Slovenská technická univerzita

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijné odbory : Informačné systémy a Softvérové inžinierstvo

Mobilný cestovný poriadok pre iPhone

Dokumentácia k inžinierskemu dielu

Tím 14 : iTEAM
Martin Jačala
Marek Brandobúr
Michal Macko
Michal Hrdina
Martin Blažko
Hana Časnochová

Vedúci tímu :
Ing. Michal Čerňanský, PhD.

tp14@googlegroups.com

OBSAH

ÚVOD

Účel dokumentu

Tento dokument bol vytvorený v rámci predmetu Tvorba softvérového systému v tíme v akademickom roku 2009/2010. Slúži ako projektová dokumentácia pre riešený projekt a jeho cieľom je priblížiť priebeh riešenia daného projektu a jeho jednotlivé etapy.

Scrum

Hneď v úvode je nutné podotknúť, že na vypracovanie projektu sme si zvolili agilnú metódu softvérového vývoja SCRUM. Agilný vývoj používa inkrementálny spôsob vývoja, časté testovanie a zároveň núti zákazníka byť zainteresovaným do vývoja. Jeho najväčšou výhodou je rýchla adaptácia na zmeny. Metóda Scrum pozostáva zo šprintov, v rámci ktorých sa vytvára inkrementálne komplexná časť softvéru. Výsledkom tohto čiastkového procesu je už otestovaná časť softvéru. Na tomto základe je aj ďalej členený dokument.

Štruktúra dokumentu

Dokument je rozdelený na niekoľko častí. V nasledujúcej kapitole sa nachádza opis jednotlivých šprintov. V prílohe A je uvedené testovanie funkcionality, ktorá vznikla v druhom šprinte. V prílohe B sa nachádzajú návrhy dizajnu.

Zadanie

Mnohí účastníci mestskej autobusovej prepravy vysoko oceňujú portál "imhd.sk" prevádzkovaný občianskym združením "mhd.sk". Portál poskytuje informácie týkajúce sa mestskej hromadnej dopravy pre viaceré slovenské mestá. Medzi jeho najvýznamnejšie služby patrí možnosť prehliadania a tlače cestovných poriadkov jednotlivých liniek a možnosť plánovania cesty mestskou hromadnou dopravou. Zaujímavým doplnením existujúcich služieb portálu by mohlo byť poskytovanie služieb pre mobilné zariadenia.

V poslednej dobe silnie ponuka aj popularita mobilných zariadení schopných vykonávať používateľom inštalované aplikácie. Možnosti mobilných zariadení sú čoraz širšie a tiež výrazne stúpa "príjemnosť" používania aplikácií (angl. User Experience). Tiež mobilní operátori poskytujú služby umožňujúce cenovo dostupné pripojenie na Internet prostredníctvom mobilných zariadení. Významným z hľadiska rozšírenia ale aj možností je zariadenie iPhone (Apple) s operačným systémom OS X pre iPhone disponujúce dotykovým displejom, GPS, akcelerometrom, a vo verzii 3GS aj kompasom. Na trhu je tiež stále väčšie množstvo zariadení s OS Microsoft Windows Mobile (napr. HTC Touch Pro 2) alebo s OS Symbian (napr. Nokia N97). Novým operačným systémom je Android OS vyvinutý spoločnosťou Google (napr. HTC Hero).

Navrhnite, implementujte a otestujte riešenie umožňujúce používateľom prístup k informáciám mestskej hromadnej dopravy prostredníctvom mobilných zariadení. Zamerajte sa na "príjemnosť" použitia služby a možnosť využitia vlastností mobilného telefónu iPhone. Analyzujte vhodnosť a možnosti rozšírenia služby aj na linky medzimestskej dopravy či železničnej dopravy.

Zvážte potrebu sa podrobnejšie venovať nasledujúcim bodom:

- Overte možnosti použitia mobilného zariadenia ako polohovacieho zariadenia, okrem zemepisných súradníc z GPS overte možnosti použitia kompasu a tiež akcelerometra, overte presnosť poskytovaných údajov a tiež možnosť ich spresnenia filtrovaním.
- Navrhnite aplikáciu pre iPhone využívajúcu polohovanie na spresnenie oblasti záujmu používateľa.
- Navrhnite spôsob ukladania informácií do vyrovnávacej pamäte aplikácie automaticky na základe preferencií používateľa (modelovanie potrieb používateľa), riešte možnosť použitia služby aj bez pripojenia na Internet.
- Navrhnite spôsob analyzovania kvality modelovania potrieb používateľa, kvality navrhnutého používateľského rozhrania, spôsob analýzy dopytov realizovaných používateľom a pod.
- Venujte sa bezpečnosti prípadných citlivých údajov a špeciálne sa zamerajte na otázku možnej straty súkromia používateľov (sledovanie ich polohy, identifikácia používateľa na základe korelácie medzi zaznamenanými údajmi a pod.).

Slovník pojmov a skratiek

Scrum – agilná metóda softvérového vývoja

GPS - satelitný navigačný systém používaný na zistenie presnej pozície

Mac OS X - operačný systém pre počítače Macintosh

iPhone SDK - softvérový balík na vývoj aplikácií pre iPhone a iPod touch

XCode – vývojové prostredie pre Mac OS X

User story – požiadavka na softvérový systém formulovaná pár vetami v každodennom

jazyku používateľa

MHD – mestská hromadná doprava

REST – z anglického Representational State Transfer

Mock – objekt, ktorý kontrolovane napodobňuje správanie reálneho objektu

POI – bod záujmu, z anglického point of interest

ŠPRINTY

1. šprint

Prvý šprint trval od 6.10.2009 do 20.10.2009. V rámci tohto šprintu sme zrealizovali dve User stories. Zaoberali sme sa vytváraním rannej verzie aplikácie a snažili sa o oboznámenie používateľov so vznikajúcou aplikáciou.

Hello world aplikácia

Používateľ spustí aplikáciu na mobilnom zariadení. Aplikácia sa bez problémov spustí a zobrazí sa okno, ktoré bude mať tri časti. V spodnej časti budú tlačidlá, v strednej obsah a v hornej kontextovo závislé možnosti. Zároveň prevezme súbor zo servera.

Analýza

Už existuje viacero služieb, ktoré poskytujú jednoduchý prístup k cestovným poriadkom MHD. Na portáli imhd.sk je možné vyhľadávať spojenia medzi zastávkami, tak isto ako prezerať, či prípadne si vytlačiť cestovný poriadok podľa čísla linky. Je tu aj statická mapa s vyznačenými zastávkami. Portál však nie je veľmi prispôsobený takým ľudom, ako napríklad turistom, ktorí nevedia názov zastávky, nevedia kde sa nachádza a ako vyzerá, prípadne vôbec netušia kde práve sú.

Na rad prichádza flexibilnosť mobilného zariadenia, ktoré je možné mať stále poruke. Myslíme si, že aplikácia podporovaná mobilným zariadením prináša zo sebou mnoho výhod oproti webovému portálu. Netreba si plánovať cestu vopred, je možné zistiť spojenie priamo v teréne.

Po zvážení sme sa rozhodli vyvíjať aplikáciu pre mobilné zariadenie iPhone od firmy Apple s operačným systémom OS X. Využijeme prostredie XCode a platformou iPhone SDK 3.0 ktoré, okrem iného, využíva pre nás podstatné GPS. Na vytvorenie grafického užívateľského prostredia použijeme nástroj Interface Builder, ktorý je súčasťou balíka určeného pre vývojárov iPhone. Distribúcia, ktorá je zabezpečená pomocou kanála App Store, umožňuje každému používateľovi prístup k aplikácii. A voľne šíriteľné aplikácie sú na App Store umiestnené bezplatne.

Návrh

Hlavným cieľom je získanie zručností pri používaní nástrojov, v ktorých budeme ďalej implementovať aplikáciu. Architektúra klient-server.

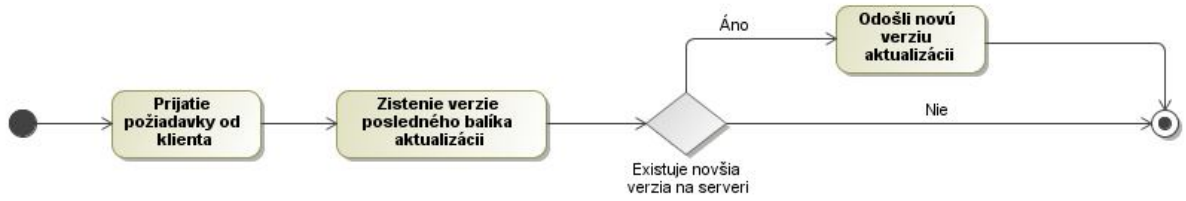
Samotná aplikácia po spustení bude obsahovať v spodnej časti obrazovky jednoduché tlačidlo, pričom po jeho stlačení sa vyšle požiadavka na sever a prevezme sa odpoveď - reťazec zo servera, ktorý sa vypíše v strednej časti obrazovky.

Implementácia

Implementáciu sme rozdelili do dvoch častí, na klientskú a serverovú časť.

Klientskú časť sme implementovali v jazyku ObjectiveC v prostredí XCode, Aplikácia bola vytvorená pomocou šablóny „View based application“, ktorá je súčasťou vývojového prostredia. Používateľské grafické rozhranie obsahovalo dva prvky – tlačidlo a textové pole.

Serverová časť je implementovaná v jazyku php. Postupnosť krokov, ktoré vykonáva server, sa nachádza na Obr.1.



Obr. 1. Postupnosť krokov na serveri.

Testovanie

Keďže ide o veľmi jednoduchú aplikáciu, test pripojenia na server spočíva v samotnom zobrazení verzie prevzatého súboru.

Propagácia produktu

Zlepšiť informovanosť záujemcov o produkt.

Analýza

Na to, aby sme získali čo najviac používateľov našej aplikácie, je nutné nájsť spôsob oslovenia tých potencionálnych. Dobré je osloviť obyvateľov Bratislavy. Zistiť zhruba koľko ľudí reálne používa iPhone, kvôli ďalšej funkcionalite a prípadnému zapojeniu používateľov. Vhodným spôsobom na interakciu je web. Webová stránka opisujúca produkt musí mať priliehavý a ľahko zapamätateľnú doménu.

Návrh

Stránka produktu bude v slovenskej aj anglickej verzii. Jej dizajn musí byť prívetivý a prehľadný. Zároveň bude optimalizovaná pre vyhľadávače. V úvodnej fáze sa na nej budú nachádzať len informatívne údaje o plánovanej funkcionalite, neskôr pribudnú ďalšie podrobnosti a možnosť prevzatia samotnej aplikácie.

Za názov sme zvolili iTransport.

Anketa bude vytvorená tak isto v slovenskej aj anglickej verzii – chceme osloviť aj zahraničných turistov. Z ankety zistíme, ako rozšírené sú mobilné zariadenia, pre ktoré implementujeme aplikáciu, a akú ďalšiu funkcionalitu, by budúci používatelia privítali. Anketa bude pozostávať z niekoľkých jednoduchých otázok.

Implementácia

Dizajn stránky sa nachádza v prílohe . Na jej implementáciu sme použili html a css.

Anketa, implementovaná v php, pozostáva z týchto otázok:

- Aký operačný systém používate v MT?
- Koľko ľudí používajúcich iPhone poznáte?

- Aká funkcionalita by sa vám páčila?

Testovanie

Aplikácia je určená pre širokú verejnosť, preto je potrebné zaručiť jej správne zobrazovanie v najčastejšie používaných webových prehliadačoch. Stránku sme otestovali pomocou nástroja Browser Shots – vybrali sme prehliadače Opera, Mozilla Firefox, Safari, Internet Explorer 7. Odhalené nedostatky pri zobrazovaní sme opravili, najviac opráv bolo nutných pre prehliadač Internet Explorer 7.

2. šprint

Druhý šprint sme začali 20.11.2009 a ukončili 3.11.2009. Cieľom bolo dopĺňanie funkčnosti do aplikácie a ďalšia propagácia. Celkovo sme opäť identifikovali dve User stories.

Funkcionalita „najbližšie spoje“

Používateľ chce informácie o odchodoch autobusov z najbližších zastávok v najbližšom čase. Klikne preto na prvé tlačidlo v spodnej (tlačidlovej) časti a v strednej (obsahovej) časti sa objaví zoznam s informáciami o spojoch. Ku každému spoju zobrazí:

- zastávku, z ktorej spoj ide
- číslo spoja (veľké a výrazné)
- smer spoja
- čas kedy ide
- čas, ktorý zostáva do odchodu

Analýza

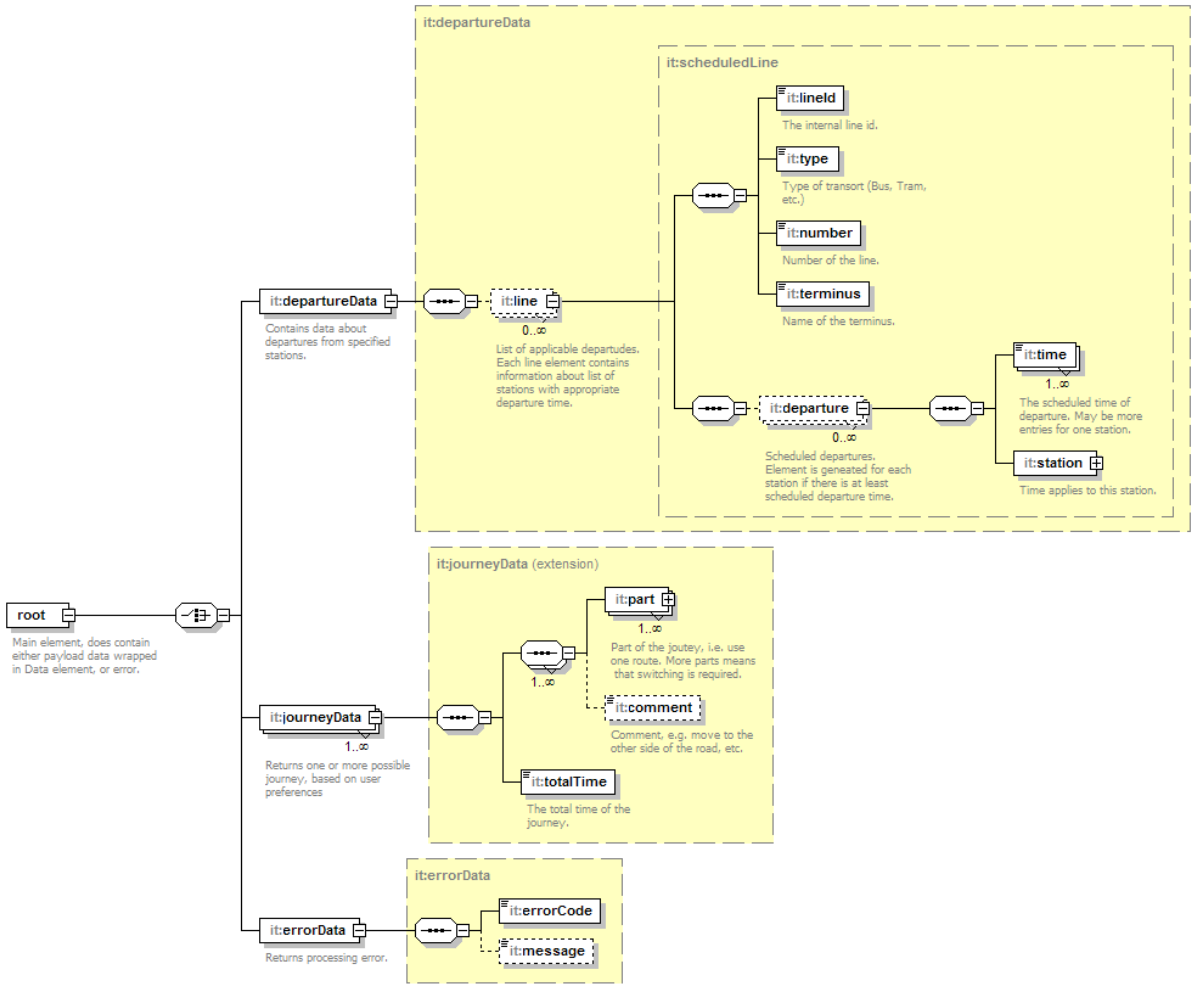
Na začiatok je nutné vytvoriť databázovú štruktúru, ktorá bude obsahovať zastávky, časy odchádzajúcich spojov a aké linky z akej zastávky odchádzajú. Pre každú zastávku musíme vedieť jej zemepisnú šírku a dĺžku. Tak isto je potrebné rozlišovať medzi zastávkami, ktoré majú rovnaký názov, ale rôznu polohu, ako napríklad zastávky Račianske mýto. Zohľadniť treba aj smer linky, nielen jej číslo. Databázová štruktúra musí byť tiež ľahko rozšíriteľná o ďalšie typy údajov, ktoré budú potrebné v nasledujúcom vývoji aplikácie.

