Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

TDK Dolgozat

TÖMEGKÖZLEKEDÉST SEGÍTŐ RENDSZER MOBIL ESZKÖZÖKRE LÁTÁSSÉRÜLT ÉS VAK FELHASZNÁLÓK RÉSZÉRE

Készítette:
Decsi István

KONZULENSEK

Dr. Németh Géza, Tóth Bálint Pál

BUDAPEST, 2011

Tartalomjegyzék

Kivonat

Napjainkra az informatika már túlnőtt a helyhez kötött, asztali számítógépek által kínált lehetőségeken. A mobil eszközök és hálózatok fejlődésével az élet számos területén jelentek meg új informatikai megoldások. Erre a fejlődésre jellemző a különböző eszközök és szolgáltatások integrálódása. Míg korábban sok feladathoz – helymeghatározás, vezeték nélküli Internet elérés, fényképezés, média lejátszás – specifikus eszközökre volt szükség, addig ma már egyetlen mobiltelefonban is megtalálhatjuk mindezeket. Ezek közül a helymeghatározás kap fontos szerepet jelen dolgozatomban.

Számos lehetőség létezik aktuális pozíciónk meghatározására. A legismertebb talán a műholdas helymeghatározás, vagyis a GPS (Global Positioning System) használata. Ez a legtöbb helyzetben megfelelő pontosságú adatokkal szolgál, azonban vannak kivételek. Zárt térben például nem használható az épületek árnyékoló hatása miatt. Ilyen területeken lehetőség van WIFI alapú helymeghatározásra, illetve a mobil hálózat cellainformációi alapján is megállapíthatjuk helyzetünket. A mai mobil eszközök már képesek ilyen technológiák felhasználására. Ennek eredményeképp egy új szolgáltatáscsoport jött létre, melyeket hely alapú szolgáltatásoknak nevezünk (Location Based Services, LBS).

Jelen dolgozatban egy speciális hely alapú szolgáltatást biztosító rendszer készítését szeretném bemutatni. A rendszer egy gyalogosok számára készített navigációs alkalmazás, melyet Android platformon valósítok meg. Az alkalmazás segítségével a felhasználók bárhol, bármikor képesek meghatározni egy tetszőleges pontra vezető legkedvezőbb tömegközlekedési útvonalat. Emellett a teljes tömegközlekedési menetrendet is elérhetik és kereshetnek benne. A kutatás és fejlesztés során különös figyelmet fordítottam arra, hogy mindenki számára kényelmesen használható rendszert készítsek. A manapság divatos érintőképernyős eszközök nem nyújtanak minden felhasználói csoport számára megfelelő kezelőfelületet. Látássérült és vak felhasználók számára a fizikai billentyűzettel ellátott készülékek használata könnyen elsajátítható, hiszen a tapintásukkal tudtak tájékozódni a készüléken, érintőképernyő esetén azonban ez számos nehézséget vet fel.

Az általam készített alkalmazásban egy, a valós életben is jól alkalmazható megoldást mutatok be erre a problémára. A specifikus elméleti megoldás alapján egy általános javaslatot teszek az érintőképernyők használhatóvá tételére vak és gyengén látó felhasználók részére. Ennek a megoldásnak az alapja egy XML leírásra támaszkodó multimodális felhasználói felület. Ez egyrészt grafikus felületbe integrált beszédinterfészt nyújt, másrészt pedig a grafikus elemek használatát biztosítja egy új megközelítéssel (a rendszer felhasználja a készülék különböző modalitásait: tapintással és hallással is érzékelhetővé teszi az érintőképernyő grafikus elemeit). Mobil környezetben mindenki számára hasznos lehet a beszédinterfész alkalmazása, hiszen nem kell minden esetben a készülék kijelzőjére figyelni, elég lehet egy headset is az alkalmazások kezeléséhez.

Abstract

Today information technology has expanded beyond the possibilities offered by the stationary, desktop computers. Thanks for the development of mobile devices and networks new IT solutions have appeared in several new areas of life. This development is characterized by integration of different tools and services. While many earlier jobs - positioning, wireless Internet access, camera, media playback – needed specific tools, now you can find all of these in most of mobile phones. Among these, the positioning will get an important role in my thesis.

There are several options for determining your current position. Maybe the best known is using satellite positioning, called GPS (Global Positioning System). It provides sufficiently accurate data for most situations but it is not the best option in all circumstances. It is not usable in closed areas because of the shielding effect of buildings. In such areas it is possible to use Wi-Fi networks and mobile network cell information to be able to determine our position. Today's mobile devices can use such technologies. As a result, a new group of services has been created, called location-based services (LBS). In these services it is crucial to determine the position of users and use it in some way.

In the present study I would like to introduce the production of a system providing a special location-based service. This system is a pedestrian navigation application, designed for the Android platform. The application allows users to determine the leading public transport route to anywhere. In addition, the entire public transport timetables can also be accessed and searched. During the research and development, special attention was paid to the fact to create a system, which is comfortable for everyone. The currently fashionable touch-screen devices do not provide suitable user interface for all groups of users. Blind and visually impaired users can easy learn using devices with a physical keyboard because they can feel the buttons; however it presents several problems on touch screen.

In my system I present a solution for this problem that is well-suited in real life. I propose a specific solution based on a general theory to let blind and visually impaired users to use touch screen. This solution is based on an XML description based multi-modal user interface. On the one hand this is a speech user interface (SUI) integrated in graphical

user interface (GUI), on the other hand, it means a new approach of the use of graphical items (the system uses the devices' different modalities: it lets you feel the touch-screen graphic elements using touch and hearing). In mobile environment, everyone can benefit from the use of speech user interfaces. You do not need to always look on the display of your device; a headset is quite possible to manage applications.

1 Bevezetés

Ma már nem egyértelmű, hogy a személyi számítógép a legfontosabb informatikai eszköz. A mobilitás egyre több ember számára lett fontos tényező. Ez a tendencia évek óta megfigyelhető a vásárlók körében, és ezt a gyártók is észrevették. A laptopok mellett megjelentek a netbookok, táblagépek és okostelefonok, bár ez utóbbi két kategória között egyre nehezebb meghúzni a határt. Hiszen a telefonok teljesítménye is rohamosan növekszik, egyre nagyobb kijelzővel rendelkeznek, sok esetben a táblagépekhez hasonló operációs rendszert futtatnak (Android, iOS), a táblagépek pedig ugyanúgy képesek mobil hálózatok elérésére, mint a telefonok.

Ennek a két eszközcsoportnak köszönhetően komoly átalakuláson mentek át a felhasználói felületek és az ehhez kapcsolódó perifériák. A gyártók igyekeznek minden eszközre a lehető legnagyobb kijelzőt helyezni, ez pedig sok esetben kiszorítja a korábban megszokott billentyűzeteket, hogy a készülékek használható méretűek maradjanak. Ennek eredményeképp ma már több érintőképernyős telefon és táblagép létezik, mint amennyit fizikai billentyűzettel is ellátnak.

Ezek az eszközök azonban nem felelnek meg minden felhasználói csoportnak. Dolgozatomban egy olyan, Android operációs rendszeren működő navigációs rendszer fejlesztését szeretném bemutatni, mely vak és látássérült felhasználók által is könnyen kezelhető felülettel rendelkezik. Egy ilyen, speciális igényeket kielégítő alkalmazás fejlesztése már önmagában is nagy kihívást jelent, hiszen teljesen újra kell gondolni a felhasználói felület kialakítását és annak felhasználását, hiszen vak felhasználók esetében nem triviális az érintőképernyők kezelése. Az alkalmazásnak minden pillanatban a lehető legtöbb információt kell eljuttatnia a felhasználóhoz, minden lehetséges eszközt felhasználva ehhez. Mindezt anélkül, hogy a legfőbb tájékoztató eszközt, a képernyőt nem használhatjuk információ továbbítására.

A dolgozat első részében megvizsgálom, hogy milyen technológiák léteznek ilyen felhasználói felületek kialakítására. Emellett áttekintem, hogy milyen navigációs és egyéb közlekedést segítő alkalmazások léteznek Android platformon, illetve ezek felhasználói felülete mennyire felel meg vak és látássérült felhasználóknak. Kitérek továbbá az Android

platformon elérhető multimodális lehetőségekre, melyek szintén segíthetik a hatékony kezelőfelület kialakítását.

Ezt követően bemutatom a tervezés folyamatát. Ezen belül ismertetem a rendszer architektúrális tervét, valamint az egyes funkciók részletes tervezését. Külön kitérek a felhasználói felület tervezésének szempontjaira is.

Ezután ismertetem magát a fejlesztési folyamatot, az egyes komponensek elkészítésének lépéseit, illetve a fejlesztés közben felmerült problémákat és azok megoldását.

Végül összefoglalom tapasztalataimat és felvázolok néhány további fejlesztési lehetőséget a témával kapcsolatban.

1.1 Problémafelvetés

Napjainkban egyre inkább előtérbe került a kérdés, hogy vajon mindenkinek megfelelnek-e az érintőképernyővel felszerelt mobil eszközök. A válasz természetesen nem. Vak és látássérült felhasználók számára korábban a billentyűk tapintása jelentette a lehetőséget mobil készülékeik kezelésére. Érintőképernyő esetén azonban ez a lehetőség hiányzik. A probléma jelentőségét mutatja, hogy több helyen is folynak a megoldására irányuló kutatások.

A Stanford Egyetem diákjai például kifejlesztettek egy alkalmazást, amellyel vak felhasználók is képesek szöveget gépelni táblagépen [4]. A megoldás ötletét a Braille írásra [13] alkalmas írógépek működése adta. Ezek lényege, hogy néhány, könnyen elkülöníthető gomb segítségével lehet gépelni. Az egyes karakterek leírása a billentyűk különböző kombinációjú lenyomásával történik. Érintőképernyő esetén azonban hiányzik a fizikai gombok tapintásának lehetősége. Ezt a problémát úgy oldották meg, hogy a felhasználó bárhol érinti meg a felületet, az ujjai alatt fognak megjelenni a gombok, melyekkel gépelhet. Innentől kezdve pedig minden egyes karaktert visszaolvas a program, a hibák elkerülése érdekében.

A menürendszerben történő navigálást szintén úgy oldották meg, hogy nem kelljen tudnia a felhasználónak, hogy épp hol érinti meg a kijelzőt. A készülék megrázásával a főmenübe jutunk, ahol elég megérinteni a képernyőt, és az egyes menüpontok sorban

jönnek az ujjunk alá. Az alkalmazás jelenleg még csak prototípus, de nagyon ígéretesnek tűnnek a megoldásai.

Dolgozatomban egy hasonló elven működő felhasználói felület létrehozását ismertetem. Ennek a felületnek is az a célja, hogy a felhasználó mindig tisztában legyen azzal, hogy épp milyen elemet érint meg a képernyőn, azonban én a fordított megoldást alkalmaztam. Száműztem a felületekről a görgetést, fixen pozícionáltam minden grafikus elemet, így a rendszer egyértelműen meg tudja állapítani, hogy a felhasználó épp mihez ért hozzá. Erről pedig beszéddel és rezgéssel is képes visszajelzést adni. Az általam kidolgozott megoldás annyival előnyösebb, hogy itt a felhasználó az elemek közti szabad navigációnak köszönhetően sokkal inkább úgy érezheti, hogy kézben tartja a program irányítását, nincs kiszolgáltatva az alkalmazásban előre meghatározott kezelési forgatókönyveknek. Tulajdonképpen úgy is tekinthető ez a rendszer, mintha egy hagyományos grafikus felület mellé kapna valakit a felhasználó, aki minden lépését figyeli és tájékoztatja a lehetőségekről.

Természetesen vannak egyéb módszerek is, melyekkel vakok és látássérültek is használhatnak mobil eszközöket. Például létezik gesztus alapú vezérlés is, ahol a képernyőn akár több ujjal végzett mozgások bírnak egyedi jelentéssel. Egy ilyen megoldást dolgoztak ki a Seattle-i Washington Egyetemen [14].

1.2 Korábbi megoldások

Ebben a fejezetben röviden bemutatok néhány, már működő alkalmazást, melyek valamilyen formában képesek megkönnyíteni a tömegközlekedési eszközök használatát. Ezeknek az alkalmazásoknak egy része csupán menetrendi adatokat biztosít, néhány viszont navigációs szolgáltatást is biztosít a felhasználóknak.

1.2.1 SmartCity

Ez egy magyar fejlesztésű alkalmazás, mely elérhető Android és iPhone platformokon. Az aktuális BKV és MÁV menetrendek alapján lehet vele útvonalat tervezni Budapesten. A megtervezett útvonalakat Google Maps-en jeleníti meg, illetve szöveges leírást is készít hozzájuk. A megjelenített útvonalak azonban nem teljesen pontosak, nem

követik végig az egyes járatok útvonalát, csak a megállókat kötik össze. Emellett böngészhetünk is az aktuális menetrendekben.

Ebből az alkalmazásból sem maradtak ki a manapság nagyon népszerű közösségi funkciók, például kommenteket lehet fűzni bizonyos helyekhez, melyeket meg lehet osztani ismerőseinkkel és persze mi is követhetjük mások írásait.

Felhasználói felülete igényesen megtervezett, könnyen kezelhető. A kellően nagyméretű grafikus elemeknek köszönhetően átlátható és könnyen használható az alkalmazás, azonban ezen kívül semmilyen más modalitást nem biztosít, hiányzik a beszédinterfész, ami megkönnyíthetné a vakok és látássérültek számára a használatot.

Hiányossága, hogy a menetrendek nem érhetők el offline módban, azonban ez is szerepel a fejlesztők későbbi tervei között, továbbá a több településre történő kiterjesztés is. További érdekesség az alkalmazással kapcsolatban, hogy a mobil kliens mellett webes felületen is kínál útvonaltervezési lehetőséget. Az alkalmazásról további információk az [1] forrásban találhatók.

1.2.2 Google Térkép Navigáció

Ez a Google Maps mobil készülékekre szánt hely alapú szolgáltatása. Meg tudjuk vele jeleníteni a saját pozíciónkat, üzleteket lehet vele keresni, követhetjük ismerőseink tartózkodási helyét is. Útvonaltervezést biztosít autósok és gyalogosok számára, akár tömegközlekedéssel is, illetve különböző térkép rétegeket tud megjeleníteni. Érdekessége, hogy képes forgalom információkat is megjeleníteni, ez segíthet az optimális útvonal kiválasztásában is.

A program felülete nagyon igényes, azonban vakok és látássérültek számára nem kimondottan alkalmas. Problémás lehet az egész képernyőt kitöltő térképen a navigálás, és meglehetősen kicsik a grafikus elemek is. Útvonaltervezés után lehetőség van szöveges és térképes navigációra is. Kiemelendő, hogy rendelkezik beszédinterfésszel, így navigáció során nem szükséges folyamatosan a képernyőt figyelni. A beszédinterfész azonban csupán a navigációs utasítások felolvasására terjed ki, így vakok és gyengén látok számára nem könnyíti meg az alkalmazás kezelését.

