Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedésmérnöki és Járműmérnöki Kar
Közlekedésüzemi és Közlekedésgazdasági Tanszék

SZAKDOLGOZAT

A Budapesti P+R Parkolás Informatikai Támogatása Applikáció Fejlesztésével

2016

Tartalomjegyzék

1. Bevezetés

A közlekedés, mint emberi szükséglet már az emberiség civilizálódásával megjelent. Ahhoz, hogy a mindenkori gazdasági rendszer működhessen, legyen az kezdetleges, mint az ókorban, vagy modern, mint napjainkban, helyváltoztatásra van szükség.

„Az emberi tevékenységek helyének térbeli elkülönülése közlekedési igényeket támaszt. A közlekedési igények az emberi, és gazdasági kapcsolatok térbeli, és időbeli vetületei. A területi funkciók megoszlása, és a társadalmi munkamegosztás forgalmi igényeket generál.” [1]

A társadalom ilyen jellegű folyamatos változása, és a homogén népességeloszlás eltolódása az ipari forradalomban erősödött meg. Az addig főként mezőgazdasággal foglalkozó lakosság nagy része gyárakban kezdett dolgozni, a gyárakat pedig a városokba, illetve azok közvetlen szomszédságába építették. Megfelelő közlekedési lehetőség híján, az embereknek be kellett költözni a városokba. Elkezdődött a városodás folyamata. A városodás, idegen szóval urbanizáció egy népességvándorlási folyamatot jelent, a vidéki részekről a városokba költöznek az emberek, azaz a városok mérete, és városlakók száma növekszik. Az idő múlásával a városok akkorára nőnek, földrajzi kiterjedését, illetve lakosságát tekintve, hogy valamilyen közlekedési rendszer megteremtése szükségessé válik. A városok szerkezete nem homogén, hanem vannak sűrűbben lakott részek, ilyen a belváros, vannak ipar negyedek, amelyek a városi forgalomra, olyan hatást gyakorolnak, hogy az oda vezető útszakaszok forgalmi terhelését megnövelik. A városközpontokban funkciótömörülés van, közintézmények, munkahelyek, és egyéb olyan helyek, amelyek forgalmi igényeket generálnak a városközpont, és a külsőbb városrészek között. Megnövekedett nappali lakos szám a városközpontokban. A funkció tömörülés miatti nagyobb forgalmi igények lebonyolítására nincs elegendő szabad terület, a városvezetésnek különböző intézkedéseket kell hoznia, ahhoz hogy a lakossági igényeket kiszolgálhassák.

A faluról való meneküléssel, s a városba áramlással sem a középosztályosodás, sem a középosztály vidékre költözése nem tud lépést tartani. A fejlődő országokra egyszerre ereszkedik rá a különböző erők nyomása: a fejlett agrártechnika s a falu nyomora, mely elüldözi a fölösleges agrárnépességét, valamint a megkésett, a múltszázadban elmulasztott nagyfokú iparosítás, amely nyomorban hagyja a tanulatlant és végül a

posztindusztriális társadalom igénye, mely újfajta struktúrát, új ipart és újfajta, értelmiségi szakemberképzést követel. [2]

„Az egyre nagyobb koncentráció révén keletkező tömeg kiszolgálási problémák mellett, növekszik az egyéni igényeket kielégítő szolgáltatások iránti tömeges kereslet is. Olyan intelligens megoldásokat kell találnunk, amelyek egyrészről igen hatékonyak és fenntarthatóak, másrészről elősegítik a gazdasági prosperitást és az egyre komfortosabb, biztonságosabb életvitelt. A jövő internete az a platform, amely ezeket a kihívásokat kezelni tudja és az okos város (Smart City) koncepció egyike azon megoldásoknak, amelyek a jövő internete segítéségével a fent felsorolt problémákra megoldást kínálnak. Ezért felértékelődött, s mára már különösen fontossá vált a jövő internetének és ezzel párhuzamosan az okos városok kialakításának egyidejű kutatása.”[3]

A városok problémáinak vizsgálatában az okos város koncepciót követve kell a vizsgálati módszereket felállítanunk. Tehát a nem mérhető jellemzőket mérhetővé tenni, a beérkező mérési adatokat pedig fel kell dolgozni, ezáltal megfigyelhető a trend jellege.

Azonban a trendeket nem csak közlekedésmérnöki szemmel kell vizsgálni, hanem gyakran szociológiai nézőpontból is. Be kell látni, hogy a városodás folyamatát, mint jelenkori trendet nem tudjuk megváltoztatni, mert sokkal nagyobb gazdasági erők idézik, elő, mint a közlekedési beavatkozások gazdasági ereje. Tehát szükséges megtalálni azokat a megoldásokat, amelyek biztosítják számunkra, hosszútávon a fenntartható fejlődés lehetőségeit.

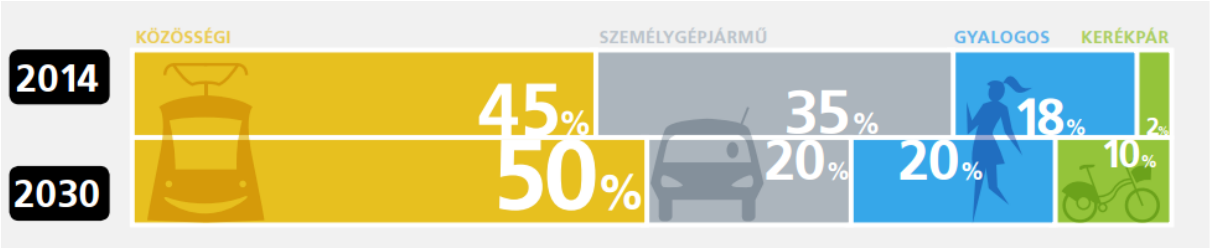
„Az IKT (Infokommunikációs Technológia) az a kulcstechnológia, amely ezeket a kihívásokat kezelni tudja és a Smart City (más néven okos város) koncepció egyike azon megoldásoknak, amelyek az IKT segítéségével a városok közlekedési problémáira hatékony választ kínálnak.” [3]

A Smart City technológiára nem csak egy-egy városban van igény, hanem a világ civilizált lakott területein szinte mindenhol szükséges, legalábbis előnyös, és fontos megjegyezni azt is, hogy ezeket a megoldásokat integráltan, interoperábilisan (rendszerek közötti átjárhatóság) kell megvalósítani. Ezt tükrözi, hogy az Európai Unió Smart City kezdeményezését a Horizon 2020 célokkal összhangban dolgozta ki.

Az okos városok kialakításának általános, koncepcionális kialakítása a következő:

- „Smart energetika
- Intelligens urbanisztika
- Intelligens közlekedés
- Klímaváltozás
- Környezetvédelem
- Életminőség – egészségügy
- Közösségi részvétel a városfejlesztésben
- Vízminőség és vízgazdálkodás
- Integrált intelligens infrastruktúra
- Smart alrendszerek informatikai támogatása” [3]

A városi közlekedés egyik minőségi jellemzője az alágazatok közötti teljesítmény megosztás, az úgynevezett modal split. A helyzet részletes felméréséhez feltétlenül szükséges ez a mutatót alaposan megvizsgálni, mind nominálisan, mind a trendszerűségére vonatkozóan. Manapság megfigyelhető, hogy a városi közlekedésben az egyéni motorizált forgalom volumene nem csökkent jelentősen. Az évek során voltak jelentős arányváltozások a modal split-ben azonban jelentősen még napjainkban sem tolódott el egyértelműen a közösségi közlekedés irányába. A modal split alakulása látható az elmúlt években a Balázs Mór terv felmérései szerint.



1. ábra: A Modal Split budapesti alakulása *[4]*

A közösségi közlekedést Budapesten és országszerte is folyamatosan támogatja a városvezetés informatikai fejlesztésekkel, újabb komfortosabb járművek beszerzésével, valós idejű utastájékoztatási anyagokkal. Célszerű olyan közösségi közlekedést kialakítani, hogy az utazók ezt válasszák. Ehhez folyamatosan kampányokat indít a városvezetés, reklámokkal, kedvezőbb ajánlatokkal próbál beavatkozni abba a kedvezőtlen folyamatba, hogy a mobilitási arány az egyéni eljutási módok irányába billent el.

„A Magyarországon ajánlott városi közlekedéspolitikának EU-konform, egységes irányelvei a következők:

- a fenntartható közlekedési igények kielégítése;
- a kiegyensúlyozott területfejlesztés támogatása;
- az igazságos piacszabályozás szabályozása;
- a közlekedési integráció támogatása;
- a minőség és a szolgáltatási színvonal javítása;
- az emberi élet és a környezet védelme;
- a teljesítményhez, a valós költségekhez igazított árak alkalmazása.” [5]

Ez gyakorlatilag részben ellentétes a közlekedés fejlesztésével, mivel általában a közlekedés fejlesztése infrastruktúrabővítést, járatsűrítést, ezáltal nagyobb forgalmat eredményez. Meg kell jegyezni, létezik olyan eset, amikor a közlekedésfejlesztés egyik célja egy adott részen a forgalomi terhelés csökkentése. A városok közlekedési rendszerének fejlesztési iránya, hogy az átlagosan megtett út hosszát minimalizáljuk, méghozzá úgy, hogy a várost tudatosan építjük fel, rendezzük be.

Gyakorlatilag a régen épült városrészeket nem igazán tudjuk átépíteni, ezért itt más megoldásokat kell keresni, átszervezéssel javítható a helyzet. Az újonnan épített városrészeket, pedig tudatosan át kell gondolni. Célszerű úgy kialakítani, hogy gyaloglási távolságban minden elérhető legyen. Azaz körülbelül 500 méteres körzetben legyen zöldterület, közintézmények, óvoda, iskola bevásárlási lehetőség. Mivel belátható, hogy nem igazán lehet ezt így megépíteni, törekedni kell arra, hogy a közforgalmú közösségi közlekedés legyen elérhető gyalogosan, vagy az előbbi intézmények elérhetők legyenek nem motorizált egyéni közlekedéssel.

Tehát úgy kell kialakítani az infrastruktúrát, hogy a közösségi közlekedés attraktivitása nagyobb legyen, mint az autós közlekedésé. Mindenki szeretné, hogy a buszmegálló a lehető legközelebb legyen hozzá, de azért ne éppen a háza előtt mert zajos. Ezt persze orvosolni lehet új közösségi közlekedési járművek beszerzésével, azonban ez nagyon drága megoldás, mert ugye az egész várost kiszolgáló járműflottát le kellene cserélni. Tehát egy élhető város kialakítása ellentmondásokkal van tele, és ezek között kell egy optimumot találni.

Egyik ilyen megoldás lehet az egyes városrészeken a közösségi közlekedés fejlesztése, illetve előnyben részesítése a motorizált egyéni közlekedéssel szemben. Ismert tény, hogy a külvárosi részeken a közösségi közlekedés színvonala alacsonyabb,

ezért egy kézenfekvő megoldás lehet az egyéni közlekedés és közösségi közlekedés kombinálása.

Ha az információtechnológiai eszközöket aktívan használjuk, akkor az egyéni közlekedőket is támogathatjuk vele. Nem azért adunk támogatást az egyéni közlekedőknek, hogy még többen válasszák a személyautós közlekedést, hanem azért, hogy különböző navigációs, valós idejű forgalmi terhelés térképet szolgáltatva, csökkentsük a dugókat, ezáltal a káros anyag kibocsájtást. Biztosítjuk a meglévő úthálózat megfelelő, nem túlterhelt, és megfelelően elosztott kapacitás kihasználtságát. Ugyanakkor fontos korlátozó tényezőket figyelembe kell venni, például a forgalomcsillapított területekre ne tereljünk extra forgalmakat a dugók elkerülése miatt. Ha az IT eszközeit említjük, itt olyan megoldásokra kell gondolni, mint a dinamikus forgalomirányítási rendszer, amikor a városi forgalomnagyságokat folyamatosan mérjük, és a mért eredményeknek megfelelően alakítjuk ki a jelzőlámparendszer programjait. Ezáltal elérhető, hogy a közösségi közlekedési járműveket bevonjuk a jelzőlámpás csomópontok hangolásába, esetleg zöld előnyitással, vagy zöldidő hosszabbítással előnyt biztosítva számukra. Hasonló jó megoldás lehet a ramp metering, vagy a parkolás menedzsment.

A járművek kihasználtsága is igen alacsony, amelyet javítani lehetne car sharing, és car pooling rendszerekkel, azonban ezek kialakítása nem része a dolgozatnak. Ezek a kifejezések az autó megosztását jelentik, ezzel növelve az autók kihasználtságát. Belátható, hogy a járművek nagyobb kihasználtságával jelentősen javítani lehetne, a városi közlekedés minőségén.

Az előbb felsorolt problémák kezelésére, nem csak a városlakókat kell folyamatosan képezni, fejleszteni, hanem magát a várost, mint területi egységet kell okosítani. Napjainkban jelentősen megnőtt a társadalmi, gazdasági, és politikai igény az okos városokra. Nem okos városokat építünk a régiek helyett, hanem a meglévő régi, jelenlegi igényeket kielégíteni nem képes városokat kell okosítani, hogy megfeleltethessük saját elvárásainknak.

Szakdolgozatomban a parkolás problémáit vizsgálom, így az okos városok koncepcióból az okos közlekedést emelem ki. A felsorolt szempontok között az egyik legfontosabb az intelligens közlekedés, melynek tipikus elemei a következőek: Pálya, Jármű, Irányítás, Szolgáltatás. [3]

Ezek nagyban javíthatják a városok közlekedési, és életviteli nehézségét, egy rendkívül fontos témát, a járművek parkolását kiemelten érintik. Dolgozatom kereteiben a parkolásra helyezem a fókuszt. Az okos városok tudományágának fejlődésével észrevették, hogy a parkolás olyan terület, amelyben nagy potenciál van. Okos közlekedésirányítás és menedzsment esetén alkalmazhatunk okos parkolási rendszereket, és dinamikus parkolóhely kiosztást, és parkolóhely menedzsmentet, illetve valós idejű szabadhely adatokat tehetünk közzé.

Smart parking rendszer:

Már kisebb forgalom esetén is problémát jelenthet a parkolóhelyek kezelése, kiosztása, biztosítása. A városi logisztika, a vegyes összetételű forgalom (kerékpárok, elektromos és hagyományos személyautók, stb. forgalma) eltérő és változó igényeket támaszt. A parkolóhelyek jelzésére, biztosítására és a foglaltság felmérésére széles körben lehet fejlesztéseket végezni az igényeknek megfelelően.

Dinamikus parkolóhely kiosztás és menedzsment:

Telefonos applikáció segítségével például lehetővé válhat a parkolóhely lefoglalása, illetve oda vezető navigációs útvonal kidolgozása, távozás után pedig a pontos parkolási időnek megfelelő fizetés. Kiemelten kezelheti a jelenleg alacsony hatékonysággal kihasznált helyeket, mint az áruszállítás, illetve a mozgássérült parkolóhelyeit.” [3]

A Smart Parking egy járműparkolási rendszer, amely segít a járművezetőnek megtalálni egy üres helyet. Minden parkolóhelyen szenzorokat használnak, amely érzékeli egy jármű ottlétét vagy hiányát, és jelzik a közeledő járművezetőnek a szabad helyek elérhetőségét. Például a 2000-res évek elején egy okos parkolási rendszer lett telepítve a Baltimore-Washington International Airport parkolójába. Az üzembe helyezés előtt a parkolóház bezárt, ha 90%-ig megtelt. Az okos parkolási rendszerrel, a garázs csak akkor zár be, ha 99%-ot elérte a telítettség. [6]

Az előbbi példa alapján kimondható, hogy a parkolás problémája egészen jól kezelhető az okos városokban, ha kihasználjuk az okos eszközök adta lehetőségeket.

1.1. A motorizált egyéni közlekedés volumenének növekedése

A motorizált közlekedési eszköz a személyautókat, és a motorkerékpárokat, tehergépjárműveket foglalja magába. A motorkerékpáros közlekedés vizsgálatától

eltekintünk, hiszen nem képezi jelentős részét a városban lezajló forgalomnak, másrészt a motorkerékpárok parkolására jelentősen eltérő szabályok vonatkoznak, emiatt a parkolásuk sem okoz jelenleg a hazai körülmények között észlelhető problémát. A városi teherforgalom erősen korlátozva van, legalábbis speciális és drága engedélyekhez kötik a behajtásukat, azonban az ezek várakozása is valós gond. Tipikus ilyen elem az áruszállítás nehézségei, melyet a City logisztika igyekszik kezelni.

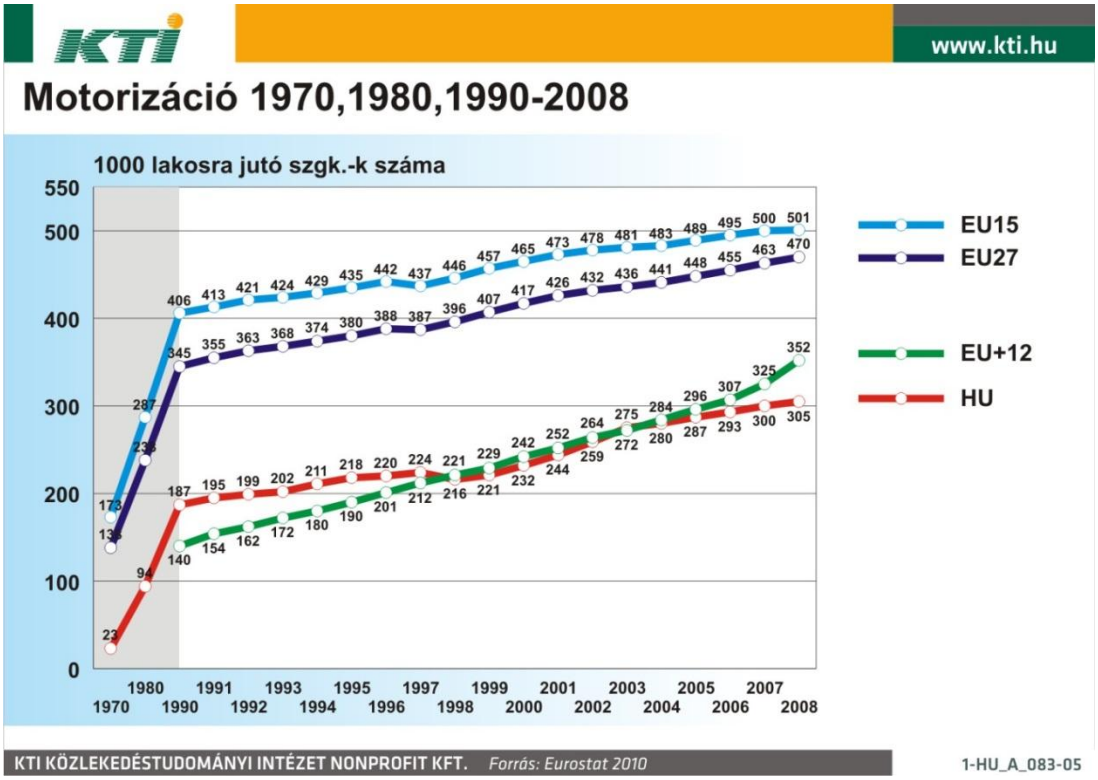
„A city logisztika magába foglalja a következőket:

Áruterítés – Áruféleségenként vagy fogyasztóként. Ezt támogatja a logisztikai centrumok, ipari parkok gyűjtő raktározása, árukezelése.

Útvonal optimalizálás - Minden város digitalizált térképe lehetővé teszi, hogy a városi közlekedés szabályainak megfelelően a legoptimálisabb útvonalon jusson el az áru az adott helyre.”[7]

Ezen indokok miatt a parkolási probléma vizsgálatába csak a személygépjárműveket vonjuk be.

Továbbiakban vizsgáljuk meg a járműállomány nagyságát és változási tendenciáját. A hazai személygépkocsi állomány fejlődése a nemzetközi trendet követi valamennyi lemaradással. Gyakorlatilag ezt azt jelenti, hogy a motorizációs fok egyre nő. Ez a trend az emberi igények folyamatos, és mára már természetes velejárója. A növekedés legfőbb okai a GDP növekedése, a fogyasztói társadalmi szemlélet elterjedése és az autó tulajdonlása, mint státusszimbólum. A motorizációs fok folyamatos emelkedését mutatja meg a 2. ábra.



