

Pannon Egyetem

Műszaki Informatikai Kar

Rendszer- és Számítástudományi Tanszék

Gazdaságinformatikus BSc

SZAKDOLGOZAT

Veszprémi tömegközlekedést támogató okostelefon

alkalmazás

Böröndi Evelin

Témavezető: Dr. Hegyháti Máté

2016

KÖSZÖNETNYILVÁNÍTÁS

...

TARTALMI ÖSSZEFOGLALÓ

Tartalmi összefoglaló...

Kulcsszavak: tömegközlekedés, Android, Veszprém, útvonaltervezés

ABSTRACT

Angol tartalmi összefoglaló...

Keywords: public transport, Android, Veszprém, route planning

Tartalomjegyzék

1. Bevezetés	6
2. Veszprém tömegközlekedése	8
2.1. Az útvonaltervezés és a Google Maps	8
2.2. Veszprém tömegközlekedése	9
2.3. Jelenlegi megoldások	11
3. Követelmények, technológiák	17
3.1. Követelmények	17
3.2. Funkcionális követelmények	18
3.3. Felhasznált technológiák	19
4. Felhasználói kézikönyv	22
4.1. Android alkalmazás	22
4.2. Főmenü	22
4.3. Útvonaltervezés	23
4.4. Menetrendek	24
4.5. Megállók	24
4.6. Kedvencek	24
4.7. Admin oldal	25
5. Fejlesztői dokumentáció	26
5.1. A szerver	26
5.2. Szerver feladatai	29
5.3. Az Android kliens	30
6. Továbbfejlesztési lehetőségek	35

Ábrák jegyzéke

2.1.	Google Maps útvonaltervezés	9
2.2.	Élő útvonalinformáció	10
2.3.	Veszprém tömegközlekedési hálózata	10
2.4.	Az 1-es buszhoz tartozó menetrend táblázat	12
2.5.	Az 1-es és a 4-es járat útvonala	13
2.6.	A BamBusz webes felülete	13
2.7.	Az alkalmazás menüje	15
2.8.	Az alkalmazás főoldala a menetrendi naptárral	16
3.1.	A Google Maps és az OSM összehasonlítása	19
5.1.	Az autentikacio folyamata	27
5.2.	A jarat entitas kapcsolatai	29
5.3.	A jarathoz kapcsolatai	30
5.4.	Kulonleges es indulasi idopontok kozotti kapcsolat	30

1. fejezet

Bevezetés

A nagyvárosokban található tömegközlekedés bonyolult és szerteágazó módon húzódik végig a város különböző pontjait érintve. A tömegközlekedéshez tartozó útvonalak, megállók és indulási idők sokaságán az tud igazán kiigazodni, aki hosszabb ideje használja már. A városba „idegenként” érkezők, turisták számára szükséges lehet egy olyan eszköz, melynek segítségével eljutnak a céljukhoz, gyorsan és kényelmesen tudják használni a város nyújtotta közösségi közlekedést.

Veszprémben jelenleg a közlekedés e fajtáját az autóbuszok szolgálják ki. Bár a megyeszékhely nem tartozik a legnagyobb városok közé Magyarországon, mégis szerteágazó menetrendet tudhat magáénak. Emellett a városban gyakran fordulnak meg egész évben turisták, egyetemisták, akik számára a legnagyobb hátrány, hogy nem ismerik a buszmegállókat, esetlegesen a saját helyzetüket sem. Egyelőre még nem található olyan szolgáltatás városunkban, ami eleget tenne annak, hogy segítse az utazóközönséget a tájékozódásban. Jelenleg több, a veszprémi tömegközlekedést segítő megoldással is találkozhatunk. Ezen megvalósítások ugyanakkor túlságosan is statikusak egy újonnan a városba látogató számára. Egyik ilyen fennálló probléma az, hogy megtudja az utas egy adott megállóban milyen buszok állnak meg, végig kell néznie az egész menetrendet, továbbá nem szolgálnak vizuális visszajelzéssel, azaz ugyan ismeri a megálló nevét, azonban nem tudja pontosan meghatározni a város mely területén található. Ilyen helyzetekben igény lenne egy olyan megoldásra, hogy térkép alapján is tudjanak tájékozódni, viszont a jelenlegi megoldások közül egyik sem felel meg az előzőekben felállított igényeknek a kielégítésére. Erre a fennálló problémára nyújt megoldást a szakdolgozatomban megvalósított Android alkalmazás, amely hordozható, megbízható és gyors formában tájékoztatja az utazni kívánókat.

A 2. fejezetben bemutatom az útvonaltervezést, mint szolgáltatást, illetve a városban működő jelenlegi megoldásokat. A 3. fejezetben ismertetem a program szükséges funkcióit, és az azokkal szemben támasztott követelményeket. A 4. fejezetben bemutatom az elkészült alkalmazást

felhasználói szinten. Az 5. fejezetben kitérek a fejlesztésre részletesebben, az alkalmazás egyes részegségeire és azok implementációjára. Végül a 6. fejezetben felvázolok pár elképzelést az alkalmazás továbbfejlesztéséhez.

2. fejezet

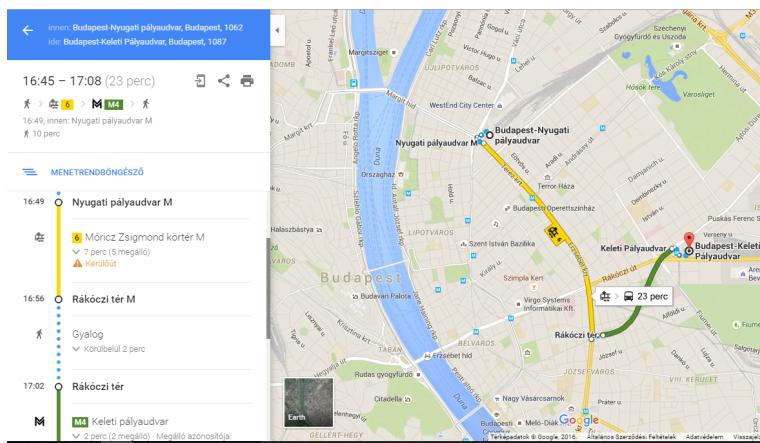
Veszprém tömegközlekedése

A fejezetben ismertetésre kerül az útvonaltervezés, mint szolgáltatás. Sok weboldal és telefonos alkalmazás segítségével tudjuk az utunkat előre megtervezni, továbbá különböző egyedi funkciókat is igyekeznek fejleszteni, ezzel csalogatva magukhoz a felhasználókat. A fejezet első részében az útvonaltervezést fogom bemutatni pár népszerű alkalmazáson keresztül, amely sikeresen elégíti ki az utazni vágyók igényeit. Ezután áttekintést adok Veszprém tömegközlekedéséről, hogy átfogó képet adjak a város közlekedési helyzetéről. Végül pedig bemutatom a jelenleg is a piacon lévő megoldásokat, amik a városban való tájékozódást segítik.

Az útvonaltervezés és a Google Maps

Útvonaltervezők nevezik az olyan szoftvereket, amelyek két földrajzi pont között keresnek optimális útvonalat egy keresőmotor segítségével. Ezen motorok gyakran intermodális működések. Már az 1970-es évektől használják a támogatás ezen fajtáját. Akkoriban ez annyit jelentett, hogy egy terminálos felhasználói interfészen keresztül csatlakozott a hívóközponthoz, és onnan érdeklödtek meg a tömegközlekedéssel kapcsolatos információkat. Miután elterjedt az a szokás az emberek között, hogy maguknak terveztek meg a nyaralásokat, és nem vették igénybe az utazási irodák ügynökeit, elkezdtek fejlődni az interneten elérhető útvonaltervezők.

A tervezők ebbe a fajtájába tartozik az akkoriban Google Transitként ismert útvonaltervező, ami napjainkban a Google Maps térképes szolgáltatás része. Az alkalmazás interneten és mobil eszközökre is elérhető, rengeteg plusz funkcióval. A térképes adatbázisát különböző partnerek segítségével szerzi be, de sok helyen (például a fejlődő országokban) a közösség frissíti a térképes adatokat. Az útvonaltervező funkció kezdetben gyalogos és autós közlekedés tervezésére volt képes, azonban 2007-ben integrálták a tömegközlekedést is az útvonaltervezésbe. Magyarországon 2011 óta kizárálag Budapesten érhető el a funkció.