A következő platformokon érhető el a rendszer: Android, BlackBerry, iPhone, Nokia S60, Windows Mobile/Phone. További információk a [2] forrásban érhetők el.

1.2.3 NDrive

Az NDrive egy portugál vállalat, mely a világ szinten is ismert a navigációs szoftverek terén [3]. Termékeik között szerepel Android, Palm OS, bada, Windows Mobile, Symbian és iPhone platformra készített autós navigációs szoftver is. A mai trendeknek megfelelően ezek az alkalmazások is rendelkeznek közösségi és egyéb online szolgáltatásokkal. A navigációhoz használt térképek az alkalmazás részét képezik, vagyis közvetlenül a felhasználó eszközén tárolódnak, így nem szükséges adatforgalom a hétköznapi használathoz.

Felhasználói felülete szépen kidolgozott, a kényelmes használathoz megfelelően nagy grafikus elemeket alkalmaztak a fejlesztők. Navigáció során biztosít TTS (Text-to-Speech) funkciót a könnyebb használathoz, azonban ez sem a kimondottan a vak és látássérült felhasználókat célzó szolgáltatás, inkább csak egy kényelmi funkció.

1.2.4 BPMenetrend

Ez a program [8] kicsit kilóg az eddigiek közül, mivel nem alkalmas navigálásra. Azonban mégis érdemes említést tenni róla, hiszen lehetőséget az a BKV menetrendjének offline böngészésére. Az menetrendi adatokat a felhasználók kézzel frissíthetik. Jelenleg Android és iPhone platformokon érhető el. Funkciói között megtalálható az idő és megálló alapján történő szűrés, a kedvenc járatok külön mentése, illetve a következő járat megjelenítése egy adott pillanatban.

Összességében egy jól átgondolt böngésző alkalmazásról van szó, mely jó példaként szolgálhat az általam tervezett szoftver böngésző funkciójához. Felhasználói felület szépen kidolgozott, könnyen meg lehet keresni az egyes járatok adatait. Legnagyobb hiányossága, hogy nem lehet vele térképen megjeleníteni az egyes járatok útvonalait.

Vakok szempontjából a legnagyobb probléma a beszédinterfész teljes hiánya. A felhasználói felület kialakítása igényes, azonban nem elég kontrasztosak az egyes vezérlők, szinte összemosódnak a képernyőn, így egyáltalán nem könnyítik meg a kezelést semmilyen felhasználói csoport számára.

1.2.5 Budapesti Menetrend

Ez szintén egy menetrendböngésző alkalmazás, mely a BKV GTFS (General Transit Feed Specification) adatait használja fel. A kezdőoldalon megjeleníthetjük kedvenc járatainkat, megállóinak. Böngészhetünk és kereshetünk az összes járat és megálló vagy az aktuálisan legközelebbi megállók listájában. Egy kiválasztott megállóban megtaláljuk az összes onnan induló járat nevét és irányát. Egy kiválasztott járat esetén pedig megadhatjuk, hogy melyik megállóból melyik irányba közlekedő járművek indulási idejét szeretnénk listázni az aktuális időponttól vagy egy kiválasztott napon.

Az alkalmazás a BPMenetrendhez hasonlóan offline tárolja az adatbázist, azonban ez azt is jelenti, hogy a felhasználók csak frissítés után értesülhetnek a menetrendi változásokról. Sajnos ez az alkalmazás sem képes térképen megjeleníteni az útvonalakat.

Felhasználói felülete nem olyan kidolgozott, mint a BPMenetrendnek, nehézkes az adatok keresése, és nem mindig egyértelmű, hogy az alkalmazás mely részén járunk. A kevésbé átlátható felület és a beszédinterfész hiánya lehetetlenné teszi a vakok és látássérültek számára a használatot.

1.3 Multimodális lehetőségek Android platformon

1.3.1 Beépített TTS és beszédfelismerés

Az Android platform már az 1.6-os verziótól kezdve lehetőséget nyújt beszéd alapú multimodális alkalmazások készítésére, hiszen már ekkor tartalmazott TTS (Text-to-Speech) szolgáltatást [9]. Jelenleg öt nyelvet támogat a rendszer, ezek az angol – külön brit és amerikai akcentussal – francia, német, olasz és spanyol. Az egyes nyelvek megfelelő működéséhez különböző erőforrásokra, szótárakra van szükség, ez azonban nem minden készülékben található meg gyárilag. A TTS használatához tehát mindenképp meg kell róla bizonyosodni, hogy a készülékünk támogatja az adott nyelvet, illetve ha nem, akkor le kell tölteni a szükséges erőforrásokat. Ha ez sikerült és inicializáltuk a TTS szolgáltatást, máris egyszerű függvényhívásokkal használhatjuk azt. A paraméterezési lehetőségeknek hála lehetőségünk van soros, illetve megszakításos működés elérésére is, vagyis hogy egy még be nem fejezett felolvasást megszakíthasson egy újabb, vagy nem.

Jóval később, a 2.1-es rendszertől kezdve érhető el a hangvezérelt billentyűzet [10]. Ennek segítségével minden olyan alkalmazásban elérhető a beszédfelismerés, ahol szövegbevitelre van szükség. Még ez a szolgáltatás sem érhető el magyarul, a jelenleg támogatott nyelvek az angol, a kínai és a japán. A minél pontosabb beszédfelismeréshez két nyelvi modellt lehet beállítani. Egyik a természetes nyelvű szövegek felismerésére alkalmas, például levelek, feljegyzések készítésénél. A másik modell az egyedi kulcsszavakból álló keresésekre lett optimalizálva, ez utóbbit használja például a Google Voice Search szolgáltatása. A webes keresés mindhárom nyelven rendelkezésre áll, azonban a teljesen szabad szöveg modellje egyelőre angol nyelvre van optimalizálva.

A beszédfelismerés nagyon bonyolult, nagy számításigényű feladat. Különösen így, hogy nincsenek előre megadott kifejezések, hanem bármit fel kell ismernie a rendszernek. Ennek megvalósíthatósága érdekében nem is magán a telefonon történik a felismerés, hanem a Google szerverein, melyekre a telefon továbbítja a felismerendő hangot egy stream formájában.

1.3.2 SVOX TTS és ASR

A svájci székhelyű SVOX AG [5] komplett beszédinterfész megoldásokat kínál mobil és beágyazott környezetekben. Rendszereik jellemzője a hordozhatóság és a skálázhatóság. Ennek megfelelően a különböző teljesítményű mobil eszközökhöz különböző szintű ASR (Automatic Speech Recognition) és TTS megoldásokat kínálnak. Érdemes kiemelni, hogy a két komolyabb képességű TTS rendszerük támogatja a magyar nyelvet.

A céget 2011-ben felvásárolta az amerikai Nuance vállalat [6], mely így felhasználja saját termékeiben az SVOX által készített beszédinterfész megoldásokat.

1.3.3 Profivox

Ez a Távközlési és Médiainformatikai Tanszéken fejlesztett magyar szövegfelolvasó, mely működik Windows Mobile és Android platformokon is. A most fejlesztés alatt álló szoftverben is ezt alkalmazom a multimodális felhasználói felület kialakításához.

Az Android alapú verzió JNI-ben (Java Natív Interfész) került megvalósításra. Az egyszerű használat érdekében tartozik ehhez egy Java osztály, melyen keresztül magas szinten lehet vezérelni a működést. Ez az osztály pufferként is funkcionál, vagyis minden beérkező szöveget eltárol, és egymás után felolvassa őket. Ez azonban nem megfelelő, ha a felhasználói felülethez akarjuk kötni a TTS-t, hiszen előfordulhat, hogy gyorsan navigálva az eszközök között nagyon elcsúszna a felolvasás a program tényleges állapotától. Ennek megoldására úgy módosítottam még a korábbi félévekben ezt az interfész osztályt, hogy paraméterezni lehessen a kéréseket egy prioritással. Prioritásos kérés esetén minden korábbi kérést, melyek a várakozási sorban voltak egyszerűen eldob a felolvasó, és az új kérés kerül felolvasásra.

1.3.4 HMM alapú beszédszintetizátor

Dolgozatom készítése közben a Távközlési és Médiainformatikai Tanszéken folyó kutató és fejlesztő munka eredményeként létrejött az első magyar, HMM (Hidden Markov Model – Rejtett Markov Modell) alapú beszédszintetizátor, mely elérhető Android platformon [11].

A HMM technológiának köszönhetően kis méretű adatbázis alapján is képes jó minőségű, a természetest közelítő beszéd előállítására ez a szintetizátor. Ezzel kiküszöböli a legjobb minőségű beszédet előállító, korpusz alapú rendszerek egyik legnagyobb hibáját, a hatalmas, esetenként több gigabájt méretű adatbázis használatát. Továbbá a korpusz alapú rendszerekkel szemben ennek a szintetizátornak a hangkarakterisztikája és a beszédstílusa is könnyen állítható anélkül, hogy nagy mennyiségű új adatot kellene az adatbázisban elhelyezni. Már néhány percnyi hanganyag elég lehet ahhoz, hogy új beszédstílushoz adaptálódjon a rendszer.

Ez a szintetizátor csak az általam végzett fejlesztés utolsó fázisában vált csak elérhetővé, így én még a Profivox segítségével valósítottam meg a TTS funkciót. A későbbiekben érdemes lehet megvizsgálni a HMM alapú szintetizátor alkalmazásának lehetőségét.

1.3.5 Beszédfelismerő

Dolgozatom készítése közben elkészült egy magyar beszédfelismerő Android platformra a Távközlési és Médiainformatikai Tanszék fejlesztésében. Az alkalmazás egy parancsszó felismerő, mely képes előre megadott kifejezéseket felismerni. Még tesztelési fázisban van a program, így egyelőre nem kerül bele az általam készített navigációs alkalmazásba, de később mindenképp érdemes lesz megvizsgálni, hogyan lehetne vele bővíteni a beszédinterfészt.

Ez a beszédfelismerő mobil környezetben néhány száz szóig használható megfelelő sebességgel. Ez a mennyiség alkalmas lehet arra, hogy egyszerű utasításokkal vezéreljünk egy programot. A felismeréshez szükséges tanító adatbázis 500 ember telefonos bemondásaiból áll. Ez nagyjából 5-6 órányi általános telefonos beszédet jelent, vagyis nem egy megadott témára lett optimalizálva.

2 Tervezés

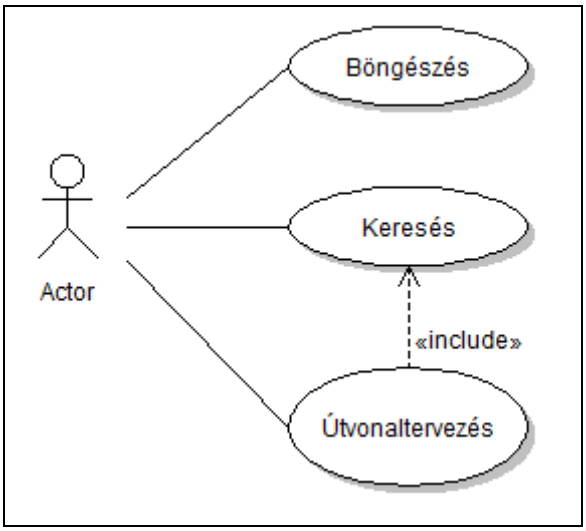
2.1 Használati esetek

Ahogy már az 1.1 fejezetben is említettem, a célom egy olyan navigációs szoftver készítése, mely megkönnyíti a gyalogosok közlekedését Budapesten. Egy ilyen alkalmazás tervezésénél nagyon fontos szempont, hogy egyszerűen kezelhető legyen, ne vesszen el a felhasználó a rengeteg funkció és beállítási lehetőség között. Hiszen az cél az, hogy megkönnyítse az utazást, nem pedig az, hogy egész úton a programra kelljen koncentrálni. Ennek érdekében csak olyan funkciókat valósítok meg, melyek valóban hasznosak lehetnek a mindennapokban. Így a felhasználó csak a tényleg szükséges beállításokhoz férhet majd hozzá. Úgy gondolom, hogy sokkal lényegesebb a megfelelően kidolgozott működés és a hatékony, egyszerű kezelhetőség, mint a rengeteg szempont szerinti beállítási lehetőség, hiszen a legtöbb felhasználó általában nem is változtatja meg a programok gyári beállításait.

Az egyszerű kezelhetőség mindenképpen fontos követelmény a vak és látássérült felhasználók szemszögéből nézve. Már a funkciók tervezése során arra kell törekedni, hogy egyszerűen kezelhető részekre bontsunk minden feladatot, ezáltal biztosítva az egyszerűbb kezelést. Ráadásul egyszerűbb lépésekhez a későbbiekben könnyebb lesz a felhasználói felületet is úgy létrehozni, hogy az könnyen kezelhető legyen. Ezeket a szempontokat figyelembe véve az alkalmazás a következő fő funkciókkal fog rendelkezni:

- BKV menetrendek böngészése
    - Járat, megállóhely és idő szerint
    - Járatok megjelenítése térképen és szövegesen
- Útvonaltervezés és navigálás
    - Aktuális pozíciótól
    - Felhasználó által megadott pontok között
    - Térképes és szöveges megjelenítés
- Keresés
    - Keresés a térképen cím vagy koordináta alapján

A fő funkciókat az 1. ábra szemlélteti. Ezek alapvetően online lesznek használhatók. Ennek oka egyrészt a nagy adatmennyiség, amit a menetrendek tárolása igényel. Igaz, hogy ma már a mobil készülékek is GB nagyságrendű tárhellyel rendelkeznek, azonban a felhasználók nem szívesen áldoznak fel több száz MB-ot egyetlen alkalmazásnak. A nagy adatmennyiség kezelése ráadásul komoly CPU teljesítményt igényel – keresések, útvonaltervezés – ami pedig a mobil eszközök készenléti idejét csökkentené drasztikusan. Vajon mennyire szívesen használna bárki is egy olyan programot, ami rövid idő alatt lemeríti a készüléket, ráadásul a tárhelyünk jelentős részét is elfoglalja? És végül, ha nem a készülékeken tároljuk az adatokat, akkor a frissítések kezelése is sokkal egyszerűbb, hiszen elég az adatokat nyújtó szerveren elvégezni a frissítést, a felhasználóknak pedig nem kell több száz MB-os állományokat letölteniük.