Zobrazovanie požadovaných informácií na obrazovke mobilného zariadenia musí byť jasné a prehľadné. Požadované informácie musia byť zobrazené ihneď po zapnutí aplikácie.

Pri návrhu protokolu komunikácie klient-server musíme brať do úvahy požiadavky na prepoužiteľnosť a škálovateľnosť riešenia.

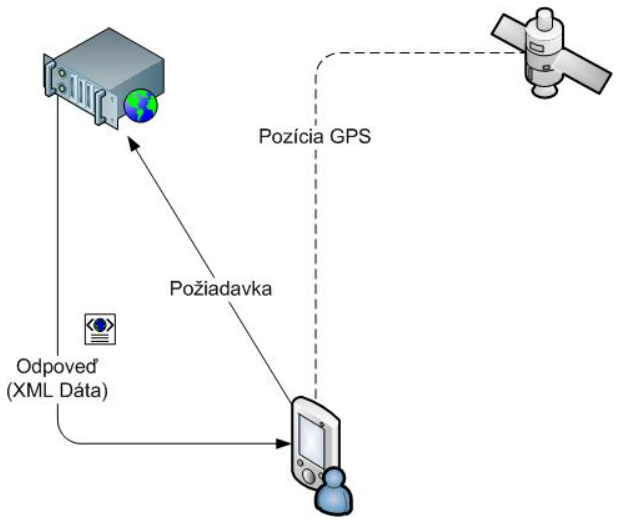
Návrh

Prvý krok pri návrhu je vytvorenie sformalizovania komunikačného protokolu medzi klientom a serverom. S ohľadom na požiadavky definované v analýze, sme sa rozhodli využiť webové služby založené na štandarde REST, teda posielanie správ pomocou XML. Formát dát je špecifikovaný pomocou schémy XSD, ktorá zároveň slúži ako referencia pri implementovaní ostatných častí (Obr. 2).



Obr. 2. XSD schéma.

Klientská časť pozostáva s aplikácie, ktorá komunikuje so serverom pomocou protokolu HTTP. Po zavolaní webovej služby a predaní parametrov dostáva odpoveď vo formáte XML (**Obr. 3**).



Obr. 3. Model komunikácie klient – server.

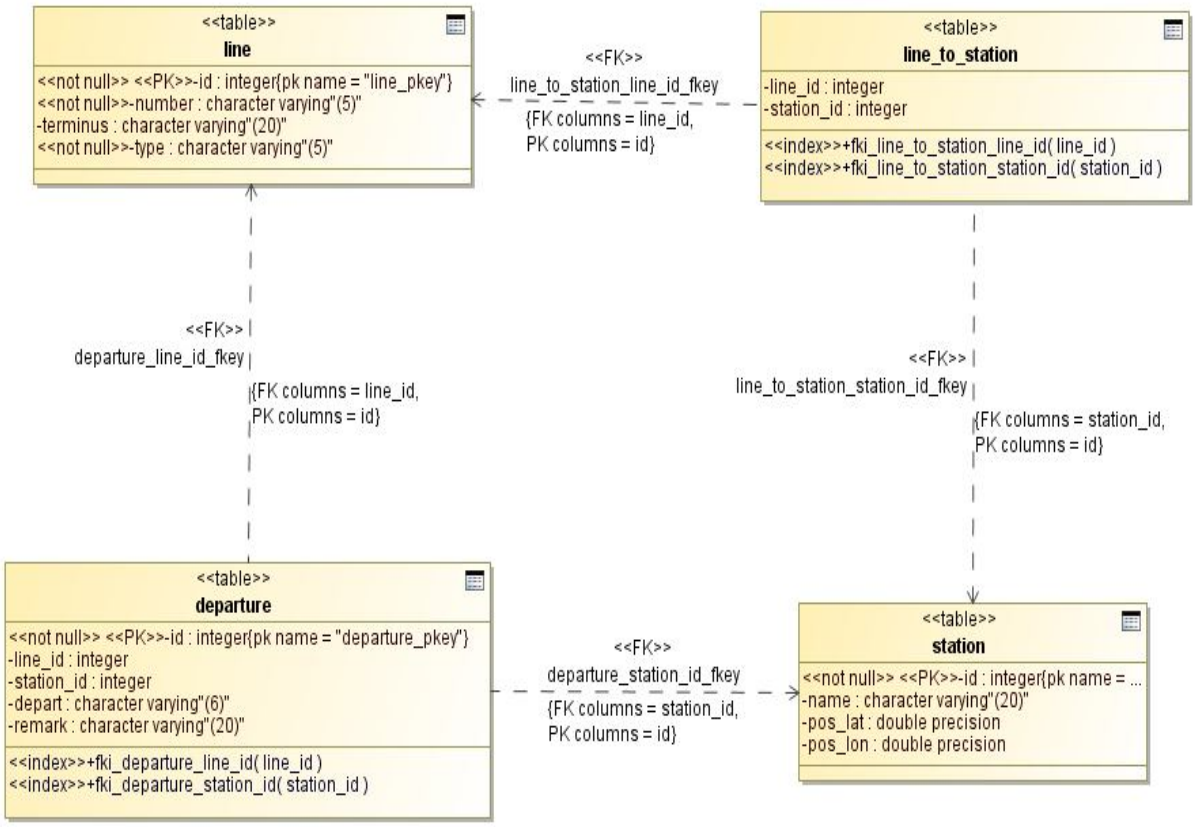
Pre samotnú funkcionalitu bude vytvorená webová službu, ktorá poskytuje informácie o odchodoch spojov z lokality v špecifikovanom čase. Pre zadanú vstupnú polohu nájde všetky zastávky v okolí definovanej veľkosti. Pre každú zastávku vyhľadá spoje obsluhujúce danú zastávku, ku každej dvojici spoj - zastávka bude nájdený najbližší odchod.

Použitie GPS v simulátore iPhonu má obmedzené možnosti, pretože simulátor udáva vždy iba jednu GPS pozíciu v USA. Pozíciu teda bude nutné simulovať pomocou dočasného objektu (Mock). Počas prvých fáz vývoja klienta bude pozícia simulovaná v blízkosti školy.

Implementácia

Serverovú časť sme sa rozhodli implementovať v skriptovacom jazyku Python, kvôli lepšej výkonnosti, nakoľko na server budú kladené vysoké nároky, súvisiace s plánovaním cesty. Za databázový systém sme si zvolili voľne šíriteľný objektovo-relačný PostgreSQL 8.4.

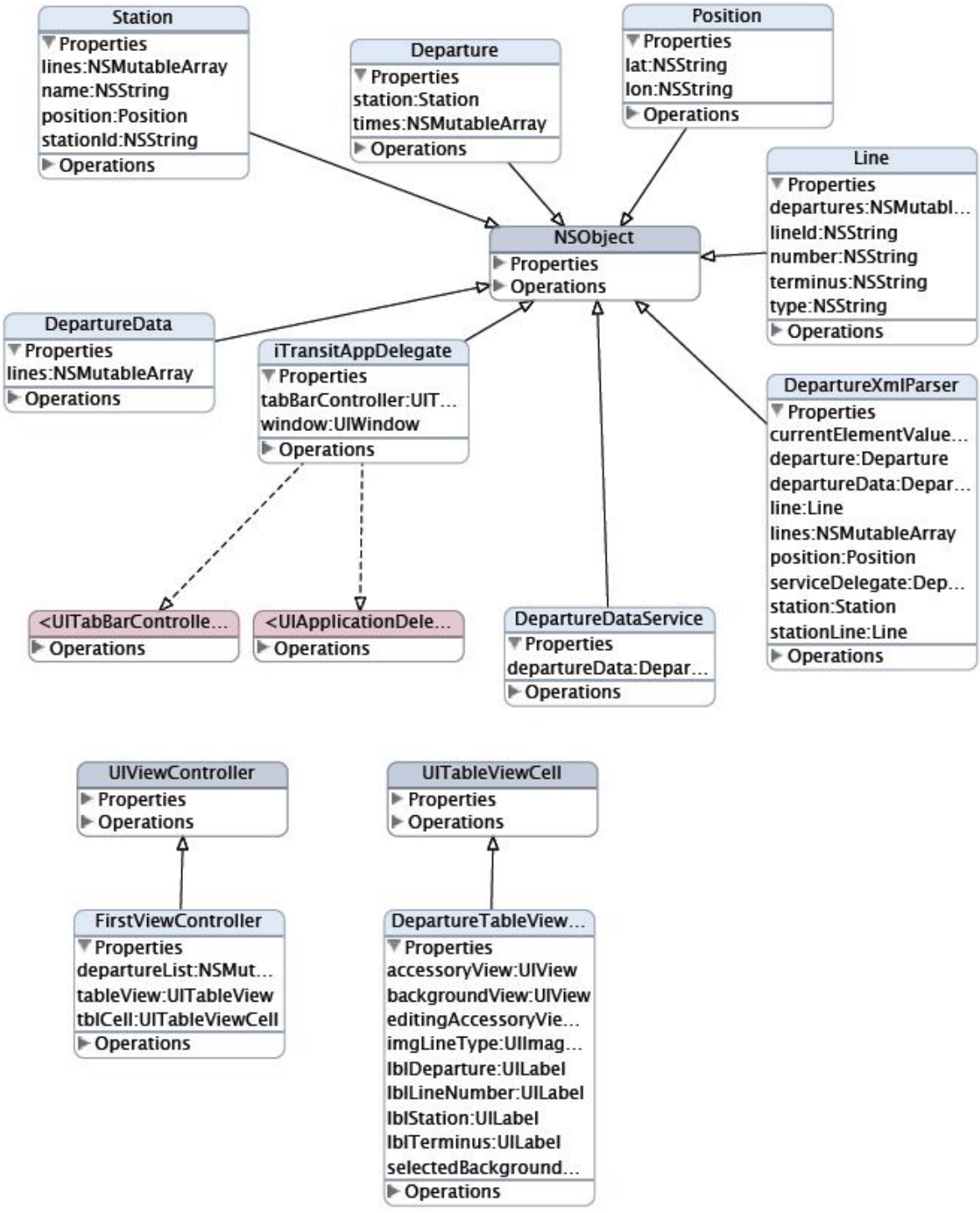
Implementácia logického modelu, špecifikovaného ako XSD schéma, je na strane databázových objektov vyjadrená pomocou fyzického dátového modelu (**Obr. 4**).



Obr. 4. Fyzický dátový model.

Klientská časť je implementovaná na základe návrhového vzoru „model-view-controller“.

Triedy Line, Position, Departure, Station a DepartureData predstavujú vrstvu „model“ v aplikácii. Sú implementované podľa špecifikácie služby definovanej v XSD schéme. Tieto objekty sú inštancované v triede DepartureXmlParser. Vrstva „controller“ manipuluje s dátami na základe požiadaviek GUI. Trieda DeparureDataService prijíma požiadavky z objektov „controller“ a slúži ako obaľovač (wrapper) na REST služby bežiace na aplikačnom serveri. Diagram tried sa nachádza na Obr. 5.



Obr. 5. Diagram tried.

Nami implementované používateľské rozhranie spĺňa požiadavky User story (Obr. 6.).

Obr. 6. Výstupná obrazovka aplikácie.

Testovanie

Testovacie scenáre, slúžiace ako integračné testy pri testovaní danej User story, sú uvedené v prílohe A.

Propagácia produktu, druhá časť

Ešte viac zlepšiť informovanosť záujemcov o produkt.

Analýza

Súčasný dizajn stránky propagujúcej produkt je nevyhovujúci, preto je nutná jeho zmena. Keďže vyvíjame aplikáciu pre iPhone, aj stránka by sa mala podobať na stránky produktov určených pre iPhone.

Doména iTransport.sk nie je voľná, preto je nevyhnutné prehodnotiť názov.

Ďalším spôsobom zvýšenia záujmu o produkt je prezentácia stránky produktu pomocou reklamy. Forma prezentácie, ktorá podľa nás najviac osloví želanú skupinu potencionálnych používateľov, je reklamný prúžok (banner), zverejnený na už existujúcej a pomerne navštevovanej webovej stránke. Ďalšou formou je plagát, ktorý by tiež upozorňoval na existenciu stránky.

Návrh

Nový dizajn stránky sa nachádza v prílohe B.

Zmena ankety spočíva v rozšírení otázok a taktiež v ich postupnom zobrazovaní užívateľovi.

Názov sme zmenili na iTransit, pričom doména iTransit.sk je prístupná.

Plagátik na podporu návštevnosti webovej stránky produktu, v prílohe B, sa vyvesí na frekventované miesta v Bratislave..

Implementácia

Ukážka reklamného prúžku sa nachádza v prílohe B. Bol vytvorený technológiou Flash. Stránka, ktorá nám umožnila jeho zverejnenie je fei.sk.

3. šprint

Tretí šprint sa začal 3.11.2009 a skončil 18.11.2009. Pracovali sme na jednej User story, ktorej cieľom bolo ďalšie dopĺňanie funkčnosti do aplikácie.

Funkcionalita „kde som“

Používateľ chce vedieť kde v lokalite je, a aké sú najbližšie zastávky poprípade iné body záujmu (POI). Klikne preto na druhé tlačidlo v spodnej (tlačidlovej) časti a v strednej (obsahovej) časti sa objaví mapa s vyznačenou aktuálnou polohou používateľa a najbližšími zastávkami.

Analýza

Pre jednoduché zobrazenie mapy v iPhone SDK existuje framework MapKit. Na zobrazenie jednotlivých POI na mape potrebujeme ich GPS súradnice. Zastávka s rovnakým menom je zvyčajne fyzicky umiestnená na rôznych miestach – minimálne rôzna pre opačné smery. Je nutné na jeden názov zastávky namapovať viacero GPS súradníc.

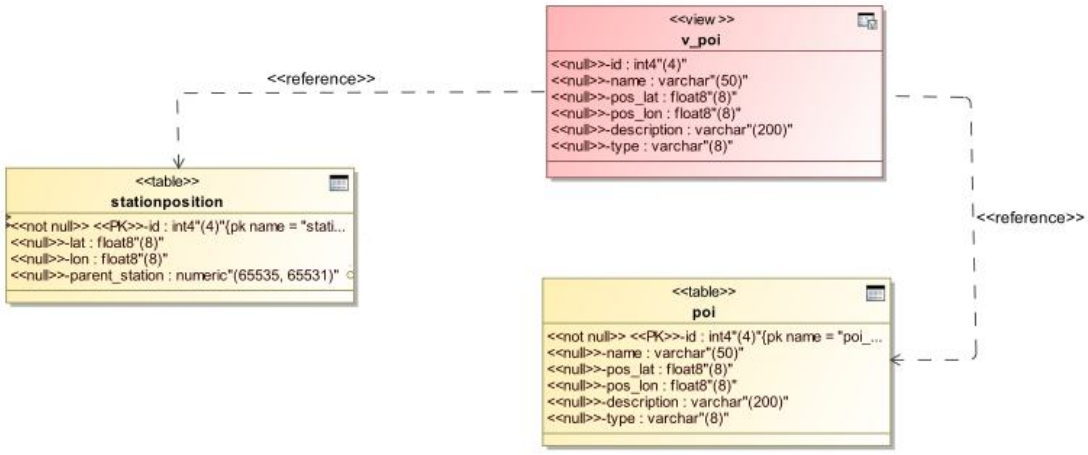
Návrh

Nová webová služba getPoi (príloha A), založená na štandarde REST, poskytuje GPS súradnice vybraného typu POI. Pre zadanú vstupnú polohu nájde všetky POI (zadaného typu) v definovanom okolí.