2.1. ábra. Google Maps útvonaltervezés

Ahogy a 2.1. ábrán is látszik, tervezéskor beállíthatjuk a közlekedési formát, hogy éppen gyalogosan vagy tömegközlekedéssel szeretnénk igénybe venni. Gyalogos és autós közlekedéskor az elérhető járdákat és autóutakat veszi figyelembe az alkalmazás, majd az eredményt kirajzolja a térképre, esetlegesen több elérhető opció esetén a többöt szaggatott vonallal jelöli. Tömegközlekedés esetén is több lehetőséget kínál fel, ezekből több szempont alapján tudjuk kiválasztani a számunkra optimálisat. Ilyen szempont lehet a legkevesebb átszállállás, illetve legkevesebb gyaloglás. A Google Maps továbbá indulási időket is rendel a járatokhoz, így képesek vagyunk tervezni mostani időpillanathoz, illetve ha később szeretnénk csak utazni, azt is beállíthatjuk. Mindemellett rendelkezik élő menetrendinformációval, amit a Budapesti Közlekedési Központ hivatalos oldaláról szerez be. Itt láthatjuk ha felújítás, útlezárás miatt nem közlekednek járatok, vagy más útvonalon járnak, emiatt más megállókat érintenek, ezt láthatjuk a 2.2. ábrán is.

Veszprém tömegközlekedése

Veszprém tömegközlekedését jelenleg a város méretéből és rendelkezésre álló infrastruktúrájából adódóan autóbuszok adják. A buszokat a Balaton Volán Zrt. biztosítja az 1960-as évek óta. Veszprém tömegközlekedésének gondolata azonban már 1884-ben megfogalmazódott Czollenstein Ferenc által, aki omnibuszokat indított Veszprém és Balatonalmádi között. A jelenlegi közlekedési forma 28 vonalat foglal magában, és a város belterületén kívül közlekedik a közigazgatásilag a városhoz tartozó településekhez is, úgymint Szabadságpuszta, Jutaspuszta, Kádárta és Gyulafirátót, valamint Csatár. Több átalakításon is átesett, amíg elérte a mai napi formáját, amit a 2.3. ábra mutat.

A menetrend integrálása lehetséges a Google Maps rendszerébe, így az útvonaltervezés funkció Veszprém városában is használható lehetne. A busztársaság részéről, egy meghatározott

11:00 – 11:24 (24 perc)



➤ M M2 ➤ M M3

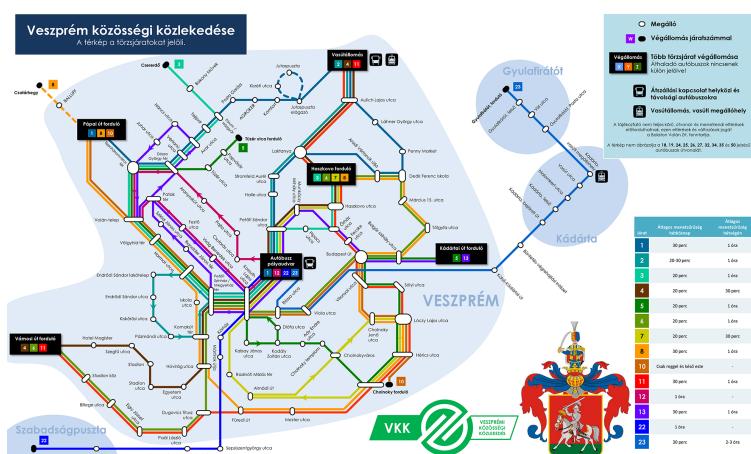
11:05, Innen: Széll Kálmán tér

5 perc 5 percenként

MENETRENDBÖNGÉSŐ

11:00	O Mammut Budapest, Lövöház utca 2, 1024
	Gyalog Körülbelül 5 perc , 400 m
11:05	O Széll Kálmán tér
M	M2 Örs vezér tere M+H 4 perc (2 megálló) - Megálló azonosítója: F02481 ⚠ Csökkentett járatszám
11:09	
11:15	O Deák Ferenc tér
M	M3 Kőbánya-Kispest M 9 perc (6 megálló) - Megálló azonosítója: F00955

2.2. ábra. Élő útvonalinformáció



2.3. ábra. Veszprém tömegközlekedési hálózata

formátumú adatbázis továbbítása szükséges a Google felé, mivel ez csak Budapesten valósult meg, így csak a fővárosban érhető el Magyarországon belül ez a szolgáltatás.

Jelenlegi megoldások

Veszprémben a tömegközlekedés támogatására jelenleg is létezik több megoldás. A busztársaság is igyekszik minél kielégítőbb segítséget nyújtani az utazóközönségének, hiszen fontos számára, hogy minél többen vegyék igénybe a tömegközlekedést. Továbbá léteznek olyan harmadik fél által készült eszközök is, amelyek szintén hozzájárulnak az információszerzéshez. Ezen alkalmazásokat fejlesztők nem profitszerzési céllal készítették el, hanem csupán önkéntes alapon, az emberek megsegítésének céljával. Funkcionalitásuk ezáltal elmarad egy céges környezetben, nagyobb fejlesztőgárda által készített szolgáltatástól, melytől nyereséget várnak a tulajdonosok. Ezek közül mutatok be pár ismertebb példát, amik jelenleg elérhetők a piacon.

ÉNYKK

Az ÉNYKK vagyis az Észak-nyugat-magyarországi Közlekedési Központ felel a tömegközlekedés üzemeltetéséért. Weboldalukon található dokumentum magába foglalja az összes buszjáratot, és az azokhoz tartozó megállókat és indulási időket. A 2.4. ábrán látható módon szerepel egy buszjárat a dokumentumban. Ez a fajta statikus megoldás általános segítséget nyújt az utazni vágyóknak, ha már rendelkeznek információval a városról, például a megállók elhelyezkedését illetően.

A társaság igyekszik több segítséget nyújtani az utasoknak, emiatt új funkciókat készítettek a weboldalra az elmúlt időben. Létrehoztak egy térképes funkciót, ahol a járatokat kiválasztva az alkalmazás felrajzolja ezen buszoknak az útvonalát a térképre. Továbbá elkezdtek fejleszteni egy útvonaltervező funkciót is, viszont ezek jelenleg kezdetleges formában működnek csak. A térképes szolgáltatásnál kiválasztották, hogy milyen buszjáratokra vagyunk kiváncsiak, és a program kirajzolja azokat a térképre, ahogy a 2.5. ábra mutatja.

BamBusz

Online felületen elérhető segítség, célközönsége főleg az egyetemisták. Kedvezőbb megoldást nyújt, mint a busztársaság oldala abból a szempontból, hogy nem kell átböngészni az egész dokumentumot az indulási időkért, hanem beállíthatjuk az indulási és az érkezési megállónkat. Ezt követően az oldal kilistázza nekünk azokat a buszokat és a hozzájuk tartozó indulási időket, amikkel eljuthatunk a célunkhoz a 2.6. ábrán látható módon. Hátránya hasonlóan a hivatalos oldalhoz, hogy ismernünk kell a megállókat, ahhoz hogy használni tudjuk.