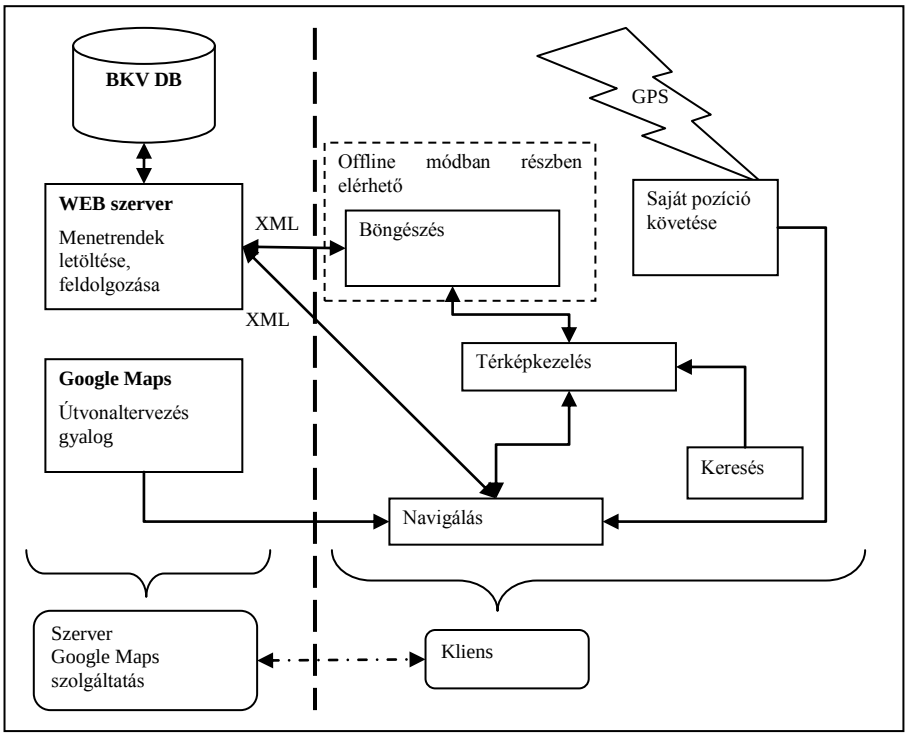


1. ábra - Fő funkciók

Természetesen bizonyos részfunkciók offline módban is elérhetők lesznek, például előre elkészített útvonalterveknek a mentése és későbbi visszatöltése. Ez lehetővé teszi, hogy ha a felhasználó nem szeretne a mobil internetet használni, akkor például az otthoni WiFi hálózaton megtervezheti az éppen szükséges útvonalakat, melyeket aztán később felhasználhat.

2.2 Rendszerterv, fő komponensek

Ahogy az előző fejezetben is leírtam, a rendszer alapvetően online működésű lesz, ami kézenfekvővé teszi a kliens-szerver architektúra alkalmazását. A menetrendi adatok egy MySQL adatbázis szerveren kerülnek tárolásra. Ehhez egy PHP-t futtató webszerver kapcsolódik, amihez HTTP kérésekkel férhet hozzá a kliens. A kliens kéréseire a szerver XML formátumú válaszokat küld az adatbázis alapján, például egy járat indulási időpontjait egy megállóból. Az adatok grafikus megjelenítését a térképkezelő modul fogja végezni. Ez kapcsolódik a menetrendböngészéshez, az útvonaltervezéshez és a keresés funkcióhoz is. A rendszerterv logikai felépítését a 2. ábra szemlélteti.



2. ábra - Rendszerterv

2.3 Funkciók részletes tervezése.

2.3.1 Menetrendek böngészése

Ennek a funkciónak a célja, hogy a felhasználó bármilyen BKV járat indulási adatait megtalálhassa. Mivel az egész menetrend adatbázis rengeteg adatot tartalmaz, meg

kell találni a módot, hogy a felhasználó kiszűrhesse a számára lényegeseket. Ráadásul olyan módon kell mindezt megoldani, hogy vak és látássérült felhasználók is könnyen eligazodjanak az egyes lépések között. Ennek érdekében úgy érdemes kialakítani ezt a funkciót, hogy minden egyes szűrési lépésnél előre megadott lehetőségek közül kelljen csak választania a felhasználónak. Minimalizálni kell az elvárt adatbevitelt. Ahol csak lehet, a képernyőn megjelenő gombokkal kell megoldani minden funkciót, melyekhez integrált beszédinterfész tartozik, így egyszerűsítve a használatot.

A böngészésre vonatkozó szűkítő feltételek megadásának tervezésekor a BKV weblapján található struktúrát vettem alapul. Ennek megfelelően a felhasználó először kiválaszthatja, hogy milyen járműtípus menetrendjére kíváncsi. Ezt követően lehet kiválasztani az adott típushoz tartozó viszonylatokat. A BKV weboldalán ezt követően már az adott viszonylat minden megállójához tartozó menetrendi adatok megjelennek egy PDF fájlban vagy HTML oldalon, ez azonban túl nagy adatmennyiség ahhoz, hogy egyszerre jelenítsük meg egy mobil kijelzőn. Ezért újabb szűkítési feltételekre van szükség. A felhasználónak lehetőséget kell adni, hogy kiválaszthassa, hogy a viszonylat mely megállójának adataira kíváncsi, továbbá azt is, hogy milyen időintervallum érdekli, hiszen még egy napon belül is rengeteg járat indulhat egy megállóból. Ha azonban lehetőséget adunk a napokon belül óra szintű szűrésre is, akkor már kezelhetőbb adatmennyiséggel lesz dolgunk.

Mikor a felhasználó kiválasztotta az őt érdeklő viszonylatot és megjelentek az ehhez tartozó megállók, az alkalmazás lehetőséget biztosít ezek térképes megjelenítésére is, így a teljes útvonalat áttekintve is választhatunk megállót.

2.3.2 Keresés

Az útvonaltervezéshez elengedhetetlen, hogy a felhasználó meg tudja adni a célpont helyét. Azonban nem várható el, hogy mindig ismertek legyenek a célpont koordinátái. Szükség van tehát egy olyan funkcióra, amivel elég a hely címét megadni, vagy akár a térképen rákoppintani egy pontra, és máris rendelkezésre állnak a szükséges koordináták. Az ehhez szükséges cím – koordináta összerendelést Androidon nagyon egyszerűen meg lehet valósítani, hiszen a Geocoder osztály tartalmaz ilyen metódusokat.

A keresés során a legnagyobb problémát vak és látássérült felhasználók számára a keresési feltételek megadása jelenti. Erre a problémára nyilván a beszédfelismerő nyújtana megfelelő megoldást, azonban ez még nem állt rendelkezésre a tervezés ezen fázisában.

2.3.3 Útvonaltervezés és navigálás

Munkám elején, a tervezési fázis során még úgy gondoltam, hogy a menetrendeket tároló adatbázis alapján fogom megtervezni az útvonalakat. Időközben azonban a Google Transit szolgáltatás Budapesten is elérhetővé vált, így ezt fogom felhasználni útitervek készítéséhez. Ezzel egyszerűen meg lehet határozni két pont között az optimális tömegközlekedési útvonalat a hozzá tartozó instrukciókkal együtt.

Navigálás során az alkalmazás képes lesz korábban elmentett, vagy frissen tervezett útvonalakon is végigvezetni. Ehhez szükség lesz a beszédinterfész használatára is az utasítások felolvasásához, hiszen elsődleges szempont, hogy vak és látássérült felhasználók is tudjanak közlekedni az alkalmazással. Amennyiben GPS segítségével folyamatosan követjük a felhasználó mozgását az útvonaltervhez viszonyítva, teljesen automatikusan végignavigálhatja az alkalmazás.

2.3.4 Offline működés

Igaz, hogy a menetrendek tárolása és az útvonaltervezés is szerver oldalon történik, mégis lehet majd Internet kapcsolat nélkül is használni az alkalmazást. Ehhez az szükséges, hogy a felhasználó előre tervezzen útvonalakat, amiket elment, és később visszatölt. Ez hasznos lehet azoknak, akik néhány útvonalat szeretnének csak használni, azonban ekkor a navigálás során nincs lehetőség újratervezésre, vagy járatok következő indulásának meghatározására, ha például lekéssük egy járat útitervben megadott indulását. Azonban még így is eljuthatunk a célunkhoz, hiszen az út során szükséges járatok listája elérhető marad a készüléken tárolva.

Ez a funkció különösen fontos lehet vak és látássérült felhasználók számára, hiszen ha elmentik a gyakran használt útvonalaikat, akkor a későbbiekben már nem lesz szükség újabb keresésekre és útvonaltervezésre. Elég csupán néhány kattintással betölteni a szükséges útvonalat, ebben pedig segít az alkalmazott XML alapú multimodális

felhasználói felület. Így sokkal egyszerűbbé válik az alkalmazás kezelése, mintha folyamatosan új kereséseket kellene végezni minden utazás előtt.

2.4 Multimodális felhasználói felület tervezése

A program felhasználói felülete a grafikus mellett beszédinterfészt is tartalmaz majd. Az 1.4.4-es fejezetben említett beszédfelismerő az alkalmazás tervezésekor még nem volt elérhető, ezért egyelőre csak TTS szolgáltatással lesz kiegészítve a grafikus felület. Magát a felületet egy, a korábbi félévekben készített XML alapú felhasználói felület leíró nyelvvel fogom definiálni. Ennek a leírásnak az a legfőbb előnye, hogy platform független, vagyis akár változtatás nélkül is át lehet majd vinni másik operációs rendszert futtató készülékekre. Eddig Android mellett Windows Mobile 6-ra készült el a rendszer implementációja.

Ezzel az interfésszel az a cél, hogy bárki könnyebben tudja használni az alkalmazást, akár látássérült felhasználók is. Számukra azonban gondot jelenthet a manapság elterjedt érintőképernyők kezelése. Ennek megkönnyítésére szeretném a már említett XML alapú felület leírás implementációját olyan módon megváltoztatni, hogy az egyes vezérlőket akár a képernyő figyelése nélkül is lehessen kezelni. Ehhez egy újabb modalitást szeretnék bevezetni a mobilkészülékek rezgő funkciójának segítségével. Az elképzelés tulajdonképpen nem más, mint hogy ha a felhasználó megérinti a képernyőt és elkezdi mozgatni rajta az ujját, akkor a telefon rezgéssel jelezze neki, hogy új vezérlő elemre ért, valamint a TTS is tájékoztassa erről. Ehhez szükség van néhány megkötésre a grafikus elemekkel kapcsolatban. Egyrészt ki kell iktatni minden mozgó elemet a képernyőről, hogy mindig egyértelműen meghatározható legyen, hogy épp melyik grafikus elemet érintjük meg. Emellett minden elemet el kell látni egy szöveggel, melyet a TTS felolvashat, amint hozzáérünk az adott vezérlőhöz. Így remélhetőleg már kellő visszajelzést biztosít a program ahhoz, hogy vak és látássérült felhasználók is kezelhessék.

3 Megvalósítás

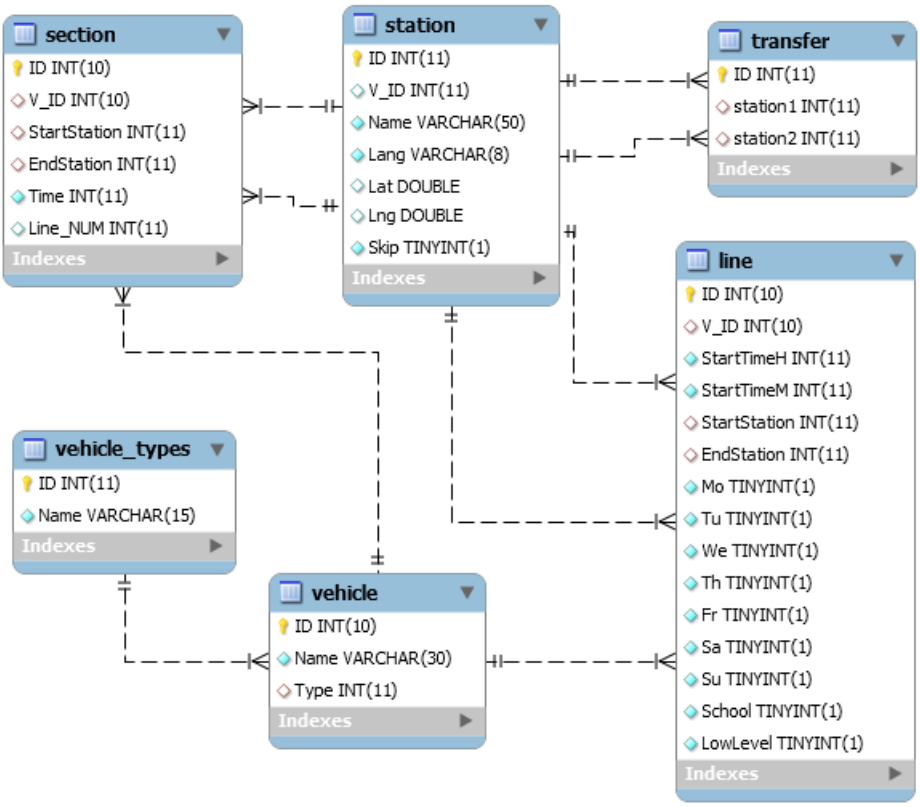
Ebben a fejezetben ismertetem a tényleges fejlesztő munka lépéseit, kezdve a szoláltatást támogató adatbázis létrehozásától, a kliens multimodális felhasználói felületének kialakításán keresztül, egészen a navigációs funkciók létrehozásáig. Bemutatom a fejlesztés során felmerült problémákat, azok megoldásait, illetve a sikeresen elkészített funkciók működését ábrákkal is szemléltetem.

3.1 Adatbázis létrehozása

A tervezés után a fejlesztési munka első lépése a funkciók megfelelő működéséhez szükséges adatbázis létrehozása volt, mely a BKV menetrendi adatait tárolja. Első célom az volt, hogy teljesen önerőből, publikusan elérhető adatok alapján hozzam létre az adatbázist, azonban ennek kivitelezése során olyan problémákba ütköztem, melyek megakadályozták ezt a tervemet.

3.1.1 BKV honlap feldolgozása

A feladat kidolgozásának első szakaszában az volt a cél, hogy a teljes rendszert publikusan elérhető információk alapján építsem fel. Ezzel az volt a célom, hogy a későbbiekben könnyen lehessen automatizálni az adatbázis frissítését, ne legyen a rendszer egy zárt adatforráshoz kötve, hanem bármikor elérhetők legyenek a legfrissebb adatok. Ekkor még nem voltak elérhetők GTFS (General Transit Feed Specification) formában a BKV adatai, ezért a menetrendi adatokat tároló adatbázist a BKV honlapján elérhető menetrendek alapján szerettem volna feltölteni. Az adatok tárolásához egy MySQL adatbázist készítettem, melyet PHP oldalak segítségével kezdtem feltölteni. Az adatbázis struktúráját az 3. ábra szemlélteti.