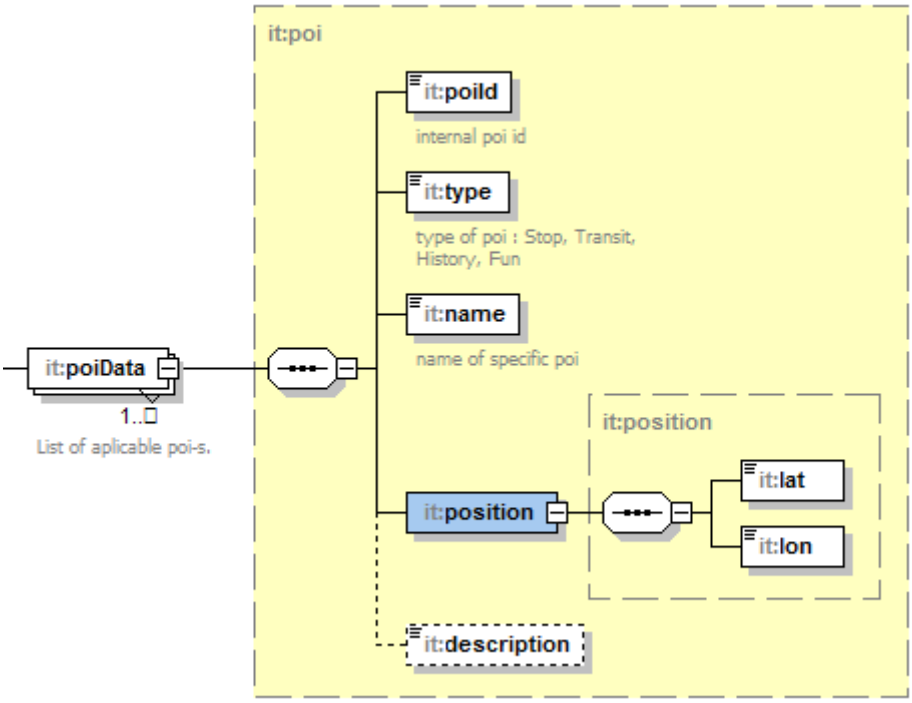
XSD schéma je rozšírená o ďalšiu časť, obsahujúcu formát dát, nutný na špecifikáciu novej služby (Obr. 8.). Nevyhnutnosťou je taktiež rozšírenie databázového modelu pre uchovávanie POI (Obr. 7.). Ku každému bodu je potrebné poznať jeho meno, GPS súradnice a typ, prípadne bližší popis.

Nemenej dôležitou časťou je aj aspoň základné naplnenie databázy POI.

Keďže použitie GPS v simulátore má obmedzené možnosti, súradnice aktuálnej pozície pri vývoji a testovaní simulujeme.



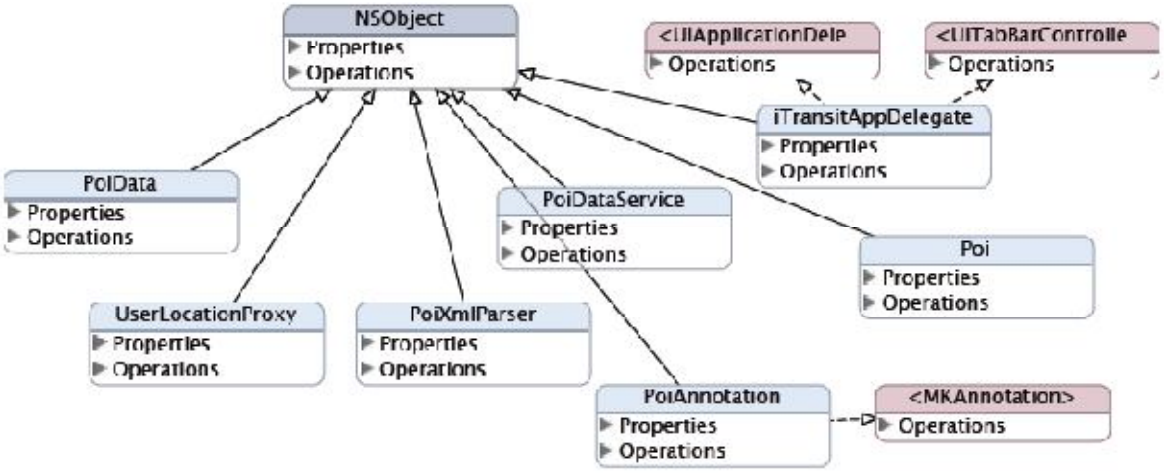
Obr. 7. Doplnenie fyzického dátového modelu.



Obr. 8. Doplnená časť XSD schémy pre službu getPoi.

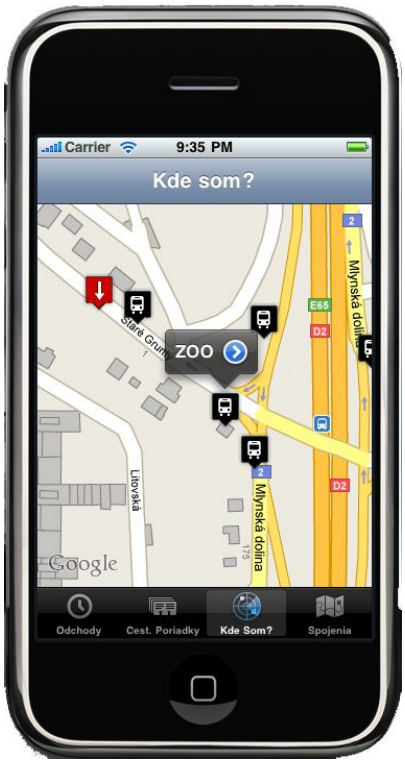
Implementácia

Klientská časť, implementovaná na základe návrhového vzoru „model-view-controller“, je v tejto fáze rozšírená o triedy POI a POIData, inštancované v POIXmlParser v rámci úrovne „model“ (Obr. 9.). Trieda UserLocationProxy zabezpečuje prípadnú simuláciu aktuálnych GPS súradníc. V prípade, že je jedna z *properties* nastavená na „real“ súradnice sú načítané pomocou GPS, ak je však nastavená na „simulator“, súradniciam pridelí preddefinovanú hodnotu.



Obr. 9. Doplnenie diagramu tried v 3.šprinte.

Jednotlivé typy POI sú na mape odlíšené rôznymi, jednoducho rozlíšiteľnými a zrozumiteľnými ikonami. Použili sme voľne šíriteľné „google maps icons“. Aktuálna poloha je vyjadrená šípkou (**Obr. 10**.).



Obr. 10. Obrazovka aplikácie pre funkcionalitu „kde som“.

V tomto štádiu aplikácia nebude umožňovať výber typov POI, ktorá sa zobrazia. Štandardne sa zobrazia všetky v blízkosti aktuálnej pozície.

Testovanie

Testovacie scenáre, slúžiace ako integračné testy pri testovaní danej User story, sú uvedené v prílohe B.

4. šprint

Štvrtý šprint sme začali 18.11.2009 a ukončili 1.12.2009. Jeho cieľom bolo ďalšie dopĺňanie funkčnosti do aplikácie. Celkovo sme identifikovali jednu User story.

Funkcionalita „plánovanie cesty“

Používateľ sa chce dopraviť na konkrétne miesto a chce vedieť informácie o najbližších spojoch. Klikne preto na tretie tlačidlo v spodnej (tlačidlovej) časti a v strednej (obsahovej) časti sa objaví "dialógové okno", kde bude môcť zadať odkiaľ ide (predvolená aktuálna lokalita podľa GPS), kam chce ísť a kedy (predvolený aktuálny čas). Po odoslaní požiadaviek sa v strednej časti obrazovky zobrazí zoznam nájdených spojení. Ku každému spojeniu (môže byť viac liniek) zobraziť:

- východziu zastávku, z ktorej ide prvá linka spojenia
- smer zastávky (prvej linka spojenia)
- čísla použitých liniek
- čas kedy ide prvá linka spojenia
- čas, ktorý zostáva do odchodu

Obmedzenia:

- iba zo zastávky na zastávku
- neriešiť GPS
- neriešiť parametre spojenia
- neriešiť optimálnosť vyhľadania spojenia
- možnosť riešiť cez iMHD

Analýza

Plánovanie spojení umožňuje naplánovať kombináciu dopravných spojov zo počiatočného miesta do cieľového miesta. Na nájdenie najkratšej kombinácie spojení je možné použiť jeden zo základných algoritmov :

- Dijkstrov algoritmus
- Bellman-Fordov algoritmus
- Floyd-Warshallov algoritmus

Vzhľadom na povahu riešenej problematiky sa najviac využíva Dijkstrov algoritmus. V moderných vyhľadávačoch spojení sa tiež používa algoritmus Diaľničné hierarchie, ktorý je niekoľko krát rýchlejší ako Dijkstrov algoritmus. Dôvodom je, že sa zbytočne neprehľadáva celá oblasť grafu reprezentujúceho sieť liniek. Pre momentálny šprint s danými obmedzeniami je však implementácia Dijkstrovho algoritmu menej náročná a postačujúca.

Návrh

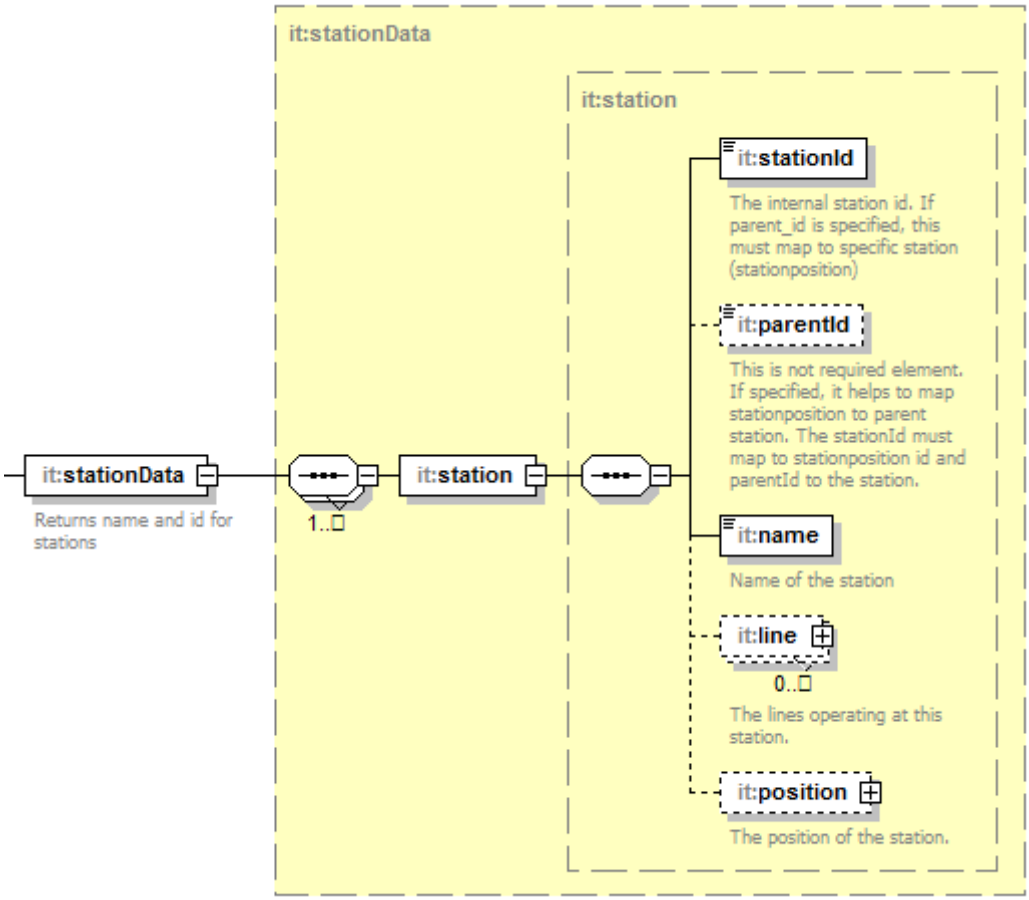
V nami zvolenom algoritme sa dĺžka cesty určuje v minútach trvania danej cesty. Výsledkom je teda najkratšie trvajúca cesta aj s miestami prestupu. Vstupom do algoritmu plánovania je graf reprezentujúci časovú dostupnosť všetkých miest zo všetkých miest, pričom pri práci s grafom sa vypočíta najkratšia cesta z každého miesta do každého miesta. Do úvahy sa berie aj možnosť pešieho

presunu medzi jednotlivými miestami, avšak iba v tom prípade, pokiaľ medzi danými miestami neexistuje dopravné spojenie.

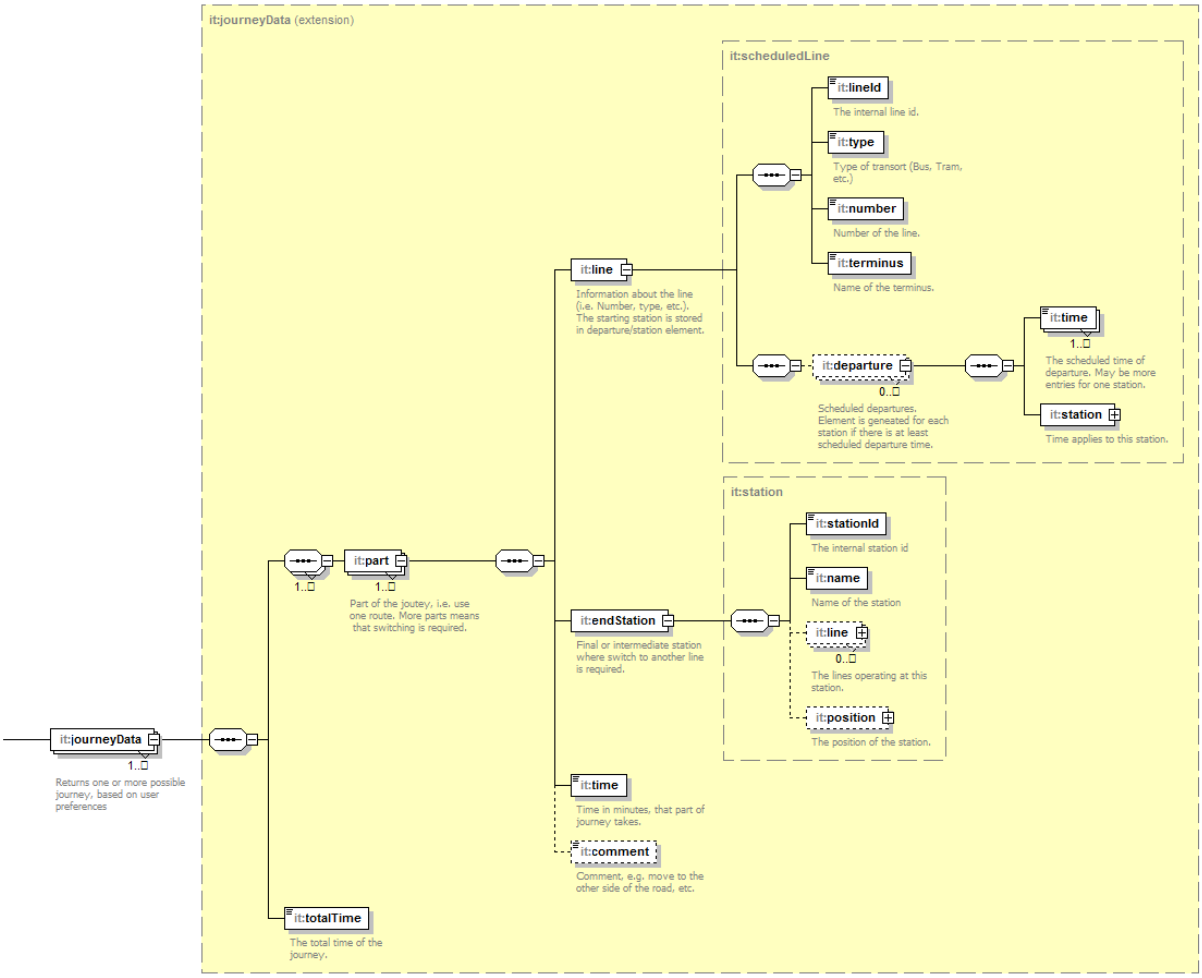
Je nutné vytvorenie novej služba, ktorá poskytne informácie o pláne cesty zo zadanej nástupnej zastávky do zadanej výstupnej zastávky v definovanom čase. Plán cesty pozostáva z čísla linky, jej najbližšieho odchodu z nástupnej zastávky, času príchodu na výstupnú zastávku, celkového času cesty, prípadne prestupných zastávok - názov zastávky, čas príchodu, číslo novej linky a čas jej odchodu.