1		Autóbusz-állomás – Laktanya – Vasútállomás – Jutaspuszta – Dózsa Gy. tér – Pápai úti ford.
Menetidő 1	Menetidő 2	MEGÁLLÓHELYEK
0	0	Autóbusz-állomás
2	2	Petőfi S. u.
3	3	Jutasi út 61.
4	4	Jutasi úti Itp.
5	5	Laktanya
7	7	Aradi Vértanúk u.
8	8	Deák Ferenc Iskola
9	9	Penny Market
10	10	Láhner Gy. u.
11	11	Vasútállomás
13	13	Kisréti u.
15	15	Jutaspuszta
17	17	Kisréti u.
19	12	Jutaspuszta elág.
20	13	Komfort
21	14	AGROKER
22	15	Posta Garázs
23	16	Fórum
24	17	Tejipar
25	18	Avar u.
26	19	Vértanú u.
27	20	Dózsa Gy. tér
28	21	Tizenháromváros tér
29	22	Pápai úti forduló

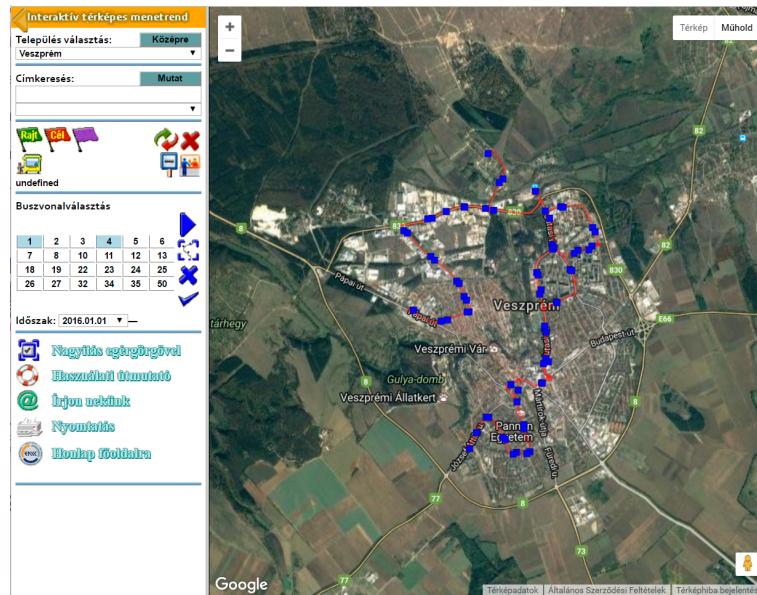
Autóbusz-állomás megállóból indul:

Óra	Munkanapokon	Szabad- és munkaszüneti napokon
	perc	perc
05 :	10, 40	35
06 :	10, 40	05
07 :	10, 35	05
08 :	10, 40	05
09 :	10, 40	05
10 :	10, 40	05
11 :	10, 40	05
12 :	10, 40	05
13 :	10, 40	05
14 :	10, 40	05
15 :	10, 40	05
16 :	10, 40	05
17 :	10	05
18 :	10	05
19 :	10	
20 :	30	30

JELMAGYARÁZAT:

Az aláhúzással jelölt járatok Jutaspuszta betéréssel közelkednek az 1-es menetidőoszlop szerint

2.4. ábra. Az 1-es buszhoz tartozó menetrend táblázat



2.5. ábra. Az 1-es és a 4-es járat útvonala

Hova szeretnél utazni
Avar utca

Honnan szeretnél utazni
Volán-telep

Munkanapokon	Szabad- és munkaszüneti napon	Iskolai előadási napon	Szabadnapokon (szombaton)
05:17 (3 busz)	05:28 (34 busz)	05:33 (34 busz)	05:37 (3 busz)
05:40 (3 busz)	05:57 (3 busz)	06:08 (3 busz)	
06:17 (3 busz)	06:37 (3 busz)	06:57 (3 busz)	07:08 (3 busz)
			07:13 (34 busz)
07:27 (18 busz)	07:31 (25 busz)	07:37 (3 busz)	07:47 (3 busz)
			07:57 (3 busz)
08:57 (3 busz)	09:08 (3 busz)	09:27 (3 busz)	09:57 (3 busz)
			10:08 (3 busz)
11:08 (3 busz)	11:27 (3 busz)	11:57 (3 busz)	12:08 (3 busz)
			12:27 (3 busz)
13:17 (3 busz)	13:28 (34 busz)	13:37 (3 busz)	13:40 (3 busz)
			13:57 (3 busz)
14:37 (3 busz)	14:57 (3 busz)	15:08 (3 busz)	15:17 (3 busz)
			15:37 (3 busz)
16:17 (3 busz)	16:37 (3 busz)	16:57 (3 busz)	17:08 (3 busz)
			17:27 (3 busz)
18:27 (3 busz)	18:57 (3 busz)	19:08 (3 busz)	19:27 (3 busz)
			19:57 (3 busz)
20:57 (3 busz)	21:08 (3 busz)	21:27 (3 busz)	21:28 (34 busz)
			21:40 (3 busz)
22:27 (3 busz)			21:57 (3 busz)
			22:08 (3 busz)

2.6. ábra. A BamBusz webes felülete

Veszprémi buszmenetrend

Okostelefonra elérhető alkalmazás, ami letisztultan, egyszerűen és gyorsan jeleníti meg a buszjáratokat külön menüpontba szedve, ahogy a 2.7. ábrán látható. Előnye, hogy akár útközben tudunk információt szerezni az autóbuszok közlekedési rendjéről. Továbbá elérhető egy éves menetrendi naptár a főoldalon, ami segítségével megállapíthatjuk hogy milyen rend szerint közlekednek a buszok adott napokon. Ugyanakkor az alkalmazás nem naprakész, a naptár a tavalyi évet reprezentálja, illetve ebből kifolyólag a buszjáratok menetrendje is az elmúlt évre érvényes indulásokat mutatja.



2.7. ábra. Az alkalmazás menüje

1 Autóbusz-állomás – Laktanya – Vasútállomás – Jutaspuszta – Dózsa Gy. tér – Pápai úti ford.

Menetidő 1	Menetidő 2	MEGÁLLÓHELYEK
0 0		Autóbusz-állomás
2 2		Petőfi S. u.
3 3		Jutasi út 61.
4 4		Jutasi úti ltp.
5 5		Laktanya
7 7		Aradi Vértanúk u.
8 8		Deák Ferenc Iskola
9 9		Penny Market
10 10		Láhner Gy. u.
11 11		Vasútállomás
13 ↘		Kisréti u.
15 ↘		Jutaspuszta
17 ↘		Kisréti u.
19 12		Jutaspuszta elág.
20 13		Komfort
21 14		AGROKER
22 15		Posta Garázs
23 16		Fórum
24 17		Tejipar
25 18		Avar u.
26 19		Vértanú u.
27 20		Dózsa Gy. tér
28 21		Tizenháromváros tér
29 22		Pápai úti forduló

Autóbusz-állomás megállóból indul:

Óra	Munkanapokon	Szabad- és munkaszüneti napokon
	perc	perc
05 :	10, 40	35
06 :	10, 40	05
07 :	10, 35	05
08 :	10, 40	05
09 :	10, 40	05
10 :	10, 40	05
11 :	10, 40	05
12 :	10, 40	05
13 :	10, 40	05
14 :	10, 40	05
15 :	10, 40	05
16 :	10, 40	05
17 :	10	05
18 :	10	05
19 :	10	
20 :	30	30

JELMAGYARÁZAT:
Az aláhúzással jelölt járatok Jutaspuszta betéréssel közlekednek az 1-es menetidőoszlop szerint

2.8. ábra. Az alkalmazás főoldala a menetrendi naptárral

3. fejezet

Követelmények, technológiák

A fejezetben bemutatásra kerülnek a funkciókövetelmények, amelyeket a fejlesztés előtt és közben figyelembe kellett venni, hogy az utazóközönség számára egy mai igényeket kielégítő, modern alkalmazás készüljön el. Cél volt továbbá, hogy a rendszer másik szereplői, az adminisztrátorok számára is könnyen kezelhető, felhasználóbarát felület valósuljon meg. A **ABRA**. képen láthatóak a rendszer szereplői és a hozzájuk tartozó használati esetek.

Követelmények

Az alábbi követelmények tartalmazzák azokat a tulajdonságokat, amiket az elkészült alkalmazásnak tartalmaznia kell.

- Sebesség

Válaszidő lecsökkentése minél alacsonyabb szintre. A felhasználóknak fontos, hogy az alkalmazás segítségével gyorsan és megbízhatóan tudjanak információhoz jutni.

- Alacsony erőforrás felhasználás

Az egyik legfontosabb követelmény, hogy az alkalmazás kevés erőforrás igénybevételével is megfelelően működjön. Ehhez szükség volt arra, hogy a telefon egy lokális adatbázist üzemeltessen. Így az alkalmazás használatakor nem kell internetkapcsolatot biztosítani az adatok elérhetőek lesznek a telefon adatbázisából.

- Megbízhatóság

Létre kellett hozni egy olyan webes felületet, ahol az adatbázis kezelhető, változások esetén pedig módosíthatóak az adatok. Ebből kifolyólag az alkalmazás minden naprakészen szolgálja az információt a felhasználóknak.