3. ábra - Saját adatmodell

3.1.1.1 Adatbázis táblák leírása

- ***Vehicle***

  Az egyes viszonylatok logikai azonosítására szolgál (például: 4-es villamos, 133-as busz, 2-es metró). Tárolja a viszonylat nevét, és típusát, valamint egy egyedi azonosítót.

- ***Vehicle types***

  Az adatbázisban elérhető jármű típusokat tárolja egy azonosítóval és a típusnévvel.

- ***Station***

  Az egyes megállóhelyeket reprezentálja. Tárolja a megállóhely nevét, külső kulcsként a viszonylatot, amihez tartozik, nyelvi információt – későbbi felhasználásra –, valamint egy jelzőbitet, mely a megállóhely aktivitását mutatja.

- ***Line***

  A valós járatokat tartalmazó tábla. Tartalmazza, hogy melyik viszonylaton, mely napokon, milyen időpontokban, milyen állomások között közlekedik járat. Minden fizikai járat egy rekordot jelent a táblában (például: 4-es villamos, hétköznaponként 10:12-kor Fehérvári út – Moszkva tér között).

- ***Section***

  A fizikai útvonalak megállók közötti szakaszai, melyek a teljes hálózatot lefedik. Tartalmazza, hogy egy viszonylat járművei milyen menetidővel közlekednek a két adott megálló között. Vagyis külön szakaszként jelenik meg, ha ugyanazon megállók között több jármű is közlekedik. Ez a későbbiekben az útvonaltervezésnél lesz hasznos, hiszen innen már adott, hogy a gráfban egy fizikai szakasz több élet is jelenthet, különböző menetidőkkel.

- ***Transfer***

  Átszállási lehetőségeket tárol két tetszőleges megálló között. Mivel egy megálló az adatbázis értelmezésében egy viszonylathoz tartozik, így ennek a táblának a segítségével egyből meg lehet határozni, hogy adott helyről milyen járművekre lehet átszállni.

Az adatbázis feltöltéséhez több PHP oldalt készítettem, melyekkel a BKV oldalon aktuálisan elérhető adatokat lehet dinamikusan az adatbázisba tölteni. A nagy adatmennyiség miatt először csak a villamos hálózat feldolgozására koncentráltam.

A *Jaratok.php* oldal kérdezi le az éppen elérhető viszonylatokat. Innen a *Menetrend.php* oldalra lehet lépni, ami egy adott viszonylat adatait jeleníti meg. Itt látható, hogy milyen megállók tartoznak az adott viszonylathoz, ezek között milyen menetidőkkel közlekednek az adott járatok, valamint hogy milyen napokon, milyen időpontokban indulnak járatok.

A *tarolt_jaratok.php* az adatbázisban már letárolt viszonylatokat jeleníti meg. Innen az *atszallas.php* oldalra lehet jutni, ahol az egyes járatok megállói közti átszállási lehetőségeket lehet megadni. A gyorsabb haladás érdekében a manuális beállítás mellett az

oldal képes a különböző viszonylatok azonos nevű megállóit is összepárosítani. Azonban még így is rengeteg időt vesz igénybe minden átszállási kapcsolat tárolása, hiszen nincs olyan adatforrás, aminek alapján automatizálni lehetne a folyamatot.

Sajnos azonban be kellett látnom, hogy ezek a módszerek nem elég hatékonyak a teljes adatbázis felépítésére. Csak a villamosok feldolgozásánál négyféle weboldal struktúrát kellett feldolgozni a *jaratok.php*-ban, sőt voltak olyan oldalak, melyek a feldolgozás közben is változtak, vagy újak kerültek fel, ami tovább nehezítette a munka automatizálását. Emiatt minden járatot egyesével kellett letárolni, és ez még csak a megállókat jelentette. Ezután még szükség lett volna a járat indulások feldolgozására, mely szintén többféle struktúrában jelenik meg a weboldalakon, illetve a már említett átszállások beállítására. Az eddig a pontig elvégzett munkához szükséges idő alapján várhatóan az több hónapot kitöltött volna az egész adatbázis elkészítése, így sajnos egyéb lehetőségek után kellett néznem és félbe kellett szakítanom ennek az adatbázis sémának a fejlesztését is.

3.1.2 BKV adatbázis alkalmazása

Az előző fejezetben leírt problémák után felvettem a kapcsolatot a BKV ZRT-vel abban a reményben, hogy hozzáférést kaphatok a cég menetrendi adatbázisához. Mint kiderült a vállalat a Google számára készít bizonyos időközönként exportokat az adatokról, méghozzá a Google által meghatározott GTFS – General Transit Feed Specification [7] – formátumnak megfelelően. Ez a forma tulajdonképpen olyan adatbázis táblákat definiál, melyekkel tetszőleges tömegközlekedési hálózat adatai tárolhatók. Ilyen adattáblák formájában, szöveges állományokban jutottam hozzá a szükséges adatokhoz. Időközben nyilvánossá tették a BKV GTFS adatok elérését. A BKK weboldaláról bárki letöltheti az épp aktuális menetrendeket tartalmazó adatbázist, így mégis teljesült az a terv, hogy nyilvános adatok alapján bármikor frissíthető legyen a rendszer.

A GTFS séma leírása nyilvánosan elérhető a weben is, így itt csak azokat a részeit szeretném bemutatni, melyek megjelentek a BKV által átadott adathalmazban. A későbbiekben ezeket a táblákat használtam fel a saját adatbázisom felépítésére, helyenként némi módosítással.

3.1.2.1 GTFS-en alapuló adatbázistáblák bemutatása

- ***Stops - megállók***

Ez a tábla tárolja az egyes megállókat. Itt egy rekord egy fizikai megállót jelent, melyhez tartozik egy név, szélességi és hosszúsági koordináták, egy megálló típus és opcionálisan egy szülő megálló. A két utolsó attribútum összefügg, hiszen egy megálló csak akkor szerepelhet szülő megállóként másik rekordoknál, ha ilyen típus van beállítva neki.

- ***Stop_times – megállási idők***

Ez a tábla tárolja, hogy egy adott megállóba egy adott viszonylat egy adott járata mikor érkezik meg, és mikor indul el onnan. Továbbá tárolja, hogy az adott rekord hányadik megállást jelent a járat egy meghatározott útján.

- ***Route_type – viszonylat típusok***

A GTFS leírás nem tartalmaz külön típus táblát a viszonylatokhoz, hanem rögzítetten definiál bizonyos jármű típusokhoz egy-egy számértéket. Ahhoz, hogy például az éjszakai buszokat külön lehessen választani a többi buszjárattól, szükséges további típusok bevezetése. Ez a tábla tárolja egyrészt a GTFS-ben meghatározott összerendeléseket az ott előforduló járművekkel, illetve két új azonosítót is bevezet, egyet az éjszakai buszok és egyet a trolibuszok számára.

- ***Routes – viszonylatok***

Ebben a táblában találhatók az aktuálisan elérhető viszonylatok. Minden viszonylathoz tartozik egy egyedi azonosító, egy rövid név, ami jelen esetben a viszonylatok számát jelenti, egy hosszú név és egy leírás, melyek a viszonylat végállomásainak nevét tartalmazzák, végül pedig egy típus azonosító.

- ***Trips – járatok***

Ez a tábla tartalmazza a konkrét járatokat. Minden járathoz tartozik egy viszonylat, amin közlekedik és egy szolgáltatás azonosító, ami megmondja, hogy milyen időszakokban közlekedik a járat. A járat irányát jelöli a headsign attribútum, mely az aktuális végállomás

nevét tartalmazza, továbbá a direction attribútum, mely vagy 0, vagy 1 értékű az irány függvényében. Egy járathoz tartozik még emellett egy shape is, amely a járat útvonalát tárolja, GPS koordináták listájaként. Így bármilyen járat útvonalát könnyen meg lehet jeleníteni térképen.

- ***Calendar – szolgáltatási idők***

Ez a tábla tárolja az egyes szolgáltatásokat. Egy szolgáltatás tulajdonképpen azt mondja meg, hogy azok a járatok, amelyek ehhez a szolgáltatáshoz tartoznak, a hetek mely napjain közlekednek, és hogy ez milyen dátumok között érvényes.

- ***Calendar_dates – szolgáltatási idő kivételek***

Az előző táblához kapcsolódó kivételeket tárolja ez a tábla. Egy rekord azt mutatja meg, hogy a hivatkozott szolgáltatás a rekordban tárolt napon az eredeti menetrendhez képest milyen eltéréssel közlekedik. Vagyis meg lehet adni, hogy egy szolgáltatáshoz tartozó járatok nem közlekednek egy olyan napon, amikor a szolgáltatás szerint kellene. Vagy akár ellenkezőleg, egy adott naphoz újabb szolgáltatások is csatolhatók. Így könnyen meg lehet oldani például, hogy húsvét hétfőn a vasárnapi járatok közlekedjenek. Elég csupán a hétfői járatokhoz kivételt megadni húsvét hétfőre, a vasárnapikat pedig hozzáadni húsvét hétfőhöz.

- ***Shapes – útvonalak***

Kapcsoló tábla a járatok és a shape pontok között a több-több kapcsolat megvalósítása érdekében.

- ***Shape_points – útvonal pontok***

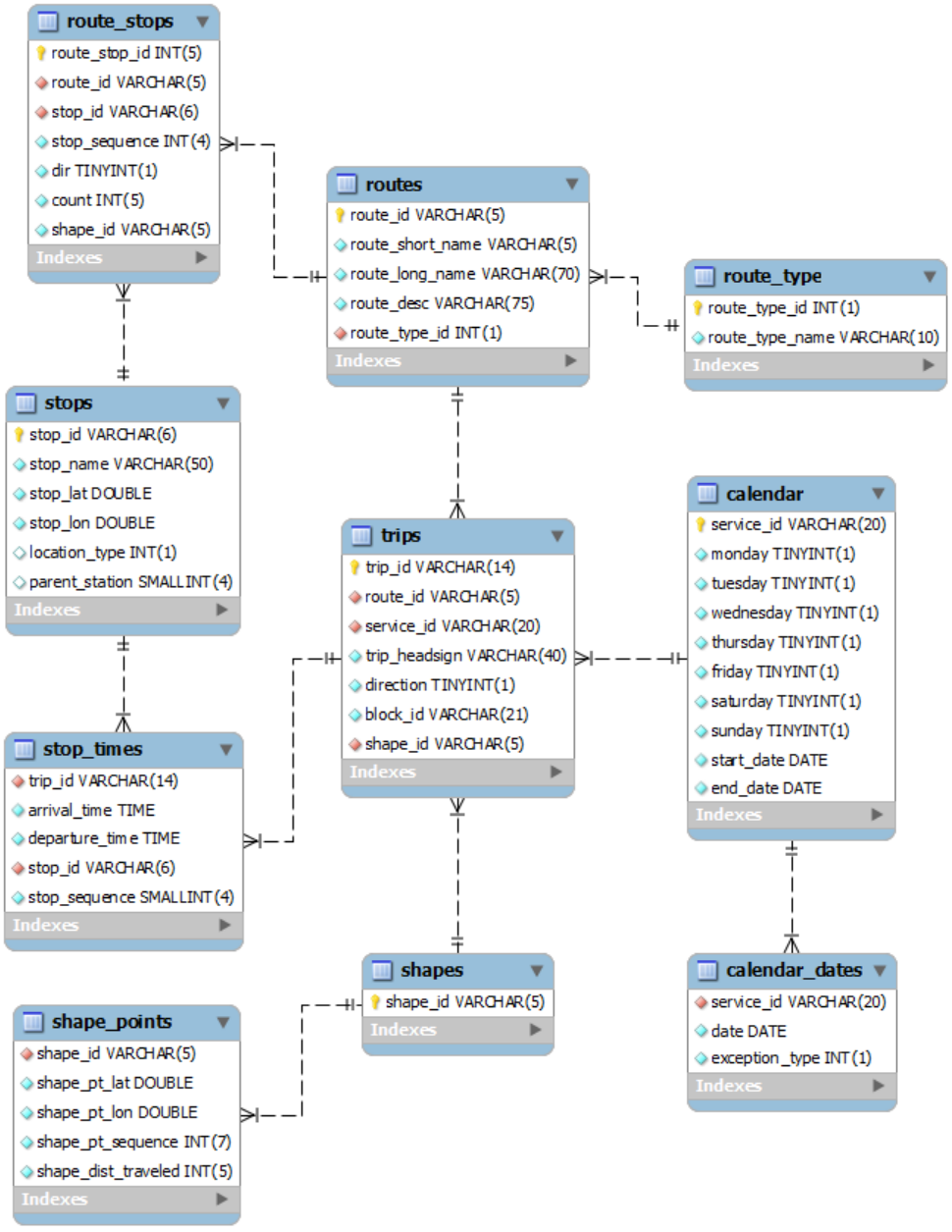
Az egyes járatok útvonalainak pontjait tárolja ez a tábla. Tárolja a pont szélességi és hosszúsági koordinátáit, sorszámát az útvonalon, valamint az útvonalon az aktuális pontig megtett távolságot.

- ***Route_stops – járatokhoz tartozó megállók***

Az egyes viszonylatokat és a hozzájuk tartozó megállókat rendeli össze közvetlenül. Erre szükség lehet a kliens alkalmazás böngésző funkciójánál, illetve útvonaltervezésnél is.

Ilyen összerendelés nem volt magában a GTFS struktúrában, viszont a kliens alkalmazás működését gyorsíthatja, ha előre le vannak tárolva a megállók, nem pedig minden egyes kérésnél kell meghatározni őket a GTFS adatokból.

A Route_stops táblával kiegészített GTFS struktúrát a 4. ábra szemlélteti.



4. ábra - GTFS alapú adatmodell

3.1.3 Adatbázis struktúrák összehasonlítása

Az általam tervezett adatbázis és a BKV-tól kapott, GTFS struktúrájú adatbázis eltérő szemlélettel kezeli ugyan a közlekedési hálózatot, mégis fel lehet fedezni hasonlóságokat a két struktúra között. Az 1. táblázatban összefoglalom, hogy a sémák mely táblái feleltethetők meg egymásnak. Ez a megfeleltetés nem egyértelmű leképzés, inkább csak a táblák funkciói közti hasonlóságot hivatott szemléltetni.

Saját modell	GTFS	Magyarázat
Vehicle	Routes	Viszonylatok reprezentálása.
Station	Stops	Megállók reprezentálása, az első esetben egy fizikai megállóhoz több Station rekord tartozhat.
Line	Trips + Calendar + Stop_times	A Line tábla összefoglalja a Trips és a Calendar szerepét, hiszen egy adott járathoz rendel indulási időket, és érvényességi időszakot.
Section + Stop	Stop_times	A Section két megállót reprezentál és a köztük levő menetidőt, így tulajdonképpen a Stop_times-hoz hasonló szerepet tölt be, ami megadja az egyes megállókba érkezés és indulás idejét.