Preto je potrebná aj zmena už existujúcej časti XSD schémy (Obr. 12.), nakoľko v pôvodnom návrhu nebol uvedený čas trvania jednotlivých spojení.

Návrh používateľského rozhrania vyžaduje výber zo zoznamu existujúcich zastávok, preto je nevyhnutné vytvorenie novej služby, ktorá poskytne zoznam všetkých zastávok s ich ID a názvami – to znamená ďalšie rozšírenie XSD schémy (Obr. 11).



Obr. 11. Rozšírenie XSD schémy.

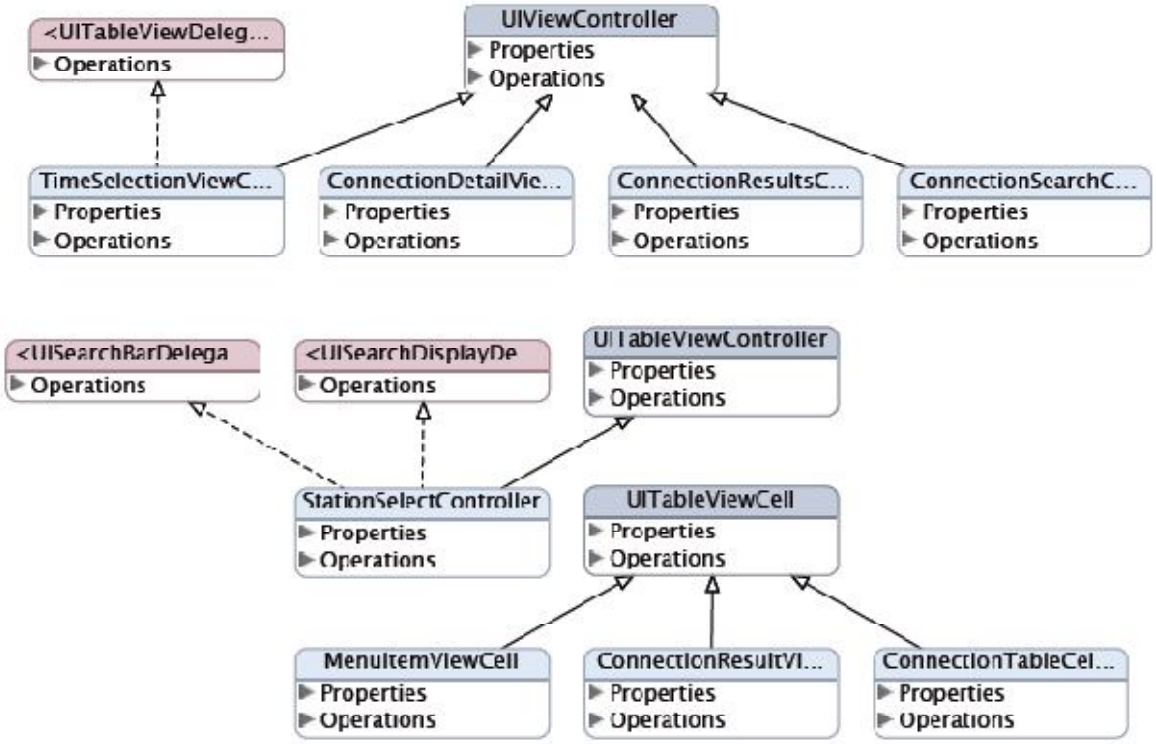


Obr. 12. Zmenená časť XSD schémy.

Implementácia

Klientská časť je rozšírená o nové triedy zabezpečujúce zobrazenie vyhľadaných spojení a zadávanie vstupných údajov(Obr. 13).

Plánovanie spojení je implementované v jazyku python a celý výpočet sa vykonáva na strane servera. Potrebné vstupné údaje sú získavané z PostgreSQL databázového systému.



Obr. 13. Diagram tried pre 4. šprint.

Obr. 14. Vstupná obrazovka pri plánovaní cesty.

Implementované používateľské rozhranie spĺňa štandardy pre dotykovú obrazovku (Obr. 14., Obr. 15., Obr. 16.).

Obr. 15. Obrazovky pre výber zastávky a výber času.

Obr. 16. Obrazovky s vyhľadanými spojeniami.

Testovanie

Testovacie scenáre sú uvedené v prílohe B.

5. šprint

Piaty šprint sme začali 1.12.2008 a ukončili 14.12.2009. Jeho cieľom bolo ďalšie dopĺňanie funkčnosti do aplikácie. Celkovo sme identifikovali tri User stories.

Funkcionalita „cestovné poriadky“

Používateľ si chce pozrieť cestovný poriadok konkrétnej linky z konkrétnej zastávky. Klikne preto na štvrté tlačidlo v spodnej (tlačidlovej) časti a v strednej (obsahovej) časti sa objaví "dialógové okno", kde bude môcť zadať linku a potom zastávku a po odoslaní požiadavky sa v strednej časti obrazovky zobrazí cestovný poriadok podobne ako je na zastávke (meno zastávky, číslo linky, odchody pracovné dni, víkendy, prázdniny, ...).

Analýza

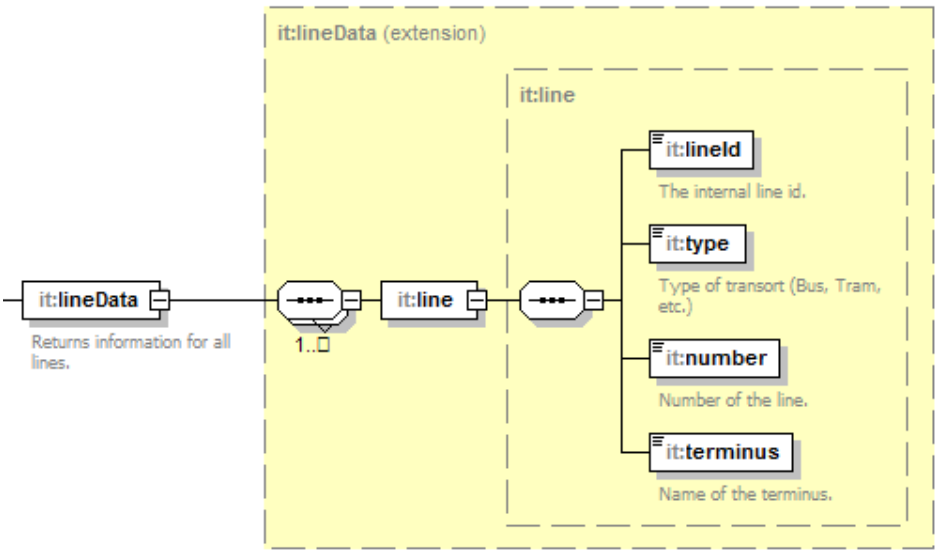
Na zobrazovanie cestovných poriadkov pre jednotlivú linku je potrebné poznať poradie zastávok spolu s ich časovou následnosťou. Taktiež musíme rozlišovať medzi odchodmi cez pracovné dni, víkendy a prázdniny.

Pre jednoduchý výber linky je požiadavkou používateľského rozhrania poznanie všetkých liniek.

Zobrazenie samotného cestovného poriadku bude pozostávať z dvoch častí. Prvou časťou budú odchody usporiadané podľa hodín, druhá časť bude trasa linky – poradie zastávok spolu s časom trvania cesty zo zvolenej zastávky na ostatné zastávky.

Návrh

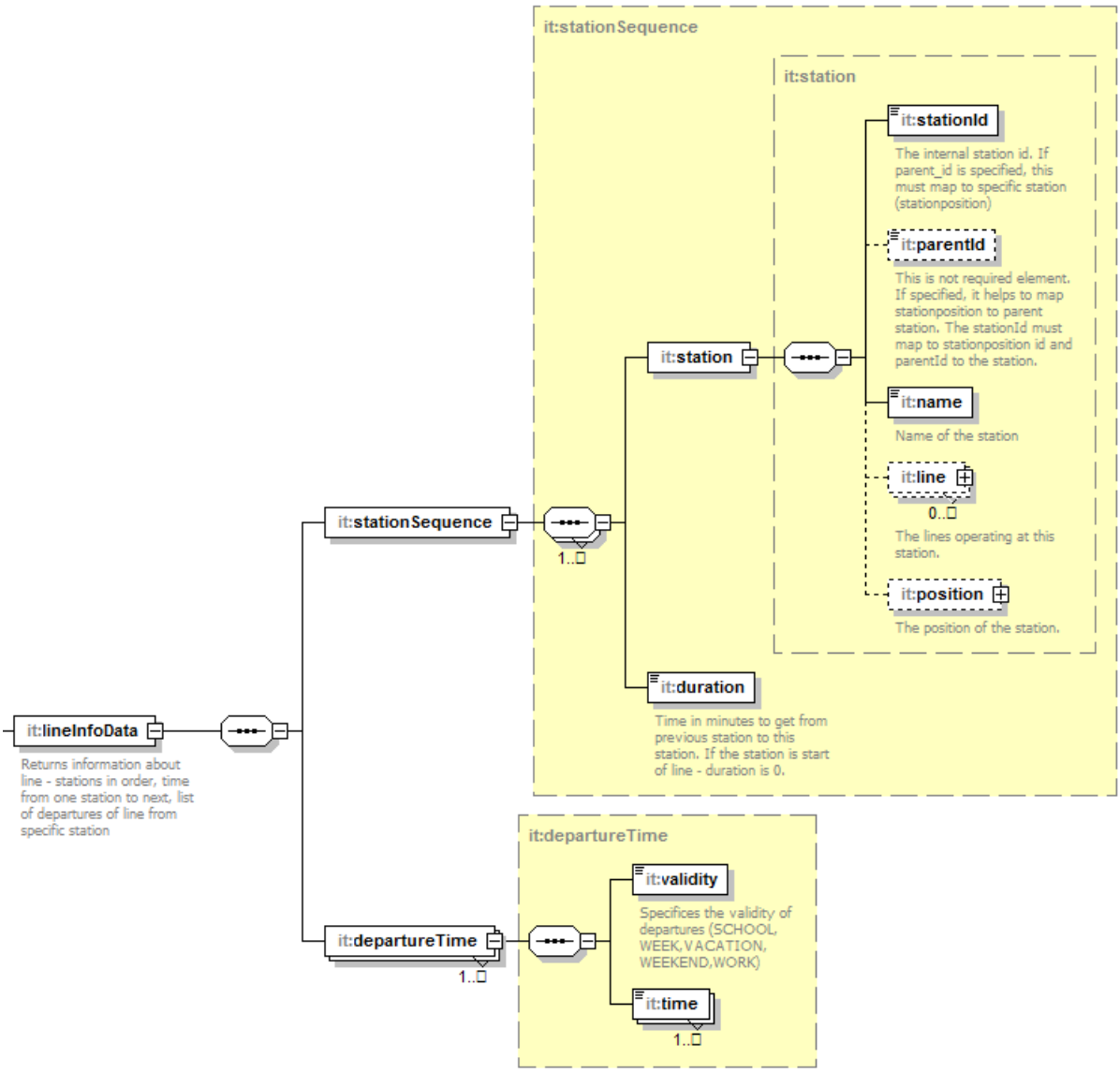
Služba getLineTimetable poskytuje informácie o odchodoch linky na konkrétnej zastávke. Pre zadanú linku a zastávku vráti všetky časy odchodov danej linky z danej zastávky, všetky zastávky patriace k danej linke a časový interval medzi nimi. Každá zastávka, cez ktorú spoj prechádza obsahuje názov a dvojicu ID (stationposition a aj station id).



Obr. 17. Rozšírenie XSD schémy pre službu getLines.

Služba getLines poskytuje zoznam všetkých liniek s ich ID, názvami, typom a cieľovej stanice. Návratovú množinu zastávok je možné obmedziť nepovinným parametrom stationId. V prípade, že je špecifikovaný, sú vrátené len spoje, ktoré prechádzajú danou zastávkou. Ak nie je špecifikovaný, služba vracia všetky spoje.

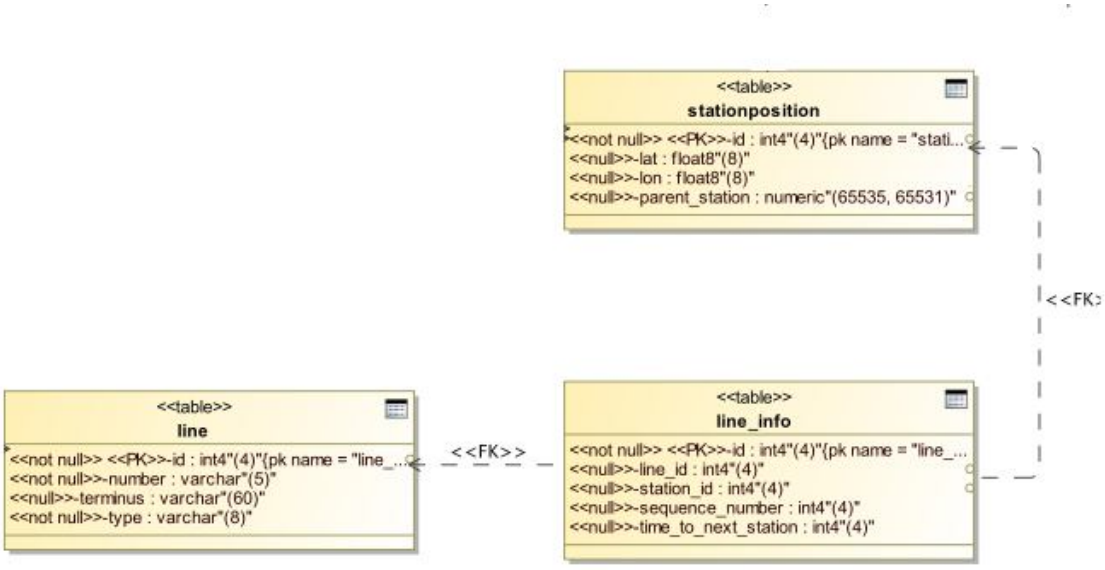
Je potrebné rozšíriť XSD schému tak, aby zahŕňala nové služby (Obr. 17., Obr. 18.).



Obr. 18. Rozšírenie XSD schémy pre službu getLineTimetable.

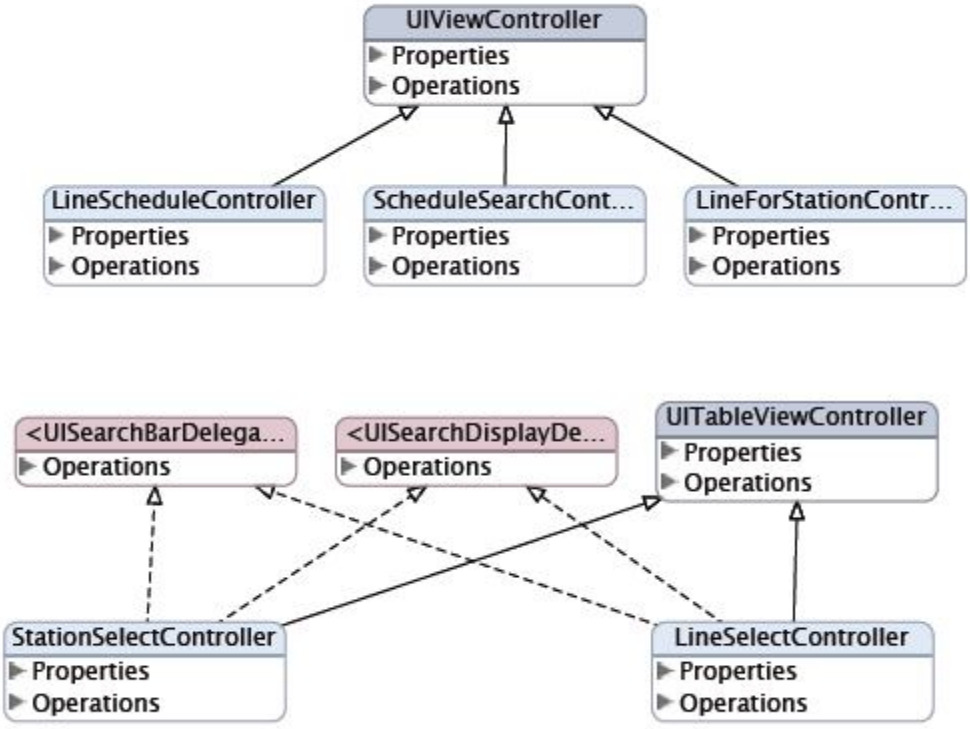
Implementácia

Databázový model je doplnený o tabuľku line_info, ktorá umožňuje uloženie postupnosti zastávok pre linku spolu s časom trvania cesty medzi jednotlivými zastávkami.