- Könnyű kezelhetőség

A felhasználók számára fontos, ha nincsenek bonyolult akciók, hanem lehetőleg az

összes funkció használata egyértelmű, ezzel is gyorsítva az alkalmazás használatát. Ha bonyos jelölések, rövidítések igénylik, akkor súgót kell hozzáadni az alkalmazáshoz.

Funkcionális követelmények

A .ábrán láthatóak a felhasználóval és az adminisztrátorral kapcsolatos használati esetek, amelyeket a következőkben fogok részletesen kifejteni.

Android alkalmazás

– Útvonaltervezés

Az útvonaltervezés az alkalmazás fő funkciója, ezért nagy hangsúlyt kellett a megvalósítására fektetni. Könnyű kezelőfelületet kívánt, egyértelmű jelöléseket amik bárki számára egyszerűvé teszik a használatát. Ez a menüpont azokat az utasokat célozza meg elsődlegesen, akik számára Veszprém ismeretlen terület. Fontos volt, hogy különböző utazási módoszatok is elérhetők legyenek, például azoknak akik buszjegyet váltanak, azok minél kevesebb átszállással kínálja az alkalmazás az útvonalat. Továbbá, mivel ez egy térképet integráló funkció, ezért megfelelő módon kellett megjeleníteni a térképet.

– Menetrend

A menetrend funkció akkor kap szerepet egy felhasználónál elsősorban, amikor nincs elérhető internetkapcsolat a mobilkészülékén. Ezen kívül azoknál akik rendelkeznek a tömegközlekedésről legalább alapszintű ismerettel, így jártasak a menetrendben és az indulási időkről szeretnének informálódni. Megjelenésénél figyelni kellett arra, hogy felülete letisztult legyen, mégis a keresett információ könnyen megtalálható legyen.

– Megállók

Az alkalmazásban szükség volt egy olyan szolgálatásra is, ahol a felhasználók tájékozódni tudnak a városban a megállókról. Mivel előfordulhat olyan eset, hogy valaki tisztában van a megálló elhelyezkedésével, viszont meg szeretné tudni, hogy milyen járatok érintik anélkül hogy az egész menetrendet át kéne olvasnia, ezért fontos volt implementálni egy ilyen funkciót az alkalmazásba.

Weboldal

- Buszjáratok kezelése
- Menetidők kezelése
- Megállók kezelése

- Ideiglenes járatmodosulások kezelése

Felhasznált technológiák

Térkép

Egy útvonaltervező alkalmazás nagyon fontos tulajdonsága a megjelenés. Egy ilyen alkalmazásnál alapvető elvárás, hogy a térképen megjelenő információ könnyen kiolvasható legyen, hogy a felhasználók könnyedén el tudják választani a lényeges információt azoktól, amelyek nem szükségesek a tájékozódáshoz. Emiatt szükségem volt egy olyan térképes szolgáltatásra, ami az előbbi céloknak megfelel. Mielőtt megkezdtem a fejlesztést, több ilyen szolgáltatást is megvizsgáltam, abból a célból, hogy a legmegfelelőbbet tudjam kiválasztani a tapasztalatok alapján a szakdolgozatomhoz. Az egyik legfontosabb szempont a kiválasztásnál az volt, hogy ingyenesen elérhető legyen a szolgáltatás. A kutatás során két fő jelöltre sikerült leszűkítenem a listát, a Google Maps-re illetve az OpenStreetMap-re. Mivel az általam készített programot Android platformra terveztem elkészíteni, ezért olyan térképes API-ra volt szükségem, amit be lehet építeni Android applikációba, és ennek a célnak mind a kettő szolgáltatás megfelelt. Mindezek mellett fontos volt számomra, hogy olyan térkép alkalmazást válasszak, ami felhasználói körökben jól ismert. Emiatt megvizsgáltam, hogy az alkalmazásoknak mekkora a felhasználói köre. Az OpenStreetMap Android alkalmazása körülbelül 5 millió felhasználóval rendelkezik, míg a Google Maps letöltése meghaladja az 1 milliárd felhasználót. Ezekből az adatokból következtetni tudtam arra, hogy a felhasználók szélesebb köre miatt a Google Maps felülete sokkal szélesebb körben ismert az emberek között.

	Google Maps	OpenStreetMap
Ingyenesen elérhető	igen	igen
Lekérdezési limit	nincs limit	1 lekérdezés/másodperc
Felhasználók száma	~1 milliárd	~5 millió
Beépíthető modul	saját modulok	külső modulok
Megbízhatóság	ellenőrzött, karbantartott	közösségi adatfelvitel

3.1. ábra. A Google Maps és az OSM összehasonlítása

A 3.1.ábrán látható szempontok figyelembe vételével végül a Google Maps-re esett a választásom, mint integrálható térképes szolgáltatás.

Android, Java

Az Android platformon futó alkalmazások elsődleges programozási nyelve a Java. A Java platformfüggetlen és széles körben elterjedt, a Sun Microsystems által fejlesztett nyelvet az

1990-es évek óta folyamatosan fejlesztik. Az Android egy Linux alapú operációs rendszer, amely több pozitívummal is rendelkezik, ilyen például a hordozhatóság és a biztonság. Futtatási sebessége a Java-hoz képes javult, hiszen a kibővített Java programozási nyelvet Android esetén egy olyan virtuális gépen futtatják, amely szerves részét képezi az Android operációs rendszernek.

Symfony

A Symfony-val készült webes alkalmazások könnyen karbantarthatóak és jól skálázhatóak, mivel a keretrendszer az MVC tervezési mintára épül. A Symfony felépítése moduláris, ennek köszönhetően könnyen bővíthető. Továbbá a Symfony egy-egy komponensét más keretrendszerrel készült alkalmazásban külön is használhatjuk, amit szintén a moduláris szerkezet tesz lehetővé. minden beérkező kérést a keretrendszer dolgoz fel, melyek az útvonalválasztás után meghívják a megfelelő kontrollerek metódusát. Továbbá a kontrollerek hozzáférnek a services.yml fájlban definiált szolgáltatásokhoz is. Az itt definiált szolgáltatások példányait, a Symfony dependency injection segítségével juttatja el a kontrollerekhez. A Symfony segítségével a dinamikus tartalom is könnyen átalakítható a benne található template nyelvek egyikének köszönhetően. Az általam választott nyelv a twig, amellyel a megjelenítés elválasztható az üzleti logikától.

MySQL

A MySQL egy SQL (Structured Query Language, vagyis struktúrált lekérdező nyelv) alapú relációs adatbázis-kezelő szerver. Teljesen nyílt forráskodú, feltételezhetően emiatt terjedt el széles körben. Egyszerűen használható és költséghatékony megoldást nyújt dinamikus webhelyek számára. Az adatbázis kezelésére a phpMyAdmin adminisztrációs eszközt használtam, amely segítségével Interneten keresztül menedzselhettem az adatokat.

Doctrine

A Doctrine keretrendszer az adatbázisban található adatok elérésére alkalmas. A segítségével PHP osztályok képezhetőek le relációs adatbázis táblákká. A keretrendszer lehetőséget ad arra, hogy az alkalmazás adatbázisa különösebb ráfordítás nélkül lecerélhető legyen. A DQL, vagyis a Doctrine lekérdező nyelv használatával lehet az adatokat elérni. A Doctrine véd továbbá az SQL injektálásos támadásokkal szemben is. A Doctrine úgynevezett Repository-kat használ az adatok lekérdezésére, amelyek lehetővé teszik, hogy alkalmazás logikája és a lekérdezés megvalósítása elvonatkoztatható legyen egymástól. Továbbá azért esett a választásom a

Doctrine-ra, mert a keretrendszer integrálva van a Symfony-ba, illetve szolgáltatásként elérhető a Doctrine EntityManager objektuma. Új entitások generálásában és az adatbázis tábláinak frissítésében is segítséget nyújt a Symfony parancssoros eszköze.

4. fejezet

Felhasználói kézikönyv

A 4.fejezetben bemutatásra kerül a szakdolgozat keretében fejlesztett Android alkalmazás felhasználói szemszögből. Ebben kifejtésre kerül milyen menüpontok találhatóak az applikációban, és ezek milyen funkcionálitással rendelkeznek. Továbbá ismertetem az adatok menedzselésére szolgáló weboldal felületét és működését, amely az admin felhasználók munkáját teszi könnyebbé.