1. táblázat - Adatstruktúrák kapcsolatai

Alapvető eltérés a két séma között, hogy a GTFS esetén minden fizikai megállónak egy megálló felel meg az adatbázisban, addig az általam tervezett modellben egy fizikai megállóhoz több logikai megálló is kapcsolódhat. Az én modellemben nem is tudtam volna egyedi megállókból kiindulni, hiszen nem volt olyan adatforrásom, ahonnan beszerezhettem volna ezeknek a listáját. Ennek eredményeképp az én adatbázisomban közvetlen kapcsolat áll fent a megállók és a viszonylatok között. Ez nyilvánvaló is, hiszen minden megálló rekord egy meghatározott viszonylathoz tartozik. Ezzel szemben a GTFS esetén hiányzott ez a kapcsolat a megállók és a viszonylatok között, melyet már a

Route_stops táblával pótoltam. Korábban csak az egyes járatokon és a megállási időkön keresztül lehetett meghatározni, hogy egy viszonylat milyen megállókat érint. Látható, hogy egy szemléletbeli különbség milyen nagy kihatással lehet egy adatbázis használatára, hiszen míg egyik esetben egy sima SELECT segítségével elvégezhető a művelet, a másik rendszerben több tábla illesztése szükséges. További előnye még az én modellemnek, hogy tartalmazza az egyes útszakaszokhoz tartozó menetidőket is, így útvonaltervezés során sokkal gyorsabban meg lehet határozni a legrövidebb utakat, egyszerűbb lekérdezésekkel.

Természetesen a GTFS modellnek is megvannak a maga előnyei. Például hatékonyan lehet vele kezelni a szokásos menetrendtől való eltéréseket a calendar_dates táblával. Továbbá az egyes járatok útvonalainál nem csak a megállók koordinátáit tárolja, hanem az útvonal egyéb részeit is, így egy térképen könnyen meg lehet azokat jeleníteni.

Összességében elmondható, hogy mindkét sémának lehetnek előnyei és hátrányai is a felhasználási módtól függően. Igaz, hogy az én modellem fejlesztése félbe maradt, de így is hasznos tapasztalatot jelentett. A GTFS modellhez való hasonlóság is arra enged következtetni, hogy amennyiben a modell fejlesztése befejeződött volna, valószínűleg megfelelő lett volna az adatok tárolására. A későbbiekben elképzelhető, hogy a két modell segítségével ki lehetne dolgozni egy olyan struktúrát, mely ötvözi a két megoldás előnyeit, és így kisebb tárhelyen, hatékonyabban lehetne tárolni az adatokat.

3.2 Szerver oldal

3.2.1 Adatbázis feltöltés és kezelés

A GTFS adatokat egyszerű szöveges fájlokban kaptam meg, melyek mindegyike egy-egy táblát tartalmazott. Először PHP segítségével töltöttem fel az adatbázist. A *parse_gtfs.php* oldalon ki lehetett választani, hogy melyik tábla adatait szeretnénk felvinni, illetve megjelenítette, hogy hány rekordot tartalmaz már az adatbázis megfelelő táblája. A táblákat betöltési sorrendjét az határozta meg, hogy melyikhez milyen idegen kulcsok tartoztak. Hiszen ha idegen kulcsot tartalmaz egy tábla, akkor nem lehet addig betölteni, amíg az idegen kulcshoz tartozó másik tábla nem áll készen.

A betöltés eszközéül azért a PHP-t választottam, mert a lehető legegyszerűbben szerettem volna feldolgozni az adatokat, és eredetileg a BKV oldaláról letöltött

információkat is ilyen módon tároltam. Azonban a nagyobb méretű táblák betöltése már túllépett a PHP képességein. Mivel először csak annyi volt a célom, hogy egyszer feltöltsem az adatbázist, egyszerűen rekordonként szúrtam be az adatokat. Ez azonban nem volt elég hatékony módszer, sok overheadet jelentett, ha több százezer rekordot tartalmazó táblákat kellett betölteni.

Az adatbázis frissítéséhez már egy ADO.NET alapú adminisztrációs alkalmazást fejlesztettem. Ezt sikerült úgy optimalizálni, hogy nagyjából fél óra alatt ki lehet cserélni a teljes adatbázist. Ezt az időt természetesen befolyásolja, hogy milyen teljesítményű szervert használunk, illetve milyen egyéb alkalmazások futnak még. A fél órás értéket egy 2,2 GHz-es Intel Core2Duo processzorral szerelt, 3 GB memóriát tartalmazó laptopon sikerült elérni. Ekkor ez a gép látta el mind a webszerver, mind az adatbázisszerver feladatokat. Webszerverként Apache 2.2.17-et használtam, melyen PHP 5.3.4 futott, az adatbázisszerver pedig a MySQL 5.1.53 verziója volt. Korábban ez a folyamat több mint 6 órát vett igénybe ugyanezen a konfiguráción, azonban akkor még egyáltalán nem volt optimalizálva a betöltés. A későbbiekben egy a Távközlési és Médiainformatikai Tanszék beszédlaborjában elhelyezett dedikált szerver fogja biztosítani ezeket a funkciókat. Ekkor már lehetőség lesz az adatbázis napi szintű frissítésére éjszakánként, melynek eredményéről e-mailben tájékoztathat a szerver.

A gyorsabb feldolgozás eléréséhez a következő optimalizálási lépéseket alkalmaztam:

- Csoportos beszúrás, 1000 rekordonként
  - Overhead csökkentés
- Automatikus commit kikapcsolása a beszúrás elején, tranzakció indítás
  - Overhead csökkentés
- Adatstruktúrák optimalizálása
  - A korábban feltöltött adatbázison végzett elemzésekkel meg tudtam állapítani, hogy az egyes mezőkbe legfeljebb mekkora adatok kerülhetnek. Ennek köszönhetően több helyen tudtam csökkenteni a szükséges méreteket. Korábban, az adatbázis első feltöltésekor nem állt rendelkezésemre ilyen információ, azért a szöveges fájlok alapján,

túlbecsléssel hoztam létre az adatbázis sémát. Ez a lépés az adatbázis méretét is csökkentette.

Ezek közül az első kettő hatására gyorsult leginkább a betöltés.

Az adminisztrációs alkalmazás segítségével bármikor le lehet tölteni a BKK honlapján már nyilvánosan elérhető, friss GTFS adatokat, be lehet tölteni az adatbázist, illetve ha kell, újra lehet generálni az egész sémát. Sőt, ez az alkalmazás a GTFS struktúra alapján generálja még a Route_stops táblát az adatbázisba, melyben eltárolja az egyes járatokhoz tartozó megállókat. Ez a megálló – járat szintű összerendelés vélhetően azért hiányzott a GTFS struktúrából, mert nem biztos, hogy egy járat minden utazás alkalmával ugyanazokat a megállókat érinti. Azonban a legtöbb utazás során meghatározott megállókat használnak a járatok. A kivételek főleg az első és utolsó utazásoknál fordulnak elő, amikor a fő útvonal mellett is megjelenik néhány megálló, vagy esetleg nem a végállomásról indul a járat. Ezeket kiszűrve azonban meg lehetett határozni a járatok megállóit irányonként elkülönítve, ami a használat során jóval gyorsabb működést eredményezhet, amikor egy járat megállóira vagyunk kíváncsiak, például böngészéskor.

A sikeres betöltés után az adatbázis mérete is megerősített abban, hogy helyes döntés volt félbehagyni a BKV weboldal feldolgozását. A korábban leírt módszerekkel feltöltött adatbázis mérete, ami a villamosok megállóit és járatait tartalmazza csak 512 KB lett, melyet körülbelül 1 hét alatt sikerült felépíteni. A teljesen betöltött GTFS adatbázis viszont több mint 430 MB lett. Optimalizálás előtt ez még majdnem 480 MB volt, és abban még nem is szerepelt a megállókat és járatokat összekötő új tábla. Ebből is látszik, hogy valóban rengeteg időt emésztett volna fel a BKV oldalak feldolgozása, ráadásul a változó oldal struktúrák folyamatos karbantartást igényeltek volna, és rontották az automatizálási lehetőségeket is. Emellett voltak olyan járatok is, melyekhez nem tartozott HTML oldal, így a teljes menetrend lefedése szinte lehetetlen lett volna. Ezzel szemben most már bármikor, dinamikusan frissíthető a teljes adatbázis. A későbbiekben az adminisztrációs alkalmazás akár egy háttérben futó szolgáltatássá is alakítható, mely folyamatosan fut az adott szerveren, és bizonyos időközönként, vagy valamilyen egyéb trigger hatására frissíti az adatbázist, kezdve a GTFS fájlok letöltésétől az adatbázis feltöltéséig. Vagyis a rendszer karbantartása teljesen automatizálható lehet.

3.2.2 Kliens oldali interfész

A kliens oldali interfész kidolgozását a megvalósított funkciókkal párhuzamosan végeztem. A tervezett feladatok közül a menetrend böngészés és az útvonaltervezés megvalósításához van szükség az adatbázis elérésére. Az adatbázis eléréséhez PHP oldalakat készítettem, melyek XML formátumban küldenek választ a kliens kéréseire.

Böngészés során a felhasználók a korábban leírtaknak megfelelően több lépésen keresztül finomíthatják, hogy milyen menetrendekre kíváncsiak.

Ezek a finomítási lépések a következők:

- Járműtípus kiválasztása
- Viszonylat kiválasztása
- Megálló és irány kiválasztása
- Időpont megadása

Ezek közül az első három lépés igényel adatbázis kommunikációt, illetve a folyamat végén a feltételeknek megfelelő járatok adatainak letöltése. A viszonylat kiválasztása után a felhasználó választhat, hogy szöveges, vagy térképes nézetben szeretne megállót választani. A térképes nézethez le kell tölteni az adatbázisban tárolt koordinátákat, melyek segítségével pontosan megjelölhető lesz az egyes járatok útvonala.

Útvonaltervezés esetén szükség lehet a Google-től visszakapott járatok típusának lekérdezésére. Ugyanis én megkülönböztetem a troli, éjszakai és a sima autóbuszjáratokat, a Google viszont nem, ez pedig problémát jelentene a járatokhoz tartozó megállók koordinátáinak meghatározásában.

A következőkben az alkalmazás működése során használt lekérdezéseket és a PHP által generált XML válaszokat mutatom be.

3.2.2.1 Jármű típusok lekérdezése

Ahhoz, hogy kiválaszthassuk a kívánt járműtípust, hozzá kell férni az adatbázis route_types táblájához. Ezt a következő lekérdezéssel lehet elvégezni:

```
SELECT route_type.route_type_id, route_type.route_type_name
FROM route_type
ORDER BY route_type.route_type_id
```

A kérést kezelő PHP oldal a lekérdezés lefutása után a következő választ adja vissza:

```
<?xml version="1.0" encoding="UTF-8"?>
      <types>
      <type id="0">tram</type>
      <type id="1">metro</type>
      <type id="2">hev</type>
      <type id="3">bus</type>
      <type id="8">troley</type>
      <type id="9">night_bus</type>
</types>
```

3.2.2.2 Viszonylatok lekérdezése

Ha kiválasztottunk egy járműtípust, akkor a kliensnek le kell kérdeznie a típushoz tartozó viszonylatokat. Ehhez a routes táblán kell egy szűrést végezni a route_type_id attribútumra. Például a villamosok viszonylatokat a következő lekérdezéssel lehet megjeleníteni:

```
      SELECT routes.route_id, routes.route_short_name, routes.route_long_name,
route_type.route_type_id
      FROM routes
      WHERE routes.route_type_id = '0'
      ORDER BY routes.route_id
```

Erre a következő válasz generálódik:

```
<?xml version="1.0" encoding="UTF-8"?>
<vehicles>
      <vehicle id="3010" l_name="Lágymányosi híd pesti hídfő H/Zugló
      vasútállomás">1</vehicle>
      <vehicle id="3011" l_name="Népliget M/Bécsi út">1A</vehicle>
      <vehicle id="3020" l_name="Jászai Mari tér/Vágóhíd H">2</vehicle>
      <vehicle id="3021" l_name="Boráros tér H/Jászai Mari
      tér">2A</vehicle>
      <vehicle id="3030" l_name="Gubacsi út/Mexikói út M">3</vehicle>
      <vehicle id="3040" l_name="Fehérvári út/Moszkva tér M">4</vehicle>
      …
</vehicles>
```

3.2.2.3 Viszonylat megállóinak lekérdezése

A viszonylat kiválasztása után a kell lekérdezni a viszonylathoz tartozó megállók listáját az újabb szűrési lehetőség elvégzése érdekében. A viszonylatokra az előzőekben lekérdezett azonosítókkal lehet hivatkozni. A felhasznált lekérdezés a következő:

```
SELECT *
FROM route_stops INNER JOIN stops ON stops.stop_id = route_stops.stop_id
WHERE route_id = 3040
AND dir = 0 ORDER BY route_stops.route_stop_id
```

Az adatbázis feltöltése után létrehozott route_stops táblának köszönhetően egyszerűen le lehet kérdezni egy viszonylat megállóit. Csupán egy illesztést kell elvégezni a stops táblához, ami a megállók tényleges adatait tartalmazza.

Ez a példa a 4-es villamos megállóit kéri le, méghozzá a Fehérvári úttól a Moszkva tér felé. A kapott válasz a következő:

```
<?xml version="1.0" encoding="UTF-8"?>
<route>
      <stop    lon="19.046916"    lat="47.474027"    name="Újbuda-központ"
id="F01998"/>
      <stop  lon="19.053788"  lat="47.474125"  name="Budafoki  út  (Szerémi
sor)" id="F01992"/>
      <stop lon="19.059613" lat="47.47694" name="Petőfi híd; budai hídfő"
id="F02225"/>
      <stop   lon="19.066665"   lat="47.480361"   name="Boráros   tér   H"
id="F01374"/>…
</route>
```

A listában szerepel a megállók neve és azonosítója, mely a további szűrésekhez szükséges, valamint a pozíciójuk. A továbbiakban a Fehérvári úti megállóra fogunk szűrni.