2. ábra: A motorizációs fok alakulása *[8]*

Vegyük számításba azt is, hogy az EU átlag alatt vagyunk, és persze a gazdaság növekedése intenzívebb motorizációs fejlődést jelent. Erre az intenzív fejlődésre a közúti infrastruktúra nem áll készen, és az előző fejezetekben elemzett okok miatt, szükséges az ennek megfelelő infrastruktúrabővítés. Infrastruktúra alatt most a parkolási létesítményeket és a városi közúthálózat kapacitását kell érteni. Az infrastruktúrabővítés önmagában még nem elegendő, emiatt szükséges a parkolás menedzsmentet bevezetni.

Szintén jelentős elem a gépjárművek kapacitás kihasználtsága. Különböző mérések alapján az átlagos járműkihasználtság 1,55 fő/jármű. Ez a szám rendkívül alacsony, melyen javítani szükséges. [9]

Ezek alapján prognosztizálható, hogy több autó esetén több parkolóhely lesz szükséges. A társadalom tagjai hamarabb engedhetik meg maguknak egy autó vásárlását, mint ahogy a városvezetés tudna reagálni a változó igényekre és végre tudná hajtani a korábbi motorizációs fokra tervezett városrész komplett átépítését. Tehát valami olcsóbb és gyorsabb megoldást kell találni, hogy a kialakulóban lévő új helyzetre időben felkészüljünk. Az egyik lehetséges megoldás a parkolás menedzsment.

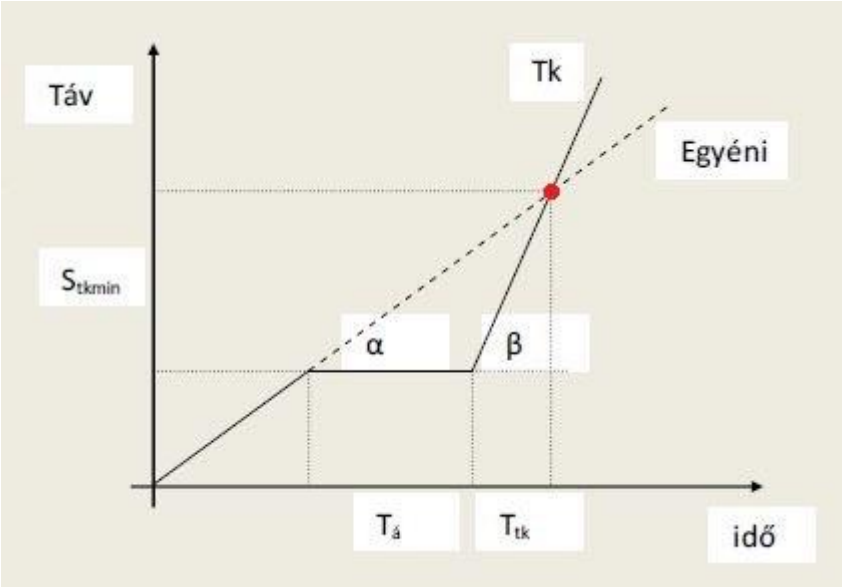
Összességében az egyéni közlekedés informatikai támogatásával a rendszerben lévő nem megfelelőségeket, kapacitáshiányokat, és felesleges mozgásokat kívánjuk eliminálni. Ha megvizsgáljuk és elemezzük a parkoló forgalmat, akkor jelentős következtetéseket vonhatunk le. Ezt tipikusan az egyéni közlekedés olyan velejárója, amelyet még a mai napig sem tudunk optimálisan kezelni, szoftveresen támogatni.

1.2. P+R parkolók

3. ábra: A P+R parkoló KRESZ tábla *[10]*

A Park and Ride fogalom azt jelenti, hogy a gépjárművet a parkolóban hagyjuk, és helyette közforgalmú közösségi közlekedéssel vagy gyalogosan utazunk tovább. Tulajdonképpen ezzel a megoldással jelentősen lehetne csökkenteni a városban megjelenő autós forgalmat, azonban nehézségei is vannak. Nehéz rávenni az autósokat, hogy a P+R mellett döntsenek. Ehhez az kell, hogy a gépjárműveket egy olyan parkolóban lehessen letenni, ahol általában ingyenesen vagy minimális díj ellenében további szolgáltatásokat nyújt. Fedett, őrzött, esetleg egyéb kényelmi szolgáltatások is lehetnek, például ingyenes Wifi használat. További kérdés a szolgáltatás ára. Mindenképpen rendkívül alacsony árakat kell biztosítani, hiszen csak így lehet vonzó. Általában bevett gyakorlat, hogy körülbelül egy vonaljegy árának megfelelő összeget határoznak meg. Azonban lehet előnyt is biztosítani, a bérlettel rendelkező utasok számára ingyenessé is tehető.

Ezen parkolók közvetlen környezetében célszerű, ha van lehetőség közösségi közlekedési eszközre átszállni. Az eljutási idő, ami az utasok nagy részét motiválja, ezért célszerű úgy építeni a parkolót, hogy az eszközváltási idő minimális legyen. Adott esetben, ha parkolóházról beszélünk, lehet mozgójárdával gyorsítani az átgyalogolást, parkolás irányító rendszerrel csökkenteni a parkoló keresési időt, esetleg lifteket alkalmazni, amelyek közvetlenül a metrómegállóba szállítják le az embereket. Az alábbi ábrán látható az egyéni közlekedés és a P+R használata közötti összefüggés.



4. ábra: az egyéni és P+R közlekedés időbeli viszonya *[1]*

A szaggatott vonal az egyéni, a folyamatos a P+R eljutási módot ábrázolja. Látható, hogy a $T_á$-val jelölt átszállási idő hossza lényeges elem. A T_{tk}-val jelölt időelem az az időtartam, amelyet a tömegközlekedési eszközön töltünk az ekvivalencia pontig. Látható, hogy a közösségi közlekedés egyenese meredekebb, tehát gyorsabb eljutást biztosít. Az egyéni, és a közösségi közlekedés metszéspontja, az úgynevezett ekvivalencia pont, vagy ekvivalencia távolság (piros pont). Ez azt a távolságot jelenti, amelynél nagyobb távolság megtétele esetén, már rövidebb időt vesz igénybe az utazás, azaz megéri használni a P+R parkolót. [1]

Ha parkolási támogatással foglalkozunk, akkor az átszállási időt tudjuk csökkenteni. Azt kell elérni, hogy ez minimális legyen. Ezt pedig parkolóházon kívüli okos támogatási elemekkel, illetve parkolóházon belüli elemekkel lehet segíteni. A külső elemek, leggyakrabban a bejáratra szerelt LED kijelzők, amelyek megjelenítik a szabad helyek számát. Egy intelligensebb megoldás, ha azt is jelzik, hogy szintenként hány szabad hely van. Az utazóknak nagy segítséget nyújt az eszközválasztásnál egy multimodális utazástervező, amely összehasonlítja az eljutási időket, akár figyelembe véve a forgalmi viszonyokat és a parkolóház kihasználtságát.

Fejlesztési lehetőséget rejt magában az utastájékoztatási rendszer (Budapest esetében a FUTÁR) összekapcsolása a P+R parkolókkal, ugyanis az utazók látnák, hogy mennyi időmegtakarítást jelenthet számukra ennek az igénybevétele.

Budapest közigazgatási területén belül a közlekedési eszközváltás biztosítására nagyobbrészt a városi gyorsvasúti és a nagyvasúti hálózatokhoz kapcsolódó P+R parkoló hálózat üzemel.

„Budapesten 2008-ban 596 481 darab személygépkocsi volt. Ez a szám a 2009-es gazdasági válságot követően 2012-re 5 százalékkal, 565 563-ra csökkent, de azután újra emelkedni kezdett, és 2015-ben ezerrel már meg is haladta a 2008-as számot.

Az agglomerációban, kisebb visszaesés után lényegében folyamatosan, 8 százalékkal emelkedett 2008-tól 2015-ig 460 435-re. 2015-ben Budapesten és agglomerációjában tehát 1 061 772 személygépkocsi volt (KSH adatok), ami azt jelenti, hogy a 2012-es 987 670 darabról három év alatt 75 ezerrel nőtt az autók száma. A fővárosi közlekedési fejlesztéseket tartalmazó Balázs Mór-terv szerint a válság előtt naponta 275 000 személygépkocsi ingázott Budapest és az agglomeráció között. Ez a szám a válság miatt 240-250 ezerre csökkent egy időre, de ma már meghaladja a válság előttit.

P+R parkoló az elmúlt 4-5 évben 1400 épült. A BKK szerint jelenleg a fővárosban mindent összevetve 5200 P+R parkolóhely áll rendelkezésre. Ebben benne vannak a kerületek és a bevásárlóközpontok által üzemeltetett P+R parkolók is. Tehát még egyszer a két szám: 275 ezer darab ingázó személygépkocsi vs. 5200 darab P+R parkolóhely” [11]

A városhatárt budapesti tartózkodási céllal naponta átlépő forgalom mintegy 275.000 szgk/nap értékre becsülhető, melynek az 5200 férőhelyes P+R rendszerű fővárosi parkoló kapacitás csak 1-2%-át tudja fogadni. látható, hogy a P+R parkoló hálózatot és a parkolás menedzsmentet fejleszteni szükséges.

1.3. Működési hiányosságok

Az autósok többnyire nem tudják, hogy hol vannak P+R parkolók, a városban, a navigációs eszközök nem terveznek kombinált autós és közösségi közlekedéssel. Másrészt az emberek nagy része még mindig nem ismeri a P+R rendszert. Ha valaki ismeri ez a több alágazatot használó eljutási lehetőséget, akkor sincs semmilyen támpont, hogy a parkolóban van-e szabad hely. Budapesten a BKK jelenleg néhány tucat parkolóhelyet üzemeltet. Ezen parkoló létesítmények adataihoz valós időben nehéz hozzáférni. Sem a BKK online felületén a http://www.bkk.hu/parkolas/

weboldalon, sem pedig egy navigációs alkalmazásban, de még csak egy LED kijelzőn sem, amely a parkoló bejáratánál van. A honlapon mindössze tájékoztató jellegű információk vannak.

Egy szemléletes példával élve a Budaörsi úton elhelyeztek egy LED táblát, amelyik a P+R parkolót javasolja a közlekedőknek. A digitális tábla adta lehetőségeket jobban ki kell használni, mert lényegesen több információ közölhető rajta a közlekedők felé.

5. ábra: LED kijelző a Budaörsi úton

A szolgáltatás meglehetősen kevés információt közöl, hiányzik az, hogy milyen feltételekkel használható a parkoló, hol van, és az sem derül ki, hogy, van-e szabad hely a parkolóban. Arra vonatkozóan pedig semmi információ nincs, sehol, hogy ezzel mennyi időt spórolhat az utas.

Részben jó példa lehet erre a 6-os főút bevezető szakaszán elhelyezett változtatható jelzésképű tábla, amelyiken megjelenítik, hogy a hidakat melyik úton keresztül hány perc alatt érhetjük el. Azonban erről hiányzik az összehasonlítás a közösségi közlekedési móddal. Meg lehetne mutatni, hogy ha a Savoya parkban leteszi az autóját, akkor tömegközlekedéssel mennyi idő alatt ér be a városközpontba.

6. ábra: VJT a 6-os út bevezető szakaszán

Ha az utas úgy dönt, hogy kipróbálja a P+R parkolást, akkor valahogy eljut a parkolóhoz, és csak ott derül ki számára, hogy van-e szabad hely. Ez egy LED kijelzőn van közölve az utassal.

Ez látható az alábbi képen, az Etele téri P+R parkoló. A kép reggel 9 körül készült, látható hogy nem sok szabad hely maradt. Ezeket a helyeket, amíg megtaláljuk a parkolóban több felesleges kört teszünk meg. A parkoló létesítményeket érdemes lenne olyan műszaki tartalommal felszerelni, amely segít az utazónak a parkolón belül megtalálni a szabad helyeket.

7. ábra: Az Etele téri P+R parkoló bejáratánál elhelyezett LED kijelző

Azonban nem csak a P+R parkolót kell vizsgálni, hanem a városban lévő parkolóházakat. Ezek döntő többségéről is elmondható, hogy a weblapjukon, alkalmazásukon nem lehet megtalálni valós időben a kihasználtság adatokat. A parkolóhelyet keresőket segíteni hivatott a www.ezpark.hu weblap. Ha alaposabban megvizsgáljuk a következő szolgáltatásokat nyújtja:

- online helyfoglalás
- online napijegyvásárlás
- online bérletigénylés
- 24 órás diszpécser szolgálat
- autóbérlés
- autószerelő műhely
- kamerarendszer
- gumis műhely
- kávézó és bár
- WiFi elérhetőség

Ha a szolgáltatásokat megvizsgáljuk, akkor látjuk, hogy széles spektrumon mozog. Ezek a parkoláson kívüli többletszolgáltatások mind szükséges alapelemek a mai társadalomban. Bár itt is látható, hogy gyakorlatilag mégsem lehet online követni a szabad parkolóhelyek számát.

1.4. Célkitűzések

A dolgozatban igyekszem választ találni a parkolási problémák eredetére, számba veszem a jelenlegi eszközöket, rendszereket, módszereket, szoftvereket, amelyekkel a probléma kezelhető, és iránymutatást adok a jövőre nézve, hogy ezt a trendet hogyan lehet megváltoztatni, vagy egyáltalán lehetséges-e.

További célom, hogy az informatika adta lehetőségeket felhasználva lefektessem egy rendszer elvi alapjait, megvizsgáljam megvalósítási lehetőségeit és korlátait. A rendszer alkalmas arra, hogy valós időben kezelje a parkolóhelyek adatait. Ezáltal valós idejű szabadhely adatokat tartalmazó adatbázismodell megvalósíthatósági lehetőség elemzését végzem el, így támogatást nyújtva az egyéni közlekedőknek.

Az adatbázisban tárolt adatok megjelenítéshez, egy mobil alkalmazás elvi alapjait is lefektetem, és megvizsgálom, hogy a szoftvertervezési folyamat hogyan néz ki erre az alkalmazásra vonatkoztatva. Definiálom az alkalmazás platformját, front-end, back-end oldalait, és az ezekhez tartozó megfelelő rétegeket, és működési struktúrát.

Az alkalmazás a járművezető számára a tudatos útvonaltervezés lehetőségét teremti meg oly módon, hogy forgalmi torlódásokat elkerüli, a parkolóhely keresést megszűntetve, egy a felkeresendő cím közvetlen közelében lévő biztosan szabad parkolóhelyre navigálja.

Megvizsgálom az általam tervezett rendszer fejlesztési lehetőségeit, és további felhasználási területeit az utazók napi tevékenységi láncának optimalizálásában.

Az alapja egy már létező algoritmus, amely a közlekedőknek lehetőséget ad arra, hogy optimálisan szervezze meg a napját, mind a felkeresendő helyeket, mind a közlekedési eszközt optimálisan válassza ki. Ezt a rendszert kiegészítem, a parkolási adattárház nyújtotta lehetőségekkel, így a napi tevékenységek optimalizálásában a parkolás is tervezhetővé válik.

2. Helyzetértékelés, Körkép hazai és világviszonylatban

2.1. Jelenlegi piaci helyzet

Napjainkban a parkolás város, ország, és világszerte általános probléma. Budapesten, kimondottan jó a tömegközlekedés és az egyéni közlekedők aránya, azonban más országokban többen használják az autós közlekedést. Ez folyamatosan környezetszennyezést, forgalmi dugókat, nehéz városi helyzeteket teremt. Próbálják a közlekedőket rábírni arra, hogy tegyék le autóikat, és válasszanak más közlekedési módot. Ennek eszközei a tiltott parkolási zónák, a fizetős zónák, a P+R előnyben részesítése.

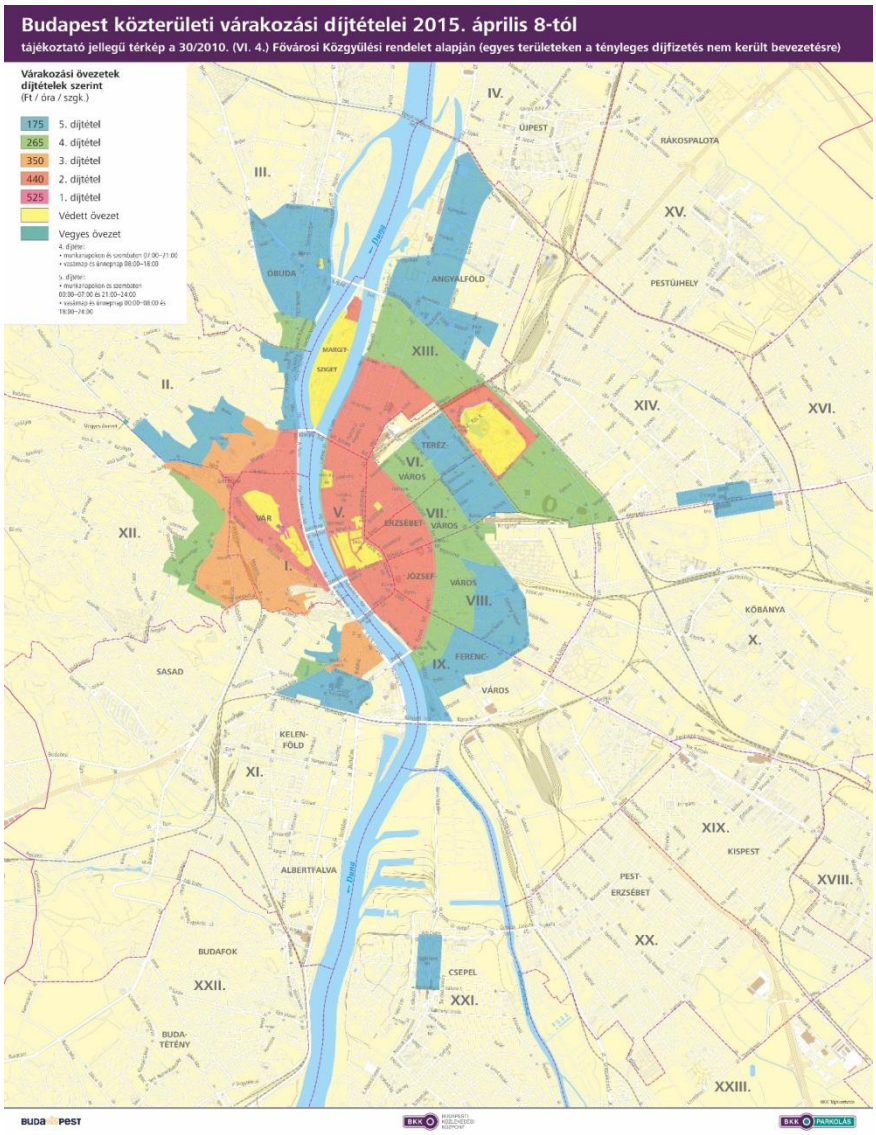
Budapesten a P+R parkolók folyamatosan épülnek, azonban gyakran alacsony szolgáltatási szintűek ezek a létesítmények. A cél az lenne, hogy a parkoláson kívül további egyéb szolgáltatások legyenek elérhetőek, hiszen így elősegíthető a közlekedési eszközváltás.

További probléma a P+R parkolók árai. Nem feltétlenül drága, bár az ingyenes parkolási lehetőség nagyobb vonzerőt jelent az autósoknak. Itt fizetünk azért, hogy otthagyhassuk az autónkat. Máshol kifizetünk egy vonaljegy árát, és a vonaljegyet is megkapjuk, hogy a közösségi közlekedéssel bejussunk a belvárosba. Az alábbi képen látható a Kőbánya Kispesti P+R parkoló reklámanyaga.

8. ábra: A KÖKI terminál P+R parkolójának reklámtáblája [12]

A KÖKI Terminál épületében a bevásárlóközpont parkolója mellett egy 330 férőhelyes P+R parkoló is működik. Azoknak érdemes ezt választani, akik egész napra szeretnék itt letenni autójukat, hogy metróval folytassák útjukat a belváros felé. Egész napra (6-22 óra között) 350 forintot kell fizetni. Ez kevesebb, mint a belvárosban egyetlen óra parkolás. [12]

A parkolók kapacitáshiányán túlmenően egyéb problémákat is észlelhetünk. Ezt próbálják ellensúlyozni a parkolási fizetőövezetek alkalmazásával. Az intézkedés lényege, hogy a városrészeket frekventáltság, turisztikai vonzás, és egyéb szempontok alapján zónákra osztják, és zónánként más-más parkolási árat szabnak meg, egyes helyeken még a várakozási időt is korlátozzák. A rendkívül kiemelt helyeket, mint a Margit sziget, Budai vár, úgynevezett védett övezetet alakítanak ki, ahol csak a lakosok parkolhatnak, illetve az egyéb behajtás külön engedélyhez kötött. Az övezetek láthatóak a 9. ábrán.



