav* elem és gyerekeinek kinézetét határozza meg.

A *form.css* a *form* elemeknek és a gyerekeiknek, míg a *table.css* a különböző *table* elemek és azok gyerekeinek kinézetét határozzák meg. A felvételük a *metadata.html*-en kívül, a head elemben történik, azokon az oldalakon, ahol van rájuk szükség.

A *stats.css* a statisztikai diagramok kinézetét határozza meg. A felvétele szintén a *metadata.html*-en kívülre esik.

3.8. Tesztelés

A tesztek manuálisan lettek elvégezve a felületen keresztül. Alább vannak részletezve a fontosabb függvények és az ellenőrzött eseteik.

3.8.1. Database

3.8.1.1. signup/regisztráció

3. táblázat: signup/regisztráció tesztesetek

Teszteset leírása	Elvárt eredmény	<i>id</i>
Normális fiók regisztráció	Hiba nélkül elkészül a fiók	2760977063
Ideiglenes felhasználói fiók tényleges regisztrálása	Hiba nélkül elkészül a fiók	1598885733
Létező fiók e-mail címére való regisztráció	Hibaüzenet küldése	-
Kitöltetlen adat	Hibaüzenet küldése	-
Nem egyező jelszavak megadása	Hibaüzenet küldése	-

3.8.1.2. login/bejelentkezés

4. táblázat: login/bejelentkezés tesztesetek

Teszteset leírása	Elvárt eredmény	<i>id</i>
Normál bejelentkezés	Hiba nélküli bejelentkezés	2760977063
Ideiglenes fiókba való bejelentkezés	Hibaüzenet küldése	2620483069
Rossz jelszó megadása	Hibaüzenet küldése	-

3.8.1.3. save_trip

5. táblázat: save_trip tesztesetek

Teszteset leírása	Elvárt eredmény	id
Normál utazás felvétele	Hiba nélküli létrehozás	62
Utazás felvétele ugyan arra az útra, ugyan akkor, más autóval	Hiba nélküli létrehozás	63
Utazás felvétele ugyan arra az útra, közeli időpontban, ugyan azzal az autóval	Hibaüzenet küldése	-
Utazás felvétele más útra, ugyan akkor, ugyan azzal az autóval	Hibaüzenet küldése	-
Utazás felvétele ugyan arra az útra, máskor, ugyan azzal az autóval	Hiba nélküli létrehozás	64

3.8.1.4. save_custom_stop

6. táblázat: save_custom_stop tesztesetek

Teszteset leírása	Elvárt eredmény	id
Saját utazási pont felvétele	Hiba nélküli létrehozás	31
Pont felvétele ugyan arra a helyre, visszaút útvonalán	Hibaüzenet küldése	-
Pont felvétele ugyan arra a helyre, más út útvonalán	Hiba nélküli létrehozás	32
Pont felvétele nagyon közeli helyre, ugyan azon az út vonalán	Hibaüzenet küldése	-
Értelmetlen névre pont létrehozása	Hibaüzenet küldése	-
Túl messze lévő pont létrehozása	Hibaüzenet küldése	-

3.8.1.5. book

7. táblázat: book tesztesetek

Teszteset leírása	Elvárt eredmény	id
Normál utazás felvétele	Hiba nélküli létrehozás	66
Normál utazás melletti párhuzamos utazás foglalása túl sok helyet foglalva	Hibaüzenet küldése	-

3.8.2. Map

3.8.2.1. make_trip_plan

8. táblázat: make_trip_plan tesztesetek

Teszteset leírása	Elvárt eredmény	<i>id</i>
Utazási terv minden megállóval	Sorban minden megállót érinti az útvonal	index.php
Utazási terv csak első és utolsó fix megállóval	Csak a két megálló van az útvonalon	63
Utazási terv 2 fix megállóval, a második nem az utolsó fix pont	Csak a két megálló van az útvonalon	58
Utazási terv 2 fix megállóval, az első nem az első fix pont	Csak az első pont és a két megálló van az útvonalon	42
Utazási terv 2 nem fix megállóval, az első nem az első fix pont közelében van	Csak az első pont és a két megálló van az útvonalon	64
Utazási terv 2 nem fix megállóval, az első az első fix pont közelében van	Csak a két megálló van az útvonalon	62
Utazási terv 2 teljes járműkapacitás használó, útban nem egymást fedő foglalással	Távolsági sorrendben vannak a megállók	59
Utazási terv 2 teljes járműkapacitás használó, útban egymást fedő foglalással, azaz az első megállója messzebbi, mint a másik felszálló pontja	A leszálló megállót előbb látogatjuk, mint a felszállót, kapacitási okok miatt. Nem távolságra optimalizált út	31
Utazási terv városon belül több nem fix megálló kijelölésével	Távolsági sorrendben vannak a megállók	65

4. Összefoglalás

4.1. Konklúzió

A webalkalmazás funkcióit tekintve kezdetleges állapotú és fejlesztésre szorul, azonban a legfontosabb része, a Google Maps API-al való kommunikációja és annak integrálása a kódba teljesült. A segítségével már ebben a pillanatban is sok dolgot meg tudunk tenni, ki tudunk számolni.