3.2.2.4 Útvonal koordináták lekérdezése

A megállók és a teljes útvonal térképes megjelenítéséhez ismerni kell az útvonalat leíró koordinátákat. Ezeket a következő lekérdezésekkel lehet elérni:

```
SELECT shape_points.shape_id, COUNT(*) as shape_count
FROM trips
    INNER JOIN shape_points ON shape_points.shape_id=trips.shape_id
WHERE direction='0' AND route_id='3040'
GROUP BY shape_points.shape_id
ORDER BY trips.trip_id, shape_points.shape_dist_traveled

SELECT shape_points.shape_pt_lat, shape_points.shape_pt_lon,
shape_points.shape_pt_sequence
FROM shape_points
WHERE shape_id = '0976'
ORDER BY shape_points.shape_dist_traveled
```

Az első lekérdezés meghatározza, hogy egy adott viszonylat utazásaihoz milyen útvonalak tartoznak. Egy viszonylathoz több útvonal is tartozhat, ám az utazások nagy része egy útvonalon zajlik, pontosan azon, amely mellett a viszonylathoz tartozó megállók találhatók. A többi útvonal egyedi utazásokhoz tartozik, például nap elején és végén, mikor a járművek kocsiszínből jönnek, vagy oda mennek, és esetleg nem is az útvonal valamelyik végállomásától közlekednek. Így ki kell szűrni a legtöbbször előforduló útvonalat, ezt a lekérdezést elküldő PHP oldal végzi el. Ha megvan a keresett útvonal, a második lekérdezés után kapjuk vissza a tényleges útvonal koordinátákat. Az erre kapott XML válasz a következő:

```
<?xml version="1.0" encoding="UTF-8"?>
<route>
       <part>
              <section>
              <checkpoint seq="100002" lon="19.046916" lat="47.474027"/>
              <checkpoint seq="100003" lon="19.047015" lat="47.47403"/>
              <checkpoint seq="100004" lon="19.047433" lat="47.474009/>
              …
              </section>
              …
      </part>
       <part>
       …
       </part>
</route>
```

A route tagon belül egyes útvonalak több részre tagolódhatnak, ezek a part tagok, melyek tulajdonképpen egy shape_id-hoz tartoznak az adatbázisban. Ezen belül az egyes section tagok mindig két megálló között szakasz koordinátáit tartalmazzák.

3.2.2.5 Megállóból induló járatok lekérdezése

A járatok lekérdezése előtt a felhasználó még beállíthatja, hogy milyen időszakban kíváncsi a menetrendre, ez azonban nem igényel adatbázis hozzáférést. Ha mindez megvan, akkor már csak le kell kérni az elérhető járatokat. Ez a megállók lekérdezéséhez hasonlóan összetett műveletet igénylő lépés. A folyamat során ugyanis meg kell határozni, hogy melyik járatok állnak meg a megadott időintervallumban a meghatározott megállóban. Továbbá azt is figyelembe kell venni, hogy ezek a járatok a beállított dátum napján érvényes szolgálgatáshoz tartoznak-e, illetve hogy nem tartozik-e hozzájuk kivétel.

A feladat összetettsége miatt két részre osztottam fel a lekérdezést. Az első fele meghatározza, hogy az adott viszonylat járatai közül melyek közlekednek a megadott napokon, figyelembe véve a kivételes dátumokat is.

Ez a következőképp néz ki:

```
SELECT trips.trip_id, trips.trip_headsign
FROM trips
       INNER     JOIN     calendar_dates     ON     trips.service_id     =
calendar_dates.service_id
       INNER JOIN calendar ON calendar.service_id = trips.service_id
WHERE trips.route_id = "3040"
       AND trips.direction = 0
       AND calendar.start_date <  '2011-10-23'
       AND calendar.end_date >  '2011-10-23'
       AND ((calendar_dates.date IS NULL
              AND calendar.sunday = 1)
              OR (calendar_dates.date=  '2011-10-23'
              AND calendar_dates.exception_type=1)
              OR (NOT( '2011-10-23' IN
                     (SELECT calendar_dates.date
                     FROM calendar_dates
                     WHERE calendar_dates.service_id = trips.service_id))
              AND calendar.sunday = 1))
GROUP BY trips.trip_id
```

Az itt bemutatott példa lekérdezés a 4-es villamos 2011. 10. 23-án, vasárnap közlekedő járatainak azonosítóit adja vissza. Ezután meg kell határozni, hogy az adott járatok közül melyek tartózkodnak a megadott időszakban a megadott megállókban. Ehhez meg kell vizsgálni az összes járathoz tartozó megállási időket és a hozzájuk rendelt megállókat.

```
SELECT stop_times.arrival_time, stops.stop_name
FROM stop_times
       INNER JOIN stops ON stop_times.stop_id = stops.stop_id
WHERE stop_times.arrival_time > "11:00:00"
       AND stop_times.arrival_time < "24:00:00"
       AND stop_times.stop_id = "F01998"
       AND stop_times.trip_id ="A77535212"
ORDER BY stop_times.stop_sequence
```

Ez a lekérdezés annyiszor fut le, ahány tripd_id-t visszaadott az előző, természetesen mindig a megfelelő paraméterrel.

Végül pedig ezt az eredményt kapja vissza a kliens:

```
<?xml version="1.0" encoding="UTF-8"?>
<trips>
	<trip name="Széll Kálmán tér M" id="A77535212">
		<stop name="Újbuda-központ" arrive="11:02:00"/>
	</trip>
	<trip name="Széll Kálmán tér M" id="A77535218">
		<stop name="Újbuda-központ" arrive="11:12:00"/>
	</trip>
	<trip name="Széll Kálmán tér M" id="A77535224">
		<stop name="Újbuda-központ" arrive="11:22:00"/>
	</trip>
	<trip name="Széll Kálmán tér M" id="A77535230">
		<stop name="Újbuda-központ" arrive="11:32:00"/>
	</trip>
	<trip name="Széll Kálmán tér M" id="A77535236">
		<stop name="Újbuda-központ" arrive="11:42:00"/>
	</trip>
	<trip name="Széll Kálmán tér M" id="A77535242">
		<stop name="Újbuda-központ" arrive="11:52:00"/>
</trip>
```

Látható, hogy valóban az összes eddigi feltételnek megfelelő adatokhoz jutottunk el. Vagyis, hogy 2011. 10. 23.-án milyen időpontokban indult a 4-es villamos Újbuda-központból a Széll Kálmán tér felé 11 és 24 óra között. Összehasonlításképp az 5. ábra mutatja a BKV holnapján található menetrendből származó adatokat.

Óra *[H/U]*	Munkaszüneti napokon (7) Perc / *minutes / Minuten*
3	
4	37, 57
5	17, 37, 57
6	17, 37, 52
7	02, 12, 22, 32, 42, 52
8	02, 12, 22, 32, 42, 52
9	02, 12, 22, 32, 42, 52
10	02, 12, 22, 32, 42, 52
11	02, 12, 22, 32, 42, 52
12	02, 12, 22, 32, 42, 52
13	02, 12, 22, 32, 42, 52
14	02, 12, 22, 32, 42, 52
15	02, 12, 22, 32, 42, 52
16	01, 08, 16, 23, 31, 38, 46, 53
17	01, 08, 16, 23, 31, 38, 46, 53
18	01, 08, 16, 23, 31, 38, 46, 53

5. ábra - 4-es villamos menetrendje munkaszüneti napokon Újbuda-központ felől [1]

[1] http://bkv.hu/villamos/4vissza.html

3.2.2.6 Viszonylat típus lekérdezése

Útvonaltervezésnél szükség lehet a visszakapott járatok típusának ellenőrzésére, ugyanis a Google Transit nem különbözteti meg az autóbuszt, a trolit és az éjszakai járatokat, az én adatbázisom viszont igen.

A lekérdezés a következő:

```
SELECT route_type.route_type_name, route_type.route_type_id, routes.route_id
FROM routes
    INNER JOIN route_type ON routes.route_type_id = route_type.route_type_id
WHERE routes.route_short_name = 4
```

Paraméterként csupán a keresett járat nevét kell megadni, mely alapján visszakapjuk a típus nevét. A válasz itt a kevés adat miatt nem XML, hanem sima szöveges formában érkezik. Visszakapjuk a típus nevét, azonosítóját és a járat azonosítóját is.

3.2.3 Google Maps adatok lekérdezése

Az útvonaltervezéshez a Google Transit szolgáltatást használtam fel. Ezt megfelelően paraméterezve megkaphatjuk két pont között az optimális tömegközlekedési útvonalat.

Egy útvonal meghatározása két fő lépésben történik. Először lekéri a program a tömegközlekedési útvonalat. Ekkor visszakapjuk, hogy a kiindulási helytől melyik megállóhoz kell elmenni, onnan milyen járatokra kell felszállni, és végül az utolsó megállóból hogy jutunk el a célhoz. Vagyis minden útvonal háromféle szakaszra osztható:

- Gyaloglás valamelyik megállóba
- Utazás megállók között
- Gyaloglás valamelyik megállóból

A megállók közötti utazás megjelenítéséhez szükséges koordináták megtalálhatók az alkalmazás saját adatbázisában, amit a megállónevek és viszonylatok alapján meg is lehet határozni. Viszont a gyalogláshoz csak utasításokat kapunk vissza, koordinátákat nem. Ezért szükség van minden ilyen szakasz esetében egy újabb lekérdezésre, ahol már gyalogos útvonalat keresünk a megálló és az adott kiindulási, vagy célpont között. Erre a

lekérdezésre KML (Keyhole Markup Language, térben ábrázolt adatok XML formájú megjelenítésére szolgál [12]) formátumban kapunk választ, ami már tartalmazza a szükséges koordinátákat, így a teljes útvonal kirajzolható.

A felhasznált lekérdezések a következők:

Teljes útvonal lekérdezése:

http://maps.google.com/m/directions?dirflg=r&hl=hu&saddr=1011+Budapest%2C+F%C5%91+utca+1-5%2C+Magyarorsz%C3%A1g&daddr=1117+Budapest%2C+Irinyi+J%C3%B3zsef+utca+42&date=2011-10-18&time=20:25

Gyalogos szakaszok lekérdezése:

http://maps.google.com/maps?dirflg=w&hl=hu&saddr=47.474359,19.052687&daddr=47.47359,19.05311&output=kml

A lekérdezésekben felhasznált paramétereket, jelentésüket és lehetséges értékeiket a 2. táblázat foglalja össze.

Paraméter	Jelentés	Értékek
dirflg	Utazás típusa (gyalog, BKV)	- r :tömegközlekedés - w: gyalog
hl	Nyelv	- hu: magyar
saddr	Indulási cím, vagy koordináta	- tetszőleges
daddr	Érkezési cím vagy koordináta	- tetszőleges
date	Utazás dátuma	- dátum
time	Utazás megkezdésének ideje	- idő
output	Kimenet	- kml: KML formátum

2. táblázat - Google Maps URL paraméterek

3.3 Kliens oldal

Amint az a feladat kiírásban is szerepel, az alkalmazásban kulcs szerepet kap a multimodális felhasználói felület. Ezzel az a célom, hogy azok a látássérült felhasználók is könnyen használhassák az alkalmazást, akiknek amúgy gondot jelenthet egy érintőképernyős készülék kezelése. Ennek megvalósítására egy, a korábbi félévek során fejlesztett XML alapú, többplatformos multimodális felhasználói felületet alkalmaztam. A rendszer fejlesztése összesen 2 éve kezdődött és jelenleg is tart. Ez alatt az idő alatt elkészült a Windows Mobile 6 és az Android platformon működő verzió, azonban még nem elég kiforrott a rendszer. Inkább csak az alap koncepció került megvalósításra, vagyis hogy ne a fejlesztőnek kelljen azzal foglalkoznia, hogy szinkronban kezelje a beszéd és a grafikus interfészeket. Helyette adjunk a kezébe egy olyan keretrendszert, ami olyan grafikus elemeket tartalmaz, melyek integráltan kapcsolódnak egy beszédinterfészhez. Az két interfész leírására XML nyelven van lehetőség, ez biztosítja a platformok közötti átjárhatóságot. A következő fejezetekben szeretném bemutatni, hogy most, az első komolyabb alkalmazás fejlesztése közben milyen korábbi hibákat javítottam ki a rendszeren, illetve hogy milyen új funkciókkal láttam el annak érdekében, hogy valóban hatékony segédeszközzé váljon a fejlesztők kezében.

3.3.1 GUI fejlesztés

Az XML alapú GUI (Graphical User Interface) leírás egyik legnagyobb hiányossága, hogy az egyes elemek helyzetét és méretét fix képernyő koordinátákkal adja meg. Ez pedig azt jelenti, hogy nem csak platformok között, de még egy platformon belül is nehéz minden készülékre alkalmazni egy leírást. Hiszen a felbontás változása esetén máris át kell írni a felületet. Továbbá ha megvizsgáljuk az Androidon elérhető lehetőségeket észrevesszük, hogy az egyetlen olyan felület, ahol közvetlenül koordináták lapján lehet elhelyezni vezérlőket, már jó ideje az SDK nem javasolt elemei közé tartozik. A rendszer működésének a tesztelésére még alkalmas volt mégis ennek az elemnek (AbsoluteLayout) az alkalmazása, most azonban olyan megoldás kell, ami jobban illeszkedik az egész Android környezethez, ugyanakkor Windows Mobile esetén is alkalmazható marad.

A megoldás meglehetősen egyszerű ötletet valósít meg. Ahol eddig fix koordinátákat tároltunk, ott adjuk meg a képernyő méreteihez képest százalékban a paramétereket. Android esetén az ilyen módszerrel definiált elemek könnyen elhelyezhetők a FrameLayout nevű elemen. Ennek az a jellegzetessége, hogy nem az elemek pozícióját lehet megadni, hanem azt, hogy a képernyő szélétől számítva mekkora hely maradjon mellettük. Ezzel könnyen meg lehet adni az elemek pozícióját, ha ismerjük az aktuális képernyő felbontást. A vezérlők méretezése szintén megoldható, amennyiben mind a vezérlők magasságát, mind a szélességüket FILL_PARENT értékre állítjuk, vagyis hogy töltsék ki a rendelkezésre álló helyet. Hiszen ekkor a bal oldali és a felső margó szabja meg az elem pozícióját, a jobb oldali és az alsó pedig a méreteit.

Ez az egyszerű módosítás rengeteg előnnyel jár a rendszer felhasználása szempontjából. Egyrészt a vezérlők felhasználhatók lesznek az összes Layout elemen, amennyiben nem XML-ből hozzuk létre őket. Hiszen nem szükséges nekik koordinátákat megadni, ugyanúgy használhatók, mint bármelyik beépített vezérlő. Továbbá azzal, hogy a képernyőhöz viszonyított méretekkel dolgozunk, egyben megoldódik a képernyő elforgatás kezelése is. Ehhez csupán létre kellett hozni egy FrameLayout leszármazott osztályt, mely az onMeasure metódust felüldefiniálja, és ebben számolja ki az egyes vezérlők konkrét pozícióját és méretét a százalékos adatokból. Ez a metódus meghívódik minden alkalommal, amikor újra kell méretezni az elemeket, így megfelel a képernyőforgatás kezelésére is.

Érintőképernyős eszközök kezelésének az egyik komoly problémája, hogy mindenképp látni kell a felületet, szemben a billentyűzetes eszközökkel, melyeket bizonyos mértékben tapintás alapján is lehet irányítani. A multimodális felhasználói felület ennek a problémának a megoldására támogatja a beszédfelismerő kezelést, azonban egyelőre nem sikerült Androidon működő beszédfelismerőt is illeszteni a rendszerbe. Így új megoldást kellett találni arra, hogy a felhasználók érzékelhessék a felhasználói felület elemeit.