Android alkalmazás

Az applikáció ikonja egy Veszprémben járműparkjában is szereplő autóbusz, az Ikarus 280. Az alkalmazás indítása után az alkalmazás lekéri az adatbázisból azokat az adatokat, amik módosítási dátuma későbbi, mint az utolsó letöltés ideje. Ezt a felhasználó egy felugró ablak képében látja, amelyen az alkalmazás közli, hogy frissítés van folyamatban, és kéri a felhasználók türelmét. Amint az adatok aktualizálása befejeződött, a felugró ablak eltűnik, átadva helyét a főmenü menüpontjainak.

Főmenü

A főmenü az alkalmazás nevét és négy almenüt foglal magába:

- Útvonaltervezés
- Menetrendek
- Megállók
- Kedvencek

A . ábrán látható az alkalmazás főmenüjének kiosztása. Az applikáció témája a lila szín és annak árnyalatai, amely lehetővé teszi a felhasználók figyelmének felkeltését, mégis letisztult külsőt kölcsönöz.

Útvonaltervezés

Az Útvonaltervezés menüpontot kiválasztva egy Google Maps térkép jelenik meg Veszprém városára fókuszálva. A képernyőn ezen kívül egy beviteli mező és három gomb található. A beviteli mezőt a felhasználók a térképen való keresésre használhatják. A keresés fő fókuszsa Veszprémre irányul, a mező automatikusan próbálja kiegészíteni a begépelezett helyszínt, Veszprém környékére összpontosítva, ahogy az a . ábrán is látható. Az alkalmazás képes más földrajzi helyre irányuló kereséseket is kiegészíteni, viszont ez a funkció - az alkalmazás minél effektívebb működése érdekében - le van korlátozva. Ahogy a . ábrán is látszik, a találatot az alkalmazás egy úgynevezett 'marker' lerakásával jelöli a térképen. A bal oldalon található gombbal a felhasználó képes a térkép nézetének megváltoztatására. Az alapérmezett mód a domborzati megjelenés, a gomb megnyomásával átválthatunk műholdas nézetre. A jobb oldalon két gomb helyezkedik el: a Saját pozíció, illetve az Útvonaltervezés. A Saját pozíció gomb használhatához szükség van GPS funkcióra a telefonban. Ha ez nincs engedélyezve, az alkalmazás egy felugró ablak segítségével átirányítja a felhasználót a GPS funkció bekapcsolására alkalmas képernyőre. Visszatérve az alkalmazásba, a térképen megjelenik a felhasználó feltételezhető helyzete egy kék ponttal jelölve. Továbbá a térképen megjelenik egy világoskék kör az előbb említett kék pont körül. A világosabb színezetű kör területén valamelyik pont a felhasználó biztos pozícióját jelöli, a kör nagysága az internet elérési módjától (mobilnet vagy Wifi) függ. Ez a funkció azoknak a felhasználóknak nyújt segítséget, akik számára Veszprém városa ismeretlen terület. A jobb alsó gombra kattintva a felhasználó átkerül az Útvonaltervezés oldalra, ahol a felső beviteli mező mellé egy újabb mező kerül. Ha az előző oldalon a mezőbe került már földrajzi hely, az alkalmazás automatikusan az alsó mezőt, az Utazás célját tölti fel vele. Az új képernyőn megjelenő ablakban is lehetőség van a saját pozíció lekérésére, ebben az esetben az alkalmazás a paramétereit a Kiindulási pont mezőbe tölti be. Abban az esetben, ha mind a kettő mezőt kitöltötte a felhasználó, akkor az Útvonaltervezés gomb ismételt megnyomásával átkerül a találati listára. Az alkalmazás az . ábrán látható módon jeleníti meg a találatokat. A képernyő felső részén a beállított indulási és érkezési helyszín van feltüntetve, alul pedig három féle szempont alapján a megoldás:

- Legkevesebb utazási idő
Ennél a lehetőségnél az alkalmazás kiszámolja a lehetséges útvonalakban a gyaloglással és utazással töltött idejét, és ezek közül a legrövidebbet jeleníti meg.
- Legkevesebb gyaloglás
A második opción azt a találatot listázza ki, amelynél a gyaloglással töltött idő a legrövidebb.
- Legkevesebb átszállás

Az utolsó alternatíva a jeggyel utazóknak kínál utazási megoldást, hiszen a legkevesebb átszállással járó utazást jeleníti meg.

A felhasználó az egyik megoldásra kattintva választhatja ki a számára optimális útvonalat. Az alkalmazás a következő képernyőn az útvonalat térképen jeleníti meg, ahogy az a .ábrán is látszik. A kezdő és a végpont jelölője piros színű, a busz útvonalát az eltérő színnel (jelen esetben lilával) jelölt megállók mutatják. Ha az útvonalban több busz is szerepel, akkor azok útvonalát egy harmadik színnel jelöli az alkalmazás. A gyalogos útvonalat az applikáció színes vonallal jelöli a térképen a piros jelölők és az első/utolsó buszmegállók között. Ha a választott megoldás mégsem felel meg a felhasználó igényeinek, a képernyő felső részén elhelyezkedő Útvonaltervezés gomb ismételt megnyomásával visszajut az utazási lehetőséget listázó oldalra.

Menetrendek

A Menetrendek a főmenü második menüpontja. Ez az opció azokat a felhasználókat segíti, akik már ismerik Veszprém városát, illetve tömegközlekedését legalább alap szinten. A menüpontra kattintva az alkalmazás kilistázza a buszjáratok számait. Amint kiválasztott a megtekinteni kívánt járatot, az applikáció bekéri a járat irányát is, majd kilistázza az indulási időket az .ábrán látható módon. Az megnyíló ablak fejléce tartalmazza az előbbi felhasználói akciók során bekért adatokat, illetve a Kedvencekhez adás funkciót egy szív ikon formájában. A képernyő bal oldalán a járat megállói láthatóak, a választott iránytól függő sorrendben. Az időpontok jobb oldalon, munkanapokra és szabadnapokra felosztva láthatóak.

Megállók

A Megállók menüpont alatt egy Google Maps térkép jelenik meg rajta Veszprém város buszmegállóival. A megállók pontos helyét piros színű jelölők jelzik a térképen, amely a közelítés mértékével arányosan csoportosítja a jelölőket. A felhasználó megkeresi a térképen azt a megállót amelyikre kíváncsi, az alkalmazás pedig kilistázza az ott megálló járatok számát, ahogy az .ábrán is látszik.

Kedvencek

A menüpontban a Menetrendek alatt Kedvencnek jelölt járatokat lehet elérni. Az alkalmazás kilistázza azokat az adott menetirányhoz tartozó járatokat, amelyeket a felhasználó kedvencnek jelölt. A funkció egy szív ikon formájában jelenik minden menetrend adatlapján. Alapérmezetten a szív üres, rájuk kattintva tudja a felhasználó a Kedvencek menüpont alá

rakni, ekkor a szív pirosra színeződik. A listából eltávolítani hasonló módszerrel lehet, a telt szívre rákattintva kikerül a Kedvencekből.