Az adatbázis és táblái az alapszintű követelményeknek megfelelnek, és a szolgáltatási kör bővítésére is kész állapotban vannak. Funkciói bővítését az *id* mezők bevezetésével nagyban megkönnyítettük.

4.2. További fejlesztési lehetőségek

A weboldal jelenlegi állapotában ugyan készen van használatra, sok további fejlesztéssel lehetne a felhasználói élményt esetlegesen csökkentő dolgokat kiküszöbölni, tovább fejleszteni. Itt van részletezve ezek közül pár nagyobb terv:

4.2.1. Fizetési mód és árak

- Az adatbázis kiegészítése az utakra számolandó árakkal.
- A program számolhatna egyedi megállási pontokra útdíjakat az alapértelmezett megállási pontoktól való távolság alapján.
- A programban lehetne fizetési mód beépítve. Esetlegesen előleg fizetésére is adna módot.
- Akciók, kedvezmények lehetőségének biztosítását is lehetne biztosítani.

4.2.2. Fiók kezelés

A felhasználók sok dolgot nem tudnak most csinálni, ami általában alapkövetelmény lenne egy jól kidolgozott weboldaltól. Ezeket a funkciókat is ki lehetne építeni, a felhasználói élmény növelése érdekében.

- Egyéb adatok (pl. telefonszám, e-mail cím) módosítása
- Értesítések küldése a kiválasztott csatornán
 - Sikeres foglalás esetén

- Az utazás előtt adott nappal emlékeztető
- Profilkép beállítása
- Fiók törlése

4.2.3. Adatbázis

- Történelmi adatok megtartása
 - Felhasználói fiókok történelme
 - Utazás törlése nem töröl, csak életben létet állít
 - A *booking-user_data* kapcsolat a foglalás pillanati adatait tárolja el
- Egyéb adminisztrációhoz kapcsolódó fejlesztések
 - Módosító adatainak felvétele *trigger*-ekkel

4.2.4. Egyéb fejlesztési ötletek

- A Google Maps API-al beépített térkép választhatna helyet kattintás hatására is.
- A foglalásnál lehetne vélhetőleg érkezési időket számolni, hogy pontosan tudja a felhasználó, hogy mikor kell készen lennie a felszállásra, és mikor számíthat a leszállásra.

5. Irodalomjegyzék

- [1] <https://mariadb.com/kb/en/data-type-storage-requirements/>
- [2] https://www.w3schools.com/php/php_form_validation.asp
- [3] <https://www.php.net/manual/en/book.mysql.php>
- [4] <https://www.php.net/manual/en/function.mt-rand.php>
- [5] <https://github.com/yidas/google-maps-services-php>
- [6] <https://developers.google.com/maps/documentation/javascript/place-autocomplete-new>
- [7] <https://developers.google.com/maps/documentation/javascript/nearby-search>

6. Ábrajegyzék

1. ábra: Navigációs sáv.....	2
2. ábra: Útvonal választása	2
3. ábra: Megállási pontok választása.....	3
4. ábra: Időpont választása	4
5. ábra: Foglalói adatok megadása.....	5
6. ábra: Utazói adatok megadása.....	5
7. ábra: Áttekintési lap és megjegyzés	6
8. ábra: Regisztrációs lap	7
9. ábra: Sikeres bejelentkezés.....	7
10. ábra: Felhasználói menüsor	8
11. ábra: Jelszó módosítás	8
12. ábra: Saját foglalási adatok	9
13. ábra: Új utazási pont felvétele	9
14. ábra: Adminisztrátori fiók regisztrálása	10
15. ábra: Utazási adatok táblázata	11
16. ábra: Útiterv	12
17. ábra: Utazás foglalási adatai	12
18. ábra: Hónap szerinti diagram	13
19. ábra: Megálló szerinti diagram	14
20. ábra: Útvonalak szerinti diagram	14
21. ábra: Adatbázis terv (eredeti)	16
22. ábra: Adatbázis terv (megvalósítás)	17
23. ábra: Consts névtér.....	21
24. ábra: HTML függvények	22
25. ábra: Database osztálydiagram	24
26. ábra: get_available_trips lekérdezés.....	26
27. ábra: Map osztálydiagram.....	27
28. ábra: metadata.html.....	31
29. ábra: navbar.php kódrészlet	31
30. ábra: Mentett adatok betöltése	32

31. ábra: Form adatok mentése	32
32. ábra: confirm függvény	38
33. ábra: Adminisztrátori jog ellenőrzése.....	38
34. ábra: :root elem	40

7. Táblázatjegyzék

1. táblázat: Böngésző kompatibilitás.....	2
2. táblázat: Rekordok méretei az egyes táblákban	18
3. táblázat: signup/regisztráció tesztesetek	41
4. táblázat: login/bejelentkezés tesztesetek	41
5. táblázat: save_trip tesztesetek.....	42
6. táblázat: save_custom_stop tesztesetek.....	42
7. táblázat: book tesztesetek	42
8. táblázat: make_trip_plan tesztesetek	43