Obr. 19. Doplnená časť fyzického dátového modelu.

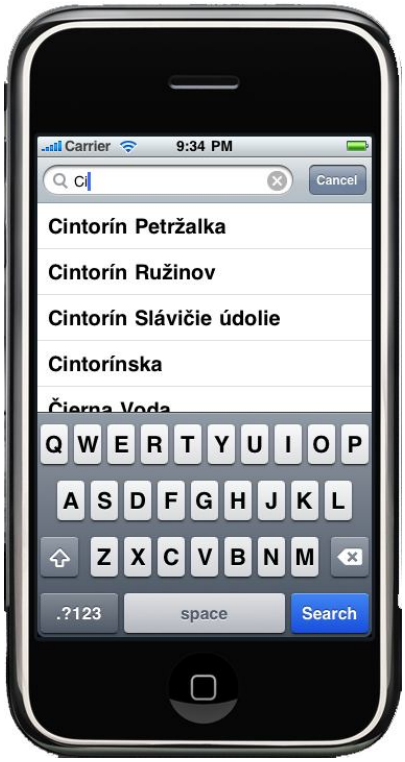
Klientská časť je v tejto etape rozšírená najmä v rámci úrovne „controller“ (Obr. 20).



Obr. 20. Diagram tried.

Implementované používateľské rozhranie funkcionality cestovné poriadky využíva štandardné prvky (Obr. 21, Obr. 22, Obr. 23, Obr. 24).

Obr. 21. Cestovné poriadky - zadávanie údajov



Obr. 22. Výber zastávky.

Obr. 23. Výber linky.

Obr. 24. Cestovné poriadky.

Testovanie

Testovacie scenáre, slúžiace ako integračné testy pri testovaní danej user story, sú uvedené v prílohe B.

Funkcionalita „informácie o zastávke“

Používateľ si prezerá mapu s lokalitou kde je, a chce získať podrobnejšie informácie o zastávke. Klikne preto v mape na ikonku zastávky, ktorá ho zaujíma. Zobrazia sa mu čísla liniek, ktoré stoja na tejto zastávke.

Analýza

Doplnenie už implementovanej funkcionality „kde som“. Pre každú zastávku potrebujeme vedieť všetky linky, ktoré z nej odchádzajú. Po kliknutí na tlačidlo v anotácii na mape sa zobrazí zoznam liniek odchádzajúcich z danej zastávky.

Návrh

Služba getStationInformation poskytuje informácie o zastávke. Pre zadanú zastávku vráti všetky čísla liniek, ktoré z nej odchádzajú. Využíva už existujúcu XSD schému.

Implementácia

Používateľské rozhranie po kliknutí na zastávku na mape zobrazuje typ, číslo a smer liniek odchádzajúcich z vybranej zastávky.

Obr. 25. Zobrazenie informácií pre zastávku.

Testovanie

Testovacie scenáre, slúžiace ako integračné testy pri testovaní danej user story, sú uvedené v prílohe B.

Funkcionalita „informácie o linke“

Používateľ si prezerá informácie o zastávke a chce získať podrobnejšie informácie o linke. Klikne preto v zozname so spojmi zastavujúcimi na zastávke na záznam s číslom linky, ktorá ho zaujíma. Zobrazí sa mu cestovný poriadok tejto linky z danej zastávky, s informáciami ako na reálnej zastávke (meno zastávky, číslo linky, odchody pracovné dni, víkendy, prázdniny, ...).

Analýza

Na zobrazenie podrobných informácií o linke využijeme už existujúcu funkciu „cestovné poriadky“, so vstupnou zastávkou a vybratou linkou.

Návrh

Po kliknutí na tlačidlo v anotácii na mape sa zobrazí zoznam liniek odchádzajúcich z danej zastávky, pre ktoré je možné zobraziť cestovný poriadok.

Implementácia

Prepojenie už existujúcich funkcií – „informácie o linke“ a „cestovné poriadky“. Používateľské rozhranie je identické s už vytvoreným (Obr. 26).

Obr. 26. Informácie o linke.

Testovanie

Testovacie scenáre sú uvedené v prílohe B.

6. šprint

Šiesty šprint sme začali 22.2.2010 a ukončili 8.3.2010. Pracovali sme na piatich User stories identifikovaných na základe dvoch požiadaviek:

- Aplikácia musí byť vždy "živá"
- Aplikácia zobrazí cestovné poriadky

Requirement : Aplikácia musí byť vždy "živá"

Aplikácia musí vždy reagovať na používateľské vstupy, hneď po kliknutí na ľubovoľné tlačidlo, voľbe v zozname možností a podobne, pričom je potrebné zobraziť zodpovedajúcu obrazovku a v prípade spracovania či čakania na dáta, zobraziť indikátor aktivity (otáčajúce sa koliesko). Ak používateľ klikne na tlačidlo späť alebo na niektoré z tlačidiel v spodnom paneli v priebehu spracovania alebo čakania, aplikácia okamžite zareaguje. Pôvodné spracovanie poprípade spojenie so serverom sa môže zrušiť, prípadne sa spracovanie dokončí respektíve dáta sa dotiahnu a výsledky sa použijú, ak sa používateľ vráti na pôvodnú obrazovku.

1. User Story : Nová obrazovka najbližšie odchody

Používateľ chce informácie o odchodoch autobusov z najbližších zastávok v najbližšom čase. Po kliknutí na zodpovedajúce tlačidlo sa zobrazia najbližšie podzastávky s najbližšími spojmi.

Analýza

Na základe prehodnotenia už vytvorenej obrazovky, sme sa ju rozhodli zmeniť. Naším cieľom je lepšia prehľadnosť informácií, ktoré poskytujeme používateľovi. Ku každej zastávke, ktorá je v blízkom okolí, teda aplikácia zobrazuje najbližšie odchody liniek z nej odchádzajúcich, je dobré poznať jej skutočnú vzdialenosť od zariadenia, tak isto ako je prehľadnejšie zobraziť oddelene jednotlivé zastávky.

Návrh

Je dôležité zohľadniť situácie na miestach s málo zastávkami - tým pádom s málo odchodmi liniek, tak isto ako prípady, ktoré môžu nastať keď zastávok a odchodov liniek bude príliš veľa. Chceme, aby používateľ mal primerané množstvo informácií. Preto navrhujeme upraviť algoritmus zobrazovania liniek a podzastávok takto:

- zobraziť všetky podzastávky zobrazenej zastávky, aj keď sú ďalej ako 500m
- ku každej podzastávke zobraziť aspoň 2 spoje
- ak nechodí žiadny spoj zobraziť prázdny riadok alebo bledým "žiadny spoj"
- ak je zobrazených "málo" zastávok zobraziť viac zastávok, aj keď sú vzdialené viac ako 500m

Implementácia

Namiesto predchádzajúceho „plain view“ sme na nové zobrazenie obrazovky použili „group view“, ktoré považujeme v tomto prípade za prehľadnejšie. Nie je viditeľný len jeden odchod pre jednu linku, ale sú viditeľné dva, spolu s minútami zostávajúcimi do odchodu. Počítanie vzdialenosti zastávky od aktuálnej polohy je realizované triedou UserLocationProxy, ktorá v prípade simulátorového módu počíta s aktuálnou polohou v Mlynskej doline. Novovytvorená obrazovka je zobrazená na Obr. 27.

Obr. 27. Obrazovka najbližšie odchody.

Testovanie

Testovanie bude prebiehať podľa zvoleného algoritmu zobrazovania

- bežný prípad, napr. lokalita FIIT, zastávka ZOO - skontrolovanie s reálnymi odchodmi
- ak žiadna zastávka v blízkom okolí - zobrazenie vzdialenejších zastávok
- ak žiadna zastávka aj v ďalekom okolí (skúša niekto zo Žiliny) - zobrazenie "žiadne zastávky v okolí"
- ak z podzastávky už nejde žiadny spoj - zobrazenie "žiaden spoj"

2. User Story : Zobrazovanie obrazovky vyžadujúcej pripojenie na server

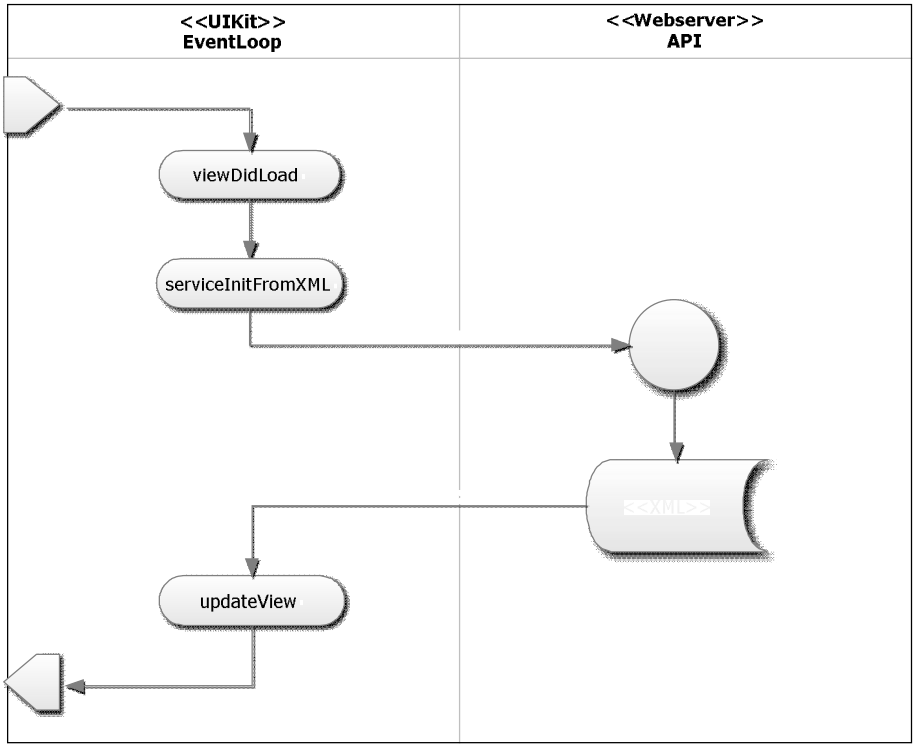
Používateľ požaduje informácie, ktorých zobrazenie vyžaduje pripojenie na server, a teda ich zobrazenie je oneskorené. Keďže sa jedná o aplikáciu určenú pre mobilné zariadenie, je potrebné zobrať do úvahy dlhší čas potrebný na vykonanie požiadavky. Klientská aplikácia toto musí zohľadniť a poskytnúť príjemné a reagujúce používateľské rozhranie aj v prípade, že je potrebné počkať na vybavenie požiadavky.

Analýza

Klientská aplikácia používa volania RESTful služieb, čím získava požadované údaje, ktoré sú následne prezentované používateľovi. Tieto volania sú realizované prostredníctvom siete Internet. Keďže sa jedná o aplikáciu určenú pre mobilné zariadenie, je potrebné zobrať do úvahy dlhší čas potrebný na

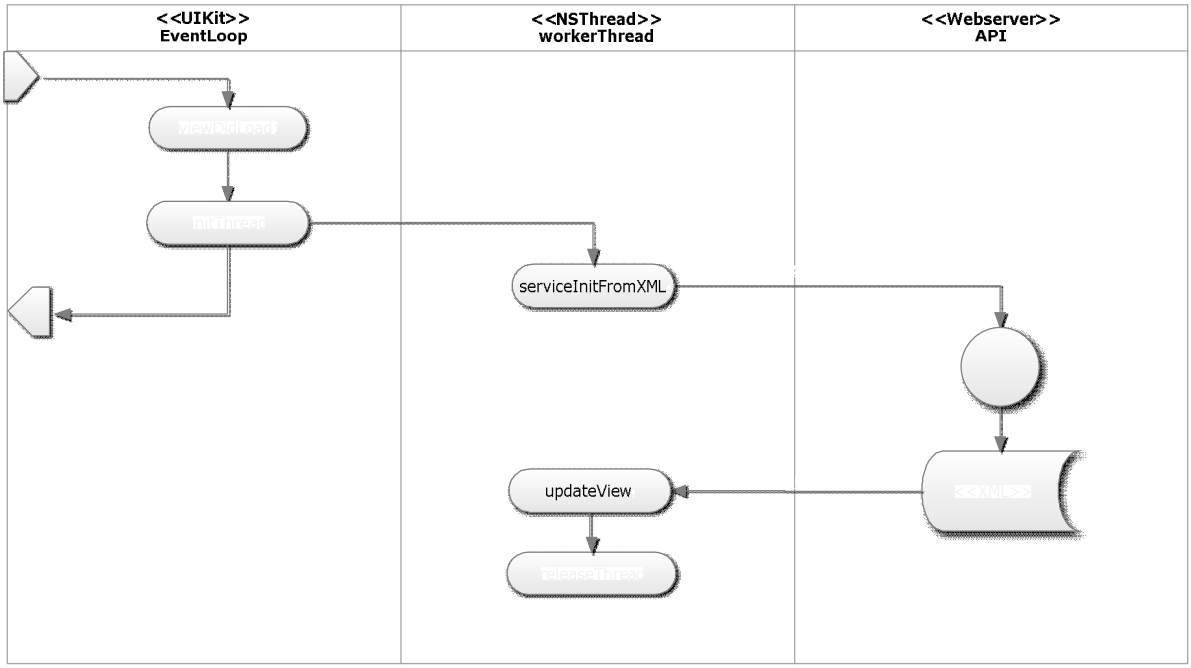
vykonanie požiadavky. Klientská aplikácia toto musí zohľadniť a poskytnúť príjemné a reagujúce používateľské rozhranie aj v prípade, že je potrebné počkať na vybavenie požiadavky.

Aktuálne sú požiadavky na server vykonávané pomocou tzv. "service wrappers", čo sú triedy určené na vykonanie požiadavky, odovzdanie parametrov, prijatie a spracovanie odpovede. Volanie týchto obaľovačov zabezpečujú rôzne udalosti, ktoré vyžadujú dotiahnutie údajov. Príkladom môže byť stlačenie tlačidla, zobrazenie obrazovky, atď. Aktuálne riešenie volá tieto metódy z hlavného vlákna – *Event Loop.* Takéto volanie má za následok pozastavenie vykonania a obslúženia všetkých ostatných udalostí, pokiaľ nie je ukončená práca služby.



Návrh

Práca používateľa nebude ovplyvnená faktom, že klientská aplikácia potrebuje získať dodatočné informácie pomocou služieb. Toto bude dosiahnuté implementáciou, kde sa všetky dáta budú získavať v samostatnom vlákne, ktoré po dotiahnutí prekreslí obrazovku a aktualizuje získané hodnoty.