Admin oldal

Az alkalmazás létrehozásakor szükség volt egy olyan kezelőfelület létrehozására is, amelyen keresztül könnyen kezelhetőek az adatbázisba feltöltött adatok. Mivel az adatok közérdekek és sok embert érintenek, fontos szempont volt, hogy csak azok férhessék hozzá akiknek van jogosultságuk. Az oldal eléréséhez a felhasználónak igazolnia kell magát egy felhasználónév-jelszó párossal, ahogy az a .ábrán is látszik. Az . ábrán látható admin oldal jelenik meg a sikeres bejelentkezés után. Bal oldalon található a menü, amiből a felhasználó kiválaszthatja azt az elemet, amit menedzselni szeretne. Alapértelmezetten a Járat menüpont jelenik meg, ez az adatbázis architektúrájának alapja. A másik alapvető entitás a Megállók, amelyek a buszmegállók pontos földrajzi helyzetét tárolják. Egy megálló felviteléhez hosszúsági és szélességi koordinátákra van szükség, amelynek felvitelét egy beágyazott Google Maps térkép segíti a weboldalon. A felhasználó kijelöli a térképen a megálló pontos helyét, aminek a koordinátái megjelennek a térkép mellett és az adminisztrátornak már csak nevet kell hozzárendelnie az új rekordhoz. Hasonló kezelőfelület található a megállók útvonalakhoz való rendelését kezelő menüpontban. Egy új útvonal létrehozása esetén az adminisztrátor a Google Maps-en elhelyezkedő jelölők közül kiválasztja a megfelelőt. Ezek a jelölők az adatbázisban található megállókat jelölik a térképen elhelyezve a könnyebb kezelés érdekében. Ha egy útvonal módosul (például egy felújítás miatt), akkor az egy új útvonalként felvezetésre kerül, az alapvető útvonal állapota pedig inaktívvá válik. Amint az adminisztrátor sikeresen az adatbázishoz adott egy buszjáratot, a megfelelő útvonalhoz és járathoz indulási időket rendel. Ezeket az időket előre definiált kategóriákba menthetjük el, ilyen kategória például a szabadnap, a munkanap vagy a tanszüneti munkanap. Mindemellett előfordulhatnak olyan időpontok a naptárban, amikor nem az alapértelmezett közlekedési rend a mérvadó, például hétköznapra eső ünnepnapokon. Mivel ezek évről évre változnak, így az adminisztrátor feladata, hogy megfelelően felvezesse ezeket a napokat az adatbázisba. Erre szolgál az Elterő közlekedési rendek menüpont, ahol a megadott időintervallumra vonatkozóan felülírhatjuk az alapértelmezett rendet.

Az adatok átláthatóságának érdekében táblázatban jelennek meg az adatbázisból lekért rekordok. Az adminisztrátor a keresés funkció segítségével kereshet az adatok attribútumai között, illetve sorba rendezheti azokat a gyorsabb kezelés érdekében.

5. fejezet

Fejlesztői dokumentáció

Az elkeszult rendszer szerver-kliens architekturara épül. Az architektura modellje a ???. Ábrán látható. A szerver felelos az adatok kezeléséért, mely a Symfony keretrendszerrel lett megvalósítva. Az adatokat a Doctrine keretrendszer segítségével eri el az adatbázisból. A rendszerben található a szerver, ami egy adminisztrációs feladatokat betölteni weboldal, és a kliens, egy Android alkalmazás, amely a felhasználók részére készült. Az adminisztrációs weboldalon az adatok a Twig sablonkezelő segítségével jelennek meg. Az Android-os kliens a kérő adatokat JSON formátumban kapja meg. A továbbiakban részletesen bemutatásra kerül a szerver és a kliens telepítése, működése.

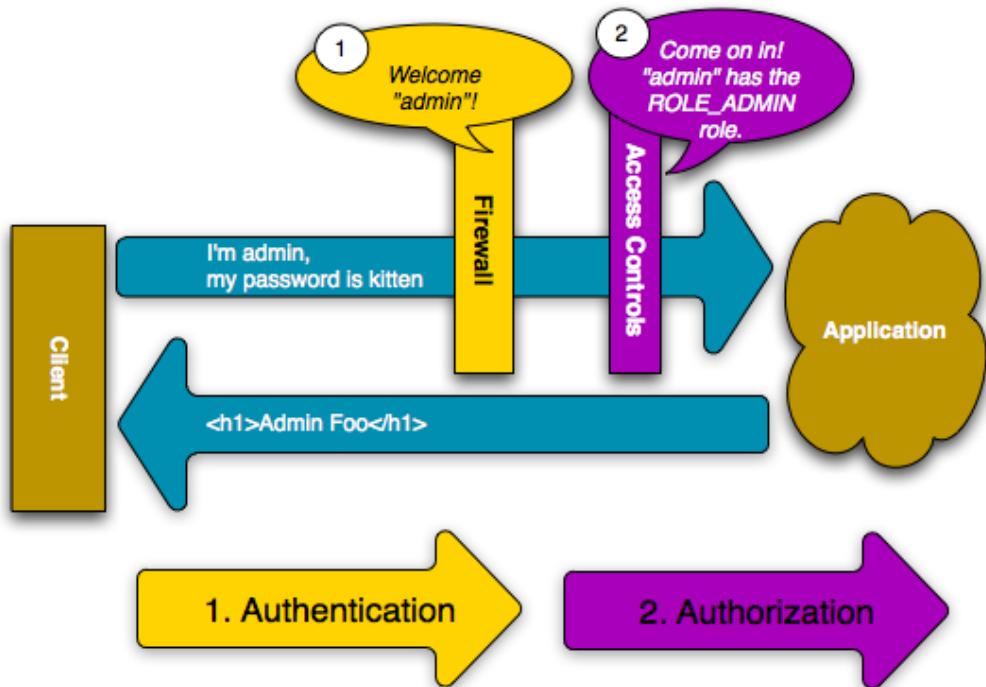
A szerver

A szerver a Symfony keretrendszer ???-as verziójára épült. A beérkező keresések áthalad a Symfony tuzfalan, ami elindítja az autentikációt. Ha sikeres volt az autentikáció, a keresés a megfelelő kontroller osztályhoz kerül feldolgozás után.

A kontroller osztályban a beírt URL-hez tartozó metodus hívódik meg. Alapesetben ezek az osztályok a *AppBundle\Controller* nevűben találhatóak. Az URL-címek metodusokra való leképzése ketkelezőleg valósítható meg a Symfony keretrendszer használatával.

- routing.yml fájlban való definíálás
- kontroller osztályokban annotaciók

Az első opció akkor elenyos, ha útvonal alapján akarjuk meghatározni, hogy mely kód részlet fog vegrehajtódni a kontrollerek megfelelő kialakításával és az annotaciók jó elhelyezésével. Másrészről a második megoldás itt elenyősebb lehet, mert ilyenkor a kódot latva a fájl elhagyása nélkül meg tudjuk állapítani, hogy milyen esetekben fog a kód részlet meghívásra kerülni. A metodusok annotációi között megszabhatunk különböző paramétereket is, melyet az alábbi kód részlet demonstrál:



5.1. ábra. Az autentikacio folyamata

```

1  /**
2  * @Route("/{id}/edit", name="line_edit")
3  * @Method({"GET", "POST"})
4  */
5  public function editAction(Request $request, Line $line)
6  {
7      // ...
8 }
```

Autentikacio

Az autentikacio a Symfony tuzfal moduljaval tortenik. Az ehhez szukseges beallitasok a `app\config` mappa `security.yml` konfiguracios fajljaban talalhatoak. A szerverhez jelenleg ket fajta felhasznalo van rendelve, melyek az elobbiekben emlitett fajlban vannak definialva. A szerver feladatkorebol kiindulva a jelenlegi igenyeket ez a ket felhasznalo kielegiti, ezert nincs szukseg ujabb felhasznalok hozzaadasara. Ez azt jelenti, hogy a felhasznalokhoz tartozo jelszavakat is ebben a fajlban taroljuk. Biztonsagi szempontokbol kritikus tenyezo, hogy ezek a jelszavak valamelyen modon titkositva legyenek. A titkositashoz a bcrypt algoritmust hasznaltam fel. A Symfony segitsegevel parancssorbol lehetseges barmilyen jelszot titkositani a bcrypt algoritmussal.

```
1  php bin/console security:encode-password
```

A parancs futása közben bekéri, mi az a jelszo, amit titkositani kell. Miután megadtuk a jelszót, a bemeneten lefuttatja az algoritmust és vegül kiirja a titkositott karakterszorozatot. Ezt a szót megadva a fajlban jelszokent, továbbra is a titkositatlan jelszoval be tud lépni a felhasználó az oldalra, viszont nem áll fent tovább a veszély, hogy illetéktelenek kezebe jutna a jelszo.

Autorizacio

Az autorizacio folyamata szintén a Symfony tuzfal moduljának segítségével valósult meg. Ez a sikeres bejelentkezést követően hajtódik végre. A folyamat célja, hogy a bejelentkezett felhasználó csak a száma által kijelölt szolgáltatásokat, oldalakat erje el. Az előző pontban említett két felhasználóhoz tehát az alábbi két szerepkör tartozik:

- Latogató
- Adminisztrátor

A latogató szerepkorhoz tartozó felhasználóknak csak az adatbázis-entitások lekerdezéséhez van jogosultsága. Az adminisztrátori engedélyel rendelkező viszont a weboldal minden szolgáltatáshoz hozzafer.