A multimodális felülethez tartozó magyar szövegfelolvasó képes arra, hogy az aktuálisan kijelölt vezérlő elemek nevét, funkcióját felolvassa. Azonban valahogy meg kell oldani a vezérlők közti navigálást a beszédfelismerés nélkül. Ennek megoldására született a „virtuális egér” vezérlés koncepciója. Az ötlet lényege, hogy iktassunk be egy új fázist az érintőképernyő kezelésébe. Ha a felhasználó megérinti a kijelzőt és elkezdi mozgatni a

kezét, akkor az működjön úgy, mint az egér a számítógépeken. Ha új vezérlő elemre ér a mozgással, akkor átkerül a bemeneti fókusz, a TTS pedig tájékoztat a változásról. További kiegészítés a rendszerhez, hogy a telefon rezgéssel is jelzi az egyes vezérlők határát, tovább javítva a visszajelzést a felhasználónak.

Az ujj mozgásának követéséhez készítettem a TouchMotionListener osztályt, mely implementálja az OnTouchListener interfészt. A működés lényege az onTouch metódus felüldefiniálása. A beérkező képernyőesemények alapján dől el, hogy a felhasználó csak koppintott, vagy mozgatta az ujját. Koppintás esetén egyszerűen tovább adja az eseményt az operációs rendszernek, mozgatás esetén viszont megvizsgálja, hogy melyik vezérlő területén történt az esemény, és ha szükséges átállítja a fókuszt. A felület használatához egyszerűen az aktuális elemeket hozzá kell adni egy TouchMotionListener-hez.

Ez a módszer azonban megköveteli, hogy minden vezérlő fix pozícióban legyen a felhasználói felületen, vagyis meg ki kell iktatni a görgetés lehetőségét. Ez például listák esetén könnyen megoldható navigációs gombok segítségével, melyekkel a listát blokkonként lehet tovább vagy visszaléptetni. Igaz, hogy ez csökkenti a ma megszokott dinamikus felhasználói élményt, viszont mindenki számára egyszerűbben használható felületet biztosít, így ez egy vállalható kompromisszumnak tekinthető.

3.3.2 SUI fejlesztés

Az XML alapú multimodális interfész fejlesztése kiterjedt a beszédinterfész (SUI – Speech User Interface) funkcióinak bővítésére is. A korábban elérhető verzióval ellentétben most már támogatja a rendszer az XML alapú dialógusok futási idejű cseréjét, illetve egyszerű felületekhez automatikus dialógusgenerálást, ami bizonyos esetekben nagy segítséget és időmegtakarítást jelenthet.

A dialógusok cseréjével sokkal jobban strukturálhatóvá válnak az alkalmazások. A fejlesztő külön dialógust hozhat létre az egyes nézetekhez, vagy az egyes funkciók felületeihez. Ezáltal nem kell egyszerre akkora adatmennyiséget kezelni, illetve a vezérlők és a dialógos elemek közti hivatkozások is egyszerűbbé válnak. A dialógus cseréje mellett arra is lehetőséget biztosít a rendszer, hogy egy korábban betöltött dialógust frissítsen a felület. Ez akkor lehet hasznos, ha a program dinamikusan generál dialógust. Ilyenkor a régi dialógusra váltás előtt elvégezzük a frissítést, és innentől kezdve már az új dialógusra

át lehet váltani ugyanazzal a hivatkozással. Az dialógusok azonosítására az elérési útjukból képzett hash kód szolgál, ezzel biztosítva az egyediséget.

A dialógusok cserélhetősége után felmerült az igény, hogy érdemes lenne valamilyen algoritmus szerint automatikus dialógusgenerálást is biztosítani a felülethez. Ennek a célja, hogy a néhány vezérlőből felépülő, egyszerű struktúrájú felületekhez ne a fejlesztőnek kelljen mindig kézzel elkészíteni a dialógust. Amennyiben megelégszünk egy egyszerű dialógus struktúrával, azt képes a rendszer elkészíteni a megadott GUI elemek alapján. Az algoritmus nagyon egyszerű, de mégis hatékony. Arra alapoz, hogy egy egyszerű felületen kétféle elem jelenhet meg. A felhasználó által kezelhető aktív elemek, például gombok, vagy passzív elemek, melyek csak információt jelenítenek meg, mint például a címke. A dialógusgeneráló függvény a megadott sorrendben végighalad az összes vezérlőn és megvizsgálja azokat. Amennyiben passzív elemet talál, egy egyszerű felolvasást szúr be a dialógusba. Ha viszont aktív elemet, akkor az elem felolvasása után egy felhasználói interakcióra várakozó input elemet is elhelyez. Emellett a bemeneti fókusz állítására szolgáló paraméterrel is ellátja ezt a részt. Ezáltal minden vezérlő megkapja a fókuszt, amint a hozzá tartozó dialóguselem feldolgozásra kerül. A fordított működés biztosítása érdekében a dialógus részeinek azonosítóit a grafikus elemekben tárolt dialógus azonosítókra állítja be a metódus. Így összességében egy olyan leírás jön létre, mely minden vezérlőhöz definiál egy dialógus blokkot, és a működés során ezek között lehet ugrálni. Ez megfelel a kézzel elkészített és a grafikus elemekhez illesztett dialógusok felépítésének. A vezérlők mellett még egy paramétert lehet megadni a függvénynek, mégpedig azt, hogy felülírja-e az esetleg már létező dialógust. Ez a megoldás növeli a funkció hatékonyságát, hiszen míg fejlesztés közben érdemes lehet mindig újrageneráltatni a leírásokat, addig lehet, hogy a végső programban már csak akkor szükséges ez, amikor még nem létezik a dialógus. Ezzel időt takaríthatunk meg a végleges program futásakor.

4 Az alkalmazás használata, megvalósított funkciók

4.1 Menetrendböngészés

A kliens indulásakor a 6. ábrán látható főmenüből választhatja ki a felhasználó a funkciót, amit használni szeretne. A képernyő teljes területét négy gomb foglalja el, melyek a program fő funkcióira továbbítanak. Jelenleg még csak a menetrendböngészés és a keresés érhető el ezek közül.

A menetrendek gombot megnyomva juthatunk el a böngészéshez tartozó első szintű szűréshez, vagyis az adatbázisban elérhető járműtípusokat megjelenítő oldalra. Ennek a felületnek az érdekessége, hogy a főmenüvel szemben nincs előre definiált XML leírása. Az adatbázisból lekérdezett adatok alapján a program építi fel a grafikus interfészt, mely itt egy LinearLayouton alapul. Ez is jól mutatja, hogy a multimodális vezérlőket ugyanúgy lehet alkalmazni, mint bármelyik beépített elemet. Az XML leírás a felület platform függetlenségét hivatott biztosítani, azonban ilyen módon is érdemes ezeket a vezérlőket alkalmazni, hiszen az integrált multimodalitás ugyanúgy működik ilyen helyzetben is. Ezen kívül ez a felület a dialógus leírását is automatikusan generálja a létrehozott vezérlők alapján.

6. ábra – Főmenü és járat típusok

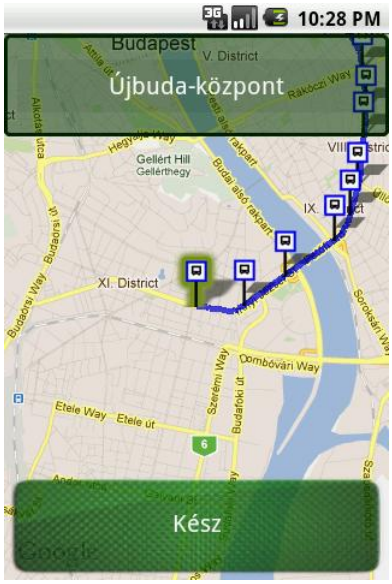
Ha a 6. ábra jobb oldalán látható listából kiválasztjuk a számunkra érdekes típust, eljutunk a böngészés következő felületre. Ezen az oldalon lehet kiválasztani a viszonylatot, melynek az adataira kíváncsiak vagyunk. Ennek a felületnek az érdekessége, hogy ugyan tartozik hozzá egy statikus XML leírás, mégis dinamikusan képes megjeleníteni az adatokat, hála a listás megjelenítésnek. Így itt a GUI leírása inkább úgy értelmezhető, mint egy séma, ami csak az elemek elhelyezkedését definiálja. Ezek mellett megfigyelhető a 7. ábrán a lista görgetésének a kiküszöbölésére használt két navigációs gomb is. Ezek a felhasználói felület fejlesztését ismertető részben leírtaknak megfelelően, blokkonként léptetik a listát a megfelelő irányba. Amennyiben az adott gomb már nem tud tovább vagy visszaléptetni, inaktívvá válik, ahogy a 7. ábrán a „Vissza” gombnál is látszik.

7. ábra - Villamos viszonylatok és a 4-es villamos megállói

A típusokat megjelenítő oldalhoz hasonlóan ez a felület is dinamikusan generált dialógust használ.

Ha kiválasztjuk a megfelelő viszonylatot, akkor a 7. ábra jobb oldalán látható felületre jutunk, ahol megálló alapján szűrhetjük a találatokat. Az adatbázis sajátosságai miatt egy megálló itt minden esetben csak az egyik irányba közlekedő járatokhoz tartozik, ezért az oldal alján helyet kapott egy irányváltó gomb is. Ennek megnyomására mindig az ellentétes irányú megállók jelennek meg, természetesen ennek megfelelő sorrendben is.

A jobb alsó sarokban található Térkép gomb segítségével nézetet válthatunk és a listás felsorolás helyett térképen jeleníthetjük meg az adott viszonylat megállóit. Ezen a felületen is kiválaszthatjuk, hogy melyik megállóból induló járatokra vagyunk kíváncsiak. Ehhez valamelyik megálló piktogramjára kell koppintani, majd a Kész gombbal tudjuk véglegesíteni a választásunkat, ahogy egy a 8. ábrán is látható.



8. ábra - Megálló kiválasztása térképen

A megálló kiválasztása után jutunk a dátum és idő beállítására szolgáló felületre, ami a 9. ábrán látható. Ennél a felületnél is az volt a szempont, hogy egyszerűen lehessen megadni az adatokat. Feltételezhető, hogy a felhasználók akkor böngészik a menetrendet, amikor valahova utazni szeretnének, ezért induláskor az aktuális dátum kerül beállításra, az időintervallum pedig az aktuális órától az adott nap 24 óráig lesz beállítva. Ezt természetesen tetszőlegesen lehet módosítani a felületen elhelyezett gombok segítségével. Ez az elrendezés jobban illeszkedik a multimodális felülethez, mintha a felhasználó tetszőleges adatokkal tölthetné fel az oldalt. Így sokkal hatékonyabban használható a beszédinterfész, és a felhasználónak is egyszerűbb a dolga, hiszen nem kell gépeléssel töltenie az időt. Ha megváltoztatjuk a beállításokat, azonban mégis a mai adatok érdekelnek, egyszerűen a „Ma” gomb megnyomásával alapállapotba hozhatjuk a felületet.

9. ábra - Időszak beállító felület és a szűrt menetrendi adatok

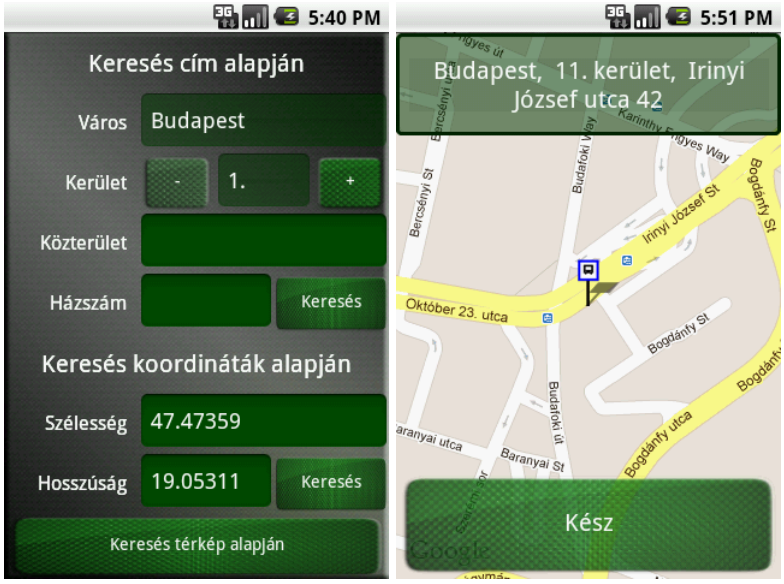
Ha megfelelően beállítottuk a kívánt időszakokat, a „Keres” gomb segítségével juthatunk a böngésző felület utolsó oldalára, ahol már az adott járat, adott megállóból történő indulásai láthatjuk időrendben felsorolva. Ezt a felületet a 9. ábra jobb oldala mutatja. A képernyő felső részén láthatjuk összefoglalva, hogy melyik viszonylat, melyik megállójából, milyen időintervallumban induló járatait jelenítettük meg.

4.2 Keresés

Útvonaltervezés előtt elengedhetetlen, hogy a felhasználó meg tudja adni, milyen címre kíván eljutni. Ennek meghatározása két lépésben történik. Először a felhasználó megadja a kívánt címet szövegesen, vagy koordinátákkal. Szöveges koordinátákkal történő megadás esetén külön lehet megadni kerületet, közterületet és házszámot. Mivel jelenleg csak Budapesten működik az alkalmazás, így a településnév nem változtatható.

Miután a felhasználó megadta a keresett cím általa ismert adatait, a rendszer megpróbálja azonosítani a megadott címet. Így elkerülhető, hogy az útvonaltervezés olyan címmel induljon, mely értelmetlen a szolgáltatás számára. Ha sikerült meghatározni a beírt adatok alapján mind a címet, mind a koordinátákat – attól függően, hogy mely paraméter hiányzott – a rendszer megjeleníti az eredményt egy térképen, illetve fel is olvassa azt megerősítésként. Ha a felhasználó által kívánt címet adta vissza a rendszer, a Kész gomb megnyomásával lehet beállítani az útvonaltervezés célpontjának, ellenkező esetben pedig a

térképen elhelyezett jelölő segítségével pontosíthatja azt. Ezt, illetve a keresési adatok megadására szolgáló felületet a 10. ábra szemlélteti.



10. ábra - Keresés funkciók tesztfelülete és a térkép

A kerület beállítása könnyen megoldható két gomb segítségével, azonban a többi paraméter beírása problémát okozhat vak és látássérült felhasználóknak. Erre a problémára megoldást a már említett Android rendszeren működő magyar beszédfelismerő beépítése jelentheti a későbbiekben.