9. ábra: A budapesti várakozási övezetek [13]

További intézkedési lehetőség a dugódíj bevezetése, azonban ennek a feltételei még nem adottak. A sok autó problémáját ez sem oldja meg a belvárosban, átmenetileg lehet, hogy csökkenti az autók számát. De a problémát csak térben áthelyezi, mert azok az autók, amelyek a belvárosba nem mennek be, le kell parkolni a fizetési határ mentén. Tehát, a zónahatár mentén szükséges további parkoló létesítmények kialakítása. Ha ezek a beruházások elkészülnek, akkor a megnövelt kapacitású parkoló hálózat, és a dugódíj együttesen már képes orvosolni a belváros parkolási, közlekedési problémait. Azonban erre a város nincs felkészülve, nincsenek parkolóházak, nincsenek szabadtéri parkolók, de még szabad területek sincsenek, ahová parkolóhelyeket lehet létesíteni.

Ha ilyen súlyos a belvárosban a parkolás kérdése, joggal vetődik fel a kérdés, hogy miért nem építenek parkolóházakat, mélygarázsokat a városban. Ezek a beruházások

rendkívül költségigényesek, és a megtérülési idejük igen nagy, emiatt a magánszektor nem túl aktív. Egy másik lényeges szempont, ha valahová olyan létesítmény épül, amelyben a parkolás lehetséges, ez egyfajta parkolóforgalmi igényt gerjeszt. Tehát a parkolóhelyek számának a növekedése a probléma megoldását elősegíti, ugyanakkor fokozza a parkolóforgalmi igényt. Szakmai álláspontom szerint további parkoló létesítmények építése a belvárosban szükséges, illetve a jelenleg meglévőket kellene még inkább tudatosítani az emberekben, és ezeket optimálisan kihasználni.

Alább olvasható, hogy manapság már a jogalkotó is kiemelt figyelmet fordít arra, hogy jogilag is egzaktul megfogalmazott elvárások legyenek, a parkolás problémájának a kezelésére az új beruházások esetén. Az alábbi jogi elvárások rendelkeznek arról, hogyan és milyen feltételek mellett kell kialakítani parkolóhelyet, illetve hány darab szükséges.

„42. § (1)145 Az új építmények, önálló rendeltetési egységek, területek rendeltetésszerű használatához… - legalább a (2) és a (4) bekezdésben előírt mennyiségű és fajtájú gépjármű elhelyezési lehetőségét, továbbá rendszeres teherszállítás esetén rakodóhelyet kell biztosítani. Meglévő építmények bővítése, átalakítása, rendeltetésük módosítása esetében… - csak a bővítésből, az átalakításból, vagy az új rendeltetésből eredő többlet gépjármű elhelyezéséről kell gondoskodni, a meglévők megtartása mellett. Védett épületek (műemlék, helyi egyedi védelem) bővítéssel nem járó átalakítása, rendeltetésmódosítása esetében…- nem kell a gépjárművek elhelyezését biztosítani.” [14]

„(3) A (2) bekezdés szerint számított minden megkezdett 50 db várakozóhelyből legalább egyet a mozgásukban korlátozottak részére kell kialakítani, amelyekből legfeljebb négy helyezhető közvetlenül egymás mellé.” [14]

„(7)151 A 10 gépjárműnél nagyobb befogadóképességű felszíni várakozó- (parkoló) helyet fásítani kell. A parkoló felületek árnyékolását biztosító fásítást − helyi építési szabályzat eltérő rendelkezésének hiányában − minden megkezdett 6 db várakozó-(parkoló) hely után 1 db, nagy lombkoronát nevelő, környezettűrő, túlkoros, allergén pollent nem termelő lombos fa telepítésével kell megoldani, minimum 1 m2 szabad földterület biztosításával, amely 1 m2 alatti területei a telek zöldfelületébe nem számíthatók be.” [14]

„(16)159 Az ellenérték fejében várakozó- (parkoló) hely értékesítésére szolgáló építmények létesítése esetén a várakozó- (parkoló) helyeket úgy kell kialakítani, hogy 100 várakozó- (parkoló) hely után legalább 10 várakozó- (parkoló) hely vonatkozásában, elektromos gépjármű töltőállomás kiépíthető legyen a burkolat megbontása nélkül.” [14]

„(17)160 Az ellenérték fejében várakozó- (parkoló) hely értékesítését szolgáló, meglévő építmények esetén minden megkezdett 100 várakozó- (parkoló) helyből 2017. január 1-jéig legalább egyet, 2019. január 1-jéig legalább kettőt elektromos gépjármű töltőállomással kell ellátni.” [14]

Rendkívül előremutató, hogy az elektromos autók töltésére is gondolnak, hiszen a gépjárművek fejlődési irányvonala abszolút az elektromos hajtás felé tendál. Az átmeneti időszakban, amíg a fosszilis tüzelőanyaggal hajtott járművekről áttérünk a tisztán elektromos hajtásra, pedig egy jó marketingeszköz. A parkolóházakat, és okos parkolókat lehet reklámozni az elektromos autókkal, mondván ingyenes töltési lehetőség, közelebb parkolunk a bejárathoz, és mindig van a számára fenntartott külön hely.

2.2. Parkolást segítő alkalmazások

Ebben a fejezetben számba vettem néhány olyan alkalmazást, rendszert, ötletet, amelyek a parkolást megkönnyítik, vagy bármilyen formában elősegítik a parkolóforgalom optimális elosztását. Alapprobléma, hogy valós időben online eszközön vagy felületen a szabad helyek adatait nem lehet elérni. Ettől függetlenül léteznek olyan rendszerek, amelyek mégis hasznosak a parkolóhelyet keresők számára.

A bejáratnál vagy annak közvetlen közelében jelenítik meg egy kijelző segítségével a szabad helyek számát. Ehhez az információhoz viszont túl későn jutunk hozzá. Egy tipikusan rossz példa az Újbudai Allee bevásárlóközpontban elhelyezett LED kijelző. Az autósok lehajtanak a meredek rámpán, egészen a sorompóig, mire el tudják olvasni, így visszatolatni nem lehetséges. Emiatt bemennek a telített parkolóba. Ha a bejárattól néhány 10 méterrel korábban lenne elhelyezve, akkor még az autós elkerülhetné a behajtást.

Az egyedi foglaltság jelzők, a parkolóházon belül segítik a közlekedőket a szabad parkolóhelyek megtalálásában. Ezek az eszközök vizuálisan hívják fel a figyelmünket a

szabad helyekre. Lényegük, hogy a parkolóhelyeken egy érzékelő van elhelyezve, amely érzékeli, hogy van e ott jármű vagy nincs. A járművek érzékelése rendszerint rádióhullámokkal történik, és a visszaverődött jelekből állapítja meg a foglaltságot. Foglaltság esetén a plafonon elhelyezett piros LED világít, üres hely esetén pedig zöld. Az autósok a közlekedőfolyosón végighaladva a zöld LED fény alapján tájékozódnak. Ez a megoldás már nagyban lerövidíti a helyet keresők megtett távolságait. Így már nem kell minden egyes folyosón végighaladni, hanem a közlekedésre kijelölt főfolyosón elegendő csak, mert onnan azonnal látható a szabad hely. Egyébként ezek a berendezések sem üzemelnek 100%-os megbízhatósággal. Alkalmanként eltévesztik el a foglaltsági állapotot. Azonban nagy magabiztossággal üzemelnek, és ez a minimális hiba, amelyik abból ered, hogy a kisebb autókkal szabálytalanul parkolnak le az adott területen, elhanyagolhatóan kicsi.

Alább a szintek blokkokra vannak osztva, és az egyes blokkok pedig a jobb alsó képen látható QR kóddal vannak megjelölve. Ezeket az emberek okos eszközeikkel leolvasva, elmenthetik telefonjukba azt, hogy hol hagyták járművüket. Ez ugyan nem a parkolást segíti elő, mégis egy említésre méltó, hasznos segítség a közlekedőknek.

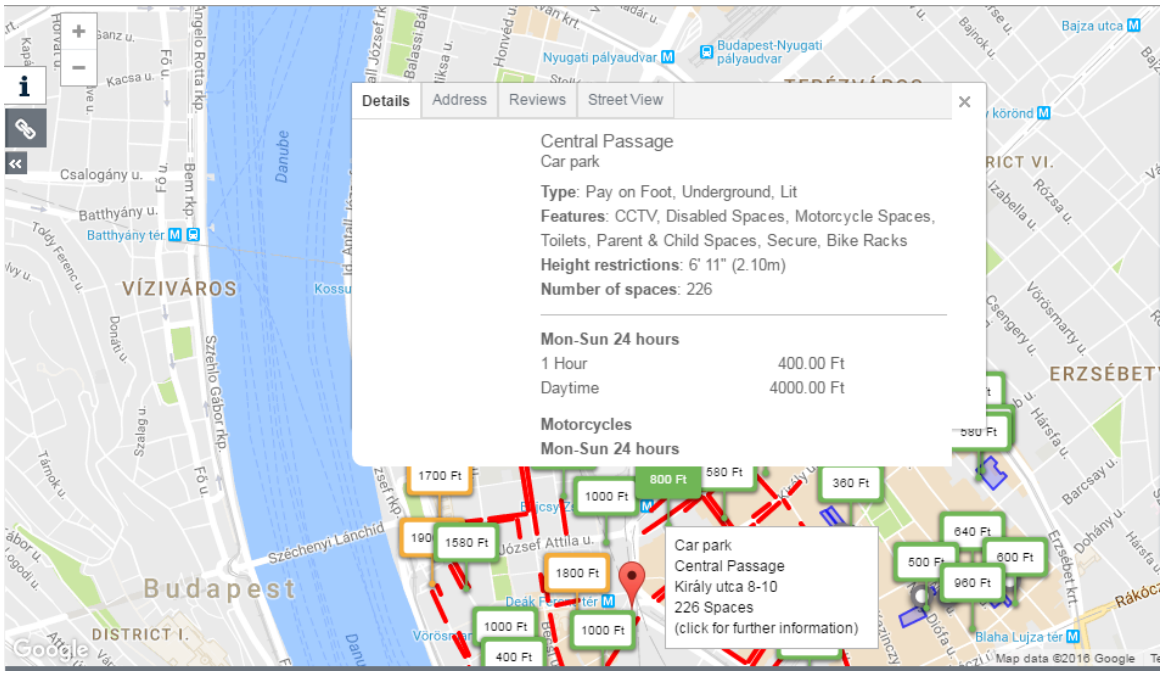
10. ábra: Az ALLEE mélygarázsának pozitív és negatív példái

2.3. Jelenlegi legnagyobb adatkezelő, adatszolgáltató

„A Parkopedia név a Parking és Wikipedia nevekből adódott. A céljuk az, hogy világszerte segítsenek megoldani, vagy megkönnyíteni a parkolás problémáját. Szeretnének nekünk segíteni, megtalálni a legolcsóbb és a legmegfelelőbb parkolóhelyet. A vállalkozás rengeteg parkolóhelyet tart nyilván, kezeli az adataikat. Teljes befogadóképesség, árak weboldal címe, térképi elhelyezkedése. Támogatja az online helyfoglalást, ott ahol maga a parkolóház is támogatja ezt. Sajnos ez az oldal sem kezel kihasználtság adatokat.

A Parkopedia több mint 40 millió parkolóhelyet tart nyilván világszerte, mely együttműködik az Apple-lel, így az online szolgáltatásban megtalálható 75 országban (Észak- és Dél-Amerikában, Ázsiában és Európában, köztük Magyarországon) nyilvántartott több mint 40 millió parkolóhely az Apple térképein is feltűnik.

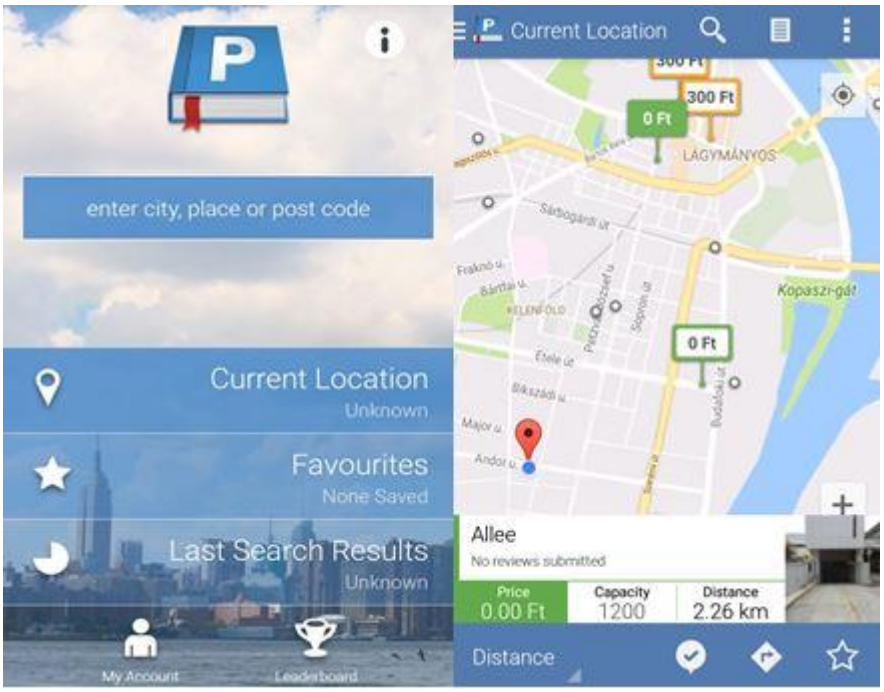
Eddig is tartalmazott parkolóházakat az Apple Maps, ezekről azonban közel sem érhettünk el annyi információt, mint amennyit a Parkopedia tartalmaz, például a nyitva tartást, az árszabást, a fizetési módokat, magassági limiteket vagy egyes esetekben akár valós idejű telítettséget. Az Apple Maps-ből a Parkopedia linkre kattintva átjuthatunk a szolgáltatás weboldalára vagy az alkalmazásba, ahol mindezeket az adatokat elérhetjük, sőt a támogatott helyszíneken még parkolóhelyet is foglalhatunk.” [15]



11. ábra: A Parkopedia internetes felülete

Ez eddigi kutatásaim során a legfejlettebb oldal és alkalmazás, amelyet találtam. Egy jól működő kereső funkció, nagy mennyiségű háttéradat, és felhasználóbarát user-interface. Ez a platform alkalmas lehet továbbfejlesztési céllal a valós idejű szabadhely adatok kezelésére. A webes felület az utazástervezési, és tájékozódási fázisban használható, azonban a közlekedők inkább az applikációt használják.

Az alkalmazás könnyen és jól kezelhető. Pont annyira, hogy autózás közben biztonságosan lehessen kezelni, mint egy navigációs programot. A parkolóhely keresés két módon indítható, a térképes keresés, vagy a keresősávba beírt címmel kereshető. A térképes felületen a jelölőkben az árak is megjelennek. Az alábbi két kép mutatja az app használatát.



12. ábra: A Parkopedia kereső, és térképes felülete

Ezek alapján egyszerűen, és könnyen található olyan parkolóház, amely a beállított gyaloglási távolságnak is megfelelő, árban elfogadható, és biztosítja a számunkra szükséges parkoláson túli egyéb szolgáltatásokat. Kiválasztva a személyes szempontokat a program meghívja a Google Maps navigációt. Ez azért előnyös, mert a navigációs programmal a mindenkori forgalmi viszonyoknak megfelelően is tervezhetjük az utunkat.

Az adatbázis felépítése lényeges kérdés. Napjainkban mindenki a maximális adatbiztonságra törekszik, és nem szívesen ad meg semmilyen adatot külső partnereknek. Azonban egy ilyen méretű adatbázist kézzel gyűjtött adatokkal nem lehet előállítani. A Parkopedia egy ingyenes megjelenési és felületet biztosít a parkolás üzemeltetőknek. Tehát minden egyes parkolóház saját maga tölti fel az adatait egy online regisztrációs felületre.

2.4. Okos parkolás Szentendrén

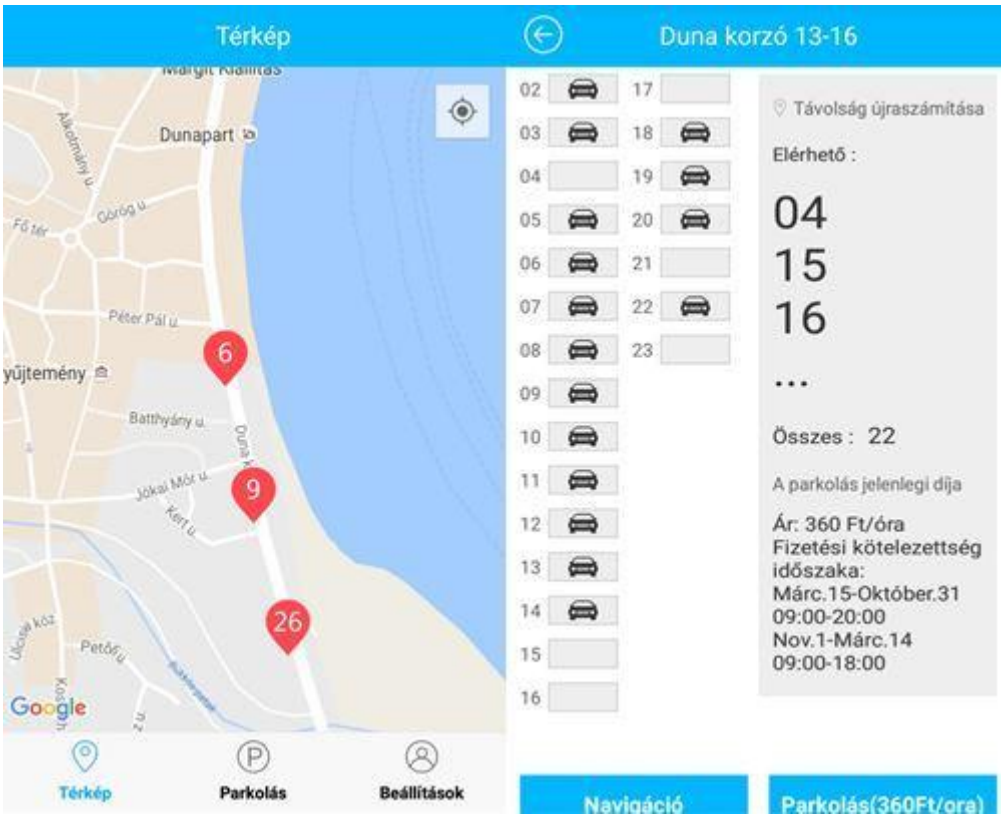
„Magyarországon egyedülálló a ZTE Hungary Kft. Szentendrén bemutatott mintaprojektje, az okos parkolási zóna, melynek lényege, hogy a parkolót keresőket mobiltelefonos applikáció vezeti a szabad helyekhez. A parkolás irányítási rendszer kiépítésével az egyes parkolóhelyek foglaltságáról szóló adatok már a behajtási ponton

elhelyezett kijelzőn tájékoztatják a járművezetőket a szabad helyek számáról, megkímélve őket a felesleges utaktól, csökkentve a zajterhelést és a légszennyezést.

A ZTE okos parkolási zóna projektje Magyarországon egyedül Szentendrén érhető el. A rendszer egyelőre 66 parkolóhelyet tart nyilván a Duna-korzón.