A szerepkorokhoz hozzá lettek rendelve meghatározott formájú URL címek.

```
1  access_control:
2      - { path: /json$, roles: ROLE_USER }
3      - { path: ^/, roles: ROLE_ADMIN }
```

Az alábbi kód részlete azt mutatja be, hogy a latogatói jogkor csak a *json* végű URL-címekhez ad elérést, miközött egy adminisztrátor minden címet elér. Ugyanezért a funkcionálitást meg lehet valósítani szintén annotaciókkal is, de a weboldal kialakítása miatt célszerűbb volt ezt a megoldást valasztani, hisz így egyetlen fajlban elegendo meghatározni a hozzaferési szabályokat, annotaciók használatával viszont minden kontroller osztályban egyenként kellett volna meghatározni ugyanazokat a szabályokat.

Adatbazis kapcsolat

Az adatbazist a Doctrine keretrendszer használva eri el a szerver. Az alkalmazáshoz tartozó entitások helye a *AppBundle\Entity* nevűter. Ezek mindegyike egyszerű PHP osztályok,

a felhasznalt keretrendszer alakitja ezeket az adatbazis tablaiva. Az osztalyban szereplő adatmezőkhoz meg kell adni az annotaciokat, hogy az átalakítás minden módon történjen meg. Az entitasok egymás között kapcsolatait a következők jelzik:

- OneToOne
- OneToMany
- ManyToOne

Az adatbazisban található adatok lekerdezés és objektumba való átalakítása a *Repository* osztályok segítségével történik. Ezek az osztályok a Doctrine *EntityRepository* osztályából származnak, mely már rendelkezik a legalapvetőbb lekerdezésekkel, mint az azonosító vagy valamilyen tulajdonság alapján történő adatlekerés. Az összetettebb lekerdezésekhez a DQL nyelvet használtam fel, mely az entitasok kapcsolatait az entitashoz tartozó osztályban megadott annotaciók alapján kezeli. Amennyiben az adatbazisból lekerdezett adatok módosultak, ahhoz, hogy ez a változás megmaradjon el kell menteni az adatbazisba. A Doctrine keretrendszerben található automatikus megoldás, megpedig az EntityManager osztály, amely erzékeli a változást és elmenti a *flush()* metodust használatával. Ha új adatot akarunk az adatbazishoz adni, azt is az EntityManager osztályal tehetjük meg. Az elkeszült új objektumot parameterként adva az osztály *persist()* metodusának, a Doctrine keretrendszer úgy kezeli a továbbiakban az objektumot, mintha az az adatbazisból lekerdezett lenne, és így a *flush()* metodussal az adatbazisba tudja irni.

Szerver feladatai

Az alábbiakban bemutatásra kerülnek a szerver főbb feladatai.

Jaratok

A városban közlekedő buszok minden- minden vonalpont között közlekednek. Elfordulhat, hogy ugyanolyan számmal ellátott busz ugyanazon vonalpontok között közlekedik, de a köztes megallok elterhetnek, vagy a köztes megallok ugyanazok, de a vonalpontok mások. Ezeknek az információk összefogására létrehozott *LineInfo* entitas felelős. Ez az entitas tárolja a két vonalpontot, a közöttük közlekedő buszjáratot, és a járat típusát. A vonalpont és a buszjáratok szintén entitasok, az ezek között fennaló kapcsolat az 5.2. ábrán latható.

5.2. ábra. A járat entitas kapcsolatai

Utvonalak

A buszjáratok utvonaluk bejárása során több, köztes megállót is érintenek. Meg kell hataroznunk, hogy egy adott busz milyen irányban haladva mely köztes megállókban áll meg. Ezt a feladatot a *LineHasStations* entitas végezi. Az entitas segítségével meg tudjuk hatarozni, hogy az előző pontban vezetett járat egy példányának a végpontjain kívül milyen más állomásai vannak. Továbbá taroljuk a köztes megálló utvonalon belül sorszámát, melyből megállapítható az utvonalon található állomások helyes sorrendje. minden egyes objektum esetén tarolásra kerül meg, hogy az előző megállóból milyen gyorsan tudunk eljutni. Természetesen a busz indulópontjának ezen attributuma nulla. Az 5.3. ábrán megtékinthető milyen kapcsolatokat tartalmaz az utvonal entitas.

5.3. ábra. A járathoz kapcsolatai

Indulási idők, különleges időpontok

Utvonaltervezés szempontjából fontos információ továbbá, hogy az adott járat az utvonalan található megállóiba milyen időpontban érkezik meg, továbbá különleges alkalmak esetén ezek hogyan változnak meg. Az indulási időpontot egy *Starttime* nevű entitas tartalmazza, melyben tarolódik továbbá az is, hogy ez az indulás mely járatra vonatkozik, valamint minden típusú datum esetén ertetendő. A különböző típusú időpontokat külön entitas tartalmazza. Az alkalmazáshoz tovább tartozik egy *Holidays* entitas is, melyben az olyan, előre nem tervezett időpontokat tartalmazza amelyek befolyásolják bizonyos járatok indulási rendjét és esetleges utvonalat is. Az előbbiekben bemutatott entitások közötti kapcsolatot a 5.4. ábra mutatja be.

5.4. ábra. Különleges és indulási időpontok közötti kapcsolat

Az Android kliens

Az alkalmazás fejlesztésehez az Android Studio fejlesztoi környezetet használtam. Az Android Studio a Gradle rendszerre épül, amelynek számos funkciója megkönnyíti az Androidos alkalmazások fordítását. A Gradle Androidos bovitmenyében található több Android specifikus parancs, amelyek közül az *assemble* parancssal készíthetjük el a futtatható és telepíthető állományt. A futtatható fajl elkesztésehez szükséges információkat a build.gradle fájlban

adhatjuk meg. Ebben talalhatoak a szukseges fuggosegek, amiket a Gradle bovitmenye letolt, es belecsomagol az .apk-ba. Tovabba itt talalhato az applikacio altal tamogatott Android verziok szama. Az altalam készített alkalmazas futtatasahoz legalabb a *minSdkVersion 14* verzio szukseges, ami az Android operacios rendszer 4.0 verziojanak felel meg. Az *AndroidManifest.xml* fajl az alkalmazas alapveto jellemzoit tartalmazza. Ilyen jellemzok tobbek kozott a szukseges engedelyek es a Google Maps-hez tartozo metainformacio. Az alkalmazas telepitesehez es futtatasahoz szukseg van engedelyekre a felhasznalo reszerol. Ezek az engedelyek a kovetkezok:

```
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
3 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Az elso hozzaferes az Internettel valo kommunikaciohoz szukseges. Szuksegunk van tovabba olyan engedelyre amivel a Sajat pozicio lekerheto es engedelyezheto a GPS a telefonon. Mivel az alkalmazasnak szuksege van Internet hozzaferesre az adatok letoltesehez, ezert az alkalmazas engedelyt ker a telefon halozati allapotnak lekeresehez is.

Helyi adatbazis

Fontos szempont volt helyi adatbazist letrehozni, mivel az alkalmazas es a szerver kozott folyamatosan felepulo kapcsolat sok eroforrast igenyel. Tovabba az olyan eseteknel, amikor a keszulek nincs halozatra kapcsolva, akkor nem erne el a szervert, ily nem toltohdnenek le a mukodesehez szukseges adatok. Ezek kovetkeztaban szukseg volt arra, hogy az applikacio offline uzemmodban is kepes legyen funkcionisan mukodni.

Megvizsgaltam tobb Android platformhoz elerheto ORM (object-relational mapping vagyis objektum-relacios lekepezes) keretrendszer, ezen belul is az ORMLite keretrendszer. Az ORMLite akkor elonyos, ha nem letezik adatbazis sema, amihez alkalmazkodnia kell. Mivel az alkalmazas eseteben mar a szerver oldalon kialakitasra kerult egy sema, emiatt a keretrendszer hasznalata nehezkes volt. Ebben a kontextusban szukseges volt, hogy manualisan szemelyre szabhassam a helyi adatbazis semajat, ily vegul a nativ SQL utasitok hasznalata mellett dontozem.

A helyi adatbazis-kezelo rendszer az SQLite, amely az Android operacios rendszerbe gyarilag beepitett adatabazis-kezelo motor. Mivel az SQLite relacios adatbazis, ezert a kommunikacio szimpla SQL utasitasokkal tortenik. Az applikacio sikeres mukodesehez szukseges egy *SQLiteOpenHelper* osztalybol orokoltetett osztaly, amelyben az adatbazis nevet, verziojat es az adatbazis letrejottekor vegrehajtando utasitasokat taroljuk. Az ososztalybol ketto metodust kotelezo implementalni, az onCreate es az onUpgrade metodust. Az onCreate

metodus segitsegevel van lehetoseg tablakat letrehozni az eppen elkeszult adatbazisba, amit az alábbi kodreszlet szemleltet.