Amennyiben a felhasználónak nincs pontos információja a címről lehetősége van arra is, hogy egyből térképen határozza meg azt.

Természetesen nem csak az útvonaltervezés célpontját lehet meghatározni a keresés funkcióval. A GPS által meghatározott pozíció helyett az útvonaltervezés kiindulópontját is meghatározhatjuk manuálisan, az előzőekben ismertetett módok valamelyikén.

4.3 Útvonaltervezés

Miután beállítottuk a kívánt úticélt és meghatároztuk kiindulási pontot, egy gombnyomással elindíthatjuk az útvonaltervezést. Alapértelmezésként a program az aktuális időpontra végzi ez a tervezést, azonban ez is módosítható tetszőlegesen a 11. ábrán látható Időpont beállítása gomb segítségével. Az időpont módosításához használt felület

úgy lett kialakítva, hogy csupán koppintásokkal bármit be lehessen állítani, eközben pedig minden lépésről visszajelzést kapjon a felhasználó. Ennek köszönhetően vak és látássérült felhasználók is könnyen kezelhetik a felületet. A Ma gomb segítségével egyszerűen vissza lehet állítani az aktuális dátumot és időpontot.

11. ábra - Útvonaltervezés adatainak beállítása

Miután minden paramétert beállítottunk, megkezdődik az útvonalterv lekérdezése több lépésben. Először a Google Directions API segítségével meghatározza az alkalmazás, hogy hány útvonaltervet tud ajánlani a megadott feltételeknek megfelelően, illetve meghatározza a hozzájuk tartozó alapvető adatokat. Ilyen az úti cél, a gyalogos illetve a tömegközlekedést használó szakaszok száma valamint az utazás időtartama.

Ezt követi az egyes utazások részleteinek meghatározása. Itt jönnek létre az utazások tényleges szakaszai a visszakapott szöveges utasítások alapján. Minden szakaszhoz tartozik egy leírás, mely a szakaszon való végighaladást segíti, egy kezdő és egy végpont, valamint egy koordináta lista az útvonalról. A kezdő és a végpont a szakasz típusától függően lehet cím, vagy egy a BKV GTFS adatbázisban tárolt megálló.

Gyalogos szakaszok esetén a szakaszhoz tartozó útvonal koordináták meghatározását a Google Maps gyalogos útvonaltervezőjétől kési le a program. Tömegközlekedést használó szakaszok esetén viszont a BKV GTFS adatbázisából határozza meg az alkalmazás az útvonal koordinátáit. Ehhez ismerni kell, hogy milyen

viszonylat, milyen megállói között közlekedünk épp. A Google Directions segítségével meghatározott adatok mindezt tartalmazzák. Egyedül a viszonylat típust kell a GTFS adatbázis alapján módosítani, ugyanis a Google Maps nem különbözteti meg a troli, éjszakai és sima autóbuszokat, a GTFS adatbázis viszont igen.

Az egyes szakaszok adatainak meghatározása után egy olyan útvonalnak kell kialakulnia, melynek minden eleme kapcsolódik az előtte és utána található szakaszokhoz. Például ha csupán egy járatot kell használnunk, akkor lesz egy gyalogos szakaszunk a kiindulási cím és a járat egyik megállója között, majd egy tömegközlekedési szakasz a járat két megállója között, végül pedig ismét egy gyalogos szakasz az előző rész második megállójától a megadott célpontig.

Ebben a folyamatban talán a leginkább időigényes feladat a szükséges járatok megállói közötti szakaszok beállítása. Ugyanis a Google Mapstől csupán egy viszonylatot és két megálló nevet kapunk vissza. Ezek alapján meg kell találni, hogy milyen koordináták tartoznak az adott megállókhoz, illetve a közöttük található útvonal koordinátáit is meg kell határozni.

A feladatot bonyolítja, hogy bizonyos szakaszadatok csak akkor határozhatók meg, ha már a soron következő szakaszt is meghatároztuk. Például egy gyalogos szakasz végén található megálló koordinátáinak megadásához ismerni kell az azt követő tömegközlekedési szakaszt. Ennek a fordítottja is előfordulhat, amikor egy tömegközlekedési szakaszt követő gyalogos szakasz indulópontját kell meghatározni a korábbi szakasz alapján.

Ha sikerült meghatározni minden útvonal adatait, egy egyszerű felületen kapjuk meg az adott úticélhoz tartozó útitervek rövid kivonatát, ami a 12. ábrán látszik. Ez tartalmazza az utazások idejét, illetve az átszállások számát.

12. ábra - Útitervek

4.4 Navigálás

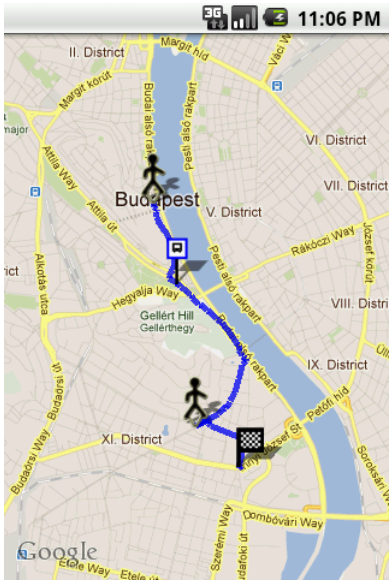
A megtervezett útvonalakat az alkalmazás képes szöveges, illetve grafikus formában is megjeleníteni. Mivel korábban minden szükséges adatot meghatároztunk, itt már nincs szükség Internet kapcsolatra.

13. ábra - Szöveges navigációs felület

Szöveges megjelenítés esetén az alkalmazás egy listában jeleníti meg az utazás egyes szakaszaihoz tartozó instrukciókat, ami a 13. ábrán látható. Ez magában foglalja

gyalogos szakasz esetében az útba eső utcákat, a becsült távolságokat, illetve a szükséges irányváltásokat. Tömegközlekedési szakasz esetén megtalálható itt a járatszám, a megállók ahol fel, illetve le kell szállni, valamint a szakaszhoz tartozó, tervezett menetidő.

A felhasználónak itt lehetősége van arra is, hogy elmentse a teljes útitervet. Ez a funkció sajnos még nem készült el, de remélhetőleg a szóbeli előadásig ez is megvalósul. Lehetőség van az útitervek térképes megjelenítésére is, ez a 14. ábrán látható.



14. ábra - Térképes navigációs felület

Az alkalmazás térképes nézetben is egyértelműen jelöli, hogy mely szakaszokon kell gyalogolni és hol lehet tömegközlekedést igénybe venni, valamint a célpont is egyértelműen látható. A valósidejű nacigáláshoz szükséges GPS kezelést megoldó modul már elkészült, azonban még nem lett az alkalmazás többi komponenséhez illesztve. Reményeim szerint ez is megvalósulhat a szóbeli előadásig.

4.5 Offline útvonalkezelés

Ahogy az előző fejezetben is említettem, sajnos még nem készült el az útvonalak mentési lehetősége, így egyelőre nem működik az offline útvonalkezelés. Azonban az útvonalakat tároló struktúra már készen áll, ha mentési funkció megvalósítása után már könnyen megoldható lesz az útitervek offline betöltése és böngészése.

5 Eredmények

A munka során komoly kihívást jelentett a BKV adatbázis létrehozása és csatolása a jelenlegi rendszerhez, a multimodális felhasználói felület kialakítása és a rendszer vak és látássérült felhasználók számára elérhetővé tétele.

Az adatbázis feltöltésével kapcsolatos első kísérlet kudarca után gyorsan sikerült létrehozni a GTFS alapú adatbázist. Később az ehhez kapcsolódó adminisztrációs alkalmazás elkészítésével pedig már könnyen frissíthetővé is vált a rendszer. Ezt az alkalmazást a későbbiekben akár egy olyan architektúra is kialakítható, mely képes folyamatosan figyelni a BKK holnapján elérhető GTFS adatokat, és meghatározott időközönként letölteni őket, majd a teljes adatbázis frissíteni ennek alapján.

A GTFS struktúra összetettsége, a nagy adatmennyiség és a ritka kivételek miatt több iterációra is szükség volt, mire minden funkcióhoz sikerült létrehozni a megfelelő lekérdezést. Lehetnek még rejtett apróbb hibák, ezeket egy teljes, átfogó teszttel lehetne csak kiszűrni, de minden eddig előfordult hibát sikerült már kiküszöbölni a rendszerből.

Nagy hangsúlyt kapott a kliens, azon belül is a multimodális felhasználói felület kialakítása. Az itt elvégzett fejlesztések egyértelműen sikeresek voltak, hiszen most már az eddigieknél is egyszerűbben lehet kialakítani az alkalmazás megjelenését az XML alapú leírás felhasználásával. Ezt a leírást, mint egy sémát felhasználva, a felületet testreszabva sikerült egy kellemes megjelenésű alkalmazást létrehozni. Az „virtuális egér” vezérlési technikával pedig sikerült kiküszöbölni a rendszerből a beszédfelismerő eddigi hiánya által okozott problémát, ami a TTS funkcióval együttműködve nagymértékben javítja a használhatóságot. Ennek segítéségével vak és látássérült felhasználók is nagy biztonsággal kezelhetik majd az alkalmazást, hiszen az minden lépésről, illetve az alkalmazás aktuális állapotáról tájékoztatást ad.

A felhasználói felület működésének pontosabb értékeléséhez szükséges lenne vak vagy látássérült felhasználók által végzett tesztek készítésére is, azonban erre sajnos még nem volt alkalom. Bízom benne, hogy a közeljövőben sikerül majd ilyen jellegű teszteket elvégezni, melyek alapján tovább lehet javítani a rendszer hatékonyságát.

Kijelenthető, hogy minden eddig elkészített funkció megfelel az elvárásoknak. A mobil kliens és a szerver közti kommunikáció lehetővé teszi, hogy a későbbiekben akár más mobil platformokon működő, vagy webes klienst is létrehozzunk a rendszerhez.

Az eddigi munkám során elkészült szoftver elemeket az 3. táblázat - Elkészült szoftverkomponensek foglalja össze.

Komponens	Forrás mérete	Komponens szerepe
NavDBAdmin	1200 sor	.NET alapú adatbázis adminisztrációs szoftver
PHP interfész[2]	Összesen 460 sor, (6 fájl)	Szerver és kliens közötti kommunikáció megvalósítása
BPNavigate	5600 sor, (34 osztály)	Android alapú kliensalkalmazás, mely a menetrendböngésző és az útvonaltervező funkciókat biztosítja
XML leírások	1780 sor	Multimodális felhasználói felületet definiáló XML-ek
COM.MMUI	200 sor kiegészítés (összesen 4850 sor)	Multimodális XML alapú felhasználói felületet tartalmazó package kiegészítése.

3. táblázat - Elkészült szoftverkomponensek

[2] Ez nem tartalmazza a BKV holnap feldolgozásához használt PHP oldalakat, hiszen ezeknek már nincs szerepe a rendszerben.

6 Jövőbeli tervek

A továbbiakban a legfőbb cél a még hiányzó funkciók, vagyis a valósidejű navigálás, GPS kezelés és az offline útvonalkezelés implementálása. Emellett még néhány beállítási lehetőséggel is ki fog egészülni az alkalmazás, például igény szerint ki- és bekapcsolható lesz a multimodális felhasználói felület. Szintén kulcsfontosságú a felhasználói tesztek elvégzése vak és látássérült felhasználókkal.

A jelenlegi funkciók jó alapot nyújthatnak a program további fejlesztéséhez. A menetrendböngésző funkciót például tovább lehetne bővíteni úgy, hogy az egyes megállókból elérhető csatlakozásokat is megjelenítse. Egy másik lehetőség a járatokhoz tartozó megállási időket ne csak a megállók alapján lehessen szűrni, ahogy jelenleg történik, hanem utazás alapján is, így végig lehessen követni, hogy egy adott megállóból adott időpontban induló járat mikor ér a viszonylat további megállóiba.

A navigálás funkciót ki lehetne egészíteni utazás közbeni újratervezéssel, vagy akár irányérzékelésen alapuló pontosabb, részletesebb utasításokkal is.

Továbbá érdemes lenne megvizsgálni a korábban ismertetett beszédfelismerő beépítésének lehetőségét, és ezzel elérni a multimodális felhasználói élmény további javítását. Emellett szükség lenne vak vagy látássérült felhasználók által végzett tesztelésre is, mely alapján tovább lehetne optimalizálni a multimodális felhasználói felületet, mind a grafikus elemek, mind a beszédinterfész szintjén.

Célom, hogy egy teljes értékű, a gyakorlatban is hatékonyan alkalmazható tömegközlekedést segítő navigációs rendszer valósuljon meg vak és látássérült felhasználók részére.

Irodalomjegyzék

[1] Ponte.hu Kft, Smartcity alkalmazás bemutatása, http://www.ponte.hu/termekeink/otelapps/smartcity.html (2011. 10. 26.)

[2] Google mobil térkép, http://www.google.com/intl/hu_ALL/mobile/maps/ (2011. 10. 26.)

[3] NDrive navigáció honlapja, http://www.ndrive.com/, (2011. 10. 26.)

[4] Érintőképernyő vakoknak, Stanford Egyetem kutatása, http://engineering.stanford.edu/news/stanford-summer-course-yields-touchscreen-braille-writer, (2011. 10. 24.)

[5] SVOX bemutatása, http://svoxmobilevoices.wordpress.com/about/, (2011. 10. 26.)

[6] Nuance vállalat, http://www.nuance.com/, (2011. 10. 26)

[7] General Transit Feed Specification leírása, http://code.google.com/intl/hu-HU/transit/spec/transit_feed_specification.html, (2011. 10. 26.)

[8] BPMenetrend, http://imind.hu/mobile/bpmenetrend/, (2011. 10. 26.)

[9] Android TTS működése, http://developer.android.com/resources/articles/tts.html, (2011. 10. 26.)

[10] Android Speech Input működése, http://developer.android.com/resources/articles/speech-input.html, (2011. 10. 26.)

[11] Tóth Bálint, Németh Géza, Hidden Markov Model Based Speech Synthesis System in Hungarian, Infocommunications Journal, Volume LXIII, 2008/7, pp. 30-34

[12] KML dokumentáció, http://code.google.com/intl/hu-HU/apis/kml/documentation/kmlreference.html, (2011. 10. 25.)

[13] Braille írás, http://en.wikipedia.org/wiki/Braille, (2011. 10. 26.)

[14] Shaun K. Kane, Jeffrey P. Bigham és Jacob O. Wobbrock, Slide Rule: Making Mobile Touch Screens Accessible to Blind People Using Multi-Touch Interaction Techniques, University of Washington, http://faculty.washington.edu/wobbrock/pubs/assets-08.pdf, (2011. 10. 26.)