Implementácia

Implementácia bude nasledovať tieto pravidlá:

- Všetky volania služby budú extrahované do samostatnej metódy v konkrétnom kontroléri. Táto metóda musí byť *public* a prijímať práve jeden parameter: (id) object
- Táto metóda vytvorí službu, príjme a spracuje dáta. Po spracovaní sú dáta posunuté do kontroléra cez referenciu *object*. Nad kontrolérom je zavolaná metóda updateView:
- Zavolanie metódy `detachNewThreadSelector:toTarget:withObject:` vytvorí nové vlákno, zavolá metódu určenú selektorom na konkrétnom kontroléri. Ako object je použitá referencia *self*.
- Kód, ktorý aktualizuje obsah obrazovky je obsiahnutý v príslušnej metóde (updateView:)

Testovanie

Overiť na každej obrazovke, ktorej zobrazenie si vyžaduje pripojenie na sever:

- po iniciovaní zobrazenia obrazovky sa zobrazí indikátor aktivity
- po obdržaní dát sa zobrazí obsah
- po obdržaní dát v nesprávnom formáte sa zobrazí chybová správa
- po neobdržaní dát v stanovenom čase sa zobrazí chybová správa
- po neúspešnom pokuse spojiť sa so serverom a zobrazí chybová správa

Requirement : Zobrazovanie cestovných poriadkov

Iniciovať zobrazenie cestovných poriadkov pomocou tlačidiel "Linky" a "Zastávky", zobrazenie zoznamov liniek a zastávok, a nakoniec cestovného poriadku linky zo zvolenej podzastávky.

3. User Story : Zohľadnenie typov dní - pracovný deň, víkend, sviatok

Používateľ chce informáciu o najbližších spojoch. K zobrazeným podzastávkam sa mu zobrazia informácie o najbližších spojoch, pričom sa zohľadňujú cestovné poriadky vzhľadom na typ dňa (pracovný deň, sviatok, víkend).

Analýza

Odchody liniek MHD v Bratislave sú závislé na type dňa. Linky majú iné odchody počas pracovných dní – pondelok až piatok, počas víkendov – sobota, nedeľa, štátne sviatky, a počas školských prázdnin. Preto je nutné tieto skutočnosti zohľadniť v databázovej štruktúre, službe a zobrazovaní cestovných poriadkov na klientskej aplikácii. Je nutné nájsť vhodný zdroj údajov, odkiaľ je možné čerpať informácie o štátnych sviatkoch, prázdninách, atď.

Návrh

Pre účel zohľadňovania typov dní v službách je vytvorená databázová štruktúra *calendar*. Táto obsahuje záznamy pre každý deň v roku s atribútmi *year*, *month*, *day* a *type*. Atribút *year* je pre jednoznačnosť, aby bolo vždy jasné, pre ktorý rok sú v tabuľke uvedené záznamy. Atribút *type* je podmnožinou rovnomenného atribútu v tabuľke *departure*, obsahuje hodnoty *SCHOOL*, *WEEKEND* a *VACATION*, ktoré sa vzťahujú na školský deň, víkend alebo sviatok a na školské prázdniny.

Typ *WEEK* zahŕňa všetky dni v týždni, *WORK* pracovné dni bez ohľadu na školské vyučovanie, *WEEKEND* sobotu, nedeľu a sviatok, *SCHOOL* pracovné dni školského vyučovania a *VACATION* pracovné dni mimo školského vyučovania.

Namiesto implementovania automatického aktualizovania údajov, sme sa rozhodli, že efektívnejšie bude manuálne naplnenie štruktúry raz ročne, čo nezaberá priveľa času a zatiaľ to pre naše potreby plne postačuje.

Implementácia

Princíp práce s touto štruktúrou je taký, že služba si zistí aktuálny dátum a podľa neho vyhľadá v tabuľke príslušný záznam a uchová hodnotu atribútu *type*. Tú neskôr porovnáva s typom dňa vzťahujúcim sa na jednotlivé odchody daného spoja a v prípade zhody pridá spoj do zoznamu spojov, ktoré sa pošlú klientovi. Podmienky porovnania sú takéto:

```
# daytype = hodnota atribútu type daného záznamu v tabuľke calendar

# daytypeCondition = podmienka pre SQL dotaz, povolené hodnoty type

#                                 v tabuľke departure

if daytype == 'SCHOOL':

daytypeCondition = '(type = \'SCHOOL\' OR type = \'WORK\' OR type = \'WEEK\')'

elif daytype == 'VACATION':

daytypeCondition = '(type = \'VACATION\' OR type = \'WORK\' OR type = \'WEEK\')'
```

```
elif daytype == 'WEEKEND':

daytypeCondition = '(type = \'WEEKEND\' OR type = \'WEEK\')'
```

Testovanie

Overiť na spoji s rôznymi cestovnými poriadkami pre rôzne typy dní:

- pre prac. deň
- pre víkend
- pre sviatok

4. User Story : Zobrazenie všetkých zastávok a následne ich podzastávok so spojmi

Používateľ chce získať informácie o zastávke. Klikne na tlačidlo "Zastávky" a zobrazí sa mu abecedný zoznam zastávok. Po kliknutí na zastávke sa mu zobrazí zoznam podzastávok a všetkých spojov z nich odchádzajúcich.

Analýza

Na zobrazenie jednotlivého cestovného poriadku je nutné vedieť zastávku a linku. Ak si používateľ nie je celkom istý jednou z informácií, chceme mu výber čo najviac uľahčiť. Preto sme sa rozhodli jedno tlačidlo *Cestovné poriadky* v spodnom paneli nahradiť dvoma – Linky a Zastávky.

V tejto User story sme sa zaoberali len funkcionalitou Zastávky.

Návrh

Uľahčenie výberu zastávky poskytne index na pravej strane obrazovky, ktorý bude pozostávať zo začiatočných písmen usporiadaných podľa abecedy. Zároveň aj zastávky budú usporiadané do skupín podľa prvého písmena. Po výbere zastávky, nasledujúca obrazovka s výberom linky bude obsahovať už len tie linky, ktoré stoja na danej zastávke. Nebudú tu linky, pri ktorých je vybratá zastávka konečnou.

Implementácia

Samotnú implementáciu indexu zabezpečujú metódy `sectionIndexTitlesForTableView` a `sectionForSectionIndexTitle`.

Obr. 28. Obrazovky Zastávky.

Testovanie

Overiť :

- na vybraných pár zastávkach, že sú zobrazené
- na pár podzastávkach, že sú zobrazené správne informácie

7. šprint

Šiesty šprint sme začali 8.3.2010 a ukončili 22.3.2010. Identifikovali sme v ňom päť User stories.

1. User Story : Používateľ chce vidieť detailné informácie o trase linky

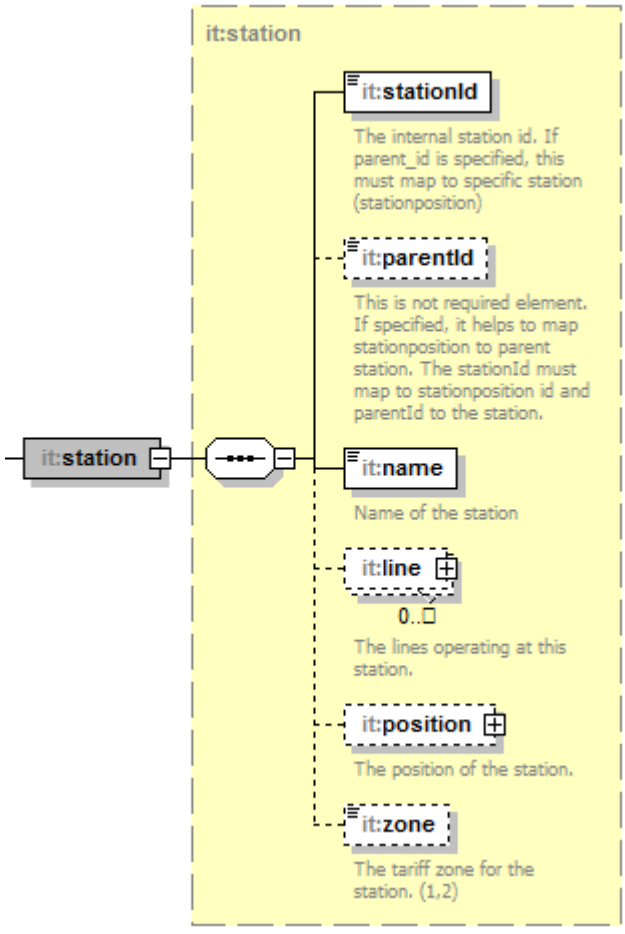
Používateľ chce vidieť informácie o jednotlivých pásmach, v ktorých sa zastávka nachádza.

Analýza

V Bratislave existujú dve pásma MHD. Je v nich rôzna sadzba na predplatné lístky, preto si myslíme, že používatelia aplikácie ocenia informáciu o tom, v ktorom pásme sa daná zastávka v linke nachádza.

Návrh

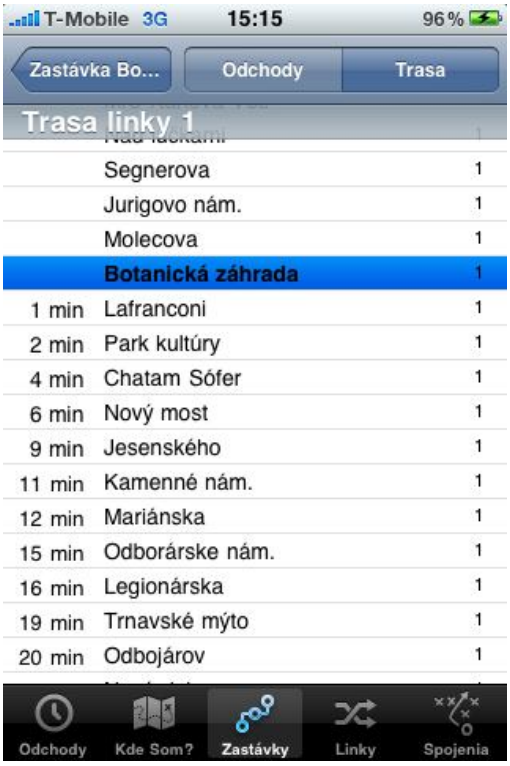
Je nutné upraviť službu getLineTimeTable aby vo výsledku vracala aj pásmo k jednotlivým zastávkam. Preto je potrebné rozšíriť XSD a databázovú schému a naplniť ju údajmi.



Obr. 29. Zmenená XSD schéma.

Implementácia

Nová obrazovka Trasy pre Zastávky je zobrazená na Obr. 30. Čas cesty v minútach od vybranej zastávky sme kvôli lepšej prehľadnosti presunuli pred meno cieľovej zastávky. Pásmo pre jednotlivé zastávky sa nachádza vpravo.



Obr. 30. Obrazovka Trasa.

Testovanie

Testovacie scenáre sú uvedené v prílohe B.

2. User Story : Zobrazenie všetkých liniek a následne ich zastávok

Používateľ chce získať informácie o spoji. Klikne na tlačidlo "Linky" a zobrazí sa mu zoznam liniek. Po kliknutí na linku sa mu zobrazí zoznam zastávok, na ktorých spoj zastavuje.

Analýza

Na zobrazenie zoznamu zastávok pre vybranú linku je nutná úprava služby getStations, ktorá vracia všetky zastávky. Jedným z riešení je pridanie nepovinného parametra id linky, pričom po jeho zadaní služba vráti len tie zastávky, cez ktoré daná linka prechádza.

Návrh

Uľahčenie výberu linky poskytuje index na pravej strane obrazovky, ktorý pozostáva z význačných liniek - napr. prvej a poslednej električky, autobusu, trolejbusu a nočnej linky v poradí. Linky sú usporiadané do skupín podľa ich typu . Po výbere linky, nasledujúca obrazovka s výberom zastávky

bude obsahovať už len tie zastávky, na ktorých stojí daná linka. Nebude sa tam nachádzať konečná zastávka linky.

Implementácia

Implementáciu indexu zabezpečujú metódy `sectionIndexTitlesForTableView` a `sectionForSectionIndexTitle`. Zastávky pre jednotlivú linku sú získané pomocou upravenej služby getStations.

Obr. 31. Obrazovka Linky.

Testovanie

Testovacie scenáre sú uvedené v prílohe B.

3. User Story : Doplnenie čísel liniek k zastávkam na mape

Používateľ si prezerá mapu, kde sú zobrazené zastávky (napr. obrazovka "kde som"), po kliknutí na zastávku sa okrem názvu zastávky v bubline zobrazia aj čísla liniek odchádzajúcich zo zastávky.

Analýza

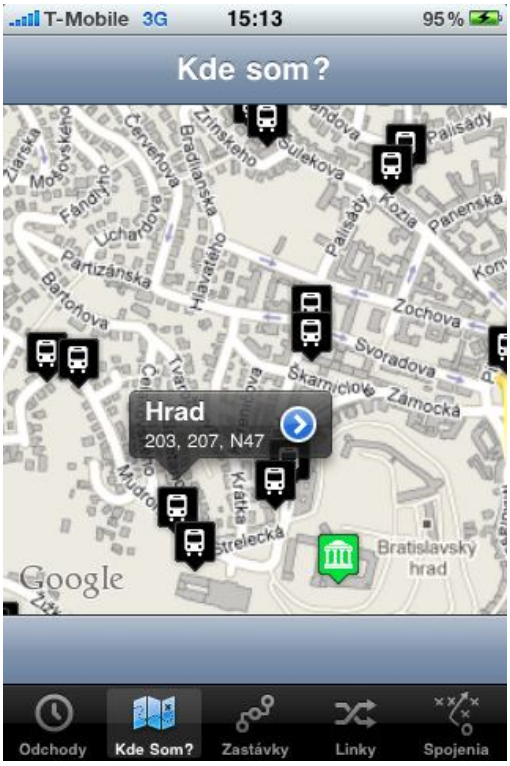
Je tu možnosť špeciálneho ošetrenia, ak zo zastávky neodchádza žiaden spoj – napr. vypísať "žiaden spoj", ale nakoniec sme sa zhodli, že lepšie bude nevypísať nič.

Návrh

Čísla liniek stojacich na danej zastávke je výhodné mať v textovom poli v databáze, kvôli jednoduchosti údržby. Zvolili sme pole „descripton“ v tabuľke POI, nakoľko vďaka tomu nie je nutné rozširovať už existujúcu štruktúru.

Implementácia

Naplnili sme pole *description* v databáze pre každú zastávku, číslami liniek, ktoré z danej zastávky odchádzajú. Následne sa zoznam získaný pomocou služby getPoi použil na vykreslenie „bubliny“ – anotácie.



Obr. 32. Obrazovka Kde som?

Testovanie

- otestovať na pár zastávkach
- otestovať na zastávke, z kt. neodchádza žiaden spoj (dočasne zrušená zastávka)
- otestovať na zastávke s "veľa" spojmi (možno Patrónka), spraviť screenshot

Requirement : Zobrazovanie vždy aktuálnych informácií

Keď používateľ ponechá obrazovku zobrazenú, musia sa informácie na nej aktualizovať priebežne. Týka sa to všetkých obrazoviek, ktoré zobrazujú informáciu vztiahnutú k aktuálnemu času, polohe, popr. natočeniu, ak sa bude používať aj informácia z kompasu.

4. User Story : Priebežná aktualizácia obrazovky s najbližšími spojmi

Používateľ ponechá zobrazenú obrazovku s najbližšími spojmi. Ako sa používateľ pohybuje, priebežne sa aktualizuje vzdialenosť od zobrazených zastávok. Ako plynie čas, priebežne sa aktualizujú časy odchodov najbližších spojov. Ak je to potrebné, na obrazovke sa zobrazí nová zastávka a tiež sa zobrazí nový spoj.

Analýza

Pri priebežnej aktualizácie podľa času je dobré zobrazovať aj čas odchodu linky, ktorá už odišla, napr. môže meškať , teda nielen koľko minút (plusových) zostáva do odchodu, ale zobrazovať aj určitý čas po odchode linky. Myslíme si, že mínusový čas, ktorý je ešte užitočný je do troch minút, potom už tento čas odchodu považujeme za irelevantný.

Pri pohybe je nutné rozlišovať pešiu chôdzu a cesta MHD, vtedy nastáva rýchlejšia zmena zastávok a údaje by sa nemuseli tak rýchlo dotiahnuť. Je potrebné zvoliť hraničnú vzdialenosť zobrazovania a zmeny zastávok.

Návrh

Údaje o odchodoch získané na začiatku aplikácia nebude obnovovať automaticky, ale až na vyžiadanie používateľa. Toto vyžiadanie by mohlo byť realizované tlačidlom „refresh“. Myslíme tým na zabránenie zbytočného prenosu údajov, keďže ide o mobilné zariadenie.