A parkolóhelyek szabad vagy foglalt voltát az aszfaltba süllyesztett érzékelők továbbítják rádiós úton a jeleket összegyűjtő három alállomás felé, s azokon keresztül a feldolgozást végző, központi alkalmazás felé, amely a felhasználóhoz juttatja az információt. A szabad helyek számáról egyrészt az út mentén elhelyezett digitális kijelző tájékoztat, másrészt a projekt mobil alkalmazása a szabad parkolóhelyre vezeti az autóst. Ezen felül a rendszer tájékoztat a díjfizetési időszakról, a díj mértékéről, és előkészíti a felhasználó számára a parkolást kezdeményező sms-t is, oly módon, hogy beírja a rendszámot, a vonatkozó parkolási zóna számát az ügyfél telefonszolgáltatójának előhívószámával.” [16]

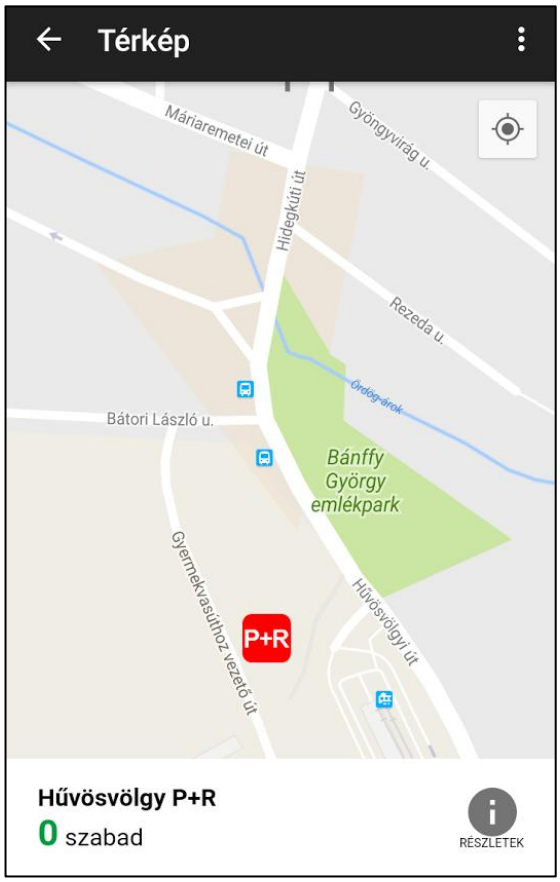
A következő képeken látható az applikáció működése.



13. ábra: Az applikáció működése

2.5. Közút Figyelő

A Közút Figyelő elnevezésű okostelefonos alkalmazás valós idejű forgalmi és parkolási információkkal segíti a fővárosi közlekedőket. Az alkalmazás segítségével megjelenített forgalomfigyelő kamerák képei alapján tájékozódhatnak az aktuális forgalmi helyzetről, a dinamikus információs táblák jelzésképei pedig aktuális forgalmi információkat nyújtanak. Emellett a szolgáltatás parkolási információkat is megjelenít, fővárosi P+R parkolók és parkolási létesítmények aktuális szabad férőhely adatai és egyéb jellemző információk publikálásával.



A cél egy korszerű közúti forgalmi információs alkalmazás létrehozása és működtetése, ezért azt folyamatosan fejlesztik. Ennek keretében bővítik a jelenlegi adatkörökben lévő objektumok számát, illetve terveink szerint bővül a megjelenített adatok köre is: elektromos autó töltőpontok helyének és foglaltságának megjelenítésével, balesetek és egyéb forgalmi zavarral járó események megjelenítésével, valamint valós idejű közterületi parkolási információkkal. A Közút Figyelő elnevezésű alkalmazás jelenleg próbaüzemben működik.[17]

14. ábra: A Közút Figyelő alkalmazás

2.6. Külföldi példák

Néhány külföldi példát is érdemes megemlíteni, és a főbb funkciókat megvizsgálni, hogy a parkolást támogató alkalmazásokról átfogó képet kapjunk. A pakolás külföldön is hasonlóan nagy probléma mint itthon. Egy 2011-es New Yorki IBM tanulmány szerit, az autók 30% már csak azért közlekedik, mert parkolóhelyet keres, amely átlagosan 20

percet vesz igénybe. Egy Los Angelesi tanulmány szerint minden nap 3600 mérföldet tesznek meg a közlekedők a parkolóhely keresés alatt, egy 15 háztömbből álló területen.

ParkWhiz: Lehetővé teszi, hogy lefoglaljuk a parkolóhelyet, és előre kifizessük a parkolási díjat, egy parkolóházban, közel a célpontunkhoz. Az alkalmazás generál egy kódot, amellyel a parkolóházban azonosítjuk a lefoglalt helyet. Az alkalmazás csak Amerikában működik.

ParkingPanda: Az alkalmazás amerikai parkolóházak egy részének adatait kezeli, tájékoztató jellegű információkat tartalmaz, szabadhely adatokkal nem rendelkezik, előre foglalási lehetőség nincs.

Parker: Ez az app segít parkolóhelyet találni, és oda navigálni. Van egy lényeges funkciója, a parkolótól a célpontig, és a célponttól a parkolóhelyig támogatja a gyalogos navigálást. Amerikában és Angliában működik több mint 30 városban. Csak app-on érhető el webes felülete nincs.

BestParking: Ez az alkalmazás a parkolás egy speciális szeletére fókuszál. 105 észak Amerikai reptéren, és környékén lehet parkolóhelyet találni a segítségével. A reptereken lehet helyet foglalni 1 naptól egészen egy hétig. [18]

Ezeket az alkalmazásokat a következő táblázatban (1. táblázat) hasonlítottam össze ergonómiai és egyéb szempontok szerint.

1. táblázat: A parkolási alkalmazások összehasonlítása

Tulajdonság / Név	App vagy weboldal	Real time szabadhely adatok	Előre foglalás	Mobil fizetés	Navigáció	Utcai vagy parkolóház
Parkopedia	mindkettő	nincs	nincs	nincs	van	parkolóház
Okos Parkolás	app	nincs	nincs	van	van	utcai
Közút figyelő	app	van	nincs	nincs	nincs	utcai
ParkWhiz	mindkettő	nincs	van	van	van	parkolóház
ParkingPanda	mindkettő	nincs	nincs	nincs	van	parkolóház
Parker	app	nincs	nincs	van	van	parkolóház
BestParking	mindkettő	nincs	van	van	van	parkolóház

3. Az alkalmazás modelljének elkészítése

A korábbiakban megállapítottuk, hogy olyan integrált rendszer jelenleg nem létezik, amely egy olyan adattárházra épül, amelyben egy felhasználói felületen keresztül informálódhatunk a szabad parkolóhelyek számáról, majd kiválasztva a hoznánk legközelebb lévőt, vagy a felkeresendő címhez legközelebb eső olyan parkolót, amelyben van szabad hely számunkra. A helyet akár le is foglalhatjuk azokban a parkolókban, amelyik támogatja ezt a funkciót, majd elindítjuk a navigációt, és a navigáció a valós forgalmi adatokat figyelembe véve elvezet bennünket az adott parkolóhelyre. A parkolóhelyre érkezve meghívja a parkoló saját alkalmazását, amely a parkolón belül vezet el bennünket a lefoglalt parkolóhelyre.

Az alkalmazás modelljének elkészítéséhez elsődlegesen koncepciót kell alkotnunk, amely tartalmazza a „flow”-ban résztvevő elemeket, tranzakciós szükségleteket, megbecsült aktivitásokat, és egzaktul lehatárolt feladatköröket.

A tervezett rendszer 3 fő koncepcionális elemből áll, melyek további részelemekre tagozódnak. A 3 fő rendszerelem:

- a parkolóház, és információs rendszere
- az általam felépített adattárház, és „webservice”
- végfelhasználói alkalmazások

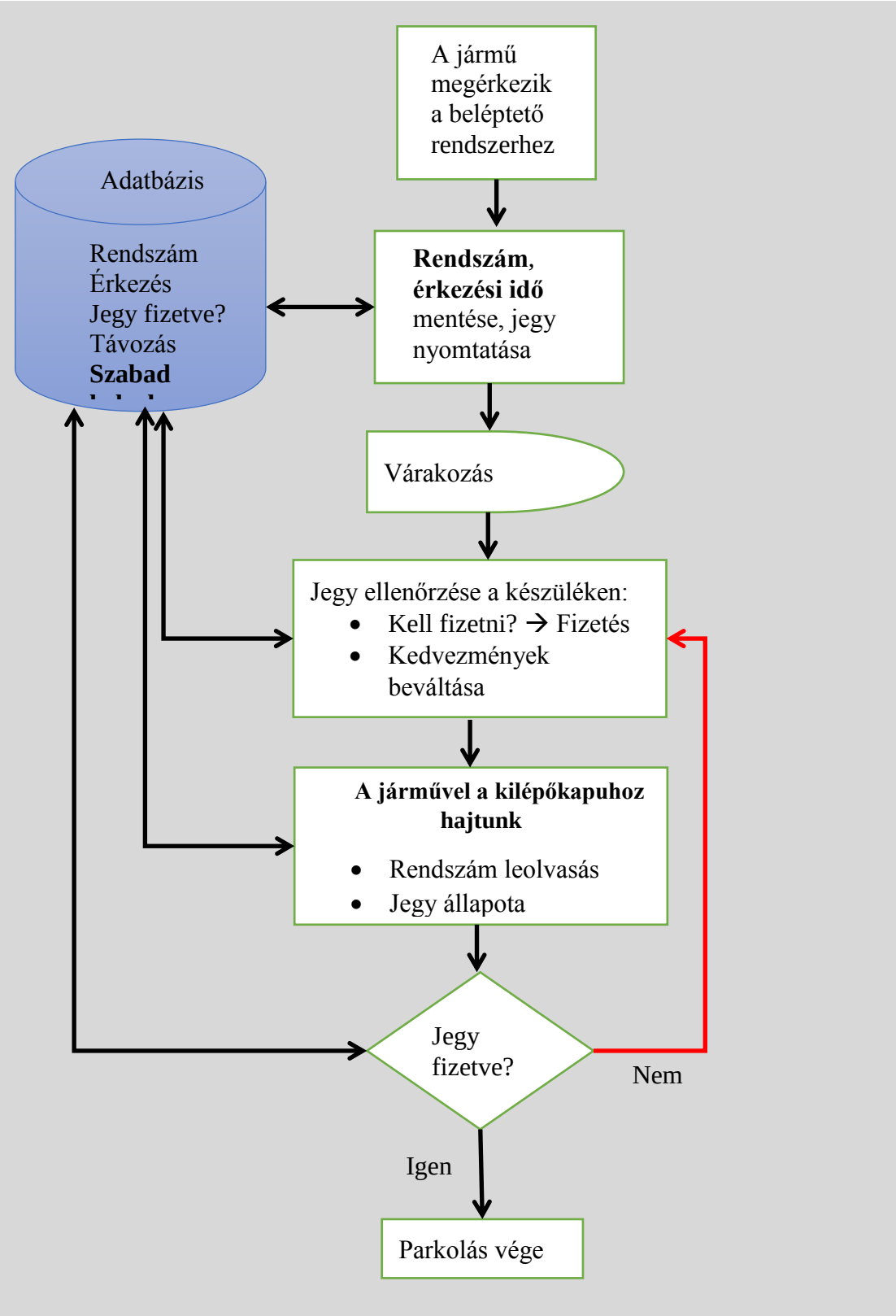
3.1. A parkolóház információs rendszerének részletes bemutatása

Az alkalmazásfejlesztés ötlete onnan ered, hogy a parkolóházak pontosan tudják, hogy hány szabad helyük van. Ezt az adatot ráadásul valós időben vagy kvázi valós idejű mérésekből ismerik. Alapinformáció a parkolóház kapacitása, és a ki és belépéseket kontrollálva megtudhatjuk a valós idejű kihasználtság adatokat. A ki és belépés „figyelése” a beléptető kapun keresztül valósítható meg, ahol általában jegynyomtatás, rendszámleolvasás történik. A kilépéskor történik a rendszámleolvasás, a fizetés ellenőrzése, sorompó nyitása. Ezen adatok birtokában pedig statisztikák készíthetőek, megfigyelhetők bizonyos trendek a parkolási szokásokat illetően. Ezek birtokában pedig előrebecslés is készíthető a szabad helyek számáról, igen jó közelítéssel.

Az alábbi folyamatábrán látható egy teljes parkolási folyamat. A komplett alkalmazás felépítéséhez szükséges ilyen mélységekig megismerni a rész összetevőket is. Látható, hogy az egész folyamat legfontosabb eleme a központi adatbázis. Ennek a parkolóház adatbázisnak minimálisan a következő elemeket kell tartalmaznia, ahhoz hogy a rendszer működőképes legyen.

- Rendszám (Úgynevezett „string" típus, kezel betűket és számokat, a hosszát nem érdemes korlátozni, sem formátumot rögzíteni, mert gondolni kell a külföldi rendszámokra is)
- Érkezés (Dátum, idő formátum, ÉÉÉÉ-HH-NN ÓÓ:PP:MM)
- Jegy fizetve? (Logikai, igen – nem állapot)
- Távozás (Dátum, idő formátum, ÉÉÉÉ-HH-NN ÓÓ:PP:MM)
- Szabad helyek (Számított érték)
- Parkolás állapota (Logikai, igen – nem állapot)

A számítógép-programozásban a sztring különböző egyszerű objektumok, leggyakrabban karakterek sorozata. [19]



15. ábra: Parkolási folyamat

Ezen adatok ismeretében már biztosan meghatározható az adott járműre vonatkozó aktuális parkolási állapot, a fizetendő díjtétel, a díjfizetés megtörténte, és a parkolási folyamat lezárható. A szabad helyek száma pedig egy számítási algoritmussal határozható meg adatbázis szinten, méghozzá a parkolóra jellemző férőhelyek számából levonjuk a folyamatban lévő parkolásokat. Ezzel a metodikával egy parkoló létesítményen belül a parkolás nyilvántartást oly módon kezelhetjük, hogy ez már kvázi alkalmas az általunk tervezett rendszer alapelemét szolgáltatni.

Fölmerül egy biztonsági kérdés is, méghozzá az, hogy hogyan rejtsük el a biztonsági kockázatot jelentő adatokat. Ilyen adatok a rendszám, és a hozzá tartozó érkezési, és távozási időpont, illetve a fizetési díjtételek járművenként. A rendszám azért fontos biztonsági elem, mert tulajdonképpen akár személyre is visszavezethető, a parkolási díjtételekből pedig következtetni lehet a parkolási bevételekre. Ezekhez az adatokhoz nincs jogunk hozzáférni, és az alkalmazásunk szempontjából teljesen szükségtelen is. Ezt a helyzetet tulajdonképpen egy adatbázis szintű kereszttáblával könnyedén lehet orvosolni. A parkolóház szoftvere az adatbázisból készíti el valós időben a számunkra szükséges kereszttáblát, a mi szoftverünk pedig csak ehhez a kereszttáblához fér hozzá.

Egy lehetséges adatbázis felépítés a parkolóház oldalán, illetve alább látható az a kereszttábla, amelyre a mi adatbázisunknak szüksége van.

2. táblázat: A parkolóház adatbázisának egy lehetséges felépítése

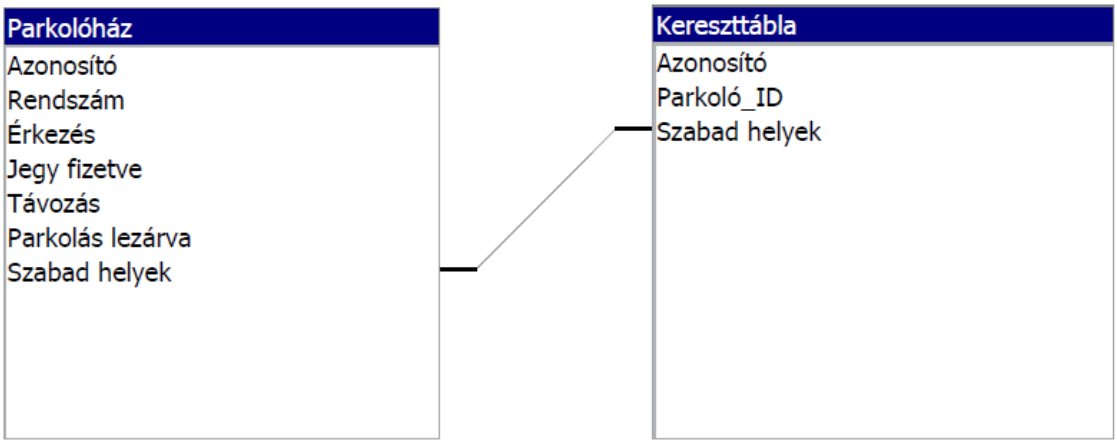
Parkolás_ID	Rendszám	Érkezés	Jegy fizetve	Távozás	Parkolás állapota
1	ABC123	2016. 10. 20. 10:20	Igen	2016. 10. 20. 11:20	Lezárt
2	ABC124	2016. 10. 20. 11:10	Igen	2016. 10. 20. 11:20	Lezárt
3	ABC125	2016. 10. 20. 13:30	Igen	2016. 10. 20. 14:00	Lezárt
4	ABC126	2016. 10. 20. 16:20	Igen	2016. 10. 20. 18:00	Lezárt
5	ABC127	2016. 10. 20. 16:30	Igen	2016. 10. 20. 17:15	Lezárt
6	ABC128	2016. 10. 20. 16:30	Igen	2016. 10. 20. 17:00	Lezárt
7	ABC129	2016. 10. 20. 16:30	Igen	2016. 10. 20. 19:00	Lezárt
8	ABC130	2016. 10. 20. 16:30			Folyamatban

9	ABC131	2016. 10. 20. 16:30	Folyamatban
10	ABC132	2016. 10. 20. 16:30	Folyamatban

3. táblázat: A szabadhelyek adatait tartalmazó kereszttábla

Parkoló_ID	Szabad helyek
00001	97

Ebben a kereszttáblában csak a parkolóháznak az azonosítója szerepel, és a szabad helyek száma. A táblák kapcsolatát a következő ábra szemlélteti.



16. ábra: Az adatbázis táblák kapcsolata a parkolóházban

Egyúttal becslést is teszünk a tranzakció adatátviteli szükségére. Ezt az információt egy „*.txt” fájlban tárolva nagyjából 50 bájt méretet foglal el. A „web request”-ekkel együtt néhány kilobájt méretű adatmennyiség szükséges ahhoz, hogy ez az adat bekerüljön az alkalmazásunk központi adattárházába.

Le kell szögezni, hogy az általam felvázolt rendszermodell, gyakorlatilag parkolóházanként eltérő, talán alapjaiban véve más lehet. Ezek részletes megismerésére biztosan nincs lehetőségünk, sem tudományos, sem üzleti céllal, hiszen a parkolási rendszert fejlesztők üzleti, és szakmai titokként kezelik azt. Abban biztosak lehetünk, hogy a szabad helyek számát azonban pontosan ismerik, mert ezt egy LED táblán általában meg is jelenítik. Tehát nekünk azt kell elérni, hogy ezeket az adatokat valamilyen módon megismerhessük. Meggyőződésem szerint ez úgy lehetséges, hogy alternatívát kínálunk számukra. Két megoldási lehetőség létezik, az hogy hozzáférést

engedélyeznek rendszerükhöz egy „interfacen” keresztül, vagy ők maguk küldik fel az adataikat automatikusan egy „webservice”-n.

Fontos a tudtukra adni, hogy ez egyfajta piaci versenyelőnyt biztosíthat a versenytársakkal szemben, mert az ügyfelek felé ezt úgy is kommunikálhatják, mit egy plusz „feature” ami egy ingyenes reklámfelületet ad a parkolóháznak, mert egy online adatbázisba kerülnek a szabadhely adatok.

A parkolóházakat is meg kell győzni, mert nekik az az aggályuk, hogy ha online látható, hogy épp nincs szabad helyük, akkor meg se próbál a járművezető helyet találni, hanem máshová megy. Ez valóban így van, és az alkalmazásnak épp ez a célja, mert így eliminálhatjuk a pakolókereséssel feleslegesen megtett kilométereket. Fontos azonban azt a pozitívumot is kiemelni, miszerint ezzel elkerülhető a parkoló túltelítődése.

Ez a megoldás működhet egy olyan parkolóhelyen is, amelyekkel Budapest belvárosa tele van. Több olyan parkolóhely található, amely egy elbontott bérlakás helyén lett kialakítva a belvárosban, és néhány évig fog üzemelni szabadtéri parkolóként. Ezeken a helyeken nem éri meg drága automatizált rendszereket kialakítani, többnyire nincsenek is. Itt egy parkolóőr egyedül kezeli a parkolóhelyet. Célom, hogy ezeket is integráljuk a rendszerünkbe. Ez egyszerűen kivitelezhető, mert egy mobil applikáció segítségével kvázi valós időben a parkolóőr tudja szinkronizálni a szabad helyek számát.

Azonban ahhoz, hogy az egész valós időben legyen kezelve, további lehetőségeket megvizsgálnunk. Mitől lesz valós idejű az adatszolgáltatás. Szükséges-e hogy minden egyes szabad hely változáskor legyen szinkronizáció, vagy meghatározott időnként, esetleg ezek kombinációja. A szinkronizálási módszerre az adattárház tervezésénél még részletesen kitérünk.