```
1  @Override
2  public void onCreate(SQLiteDatabase sqLiteDatabase) {
3      try {
4          sqLiteDatabase.execSQL(DataTypeEntry.CREATETABLE);
5          /* Create other tables similar to the above one. */
6      } catch (SQLException e) { /*...*/ }
7 }
```

Ugyanakkor az onUpgrade metodus abban az esetben fut le, ha megnoveltuk az adatbazis verziojat. Ez a verzioszam novekedes azt jelzi, hogy olyan valtozas tortent az adatbazisban, amely visszafele nem kompatibilis, es egy transzformacio szukseg es a regi es az uj sema kozott. Az *SQLiteOpenHelper* osztaly az SQL utasitasokat tranzakcionalisan hatja vegre. Egy tranzakcio tartalmazhat tobb SQL utasitast is, amelyek vegrehajtasa egyszerre tortenik. Ha egy SQL utasitas sikertelen, akkor az osszes utasitas visszavonasra kerul.

Az adatbazis minden tablajahoz egy statikus osztaly lett letrehozva, melyek egy DatabaseContract nevu osztalyban tarolodnak. minden belso osztaly implementalja a BaseColumns osztalyt, melynek koszonhetoen megkapjak az egyedi azonositasra hasznalatos `_id` mezot. Erre egy pelda az alábbi kodreszletben lathato. A további attributumokat is implementalasra kerultek az adott osztalyokban, majd a mezok alapjan elkeszult a tabla letrehozasahoz szukseges SQL parancs.

```
1 static class DataTypeEntry implements BaseColumns{
2     static final String TABLE_NAME = "date_type";
3     static final String COLUMN_NAME_TYPE_NAME = "type_name";
4     static final String COLUMN_NAME_ENABLED = "enabled";
5     static final String CREATETABLE = "CREATE TABLE IF NOT EXISTS "+
6         TABLE_NAME+"( "+_ID+" INTEGER PRIMARY KEY NOT NULL, '"++
7             COLUMN_NAME_TYPE_NAME+" TEXT NOT NULL, '"+COLUMN_NAME_ENABLED+" INTEGER
8             DEFAULT 1);";
9 }
```

Az applikacio lokalis adatbazisaban levo adatok POJO-ban tarolodnak, minden tablahoz kulon POJO tartozik. A POJO egy olyan Java objektum, amely nem rendelkezik specialis tulajdonsaggal.

Ahhoz, hogy az alacsonyabb szintu SQL utasitasok elkulonuljenek az uzleti logikatol, letrehoztam egy *DataManager* nevu absztrakt osztalyt, melynek tipus parametere egy POJO objektum. Mivel az osztaly absztrakt - ily nem peldanyosithato - ezert ebben az osztalyban kaptak helyet az olyan metodusok, melyek minden tipusu POJO-ra altalanos ervenyes seguek. Az alábbi kodreszletben a *DataManager* osztaly implementacioja lathato, ahol a `createOrUpdate` es a `queryForAll` metodusok az adatbazis tablainak kezeleseert felelosek.

```

1 public abstract class DataManager<T> {
2     DatabaseHelper helper;
3
4     /* ... */
5     public abstract void createOrUpdate(T data);
6     public abstract List<T> queryForAll();
7 }

```

Ebbol az osztalybol szarmaznak a specifikus osztalyok, melyeken keresztul adott tipusu POJO-k kerulnek lementesre vagy visszakerdezesre a helyi adatbazisbol. Az ososztalyban talalhato metodusokat mindenkepp implementalnia kell minden szarmaztatott osztalynak. Az olyan funkcionalisok eseten - amelyek nem minden POJO eseten voltak ertelmezhetoek - a specifikus osztalyban kerultek definialasra. Az alkalmazas mindenhol ezeket a manager osztalyokat hasznalja, ahol az adatbazissal valo interakcio szukseges.

Szerver-kliens kapcsolat

A helyi adatbazisban talalhato adatok csak egy adott ido-intervallumban ervenyesek, a menetrend valtoztatasaval ervenyeket vesztek. Ezert bizonyos idokozonkent frissiteni kell a benne szereplo informaciokat. Az adatok aktualizalasa az alkalmazas inditasakor tortenik, erre szolgal a korábban bemutatott szerver, ahonnan az alkalmazas az aktualis informaciokat kapja. A szerver REST uzenetekkel valaszol a beerkezo keresekre. A kommunikaciót a Retrofit nevu REST klienssel oldottam meg, melynek segitsegevel konnyu a szervernek kereseket kuldeni. A *NetworkManager* osztaly kepes lekerdezni, hogy az adott eszkoz rendelkezik-e jelenleg barmilyen internet eleressel. Ha a keszulek nem rendelkezik internet eleressel, akkor csak a helyi adatbazist hasznalja. Ellenkezo esetben pedig elkezdodik az adatok frissitese az applikacio elinditasaval. A minel kisebb adathasznalat erdekeben a kliens tarolja, hogy mikor tortent az utolso frissites. Az applikacio inditaskor elkului a legutolso frissitesi datumot a szervernek, a szerver pedig csak azokat az adatokat kului vissza, amelyek ujabbak, mint a kapott idopont. Miutan az adatok aktualizalasa sikeresen megtortent, a frissites idoponja felulirasra kerul a helyi adatbazisban. Az IDownloader interfesz a A *BaseDownloader* absztrakt osztaly implementalja, melynek konstruktoraban kerulnek inicializalasra a Retrofit szamara szukseges osztalyok. Lehetosegunk van minden kerest szemelyre szabni. Ezt felhasznalva juttatjuk el a szerver szamara a felhasznalo azonositasara szolgalo adatokat es a legutolso frissitesi idopontot. Az IDownloader interfesz egyetlen metodus-definiciot tartalmaz:

```

1 public interface IDownloader {
2     void download();
3 }

```

A *BaseDownloader* osztaly tartalmaz meg egy `saveToDatabase` metodust, amely az adatok helyi adatbazisba torteno lementesere alkalmas. A *BaseDownloader* osztalybol szarmaztatott gyerkosztalyok implementalják az elozoekben emlitett ket metodust.

A Retrofit konfiguralasa soran egy interfesz kerult megvalositasra, amelyben metodus-definicio formajaban felsorolasra kerulnek azok a hivasok, amelyeket a Retrofit kepes vegrehajtani.

```
1 public interface MyApiEndpointInterface {
2     /* ... */
3     @GET("stations/json")
4     Call<List<Stations>> getStations();
5 }
```

A fenti peldan lathato, hogy a megallok eleresethez annotacio formajaban megadasra kerult az eleresi utvonal. Ennek visszateresi erteke egy olyan lista lesz, amely megallokat tartalmaz. A listaban talalhato megallokat a Retrofit automatikusan sserializalta ki a szervertol kapott JSON valaszbol. A sserializalashoz az alabbi entitasban talalhato annotacios konfiguraciót veszi alapul.

```
1 public class Stations {
2     /* ... */
3     @SerializedName("lat")
4     @Expose
5     private double lat;
6 }
```

A felhasznalo felulek kialakitasa

A kod karbantartagosaganak erdekeben a felhasznalo felulek kialakitasa az MVP (Modell-View-Presenter) szoftvertervezesi mintara epul. Elonye, hogy az uzleti logika elhatarolodik a felhasznalo felulettol, melynek koszonhetoen a kezelofelulet modosithato a logika megvaltoztatasa nelkul. Az alkalmazas minden kepernyo-nezetehez tartozik egy ugynevezett activity. Az activity a felhasznalok es az applikacio kozotti legfobb alapelem.

6. fejezet

Továbbfejlesztési lehetőségek

Irodalomjegyzék

MELLÉKLET

A mellékelt CD könyvtárszerkezete