Iba pri pohybe alebo po stlačení tlačidla „refresh“ sa na obrazovke môže zobraziť nová zastávka , prípadne nový spoj.

Implementácia

Aktualizácia na základe času využíva triedu NSTimer. Trieda UserLocationProxy je využívaná na aktualizáciu obrazovky na základe polohy. V hornej časti obrazovky sa nachádza tlačidlo „refresh“. Minúty do odchodu sa časom menia, ak nastane situácia že odchod spoja je už tri minúty po aktuálnom čase, tento odchod sa už nezobrazuje.

Obr. 33. Obrazovka Najbližšie odchody.

Testovanie

- overiť pri pohybe aktualizáciu vzdialenosti pri zobrazených zastávkach
- overiť pri pohybe pridanie novej zastávky ak vzdialenosť od zastávky klesne pod stanovenú hodnotu
- overiť pri pohybe odstránenie novej zastávky ak vzdialenosť stúpne nad stanovenú hodnotu
- overiť pri plynutí času zmeny časov najbližších odchodov
- overiť pri plynutí času odstránenie už odídenej a pridanie novej linky do odchodov

5. User Story : Zobrazenie najbližších odchodov v obrazovke s informáciami o zastávke

Používateľ klikol na mape na zastávku (alebo skôr podzastávku) a zobrazil sa mu zoznam liniek odchádzajúcich z tejto podzastávky. Pri každej linke sa tiež zobrazia dva najbližšie odchody.

Analýza

Naším cieľom je zjednotenie s obrazovkou „Najbližšie odchody“.

Návrh

Rovnako zobrazovať informácie o linke s možnosťou podrobnejších informácií, teda poskytnúť jednoduchý odkaz na cestovné poriadky.

Implementácia

Pri implementáciu sme využili už navrhnutú obrazovku „Najbližšie odchody“.

Obr. 34. Obrazovka Informácie o zastávke.

Testovanie

- overiť na pár vybraných "bežných" spojoch a zastávkach
- overiť správne rozlišovanie pracovný deň, víkend, prázdniny
- špeciálne overiť prípad, keď spoj ide až o niekoľko hodín
- špeciálne overiť prípad, keď spoj ide až nasledujúci deň
- špeciálne overiť výstup, ak spoj v rozumne blízkej dobe nepremáva, napr. ide až o tri dni, alebo nepremáva vôbec

8. šprint

Osmy šprint sme začali 22.3.2010 a ukončili 5.4.2010. Identifikovali sme v ňom tri User stories.

Requirement : Plánovanie spojení

Plánovanie spojenia umožňuje po zadaní štartovacej a cieľovej lokality naplánovať optimálne spojenie aj prostredníctvom kombinácie viacerých liniek. Funkcionalita súvisiaca s plánovaním spojení:

- zobrazenie dialógu (cieľ, štart, čas a dátum, parametre: max. počet prestupov, rýchlosť presunu, čas potrebný na prestup a pod.)
- možnosť zadávať cieľ a zdroj pri zobrazovaní zastávky a rýchly prechod na plánovanie spojenia
- určenie spojení prostredníctvom jednoduchého algoritmu (etalónu - Dijkstra)
- rýchlejšie určenie spojení prostredníctvom jednoduchého algoritmu (napr. modifikácia Dijkstru, napr. A* alebo hocijaká iná heuristika)
- rýchle zistenie spojenia prostredníctvom zvoleného algoritmu
- rýchle zistenie spojenia prostredníctvom s využitím paralelizmu
- obrazovka s používateľom zadanými lokalitami a k nim prislúchajúcimi spojeniami v aktuálnom čase.

1. User Story : Vytvorenie obrazovky pre nájdené spojenia

Vytvorenie/zrevidovanie obrazovky pre nájdené spojenia - používateľ vyhľadá spojenie a výsledok sa mu zobrazí na obrazovke.

Analýza

Na základe algoritmu vyhľadávania spojení vo výbere vyhľadania spojení je možnosť zvoliť si okrem východzej, cieľovej zastávky a času odchodu aj maximálny počet prestupov a rýchlosť chôdze. Preto je nutné zrevidovanie obrazovky Spojenia.

Návrh

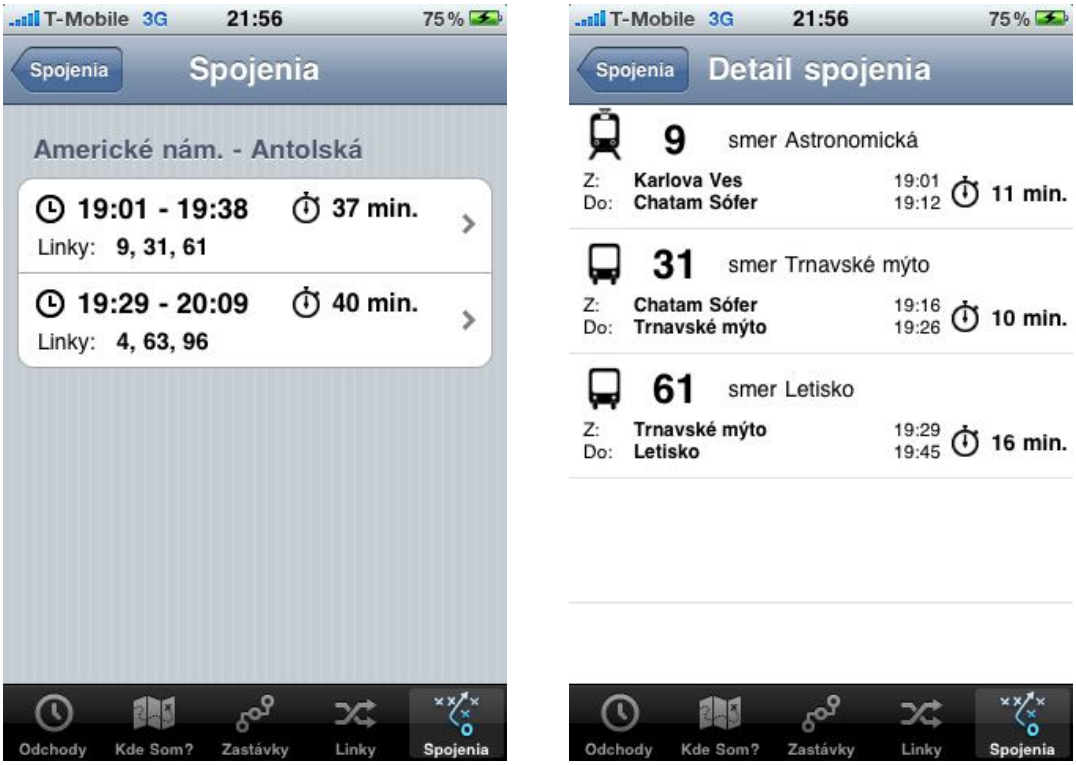
Celá funkcionalita Spojenia bude mať tri základné obrazovky:

- Výber spojenia
- Vyhľadané spojenia
- Detail spojenia

Implementácia

Novovytvorené obrazovky sú zobrazené na Obr. 35 a Obr. 36. Po vyhľadaní spojení je možné si pozrieť detaily jednotlivých liniek v obrazovke Detail spojenia. Tu je uvedený aj čas cesty konkrétnymi linkami.

Obr. 35. Obrazovka Spojenia.



Obr. 36. Obrazovky Spojenia a Detail spojenia.

Testovanie

Testovacie scenáre sú uvedené v prílohe B.

2. User Story : Implementácia funkcionality Plánovanie spojení

Implementujte funkcionalitu Plánovanie spojení na serverovej strane, pozostávajúcu z databázovej vrstvy, samotnej funkcionality, prepojeniu so službou bežiacou na serverovej strane a implementácii samotnej služby.

Analýza

Cieľom je navrhnúť algoritmus vyhľadávania kombinácie dopravných spojov, pričom v prvotnej fáze je prihliadané na fakt, že prototyp bude pracovať nad databázou obsahujúcou informácie o cestovných poriadkoch mestskej hromadnej dopravy a výsledkom výpočtu daného algoritmu v prototype bude kombinácia dopravných spojov mestskej hromadnej dopravy v hlavnom meste Slovenskej republiky, respektíve v meste Bratislava. Vstupom daného algoritmu bude miesto odchodu, miesto príchodu, dátum a čas požadovaných informácií, maximálny akceptovateľný počet prestupov a maximálna akceptovateľná dĺžka pešieho presunu v minútach medzi jednotlivými zastávkami, pričom za rýchlosť chôdze sa považuje rýchlosť 3km/h. Posledné dva vstupné parametre sú pre používateľa nepovinné a v prípade, že sa ich rozhodne nevyplniť, tak pre výpočet algoritmu sa použijú predvolené hodnoty. Pre maximálny akceptovateľný počet prestupov je určená predvolená hodnota 2 a pre maximálnu akceptovateľnú dĺžku pešieho presunu medzi zastávkami je určená predvolená hodnota 5 minút. Výstupom algoritmu bude kombinácia, prípadne kombinácie dopravných spojov, ktoré budú začínať v miestne odchodu a končiť v mieste príchodu. Začiatočný čas a dátum týchto kombinácií bude rovnaký alebo neskorší ako zadaný čas a dátum.

Návrh

Pri implementácií algoritmu budú východiskom znalosti z grafových algoritmov, najmä dijskstrov algoritmus, avšak uskutoční sa optimalizácia vzhľadom na povahu úlohy. Podkladom pre výpočet algoritmu je tabuľka obsahujúca informácie o priamych dopravných spojoch medzi východiskovými zastávkami (jednotlivé riadky tabuľky) a cieľovými zastávkami (jednotlivé stĺpce tabuľky). Princíp algoritmu spočíva vo vytvorení určitého počtu jednorozmerných polí, pričom daný počet závisí od maximálneho akceptovateľného počtu prestupov, kde navyše existuje inicializačné jednorozmerné pole reprezentujúce, že nebol uskutočnený žiadny prestup. Položkami jednotlivých polí budú jednotlivé zastávky, pričom hodnotami daných položiek budú kombinácie dopravných spojov, ktorými bolo možné uskutočniť presun do konkrétnej zastávky. Za prestup sa nepovažuje peší presun medzi jednotlivými zastávkami. Pri získavaní výsledku sa v každom kroku (prestupe) vykonajú tri činnosti a to nájdenie zastávok, do ktorých je možné uskutočniť peší presun, nájdenie priamych dopravných spojov do zastávok, ktoré používateľ označil za cieľové a v prípade, že je akceptovaný ďalší prestup, tak nájdenie priamych spojov do ostatných zastávok. Za východiskové zastávky sa v každom kroku považujú také, ktoré sa nachádzajú v predošlom kroku, respektíve jednorozmernom poli. Do inicializačného poľa sa na začiatku výpočtu vložia východiskové zastávky vložené používateľom. Výsledkom sú hodnoty položiek, ktoré sú reprezentantmi cieľových zastávok každého jednorozmerného poľa vrátane inicializačného, pričom sa uskutoční výber najlepších, z pohľadu najskoršieho príchodu, požadovaného počtu kombinácií dopravných spojov.

Implementácia

Implementácia vychádza z návrhu algoritmu, pričom pri hľadaní hodnôt položiek jednotlivých jednorozmerných polí sa uchováva iba najlepšia hodnota. Daná hodnota sa získava postupne a získaním údajov z tabuľky, v ktorej sa nachádzajú informácie o priamych dopravných spojoch medzi jednotlivými zastávkami, vytvorením kombinácie predošlých dopravných spojov a nájdeného

dopravného spoja a porovnaním danej kombinácie s už existujúcimi kombináciami, pričom za parameter porovnávania sa považuje skorší príchod do cieľovej zastávky.

Použité štruktúry a samotný algoritmus sa nachádzajú v prílohe D.

Testovanie

Testovanie základných metód algoritmu na získaných dátach, dodatočných metód na získaných dátach, testovanie implementácie algoritmu na operačnom systéme Linux.

3. User Story : Práca s mapou

Poskytnúť používateľovi možnosť pracovať s mapou. Možnosť návratu na aktuálnu pozíciu, prepínanie zobrazenia mapa/satelit.

Analýza

V zobrazení mapy vo funkcionalite „Kde som?“ chýba možnosť náhľadu iného typu mapy, tak ako je to štandardné. Tak isto je nedostatkom nemožnosť vrátiť sa na aktuálnu polohu automaticky, ak si prezeráme mapu.

Návrh

Do panelu je nutné pridať tlačidlá pre návrat na aktuálnu polohu a prepínanie typu mapy.

Po stlačení tlačidla návratu sa do stredu mapy zobrazí aktuálna poloha. V prípade zmeny polohy je mapa automaticky aktualizovaná. Ak používateľ posunie mapu na iné miesto, prestane sa aktualizovať aktuálna poloha. Správanie by malo byť rovnaké, ako v aplikácii "Maps".

Implementácia

Do obrazovky s mapou je pridaný spodný panel. Zároveň je zabezpečené zobrazenie/skrytie panelu pri príchode/opustení obrazovky. Na vykreslenie typov máp využívame framework MapKit.

Obr. 37. Obrazovka Kde som?

Testovanie

Testovacie scenáre sú uvedené v prílohe B.

9. šprint

Deviaty šprint sme začali 5.4.2010 a ukončili 12.4.2010. Pokračovali sme v ňom v práci na naplnení požiadavky plánovania spojení. Nakoľko tento šprint sme skrátili na týždeň identifikovali sme iba jednu User Story.

1. User Story : Plánovanie spojení

Plánovanie spojenia umožňuje po zadaní štartovacej a cieľovej lokality naplánovať optimálne spojenie aj prostredníctvom kombinácie viacerých liniek.

- rýchle zistenie spojenia prostredníctvom zvoleného algoritmu
- rýchle zistenie spojenia prostredníctvom s využitím paralelizmu

Analýza

Analyzovali sme korekciu rozdielov algoritmu pri spustení na operačnom systéme Windows a Linux a pripojenie algoritmu v programovacom jazyku C na službu implementovanú v programovacom jazyku Python.

Prebehla analýza využitia viacerých jadier procesora na pre výpočet algoritmu.

Návrh

Optimalizácia algoritmu na základe novovytvorenej tabuľky nezvyčajných trás liniek – napr. cesta do vozovne.

Implementácia

Algoritmus sme pripojili k službe s využitím framework-u ctype. Opravili sme nájdené chyby, načítanie aktualizovaných dát z databázy a uloženie do dátového súboru.

Testovanie

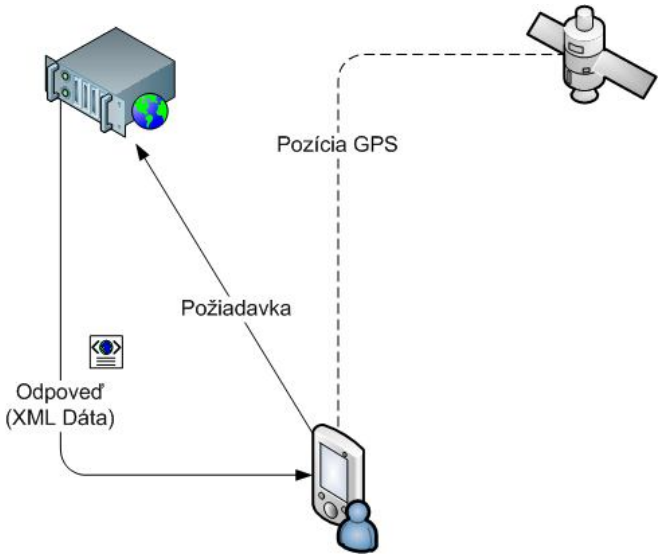
Testovacie scenáre sa nachádzajú v prílohe B.