3.2. Adattárház felépítése

A következőkben megvizsgáljuk, hogy milyen nehézségekkel kell megküzdenünk egy adattárház felépítéséhez, illetve hogyan tervezzük azt meg. A központi kiszolgáló rendszerünk, az adattárház több különböző fő elemből tevődik össze. Ezek az adatbázis és a webservice-ek. Az adatbázis az az elem, ahol a beérkező adatokat rendszerezett módon, redundancia nélkül tároljuk. A webservice-ek ahhoz kellenek, hogy az adatokat

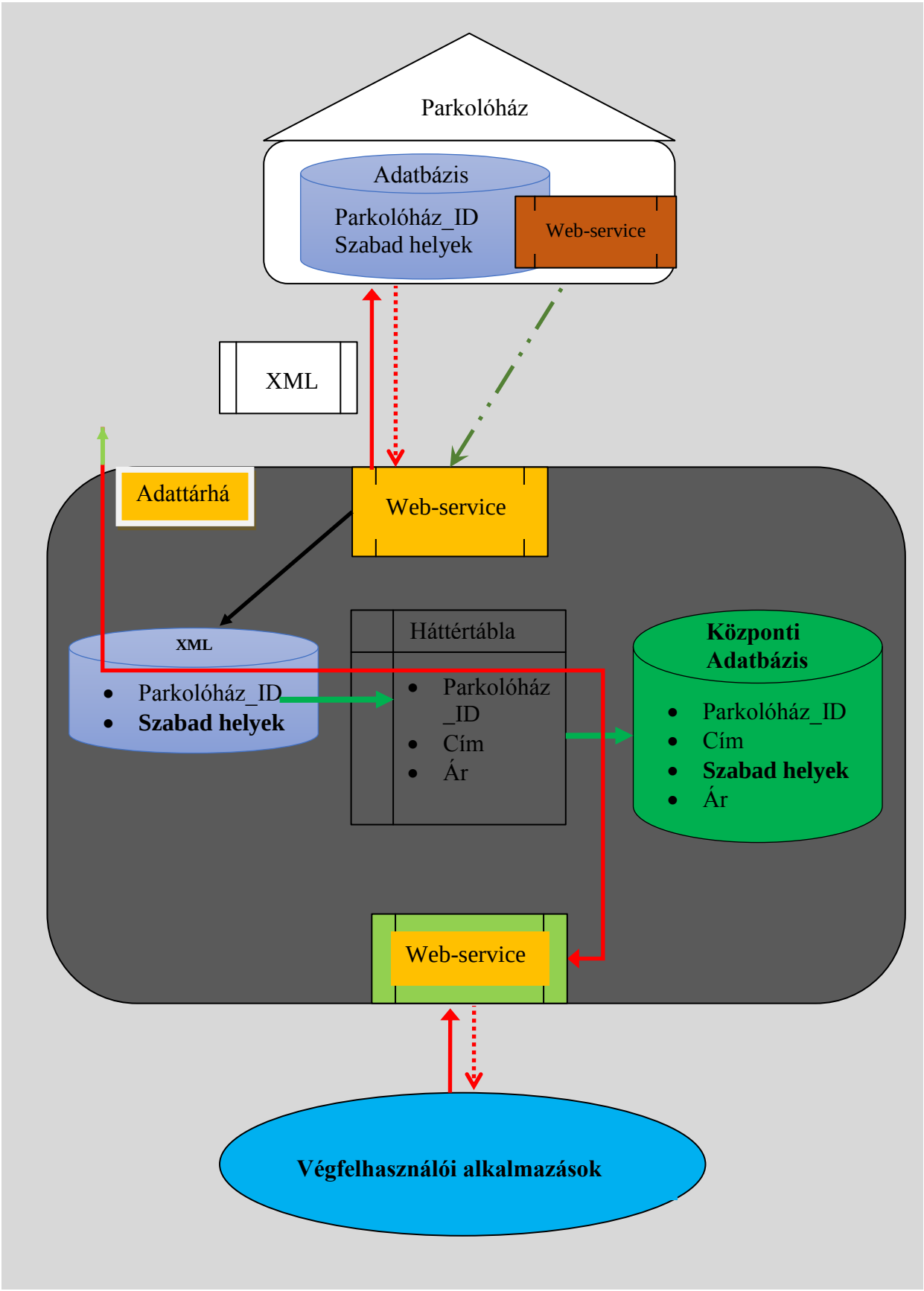
tudjuk fogadni, illetve a koncepciótól függően lekérdezni. Ezek után az adatbázisban eltárolni, majd ezeket az adatokat kiszolgáltatni a végfelhasználói alkalmazás felé. Ennek a logikai elrendezését láthatjuk a következő ábrán.

Az első fontos lépés az adatbázis megtervezése. Itt olyan koncepcionális kérdésekről kell dönteni, mint milyen adatokat tároljunk el. Milyen legyen a struktúra, az egyes elemek hogyan legyenek definiálva. Illetve az is meg kell határozni, hogy az adatbázis szerver milyen legyen.

Adatbázis-tervezés kapcsán tipikusan relációs adatbázis-tervezést szokás érteni két okból is. Egyrészt ami a relációs adatbázisokra helyes tervezési módszer és nézőpont, az jellemzően a többi adatmodell alapján felépített adatbázisokra is alkalmazható, másrészt mert ez leggyakoriabb adatbázisfajta.

A helyes tervezés általában nem könnyű feladat, számos, helyenként ellentétes szempontot érdemes mérlegelni döntéseink előtt. Amire célszerű tekintettel lenni:

- A létrehozott adatszerkezetben kódolt jellemzők és adatok a leírandó világot az adott célnak, nézetnek megfelelő teljességében reprezentálják.
- Ne lehessen olyan lekérdezést megadni, amely a leírandó világgal nincs összhangban.
- Adat (információ) ne vesszen el.
- Az adatot és annak összetartozó részeit a lehető legkevesebb helyre kelljen beszúrni, illetve helyen kelljen módosítani vagy törölni.
- Az adatbázis lekérdezésének hatékonyságát az adatszerkezet kialakítása ne rontsa jelentős mértékben. [20]



17. ábra: Az adattárház részletes logikai felépítése

Az adattárház az adatait a parkolóházaktól direkt vagy indirekt módon nyerheti ki. Az indirekt mód az, amikor az adattárház a webservice-n keresztül egy kérést küld a parkolóház szoftverének, hogy szinkronizálja a kereszttáblát. Ez látható a piros nyilakkal jelölve.

„A webszolgáltatás (angolul webservice) alkalmazások közötti adatcserére szolgáló protokollok és szabványok gyűjteménye. Különböző programnyelveken írt és különböző platformokon futó szoftveralkalmazások számítógép-hálózatokon (mint az internet) keresztül történő adatcserére használják a webszolgáltatásokat, az egy gépes folyamatközi kommunikációhoz (IPC) hasonlóan. Ezen interoperabilitás (például Java és Python, illetve Windows és Linux között) a nyílt szabványok használatának eredménye.”[21]

A webszolgáltatások előnyei:

- „A webszolgáltatások együttműködést (interoperabilitást) biztosítanak a különböző platformokon futó szoftver alkalmazások között.
- A webszolgáltatás nyílt szabványokat és protokollokat használ. A protokollok és adatok minden lehetséges helyen szöveg alapúak, így egyszerűsítve a fejlesztők feladatát.
- A HTTP használatával a webszolgáltatások keresztüljutnak a legtöbb tűzfalon a tűzfal paramétereinek megváltoztatása nélkül.” [21]

Ezzel a megoldással, csak egy kvázi valós idejű rendszert nyerünk, mert nem akkor történik a request és a válasz küldése, amikor a szabad helyek száma változott, hanem meghatározott időközönként. Természetesen ezt az időközt mi választjuk meg, így ha kellően kicsire választjuk, akkor egészen jól közelítjük a valós idejű adatnyerést. Gondolni kell azonban arra is, hogy egy éjszakai holtidőszakban, vagy egy csúcsidőszakban nem célszerű ugyanazt a frissítési időközt használni, emiatt egy óracsoportos beállítást javasolok. Méghozzá csúcsidőben 5 perc, csúcsidőn kívül 15 perc, éjszaka pedig 30 perc.

A „request – response” egy alapvető metódus, amit a számítógépek arra használnak, hogy egymással kommunikáljanak [22]

Az adattárház irányítja az adatok begyűjtésének ütemezését. Mivel a web-szolgáltatás a tárházhoz tartozik, ezért itt egyszerűen konfigurálható az adatkérés üzemezése. Ha valamilyen hiba folytán nem érkeznek meg az adatok, akkor a szerver

oldali hiba könnyebben feltárható vagy kizárható, ugyanis ha a request elküldésre kerül, akkor a hiba nem a mi oldalunkon áll fenn. Ehhez a megoldáshoz azonban a parkolókat fel kell készíteni az általunk küldött üzenetek fogadására, illetve a kereszttáblák küldésére. Ugyanakkor ahhoz, hogy az adattárház request-je megérkezzen a parkolóház szoftverébe, a tűzfalon, és a biztonsági rendszereken meg kell nyitni bizonyos portokat.

Ez egyfajta biztonsági kockázatot is jelent, emiatt vélhetően az üzemeltetők a másik adatgyűjtési eljárást részesítik előnyben. Tehát a parkolóház fogja bizonyos időközönként szinkronizálni a kereszttáblában lévő adatokat. Itt is szoftverfejlesztést kell végezni, hogy a parkolóház a kereszttáblát elkészítse és elküldje a megfelelő helyre. Ebben az esetben külső fél nem fér hozzá a rendszerhez, mert a szoftver csak egy kis táblázatot ad ki, amely az adattárházban lévő háttértábla ismerete nélkül értelmezhetetlen. A szinkronizálás sűrűségének kérdése itt is ugyanolyan lényeges elem. A túl gyakori és a túl ritka az előző bekezdésekben ismertetett problémákat hordoz magában.

Ennek az elvi kérdésnek a megoldására, és egy ténylegesen valós idejű rendszerhez egy másik elvet határoztam meg. A parkoló szoftver maga küldi egy speciális ütemezés szerint az adattárháznak. Ez a direkt mód, amikor a rendszer változást érzékel a szabad helyek számában, tehát behajtás vagy kihajtás történ a parkolóban, akkor indít egy szinkronizációt. Ezzel valós idejűvé tettük a rendszert.

Felvetődik a kérdés, hogy honnan tudjuk, hogy a legutolsó szinkronizáció alkalmával betöltött adat még aktuális, vagy esetleg hiba miatt nem tud a garázs szinkronizálni. A hibajelenségek kivédésére egy vegyes adatgyűjtési logikát javaslok. Tehát a parkolóház szinkronizál, amikor változás történik a szabad helyek számában, és az adattárház pedig egy óránként küld egy request-t. Ha erre nem jön válasz, akkor valamilyen hiba keletkezett, így a rendszer felismeri az esetleges meghibásodásokat.

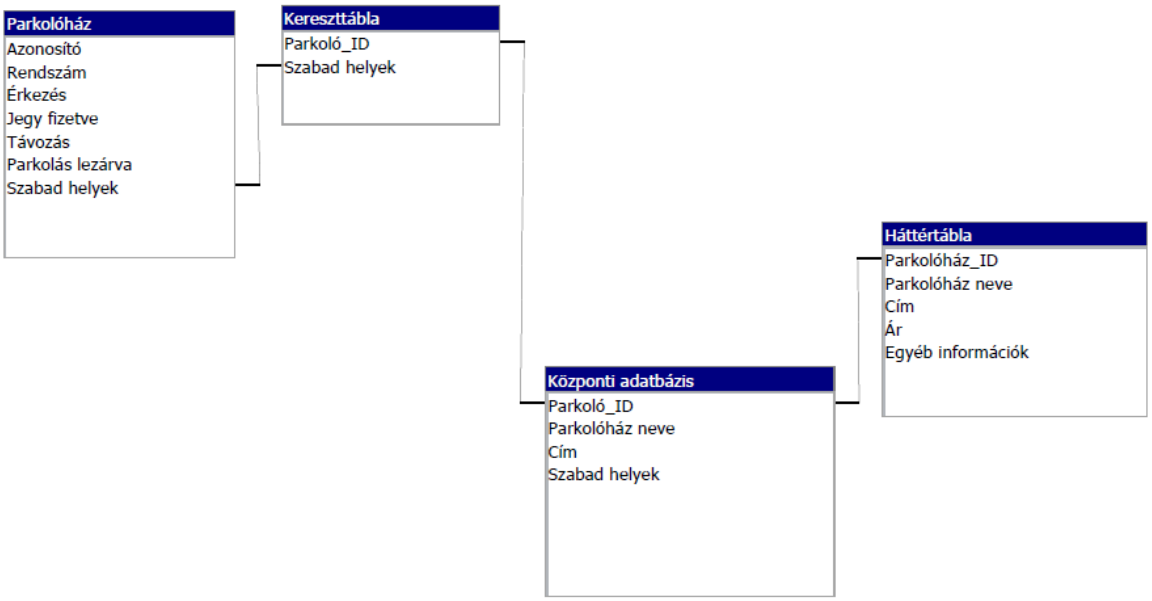
Az adatok továbbítására a XML formátumot javaslom, így olyan szöveges formátumokat állíthatunk elő, amelyek alkalmasak adatok strukturált leírására.[23]

„Az XML (Extensible Markup Language, Kiterjeszthető Jelölő Nyelv) a W3C által ajánlott általános célú leíró nyelv, speciális célú leíró nyelvek létrehozására. Az SGML egyszerűsített részhalmaza, mely különböző adattípusok leírására képes. Az elsődleges célja strukturált szöveg és információ megosztása az Interneten keresztül.” [24]

Az adattárházba a webservice-n kerszül megérkezik az XML adatcsomag. Ez tartalmazza a parkolóház_ID-t és szabad helyek számát. Ezek az adatok önmagukban még nem értelmezhetők és nem szolgáltathatók tovább.

Egy háttér adatbázis segítségével ezeket az adatokat átemeljük a központi adatbázisba. A háttér adatbázisban a parkoló_ID alapján azonosításra kerül a parkolóház neve címe, és az egyéb információk, míg a szabad helyek adatai pedig direktben emelődnek át a központi adatbázisba az XML fájlból. Az elsődleges kulcs minden esetben a Parkoló_ID. Ebből az azonosítóból csak egy van, és mind egyértelműen azonosít egy-egy parkolóházat.

Ezeket az adattábla kapcsolatokat láthatjuk a következő ábrán.



18. ábra: A központi adattárház adatbázisai, a Központi adatbázis, és a Háttértábla

Itt látható a teljes adatfolyam. A parkolóházban a Parkolóház adattáblából képződik egy Kereszttábla, ez kerül átküldésre a parkolóházból az adattárházba. A Parkoló_ID és a Szabad helyek a Kereszttáblából emelődnek át a Központi adatbázisba, a Háttértáblából a Parkolóház_ID mint elsődleges kulcs alapján pedig a Cím, Ár, és Egyéb információk kerülnek át a Központi adatbázisba.

Ezzel létrejött egy valós idejű adatbázis, amely tartalmazza a parkolóházak adatait, és a szabad helyek számát. Ezt már csak továbbítani kell a végfelhasználók felé, amely egy másik webszolgáltatáson keresztül történik.

A háttértábla felépítése:

- Parkolóház_ID: Szám típusú meghatározott hosszúságú elem, a parkolóházak egyedi azonosítására
- Parkolóház neve: „string” típusú elem
- Cím: A parkolóház címe, „string” típusú elem
- Ár: szám típus
- Egyéb információk: „string” típus

Központi adatbázis felépítése:

- Parkolóház_ID: Szám típusú meghatározott hosszúságú elem, a parkolóházak egyedi azonosítására
- Parkolóház neve: „string” típusú elem
- Cím: A parkolóház címe, „string” típusú elem
- Szabad helyek: szám típusú elem

„Kis mennyiségű adat tárolására általában elegendő a sokak által jól ismert táblázatkezelő programok által nyújtott szolgáltatás. Ám nagy mennyiségű adat esetén már körülményessé válik a kezelésük.” [25]

Az adatbázis kezelő szervernek a MySQL-t javaslom.

A MySQL egy többfelhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver. A MySQL az egyik legelterjedtebb adatbázis-kezelő, aminek egyik oka lehet, hogy a teljesen nyílt forráskódú LAMP (Linux–Apache–MySQL–PHP) összeállítás részeként költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására. [26]

Az adatbázisunk felépítése után megosztjuk az adatainkat a végfelhasználókkal. Itt jön képbe az, hogy az adatbázis kezelő rendszernek szükséges több szálon futnia, hiszen egyszerre több lekérdezést fog végezni, több felhasználót fog kiszolgálni. Felmerül a kérdés, hogy az adatok kiszolgálásánál a „push”, vagy a „pull” elvet követjük, esetleg támogatjuk mindkettőt, mint az adatgyűjtésnél.

A push elv az azt jelenti, hogy egy másik adatbázisnak vagy alkalmazásnak átadjuk a teljes tartalmat valamilyen metodika szerint. A pull elvet követve, pedig valaki lekérdezi a számára szükséges információt. Ezeket a felhasználási lehetőségeket a következő fejezetben taglalom.

4. A szoftverfejlesztés folyamata és az applikáció megvalósítása

Az adatbázisban összegyűjtött valós idejű adatokat strukturált módon kell megjeleníteni a végfelhasználóknál, alkalmassá téve az adatok egy szűrt részének megjelenítésére. Meg kell vizsgálnunk, hogy az adatokat milyen céllal, és hogyan akarjuk leszűrni.

Dolgozatom kiinduló eleme az volt, hogy a közlekedőknek szeretnék adni egy olyan szoftveres szolgáltatást, amellyel tájékozódhatnak a szabad parkolóhelyek helyek számáról és hollétéről, ezáltal csökkentve a parkolóhely keresés nehézségeit.

Egy szoftverfejlesztés a használati esetek, úgynevezett usecase-ek definiálásával kezdődik. A usecas-ek az alkalmazásnak a felhasználói oldalon vett lehetséges felhasználási módjai. A használati esetek leírják, hogy mit csinál a rendszer a végfelhasználó szemszögéből. Ezeket mindig a végfelhasználói oldalon vizsgáljuk, és onnan specifikáljuk.

Az alkalmazás két használati esetet valósít meg, melyek a következő táblázatban láthatóak.

4. táblázat: Az alkalmazás használati esetei

Használati esetek (Usecase-ek)	Használati esetek leírása
Keress parkolóhelyet a közelben!	A készülék GPS koordinátáit lekérdezzük, és a real time háttér adatbázisból lekérdezzük a hozzánk legközelebb eső olyan parkolóházat, ahol van szabad hely számunkra.
Az elérendő cím közelében keress parkolóhelyet!	A készülék a felkeresendő cím közelében megkeresi a legközelebb eső szabad hellyel rendelkező parkolóházat.
(Beállítások)	(maximális gyaloglási távolság a parkoló és az keresett cím között; parkolóhely ára; mélygarázs, vagy parkolóház; hány opcionális parkolót listázzon az app kereséskor) milyen navigációs appot hívjon

Természetesen ezeket a funkciókat a legegyszerűbb felhasználói felületre kell elkészíteni, mivel tulajdonképpen ez egy navigációs alkalmazás. Ezeknél az alkalmazásoknál pedig fontos követelmény, hogy a felhasználó felület könnyen kezelhető legyen, mert ez nem vonhatja el a figyelmet a vezetésről.

A használati eseteket gyakorta megjelenítik úgynevezett használati eset diagramban, aminek a célja a követelmények rögzítése. A középpontban a rendszer által végrehajtható funkciók vannak, a rendszerszek teljes leírására használható. Ez rögzíti, hogy milyen funkciói legyenek a rendszernek, és a funkcióknak mit kell pontosan tudnia

A használati eset diagram elemei:

Aktorok azonosítása:

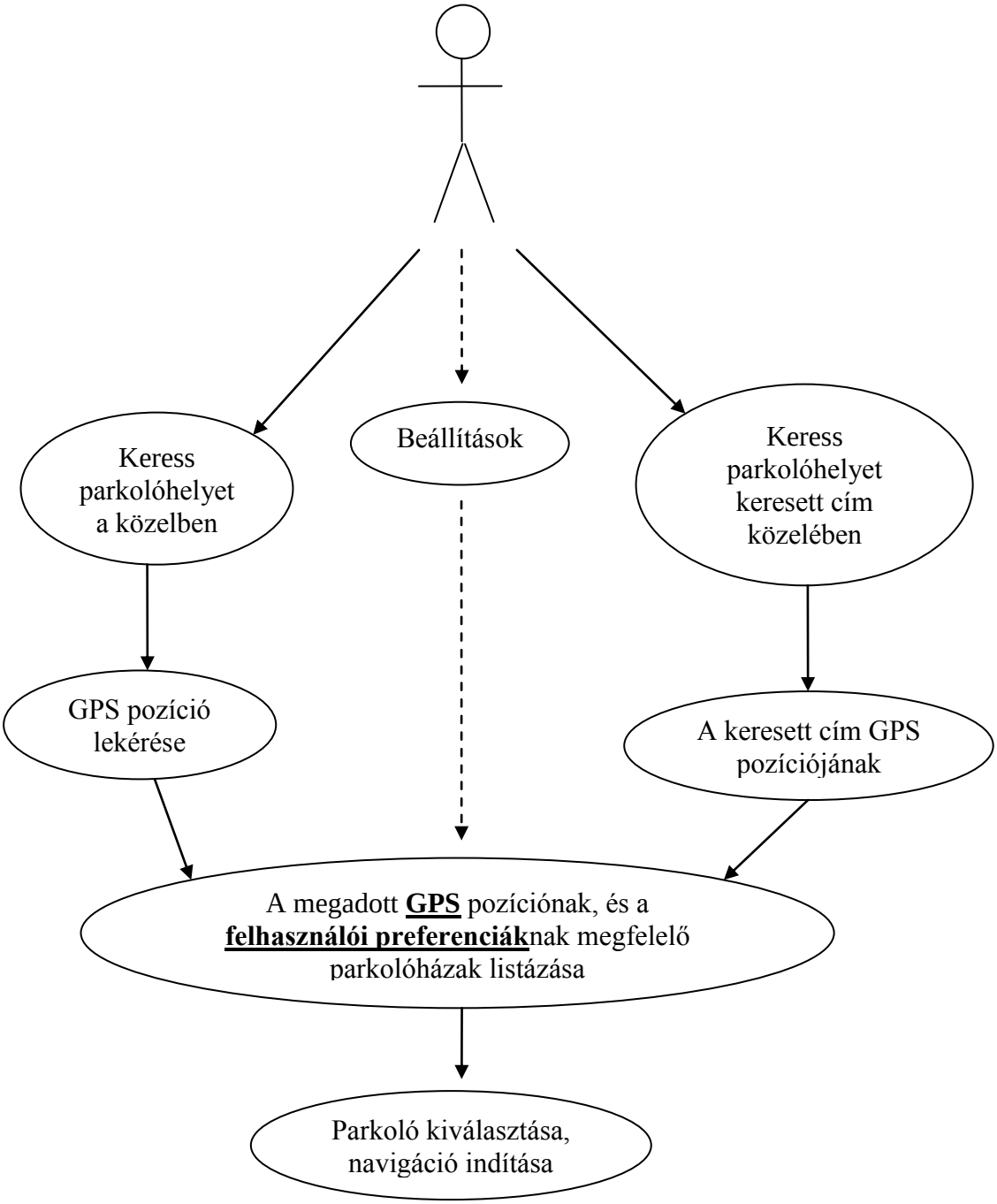
- kik a rendszer használói (user)
- ki felel a rendszer karbantartásáért
- mik a rendszer által használt erőforrások
- mik a rendszerhez kapcsolódó más rendszer(ek)

Használati esetek azonosítása:

- mire használják a rendszert
- mit csináljon a rendszer
- hogyan használják a rendszert
- mit tudjon a rendszer [27]

Ezek definiálása után pedig a használati esetek és az aktorok kapcsolatát kell meghatározni.

A rendszer használóit definiálni kell. Itt csak „user” szint van, adminisztrátor nem szükséges, hiszen az app lehetővé teszi, hogy minden felhasználó saját maga admin-ja legyen, mert a beállításokban a keresési paramétereket, a user egyéni preferenciája szerint alakítja. A következő képen láthatóak a használati esetek lefolyásai a folyamatos nyilak szerint, a szaggatott nyilak a beállítások adattartalmának az átemelését jelentik.



A használati eset ábra az összes lehetséges folyamatot leírja.

A használati esetek leírása után definiálni kell a rendszer határait és a rendszer építőegységeit. A szoftvertervezés során már a kezdetekkor két fő részre bontjuk a produktumot. Ez az úgynevezett front-end és back-end oldalak, tehát a végfelhasználói és a háttér (szerver) oldal. „A back-end réteg feladata a front-end réteg felől érkező adatok feldolgozása, illetve a keletkezett eredmény a front-end számára történő

visszajuttatása.” [28] „A front-end az adott rendszer legfelsőbb, a felhasználóval vagy a csatlakoztatott további rendszerekkel a kapcsolatot tartó rétege.” [29]

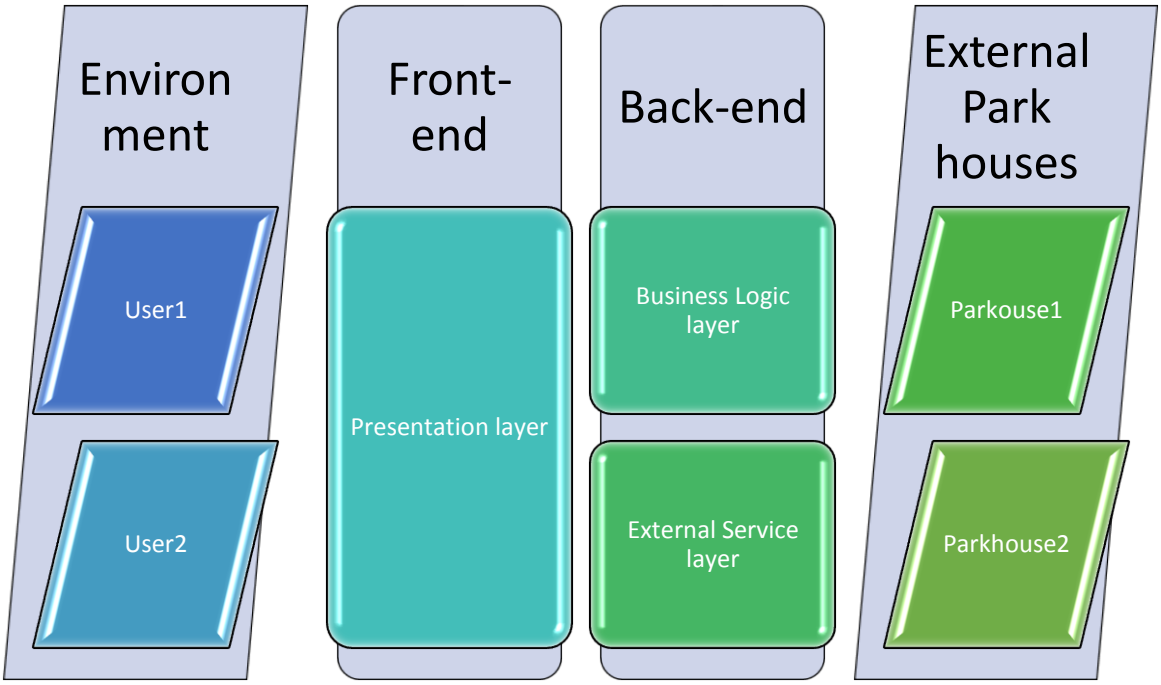
A front-end réteg feladata a rendszerből kijutó adatok prezentálása, illetve a bejövő adatok fogadása a felhasználó vagy a csatlakoztatott rendszer felől. A végfelhasználói oldal magában foglal mindent, amit a használója lát. A back-end oldal minden olyan elem, amelyet a felhasználó közvetlenül nem lát, ezek a szerverek, adatbázisok. A háttér oldalon dinamikus tartalommal dolgozunk, míg az ügyfél oldalon statikussal. A két terület között azért éles határt nem lehet húzni, mert a kettő folyamatos kapcsolatban áll egymással, és folyamatosan kommunikálnak. [30]

A szoftvertervezésnél több réteg van a hardver és a felhasználó között. Ezek a rétegek kommunikálnak egymással és a front-end és a back-end réteg között is. A front-end réteg a mögöttes tartalom egy felhasználó számára érthető absztrakciója. „A többrétegű architektúra a szoftverfejlesztésben alkalmazott kliens-szerver architektúra, melyben a megjelenítés, az adatkezelés és az üzleti logika különálló folyamatokra van bontva. Az alkalmazás rétegekre bontásával a fejlesztők könnyen karbantartható és fejleszthető rendszereket hozhatnak létre, mivel elegendő egy-egy réteget javítani, illetve bővíteni az egész alkalmazás módosítása helyett.” [31]

Az alkalmazásomat 3 rétegű architektúrában kell elkészíteni. Ezek a rétegek a megjelenítés (presentation), az alkalmazási (application) és az adat (external service).

A megjelenítési réteg a front-end oldalon a megjelenítésért felel. Ez maga a felhasználói felület, és a felhasználó, valamit a rendszer közötti kommunikációs csatorna. Eljuttatja a megfelelő információt az alkalmazási rétegnek, ahol az adatfeldolgozás és a probléma megoldása történik. „A megjelenítési réteg az első, ahol az adatra már nem csak úgy tekintünk, mint egy köteg 0 vagy 1-esre. Foglalkozik például a karakterlánc értelmezésének problémakörével is.”[32] Az alkalmazási réteg csak előkészíti a továbbítandó, vagy megjelenítendő adatot, és ezt a feladatot a megjelenítési réteg végzi el. További feladata a titkosítás is. Ennek a rétegnek a további feladata, az adatstruktúra értelmezése, például az XML struktúráé, hiszen ezt a struktúrát javasoltam az építőelemek közötti kommunikáció alapjának. [32] Ez a réteg alkotja meg a felhasználó felületet, amelynek tartalma lehet statikus vagy dinamikus. Az általam tervezett alkalmazásban dinamikus réteg szükséges, ugyanis több lekérdezést

kell végrehajtani, majd a megfelelő parkolóház kiválasztása után, navigálás közben is folyamatosan figyelni kell a szabadhely adatokat.



A réteges architektúrák jellemzője, hogy minden réteg egy-egy jól definiált feladatot lát el, a közvetlenül alatta lévő réteget felhasználva, ezért fontos érteni az ábrán szereplő rétegek szerepét.

Az (External Service layer) adatréteg feladata az adatok perzisztens tárolása, és az azokon végezhető elemi műveletek – vagyis a létrehozás, lekérdezés, módosítás és törlés – támogatása. Ez a réteg, ahogyan a neve is jellemzi felelős a külső adatok beszerzésért. Tehát a parkolóházban keletkezett szabadhely adatokat egy gyűjti be, és strukturálja, illetve lekérdezhetővé teszi az alkalmazási réteg számára. Ezt a réteget relációs adatbázis segítségével valósítjuk meg. A külső adatforrásokhoz való kapcsolódást több modul biztosítja, ugyanis a korábbiakban már felvezettem, hogy az adatokat az adattárház is lekérdezheti, vagy a parkolóház szinkronizálhatja. Ezen eltérő ütemezési mintáknak megfelelően több modul szükséges, amely fogadja az adatokat a különböző platformokról. Az adatrétegre épül az üzleti logikai réteg, amely a konkrét alkalmazási terület igényeinek megfelelő funkcionalitást biztosítja oly módon, hogy az üzleti szabályok figyelembevételével hívja meg az adatréteg szolgáltatásait. [33] Az üzleti rétegben történik a problémamegoldás. Begyűjti az utas pozícióját és a beállításaiban előre definiált korlátokat (maximális gyaloglási távolság, stb.) ezeknek

megfelelő parkolóházak listáját lekérdezi az adatrétegből, majd átadja a megjelenítési rétegnek.

A rétegek közötti kommunikáció az úgynevezett API-n keresztül valósul meg. „Az alkalmazásprogramozási felület vagy alkalmazásprogramozási interfész (angolul application programming interface, röviden API) egy program vagy rendszerprogram azon eljárásainak (szolgáltatásainak) és azok használatának dokumentációja, amelyet más programok felhasználhatnak. A rendszeren belüli kommunikációhoz azért érdemes meghatározott API-kat elkészíteni, mert így a modulok közötti kommunikáció rögzített lesz, és a későbbi modulfejlesztések alkalmával tudjuk az elvárt adatstruktúrát a modulok közötti sikeres kommunikációhoz. Az External Service layer API-jának elkészítése a rendszer működőképességének lételeme. A rendszerünkhöz több külső API elkészítése szükséges, mert a parkolóházak belső rendszerei eltérőek, így a megfelelő hozzáférés kialakításához ez elengedhetetlen. Minden olyan parkolóház, amelyet mi kérdezünk le, különböző interfészt igényelhet. Azoknál a helyeknél ahol maga a parkolóház szinkronizál, elegendő egy interfész, mivel megkövetelünk tőlük egy megadott adatstruktúra szerinti szinkronizációt. Ezek alapján megkülönböztetünk úgynevezett külső és belső API-kat, a belső a rétegek közötti kommunikációs szabványrendszer, a külső pedig az adatréteg és a parkolóházak közötti.

További kardinális kérdés maga a felhasználói felület és a design és az implementációs forma. Szükséges döntést hoznunk, hogy natív alkalmazást vagy HTML5 webes felületet készítsünk. A natív alkalmazás egy olyan program, amelyet egy megadott platformra, vagy eszközre készítettek. Tehát platformfüggő. Mivel egy adott platformra készül, ezért képes kapcsolatba lépni a rendszer más elemeivel is, és egyéb szoftverekkel, például GPS használata.

A HTML5 lényege, hogy egy weboldal, webalkalmazás készül vele, ezáltal platform független. Megjeleníthető okostelefonon, táblagépen és asztali gépen is. Ez a szabvány alkalmas a szenzoradatok (pl.: GPS) kiolvasására, így ez is alkalmas lehet az alkalmazásunk elkészítésére, ezen felül még asztali gépen is megtekinthető lenne, bár ennek a közlekedés közben nincs sok haszna.

Az érdemi döntés meghozásához hasonlítjuk össze az előnyöket és a hátrányokat. Az ikonra nyomva a natív alkalmazás azonnal indul és általában valamilyen korlátozott funkcionalitással offline is működőképes. A böngészőben futó webes appok betöltése

sokkal bonyolultabb, el kell indítani a böngészőt, majd meghívni a megfelelő URL-t, a működéshez pedig az internetes hozzáférés alapvető szükséglet. Jelenleg a natív alkalmazások lényegesen gyorsabbak, reszponzívabbak, mint a webappok. Ezzel szemben „A reszponzív weboldal (HTML5) egy olyan megközelítéssel tervezett weboldal, amelynek a célja az, hogy optimális megjelenést biztosítson - könnyű olvashatóság, egyszerű navigáció a lehető legkevesebb átméretezéssel és görgetéssel - a legkülönfélébb eszközökön az asztali számítógép monitorjától egészen a mobiltelefonokig.” [34] Mind a megjelenítés, mind a betöltés gyorsabb, az erőforrásokkal pedig kíméletesebben bánik. Hozzáférés az alacsonyabb szintű funkciókhoz: GPS, mikrofon, háttérben zajló szinkronizáció egyszerűbben kezelhető natív alkalmazáson. [35]

Ezek alapján a programnak a natív alkalmazást javaslom, mert alkalmas a szenzoradatok lekérdezésére, ami alapvető kritérium (pl. GPS pozíció lekérdezése). A megfelelő parkolóház kiválasztása után pedig meg kell hívni az előre definiált navigációs appot. Ez a kritérium dönti el a kérdést a natív és a HTLM5 között, hiszen a HTML5 nem képes az eszközön lévő egyéb alkalmazásokat meghívni.

Napjainkban gyakori probléma, hogy a piacon többféle alkalmazás van, ami ugyanazt a szolgáltatást nyújtja, csak a felhasználói élményben van némi eltérés. Teljesen felesleges a jelenleg működő több szoftver mellé fejleszteni még egyet. Fontosabb az, hogy hiányszolgáltatást pótoljunk, és a meglévőket integráltabbá, és még jobbá tegyük. Ezért a saját alkalmazás fejlesztésétől eltekintek, és javaslatot fogalmazok arra, hogy hogyan lehetne az adattárház nyújtotta lehetőséget kiaknázni a piacon jelenleg meglévő elemekkel.

Egy ilyen felhasználási alternatíva lehet az, hogy a 12. képen bemutatott Parkopedia alkalmazásnak átadni a szabadhely adatokat. Ez egy jól működő, praktikus, jól használható alkalmazás, mely alkalmas lehet parkolóhely adatok kezelésére. Ennek a megoldása két módon valósítható meg. Egyenkénti lekérdezéssel, tehát amikor a felhasználó rászűr egy adott parkolóra, akkor az előugró menüben megjelennek a szabadhely adatok is, úgy hogy azt az adatot nem a Parkopedia szerveréről, hanem az adattárházból tölti le. Ez a nehezebb megoldás, mivel az applikációnak tulajdonképpen két szerverrel kell kommunikálnia, így nagyobb adatforgalmat jelent a lekérdezés.

Másik megoldási lehetőség, hogy az adattárházhoz a Parkopedia adatbázis szinten hozzáfér, és a szabadhely adatokat innen szűri le a szerver. Ennek a mobil adatforgalmi igénye szinte megegyezik a jelenlegivel, mert gyakorlatilag csak egy adattal kér le többet a telefon ugyanarról a szerverről.

Megjegyeztem, hogy a meglévő rendszereket integráltabbá kell tenni. A parkolási folyamatba is szükséges beépíteni az alkalmazások közötti kommunikációt és az integrált működés lehetőségeit. A Parkopedia alkalmazással kiválasztjuk a megfelelő parkolóhelyet, úgy hogy a szabadhely adatokat is figyelembe vesszük. A navigáció automatikusan elindul, az Google Maps app hívódik meg, és automatikusan átkerülnek a célpont adatai. Egy fejlesztési lehetőség, az integráltság jegyében, hogy a navigációval megérkezve a kiválasztott parkolóházhoz, induljon el egy olyan alkalmazás, amelyik a parkolóházon belül segít megtalálni a szabad parkolóhelyeket. Ilyen alkalmazást mutattam be a 15. ábrán. Ezáltal a parkolóhely keresés problémáját lényegesen könnyebbé tettük, mert azonnal olyan helyre navigáltuk a járművezetőt, ahol biztosan volt szabad hely, a navigáció figyelembe vett valós forgalmi adatokat, majd a parkolóban elvezettük a legelső szabad helyre.

Az adatokat, végül pedig statisztikai céllal is felhasználhatjuk. Ehhez azonban a beérkező információkat időbélyeggel kell ellátni. Ez azt jelenti, hogy a Központi adatbázis módosulni fog, kiegészül egy vagy több oszloppal. Ezáltal a parkolás, mint általános probléma statisztikai jelleggel leírhatóvá válik, megfigyelhetjük a parkolási szokásokat, trendeket. Ezeked a trendszerűségeket elemezve, pedig felkészülhetünk a jövőben parkolási helyzet kezelésére.

5. A napi tevékenységi láncok optimalizálási eljárásának kiegészítése parkolóház keresési funkcióval

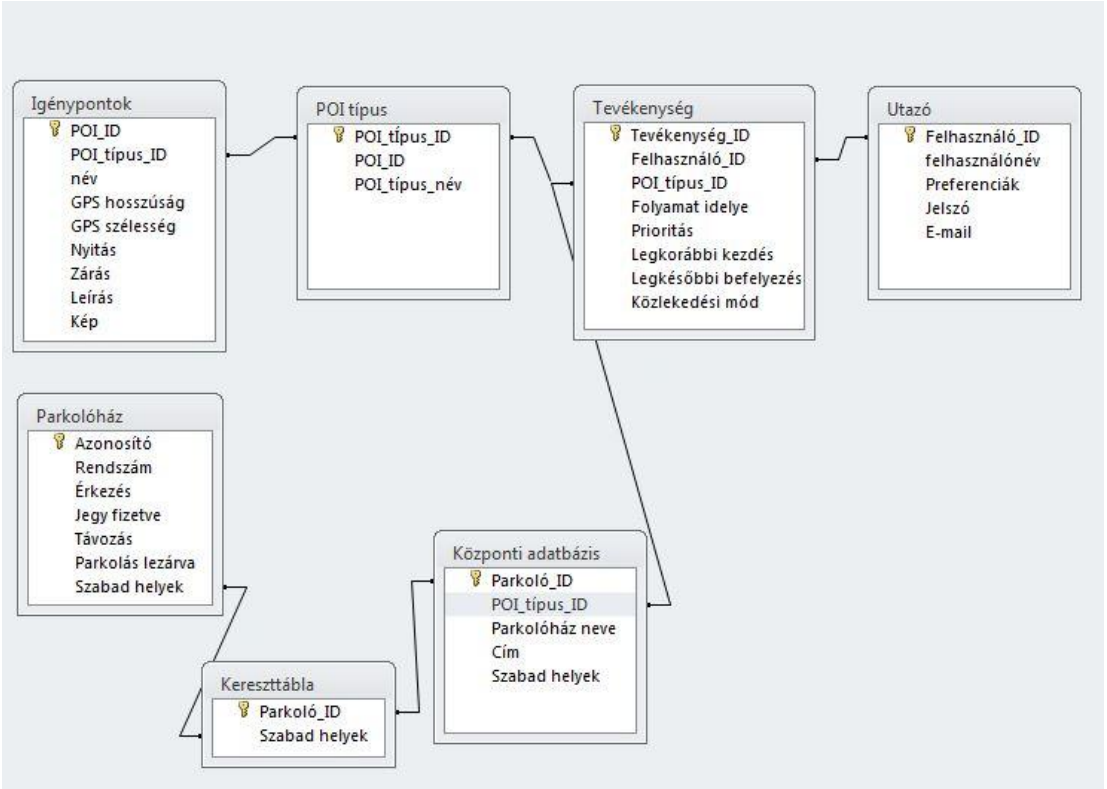
Az utazók legfontosabb szempontjai, hogy az utazást tudatosan megtervezhessék, megtekinthessék az alternatív útvonalakat, és az utazási időket összehasonlíthassák. Ezeknek a preferenciáknak az ismeretében, megalkotható egy olyan módszer, amellyel optimalizálhatják a napi aktivitási láncokat.

Tehát, nem csak egy adott utazást lehet optimalizálni, okos eszközökkel, és valós idejű adatbázisokkal, hanem akár az utazó teljes napját. Tulajdonképpen az utas így jobban jár, hiszen nem rész optimumok összegeként tölti a napját, hanem a teljes tevékenységi lánc optimalizálásra kerül.

Az alapprobléma az, hogy össze kell rendelni az utas aktuális pozícióját, és a látogatott helyeket egy speciális szempontrendszer szerint. Ez lehet az utazási idő, költség, átszállások száma, vagy ezek kombinációja. A tevékenységek kiválasztására, és sorba rendezésére a TSP (Travelling Salesman Problem) eljárás ad megoldást. Alapjában a TSP eljárást a logisztikai rendszerekben használják, de itt az áruk helyett látogatott pontokat használunk, így az eljárásban nincs szignifikáns eltérés. A fő optimalizálási paraméter az utazási idő. A mi esetünkben egy extra korlátozó tényezőt kell definiálni, mert a boltok, és intézmények nyitvatartási idejét figyelembe kell venni.

Az első lépés a tevékenységi láncok modellezésénél az adatbázis struktúra megalkotása, és az adatok tárolása. A modell tartalmazza az Utazó táblát, ahol az utazó adatait, preferenciáit tároljuk. Az információkat a felkeresendő pontokról az Szükséges pontok tábla tartalmazza, amelyben a helyekre vonatkozó adatok vannak. Ezek az érdeklődési pontok osztályozódnak be a POI (Points of Interest) táblába, amely segít alternatív ügyintézési pontokat találni. A legfontosabb a Tevékenység tábla, ahol az utazók megalkotják a napi tevékenységi láncukat. Minden tevékenység egy megadott POI típusba tartozik. A prioritás, a folyamati ideje, és a keresleti időablak, a legkorábbi kezdéssel, és a legkésőbbi befejezéssel rögzítésre kerül. Az Utazó táblát és a Tevékenység táblát az utasok töltik ki. Ez azt feltételezi, hogy az utazó tudja, hogy bizonyos napokon milyen tevékenységei vannak, és tud ebből egy listát készíteni. A Szükséges pontok és a POI tábla külső adatbázisból kerül feltöltésre.

A felső sorban lévő táblák az optimalizációs szoftver táblái, míg az alsó táblák a parkolási adattárház táblái. Az alap POI tábla kiegészítendő a parkolóházak adataival. A parkolóházak nevével, helyadataival, és szabadhely adataival. Az összerendelés logikai alapját, az jelenti, hogy egy újabb POI típust hozunk létre, ez pedig a parkoló. Így POI táblába a parkolási adattárház teljes lényegi tartalma átkerülhet. Gyakorlatilag létrejön egy újabb POI típus, amely a parkoló létesítményeket összegzi, így a modell a parkolást is tervezi a lánc optimalizálásakor. Azért javaslom a parkolási adatbázist integrálni a POI táblába, mert így az eljárásban a programkódon nem kell módosítani, csak paramétereket kell megváltoztatni. Tehát egy újabb korlátozó paramétert kell bevezetni, autós közlekedés esetén a látogatott hely közelében választani kell egy parkolót is. Ez egy újabb igénypontként kerül be a tevékenységi láncba. A tevékenység tábla újabb rekorddal fog bővülni, mégpedig a közlekedési móddal. A közlekedési módban lesz eltárolva az, hogy milyen módon utazunk. Ez azért fontos, mert innen tudja az algoritmus, hogy kell-e parkolóházat keresnie. Csak akkor keres parkolóházat, ha autóval közlekedünk. A tevékenység elvégzésére kiválasztásra kerül egy alkalmas igénypont, ennek a helye ismert, így ehhez választható szabad parkolóház.



19. ábra: A parkolási adattárház és az optimalizáló algoritmus összekapcsolása

A parkoló_ID egyértelműen azonosítani fogja a parkolóházat, és a hozzá tartozó címet, és szabad hely adatokat.

A modellben a felkeresendő helyek gráf csúcsaiként értelmezhetőek, és a gráf élei az útvonalak a helyek között. A költségmátrixot az élek definiálják, és az elemei az utazási idők a felkeresendő pontok között, ez mintegy hasznossági függvényként viselkedik. A cél a költségmátrix minimalizálása.

A következő korlátokat kell definiálni, az alap TSP probléma megoldásához:

- Az utazó elhagy minden egyes címet
- Az utazó érint minden egyes címet
- a költségmátrix elemei nem negatívok

Továbbá a következő korlátokat kell teljesíteni, ha az időablakokat is figyelembe vesszük.

- a valós időablak két időpont különbségeként van definiálva, amelyek az ügyintézési pont nyitvatartási idejéből, és a folyamat idejéből áll. A folyamat ideje az az idő, amelyre szüksége van az utasnak ahhoz, hogy megoldja a feladatát (Pl.: bevásárlás). A folyamat ideje statikus.
- minden egyes időablak legalább olyan hosszú legyen, mint a folyamat ideje
- Az igénypont akkor megfelelő, ha a valós időablak és a folyamat ideje illeszkedik a hely nyitvatartási idejéhez.

A TSP eljárás rugalmasnak tekinthető, ha az utazók célpontjai, szubjektív igényei, és önkényesen helyettesíthető egy másik ponttal, amely ugyanazzal a funkcióval bír. Ez az egyik kulcsa, hogy a napi tevékenységi láncot optimalizáltabbá tegyük.

A tevékenységi lánc tartalmazza minden rendszeres (iskola, munkahely) és nem rendszeres (étterem) tevékenységeit az utazónak. A rendszeres tevékenységek térbeli és időbeli paraméterei általában rögzítettek, amíg sok esetben a nem rendszeres tevékenységek rugalmasak. Minden tevékenyég prioritása jellemzi annak fontosságát. A fontosság megállapítása az utazó döntése.

A tevékenységek prioritása a következő értékeket veheti fel:

1: fix, amely rendszeres, az előre megadott helyen, és időablakban. A legkorábbi kezdés és a legkésőbbi végzés időpontja adott, illetőleg az igénypont fix és a látogatás az utazó döntése.

2: térben rugalmas, amely azt jelenti, hogy az igénypont időablaka fix, de a hely nem. Minden közeli igénypontot egy algoritmus keres meg, amelyek egy előre beállított távolságon belül vannak, és útba esik, és ugyanabba a POI kategóriába tartoznak. Ezeket az alternatív igénypontokat megkeressük, minden fix igénypont körül. A vizsgált alternatív igénypontok közül azokat vesszük figyelembe, amelyek átlagos távolsága kisebb, a fix ponthoz képest.

3: időben rugalmas, ahol a pozíció fix, de a bármikor mehetünk nap közben. Ekkor az időben rugalmasok közül csak azokat vesszük figyelembe, ahol a befejezési idő még megfelelő az utasnak.

4: teljesen rugalmas, ami azt jelenti, hogy tevékenységet eltolhatjuk egy másik napra, ha szükséges. Az algoritmus figyelembe veszi a preferenciákat. Ha a preferenciák nem teljesülnek, az algoritmus kihagyja a teljesen rugalmas tevékenységeket, és újraszámítja az egész napot. Iteratívan a teljesen rugalmas tevékenységek kihagyhatók, egészen addig, amíg az elvárások nem teljesülnek, vagy az összes teljesen rugalmas tevékenységet el nem hagytuk.

A helyváltoztatás 3 módon történhet, autó, közösségi közlekedés, és ezek kombinációja, azaz közösségi közlekedés és car-sharing rendszer. A parkolási adattárház, csak az autós közlekedés esetén használandó, ezért most a közösségi közlekedés, és a kombinált móddal nem foglalkozom.

Autós mód:

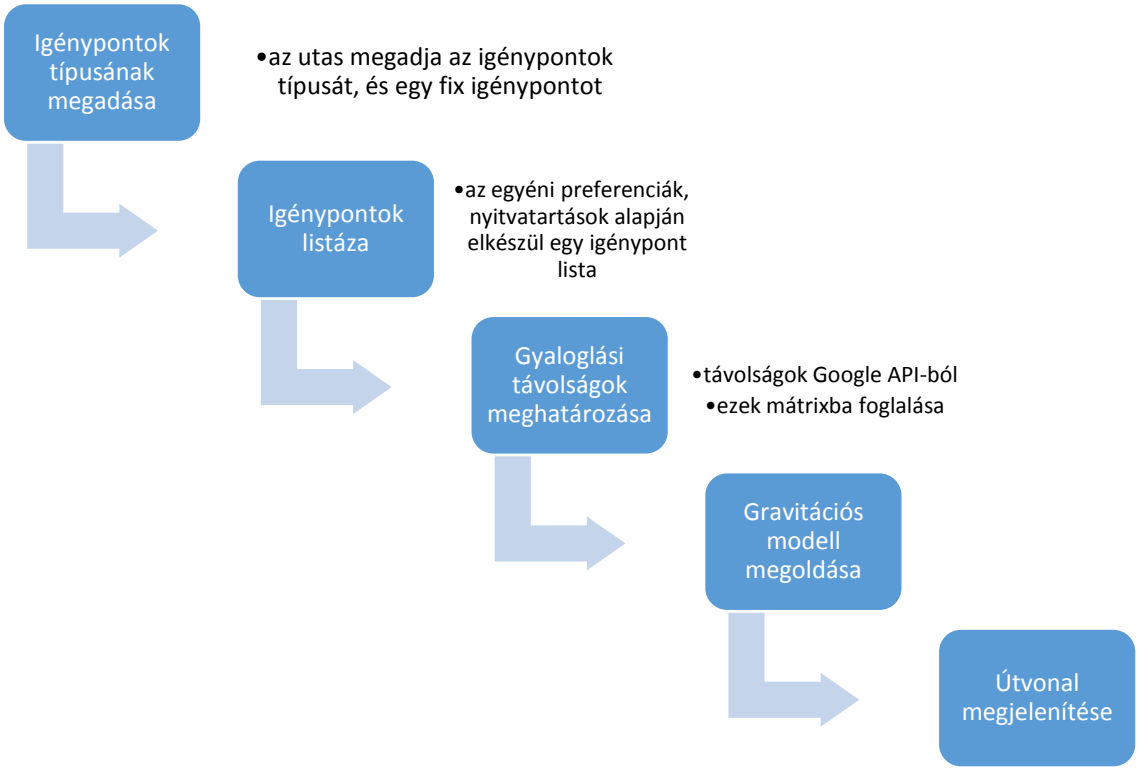
Az autós utazási időt a Google API alapján veszi figyelembe a modell, ahol az átlagos gyaloglási idő paraméterként 5 perc az adott parkolóhelyig. Azonban, ez egy átlagos értéket ad, amelyik nem speciálisan az adott útvonalhoz tartoznak, az adatbázisból lekért információk alapján a parkolóháztól történő gyaloglás is tervezhetővé válik.

Ebben az utazástervezési módban használható az általam elképzelt parkolási adattárház. Jelenleg a modell nem tervez a parkolóhely kereséssel, csak egy 5-perces időtartamot vesz figyelembe a parkolóhely és a felkeresendő cím között, és ezt, mint egy időtöbbletet adja hozzá az utazási időhöz. Ez tulajdonképpen egy statikus

pontatlansága az algoritmusnak, hiszen a parkoló kereséssel töltött idő ettől jelentősen eltérhet. A parkolási adattárház nyújtotta többletinformációval a napi utazási lánc optimalizálásánál már a parkolás is tervezhetővé válik, és ez a tényező akár dönthet is arról, hogy közösségi közlekedést vagy autós eljutást választunk.

A lényeg, hogy egy utas által megadott igénypont keresése esetén annak közelében szükséges keresni egy szabad parkolóhelyet, majd így meghatározni az utazási időket. A sorrend egy újabb korlátozó tényező a modellben, ugyanis az igénypont közelében először a parkolóhelyet kell biztosítani, majd aztán az igénypontra menni, majd a tevékenység végeztével, nem az igényponttól folytatni az utazástervezést, hanem a parkolótól.

A metodika akár meg is fordítható, azaz egy megfelelően kiválasztott parkolóház közelében kereshetünk alkalmas igénypontokat. A megfelelő parkolóház kiválasztása a POI táblából történik, úgy hogy legalább egy fix pontnak kell szerepelni az utazó igényei között. A fix pont térben és időben rögzített, így ez már behatárolja a parkolóházak lehetséges halmazát a gyaloglási távolságon belül. Ezt a felhasználási mód az esetek kisebbik hányadában lehet célravezető, mégpedig azokban az esetekben, amikor a napi tevékenységi lánc nagy része, vagy teljes egésze nem rendszeres tevékenységből áll. Korábban definiáltuk, hogy ezek a nem rendszeres tevékenységek leggyakrabban a szabadidős és a szórakozási, esetleg ügyintézési tevékenységek. Tipikusan jó példa erre egy szórakozással eltöltött szombat délután. Az algoritmus működése a következő ábrán látható.



20. ábra: A parkoló választás folyamata

Készítettem egy esettanulmányt egy ilyen optimalizálási esetre.

Az utazó igényei:

- Karácsonyi vásár megtekintése a Vörösmarty téren (Fix igénypont)
- Mozi (rugalmas igénypont)
- Elit kávézó (rugalmas igénypont)
- Thai étterem (rugalmas igénypont)

A karácsonyi vásár már teljesen meghatározza a helyet, viszont a többi tevékenység teljesen rugalmas, így az algoritmus a POI táblában választja ki ezeket egy megadott távolságon belül. Mivel a hely viszonylag behatárolt, és a szombat délutánt nem szeretnénk utazással tölteni, ezért a modell úgy választ parkolóhelyet, és egyéb igénypontokat a Vörösmarty tér közelében, hogy mindegyik hely gyaloglási távolságon belül legyen. Ebben az esetben az optimalizálás egy gravitálási feladattal kezdődik. Tehát a megfelelő igénypontokat úgy kell kiválasztani, hogy elég közel essen a parkolóházhoz. Az algoritmus a POI adatbázisból kiválaszt néhány az előre definiált korlátoknak megfelelő helyet. Ezeknek a listája alább látható.

5. táblázat: A lehetséges igénypontok listája

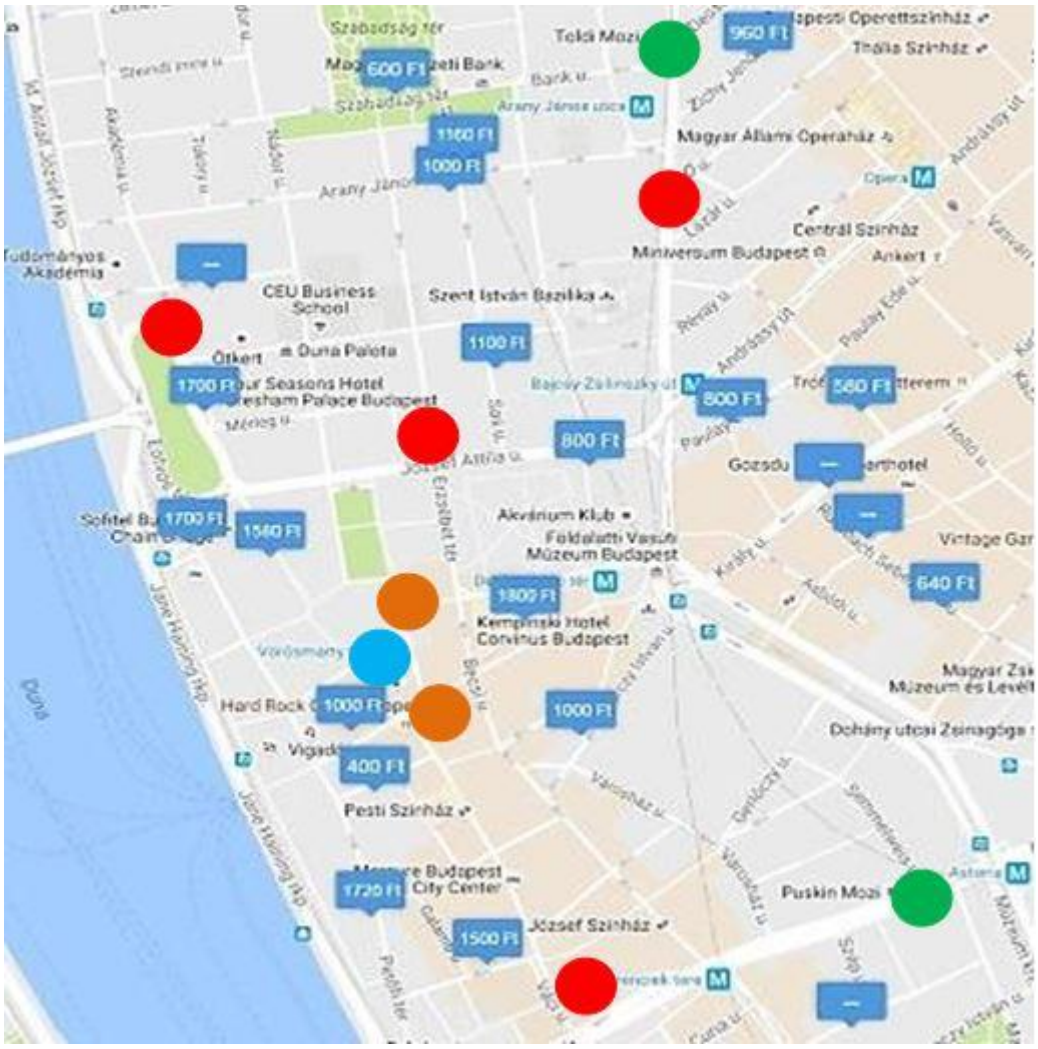
Igénypont neve	Igénypont címe	Igénypont típusa
Hard Rock Cafe Budapest	Budapest, Deák Ferenc u. 3, 1051	Elit kávézó
Gerbeaud Kávéház	Budapest, Vörösmarty tér 7-8, 1051	Elit kávézó
Vörösmarty tér	Budapest, Vörösmarty tér, 1051	Fix pont
Puskin Mozi	Budapest, Kossuth Lajos u. 18, 1053	Mozi
Toldi Mozi	Budapest, Bajcsy-Zsilinszky út 36-38, 1054	Mozi
Bangkok Thai Étterem	Budapest, Só u. 3, 1056	Thai étterem
padthai wokbar	Budapest, Október 6. u. 4, 1051	Thai étterem
Bob Bár	Budapest, 1051, Széchenyi István tér 7, 1051	Thai étterem
Fancu APART	Budapest, Bajcsy-Zsilinszky út 19a, 1065	Thai étterem

A fix pont körül kiválaszt néhány gyaloglási távolságban elhelyezkedő parkolóházat listáz ki a rendszer. Ekkor még foglaltsági állapotot nem figyeli.

6. táblázat: A lehetséges parkolóházak

Parkolóház neve	**Parkolóház Címe**
Piarista utca 4. parkoló	**Budapest,** Piarista u. 4, 1052
Mariott garázs	Budapest, Apáczai Csere János u. 2, 1051
Aranykéz Parkolóház	Budapest, Aranykéz utca 6, 1052
Vörösmarty Garázs	Budapest Deák Ferenc utca 4, 1052
Szervita Parkolóház	Budapest, Szervita tér 8, 1052 Magyarország
Kempinski Hotel Mélygarázs	Budapest, Miatyánk u., 1051 Magyarország
Mélygarázs Budapest Erzsébet tér	Budapest, Erzsébet tér, Mélygarázs, 1051 Magyarország
Gresham Garage	Budapest, Széchenyi István tér 5-6, 1051 Magyarország
Bazilika Mélygarázs Budapest	Budapest, Sas u. 3, 1051 Magyarország
Sas utca	Budapest Sas utca 27, 1051
Lipót Garázs	Budapest, Szabadság tér, 1054 Magyarország

A piros színek jelölik az éttermeket, a zöldek a mozikat, a barna pedig a kávézókat. A kék villám alakú jel a Vörösmarty teret, a kék téglalapok pedig a parkolóházakat.



21. ábra: A lehetséges igénypontok térképen

Első ránézésre több parkolóház is jónak tűnik, azonban az algoritmussal megkeressük a legjobbat. Ebben a szimulációban egy gravitálási feladattal oldottam meg, azaz az kilistázott igénypontok távolságát lemértem minden parkolóháztól. Ezeket a távolságokat, pedig igénypont típusonként egy mátrixba foglaltam. Az eljárás célja az, hogy kiválasszuk a legmegfelelőbb parkolóházat, és az igénypontokat, az optimalizálási cél pedig a gyaloglási távolságok minimalizálása.

Gyaloglási távolságok meghatározása:A gyaloglási távolságok meghatározásához Microsoft Excel Makrót írtam. A Google API minden platform felé nyilvános, és ingyenes, lekérdezési lehetőséget biztosít a különböző adatok kinyeréséhez. A gyaloglási távolságokat egy egyszerű html web request-el kérdeztem le a Google API-ból. A forráskódja alább látható.

```
'Calculate Google Maps distance between two addresses
```

```
Public Function GetDistance(start As String, dest As String)
   Dim firstVal As String, secondVal As String, lastVal As String
   firstVal = "http://maps.googleapis.com/maps/api/distancematrix/json?origins="
   secondVal = "&destinations="
   lastVal = "&mode=walking&language=pl&sensor=false"

   Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")
   URL = firstVal & Replace(start, " ", "+") & secondVal & Replace(dest, " ", "+")
& lastVal
   objHTTP.Open "GET", URL, False
   objHTTP.setRequestHeader "User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.0)"
   objHTTP.Send ("")
   If InStr(objHTTP.ResponseText, """distance"" : {") = 0 Then GoTo ErrorHandl
   Set regex = CreateObject("VBScript.RegExp"): regex.Pattern = """value"".*?([0-
9]+)": regex.Global = False
   Set matches = regex.Execute(objHTTP.ResponseText)
   tmpVal = Replace(matches(0).SubMatches(0), ".",
Application.International(xlListSeparator))
   GetDistance = CDbl(tmpVal)
   Exit Function

ErrorHandl:
   GetDistance = "ERROR"
End Function
```

A Public Function egy minden munkafüzet számára elérhető függvény definiálását jelenti, majd a zárójelben megadjuk a függvény paramétereit, ezek a címek.

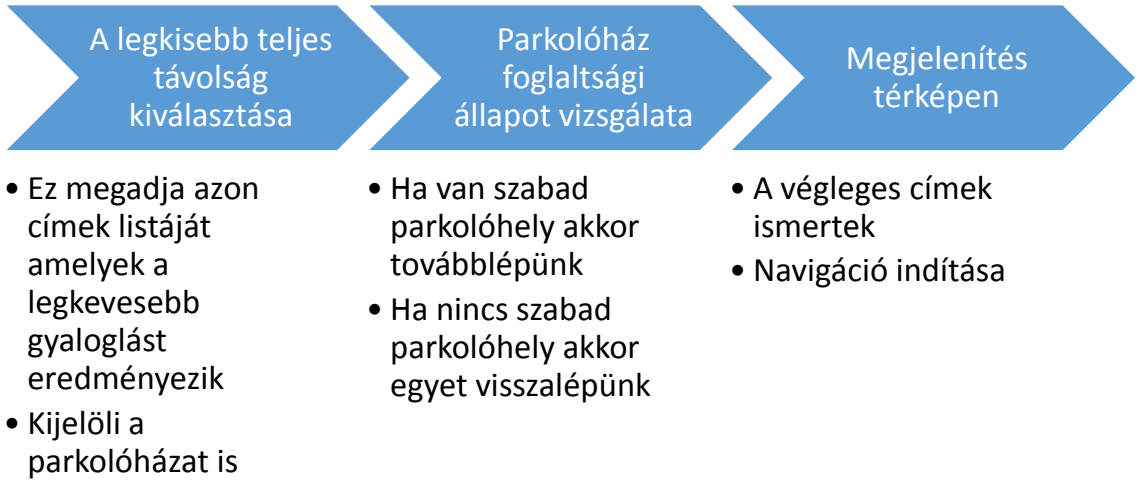
A firstVal, secondVal, lastVal a lekérdezés paramétereit tartalmazó változók, a lekérdezési URL, az irány, és az eljutási mód. A lastVal paramétert szükséges állítani autós ("&mode=car&), vagy gyalogos ("&mode=walking&) távolságok méréséhez.

A következő blokkban definiáljuk a http objektumot, amely megvalósitja az URL alapú lekérdezést.

Az utolsó blokkban pedig egy hibakezelési ág van. Az ErrorHandl(er) címkéhez akkor ugrik a program, ha a lekérdezés eredményére üres tartalom, vagy 0 érték tért vissza.

Ez a függvény két pont közötti átlagos gyaloglási távolságot adja eredményül. Az gyaloglási távolságokat tartalmazó költségmátrix alább látható. A sorokban a típusonként csoportosított igénypontok, míg oszloponként a parkolóházak vannak. A mátrix elemei a megadott sor és oszlopban lévő címek közötti gyaloglási távolságok. A cél az, hogy olyan parkolót és igénypontot válasszunk, amely a legkevesebb gyaloglást eredményezi. Tehát az oszlopokban szereplő értékeket kell megfelelően összegezni. Ez azt jelenti, hogy az igénypontok típusainál a minimális távolságokat kell kiválasztani, majd ezeket összegezni, így a **teljes távolság** sorban a legkisebb érték lesz az optimális garázs a távolságok szerint.

Optimalizálás:



22. ábra: Az optimalizálási folyamat

A szürke hátterű sorban a **816 perc**es érték a legkisebb. Ehhez az oszlophoz a **Kempinski Hotel Mélygarázs** tartozik. A garázst már tudjuk. Most kell megvizsgálni a parkolási adattárházban, hogy van-e szabad hely az adott garázsban, ha van, akkor ez megfelelő. Ha nincs szabad hely, akkor a teljes távolság sorban a következő legkisebb érték határozza meg, a következő lehetséges garázst. Ha sikerült megtalálni egy szabad garázst, akkor a hozzá tartozó igénypontokat kell kiválasztani, tehát az igénypontok típusa szerint a legközelebbit.

Ebben a példában a Hard Rock Cafe, a Vörösmarty tér, a Puskin Mozi, és a Padthai wokbar került kiválasztásra. Ezután pedig a kiválasztott címeket megjelenítjük térképen.

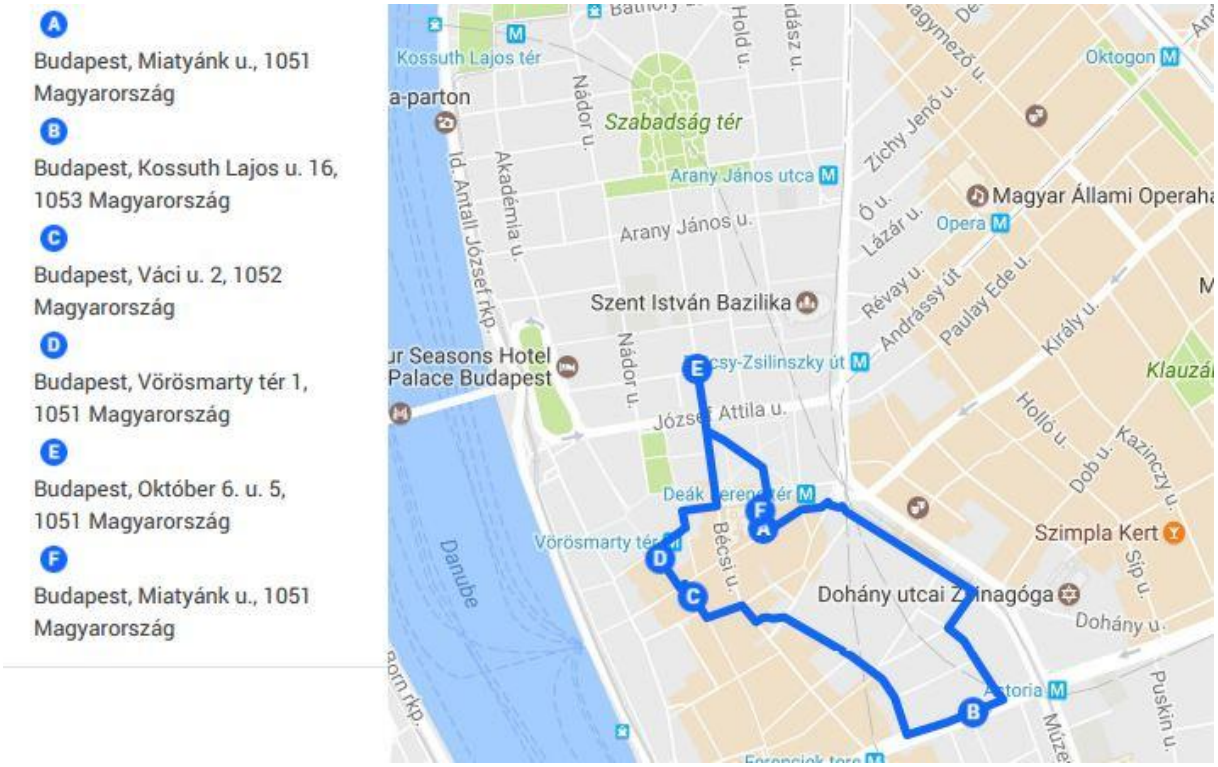
7. táblázat: Az igénypontok és a parkolóházak gravitációs mátrixa

		Piarista utca 4. parkoló	Mariott garázs	Aranykéz Parkolóház	Vörösmarty Garázs	Szervita Parkolóház	Kempinski Hotel Mélygarázs	Mélygarázs Budapest Erzsébet tér	Gresham Garage	Bazilika Mélygarázs Budapest	Sas utca	Lipót Garázs
Hard Rock Cafe Budapest	**Elit kávézó**	622	541	354	386	235	143	163	742	383	690	918
Gerbeaud Kávéház	**Elit kávézó**	540	413	242	195	330	310	351	641	605	912	1064
Vörösmarty tér	**Fix pont**	506	379	208	162	297	299	341	607	594	901	1054
Puskin Mozi	**Mozi**	536	785	724	824	531	795	810	1368	1030	13337	1550
Toldi Mozi	**Mozi**	1357	1364	1151	1126	1015	866	825	931	687	381	423
Bangkok Thai Étterem	**Thai étterem**	805	1036	1095	1195	1082	1310	1352	1814	1603	19100	2062
Padthai wokbar	**Thai étterem**	873	790	578	553	522	374	333	366	235	341	492
Bob Bár	**Thai étterem**	1055	837	752	667	873	737	696	40	585	512	662
Fancu APART	**Thai étterem**	1149	1156	943	919	807	658	617	871	532	321	549
	Min Elit kávézó	540	413	242	195	235	143	163	641	383	690	918
	Min Fix pont	506	379	208	162	297	299	341	607	594	901	1054
	Min Thai étterem	805	790	578	553	522	374	333	40	235	321	492
	Teljes távolság	1851	1582	1028	910	1054	816	837	1288	1212	19112	2464

A megfelelően kiválasztott címek a következő táblázatban és képen láthatóak.

8. táblázat: Az optimalizálás eredménye, a címek, amelyet a szoftver javasol

			Kempinski Hotel Mélygarázs
Név	Cím	Igénypont típus	Budapest, Miatyánk u., 1051
Hard Rock Cafe Budapest	Budapest, Deák Ferenc u. 3, 1051	Elit kávézó	143 m
Vörösmarty tér	Budapest, Vörösmarty tér, 1051	Fix pont	299 m
Puskin Mozi	Budapest, Kossuth Lajos u. 18, 1053	Mozi	795 m
Padthai wokbar	Budapest, Október 6. u. 4, 1051	Thai étterem	374 m



23. ábra: Az optimalizált címek térképes megjelenítése

6. Összegzés

A dolgozatban igyekeztem választ találni a parkolási problémák eredetére, számba vettem a jelenlegi eszközöket, rendszereket, módszereket, szoftvereket, amelyekkel a probléma kezelhető. Megvizsgáltam a piacon jelen lévő szoftvereket, módszereket, ötleteket, és rendszereket. Megállapítottam, hogy több egészen kiforrott rendszer van, amely hatékony, felhasználóbarát és alkalmas a parkolás menedzselésére.

Megalkottam egy olyan parkolási adatlekérdező applikáció logikai alapjait, amely az utas egyéni igényinek megfelelően lekérdezi a parkolóházak szabadhely adatait. Ebből az utas egyéni döntése szerint választhat, majd oda navigálhat. A járművezető számára a tudatos útvonaltervezés lehetősége adódik meg oly módon, hogy forgalmi torlódásokat elkerülje, a parkolóhely keresést megszűntetve a felkeresendő cím közvetlen közelében lévő biztosan szabad parkolóhelyre navigálja. Továbbá megfogalmaztam az igényt arra, hogy olyan integrált rendszereket építsünk, amelyek több rendszer összehangolt működéséből egy egységes egyrészt teremt.

Javaslatot tettem a közlekedők napi tevékenységi láncának az optimalizálási eljárásának a fejlesztésére. Egy korábbi algoritmus képes volt rá, hogy az utazó egyéni preferenciái szerint optimális napi tevékenységet konstruáljon. Ez megválasztotta az utazónak szükséges címeket, eljutási módokat, úgy hogy a napját a legrövidebb idő alatt befejezhesse, miközben minden feladatát elvégezte. Ezt a rendszert kiegészítettem a parkolási adatok kezelésével. Ezáltal a napi feladatok szervezésénél az algoritmus a parkolással is képes tervezi, ami pontosabb előrebecslést eredményez. Korábbiakban az algoritmus csak egy statikus időtöbblettel vette figyelembe a parkolást, így viszont valós idejű kalkulációt sikerült készíteni.

Továbbá a szervező algoritmus, egy másik szervezési metodikáját is kidolgoztam, az úgynevezett fordított optimalizálást. Ekkor a parkolóházhoz közel választunk címeket, és gyaloglással közlekedünk. Erre az esetre esettanulmányt készítettem, a megválasztott preferenciák alapján egy minta adatbázist csináltam, majd ezek távolságait a Google API-ból lekérdeztem, felépítettem a gyaloglási mátrixot, majd gravitálással összerendeltem az optimális parkolóházat és a megfelelő igénypontokat, majd ezt térképen megjelenítettem.

7. Felhasznált irodalom

[1] **Közlekedési Hálózattervezés előadási diasor** – Dr. Tóth János, 2013

[2] **A hagyományos várostól a regionális városig** - Batár Attila, http://www.c3.hu/~eufuzetek/index_2021.php?nagyra=konyvespolc/7batar.html Letöltve: 2016:10.31.

[3] **Future Internet and Smart Cities, avagy a jövő internete és az okos városok** KOVÁCS KÁLMÁN, BAKONYI PÉTER, Budapesti Műszaki és Gazdaságtudományi Egyetem, Egyesült Innovációs és Tudásközpont

[4] **Balázs Mór Terv** https://www.bkk.hu/wp-content/uploads/2014/06/BMT.pdf Letöltve: 2016.10.31.

[5] **Városi és Térségi Közlekedés** – Holló Péter, K. G. G. É. P. J., 2000

[6] **Definition of Smart Parking** http://www.pcmag.com/encyclopedia/term/65086/smart-parking Letöltve: 2016.10.31.

[7] http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011 0085_logisztikai_alapismeretek/ch01s07.html Letöltve: 2016.10.25.

[8] http://www.kti.hu/index.php?mact=Album,m5,default,1&m5albumid=123&m5page=3&m5returnid=503#link

[9] http://nhts.ornl.gov/tables09/fatcat/2009/avo_TRPTRANS_WHYTRP1S.html Letöltve: 2016:10.31.

[10] http://szolnokinaplo.hu/ckfinder/userfiles/images/f.jpeg

[11] http://index.hu/belfold/budapest/2016/10/24/budapest_egyre_nagyobb_dugokra_szamithat/ Letöltve: 2016.10.25

[12] http://www.bkk.hu/parkolas/pr-parkoloink/kokiterminal/ Letöltve: 2016.10.08.

[13] http://www.bkk.hu/parkolas/kozteruleti-parkoloink/varakozasi-hozzajarulasok/ Letöltve: 2016.10.08.

[14] **Részlet a 253/1997. (XII. 20.) Korm. rendeletből**
http://net.jogtar.hu/jr/gen/hjegy_doc.cgi?docid=99700253.KOR
Letöltve: 2016.10.08.
[15] http://imagazin.hu/apple-maps-parkopedia/
Letöltve: 2016.10.09.
[16] http://szentendre.hu/okos-parkolas-szentendren-a-zte-mintaprojektje/
Letöltve: 2016.10.09.
[17] http://budapestkozut.hu/kozut-figyelo
Letöltve: 2016.11.27.
[18] http://www.forbes.com/sites/emmajohnson/2014/12/18/5-parking-apps-that-help-you-save-time-and-money/#46a1e2154c0c
Letöltve: 2016.12.15.
[19] https://hu.wikipedia.org/wiki/String
Letöltve: 2016.10.30.
[20] https://hu.wikipedia.org/wiki/Adatb%C3%A1zis-tervez%C3%A9s
Letöltve:2016.10.30.
[21] https://hu.wikipedia.org/wiki/Webszolg%C3%A1ltat%C3%A1s
Letöltve: 2016.10.30.
[22] https://en.wikipedia.org/wiki/Request%E2%80%93response
Letöltve: 2016.10.30.
[23] http://www.w3c.hu/forditasok/XML_10_pontban.html
Letöltve: 2016.10.30.
[24] https://hu.wikipedia.org/wiki/XML
Letöltve: 2016.10.30.
[25] http://www.tilb.sze.hu/tilb/targyak/NGB_TA027_1/SQL.pdf
Letöltve: 2016.10.30.
[26] https://hu.wikipedia.org/wiki/MySQL
Letöltve: 2016.10.30.
[27] http://people.inf.elte.hu/molnarba/Informaciorendszerek_ELTE/Use_case/he.pdf
Letöltve: 2016.12.12.
[28] https://pcforum.hu/szotar/?term=backend
Letöltve: 2016.12.12.
[29] https://pcforum.hu/szotar/front-end
Letöltve: 2016.12.12.

[30] https://www.google.hu/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=frontend%20backend
Letöltve: 2016.12.12.

[31] https://hu.wikipedia.org/wiki/T%C3%B6bbr%C3%A9teg%C5%B1_architecht%C3%BAra
Letöltve: 2016.12.12.

[32] https://hu.wikipedia.org/wiki/Megjelen%C3%ADt%C3%A9si_r%C3%A9teg
Letöltve: 2016.12.12.

[33] http://www.szak.hu/konyvek_htm/sample_chapters/java/chap1.pdf
Letöltve: 2016.12.12.

[34] https://hu.wikipedia.org/wiki/Reszponz%C3%ADv_weboldal
Letöltve: 2016.12.12.

[35] http://www.hwsw.hu/hirek/55033/nativ-mobile-web-fejlesztes-app-amp-service-worker.html
Letöltve: 2016.12.12.

8. Ábrajegyzék

9. Táblázatjegyzék