OPIS VYTVORENÉHO PROTOTYPU

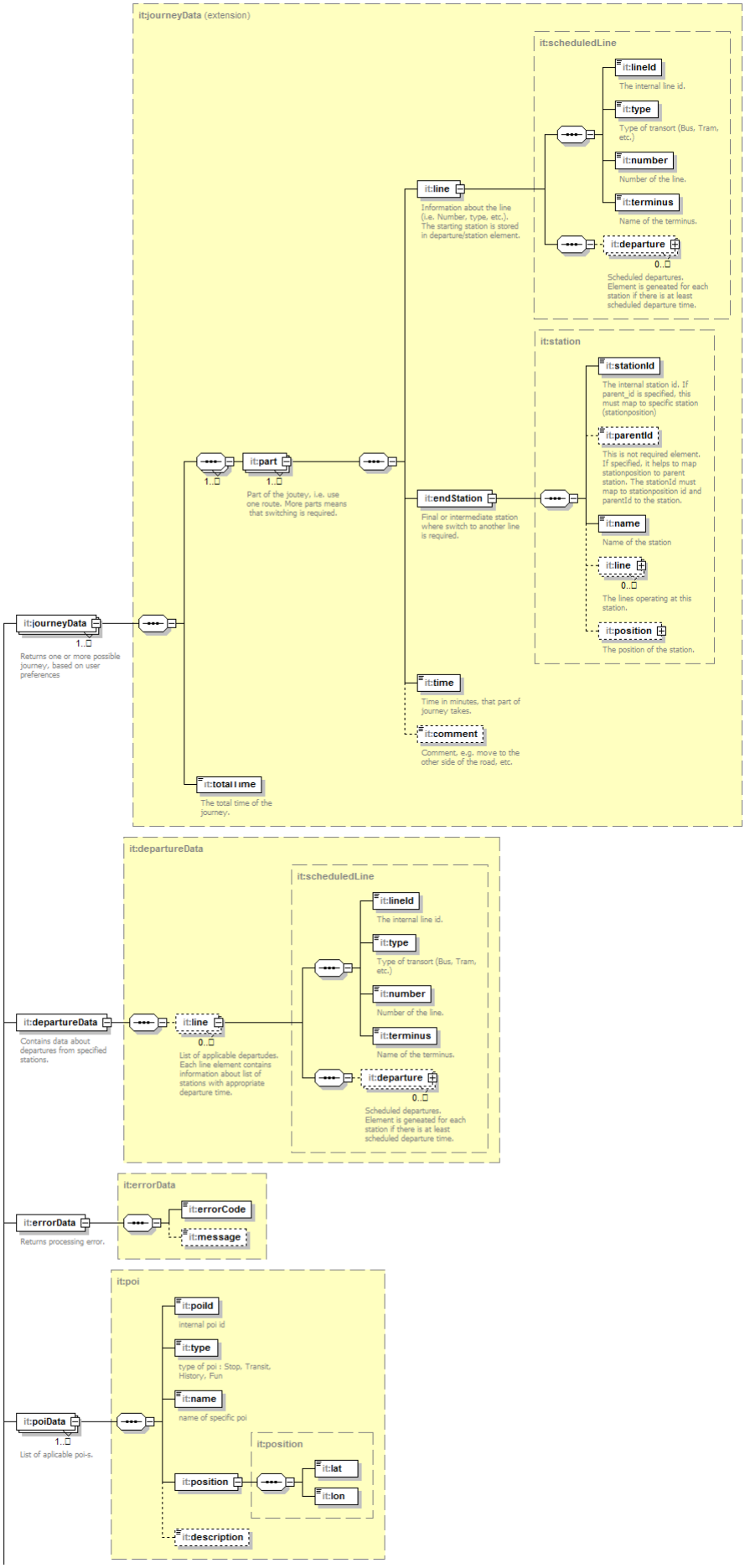
iTransit je aplikácia pre iPhone 3Gs, ktorá poskytuje mobilné cestovné poriadky pre bratislavskú MHD. Pri jej tvorbe sme sa zamerali na prívetivé používateľské rozhranie.

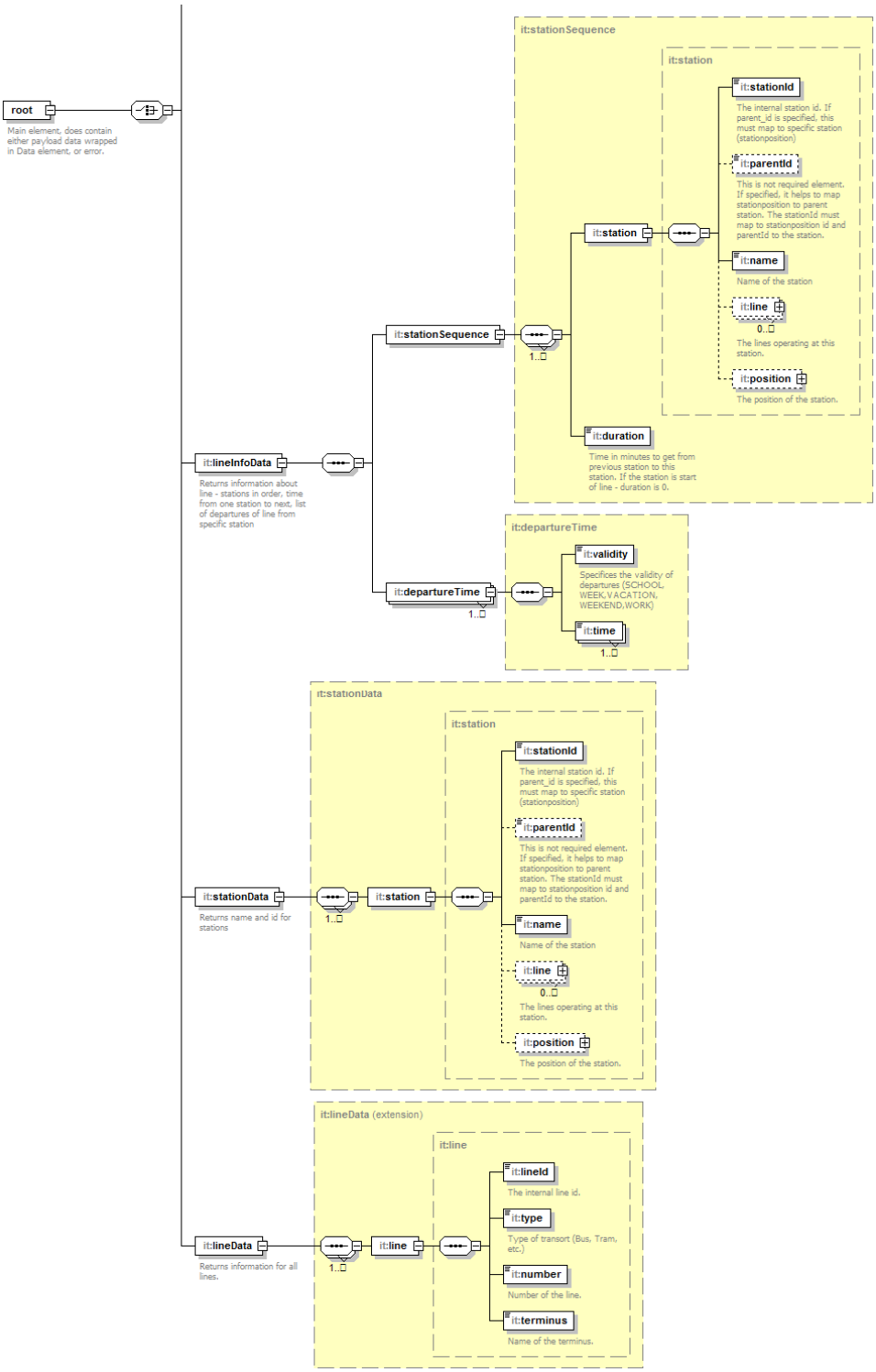
Architektúra

Prototyp využíva klient-server architektúru. Klientská časť pozostáva s aplikácie, ktorá komunikuje so serverom pomocou protokolu HTTP. Po zavolaní webovej služby a predaní parametrov dostáva odpoveď vo formáte XML. Využívané webové služby sú založené na štandarde REST. Formát dát je špecifikovaný pomocou schémy XSD (Obr. 39.). Na ukladanie a správu dát používa databázový systém PosgreSQL (Obr. 40.).

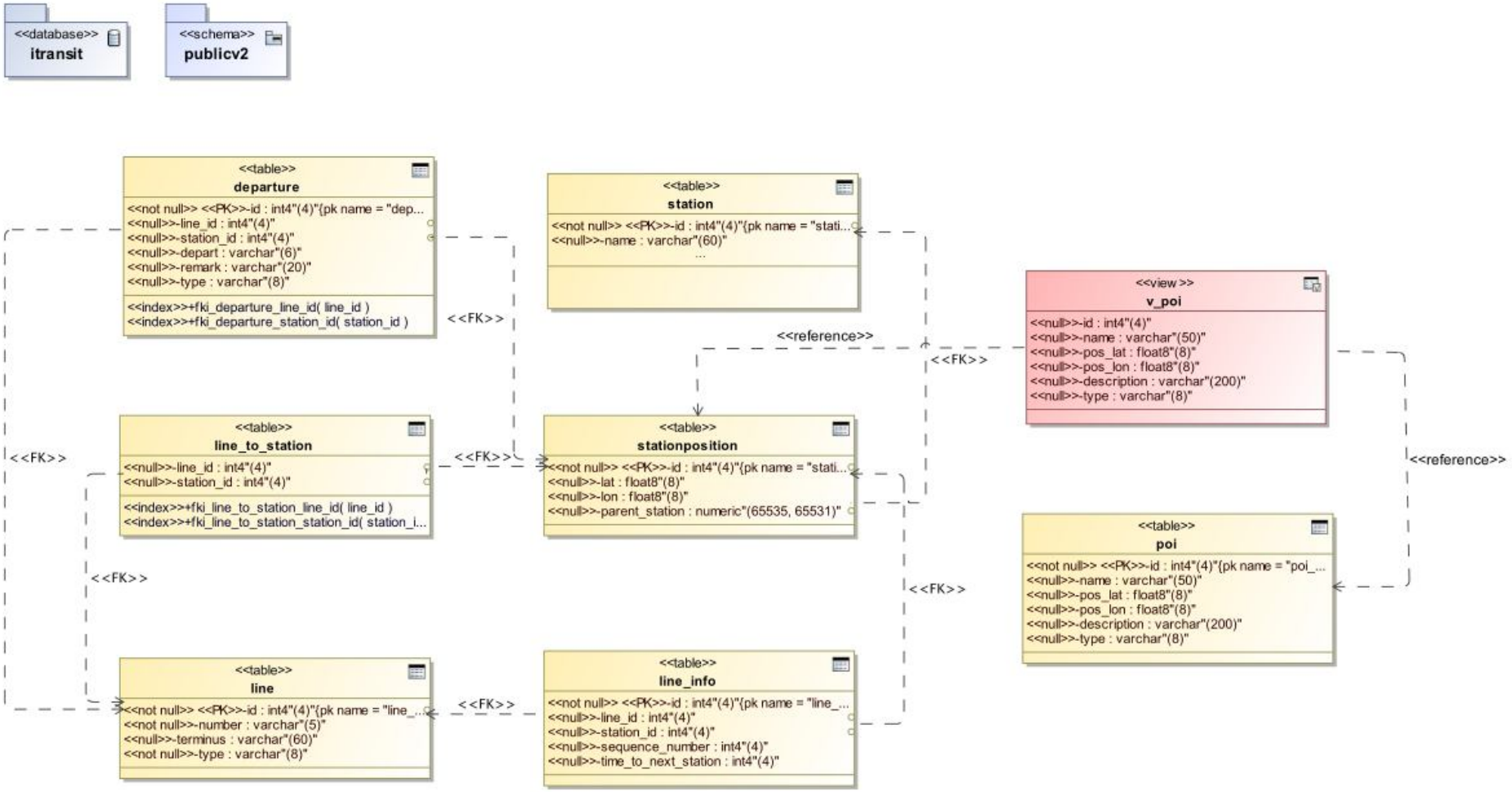


Obr. 38. Model komunikácie klient-server.


it:journeyData (extension)
it:scheduledLine
it:lineId
The internal line id.
it:type
Type of transort (Bus, Tram, etc.)
it:number
Number of the line.
it:terminus
Name of the terminus.
it:departure
0..□
Scheduled departures. Element is geneated for each station if there is at least scheduled departure time.
it:line
Information about the line (i.e. Number, type, etc.). The starting station is stored in departure/station element.
it:station
it:stationId
The internal station id. If parent_id is specified, this must map to specific station (stationposition)
it:parentId
This is not required element. If specified, it helps to map stationposition to parent station. The stationId must map to stationposition id and parentId to the station.
it:name
Name of the station
it:line
0..□
The lines operating at this station.
it:position
The position of the station.
1..□
it:part
1..□
Part of the joutey, i.e. use one route. More parts means that switching is required.
it:endStation
Final or intermediate station where switch to another line is required.
it:time
Time in minutes, that part of journey takes.
it:comment
Comment, e.g. move to the other side of the road, etc.
it:totalTime
The total time of the journey.
it:journeyData
1..□
Returns one or more possible journey, based on user preferences
it:departureData
it:scheduledLine
it:lineId
The internal line id.
it:type
Type of transort (Bus, Tram, etc.)
it:number
Number of the line.
it:terminus
Name of the terminus.
it:departure
0..□
Scheduled departures. Element is geneated for each station if there is at least scheduled departure time.
it:line
0..□
List of applicable departudes. Each line element contains information about list of stations with appropriate departure time.
it:departureData
Contains data about departures from specified stations.
it:errorData
it:errorCode
it:message
it:errorData
Returns processing error.
it:poi
it:poiId
internal poi id
it:type
type of poi : Stop, Transit, History, Fun
it:name
name of specific poi
it:position
it:lat
it:lon
it:position
it:description
it:poiData
1..□
List of aplicable poi-s.



Obr. 39. XSD schéma.



Obr. 40. Fyzický dátový model.

Používateľská príručka

Pre plnohodnotné použitie aplikácie je nutné pripojenie na internet.

Medzi jednotlivými funkciami sa dá prepínať pomocou tlačidiel v spodnom paneli. Zvýraznené tlačidlo označuje aktuálnu funkciu. Po spustení je aktívna funkcia Odchody.

Obr. 41. Spodný panel s tlačidlami.

Najbližšie Odchody

Po zapnutí aplikácie sa automaticky zobrazia najbližšie odchody MHD z najbližších zastávok v okolí, zistené pomocou GPS, v najbližšom čase. Ku každému spoju sa zobrazí

- zastávka, z ktorej spoj ide,
- číslo spoja,
- smer spoja,
- čas kedy ide,
- čas, ktorý zostáva do odchodu,
- typ spoja (autobus, električka, trolejbus, nočný autobus) znázornený ikonou

Obr. 42. Najbližšie odchody.

Čas zostávajúci do odchodu spoja sa aktualizuje automaticky, rovnako ako vzdialenosti. Aktualizovať údaje na obrazovke je možné tlačidlom v pravom hornom rohu.

Kde som?

Zobrazí sa mapa s vyznačenou aktuálnou polohou, vyznačenými najbližšími zastávkami a ďalšími význačnými objektmi, ako napr. kultúrne pamiatky, reštaurácie, atď. Aktuálnu polohu zistenú podľa GPS reprezentuje červená šípka.

Obr. 43. Kde som?

V dolnom paneli sa dá prepínať medzi rôznymi módmi zobrazenia mapy : Satelit, Hybridná, Mapa. Tlačidlo naľavo umožňuje návrat v mape na aktuálnu polohu.

Po kliknutí na ikonu šípky zastávky zobrazenej na mape sa zobrazia čísla liniek, ktoré na tejto zastávke stoja. Po kliknutí na šípku sa zobrazia dva najbližšie odchody pre každú z liniek. Podobne sa zobrazí celkový cestovný poriadok danej linky pre vybranú zastávku.

Obr. 44. Informácie o zastávke.

Zastávky

Po kliknutí na tretie tlačidlo v spodnej časti sa zobrazí obrazovka pre výber zastávky pre zobrazenie cestovných poriadkov. Na zobrazenie konkrétneho cestovného poriadku je nutné zadať zastávku a linku.

Výber zastávky môže uľahčiť index so začiatočnými písmenami, ktorý sa nachádza na pravej strane obrazovky, prípadne použitie rýchleho vyhľadávania, ktoré sa nachádza v hornej časti. Zastávky sú zoradené podľa abecedy.

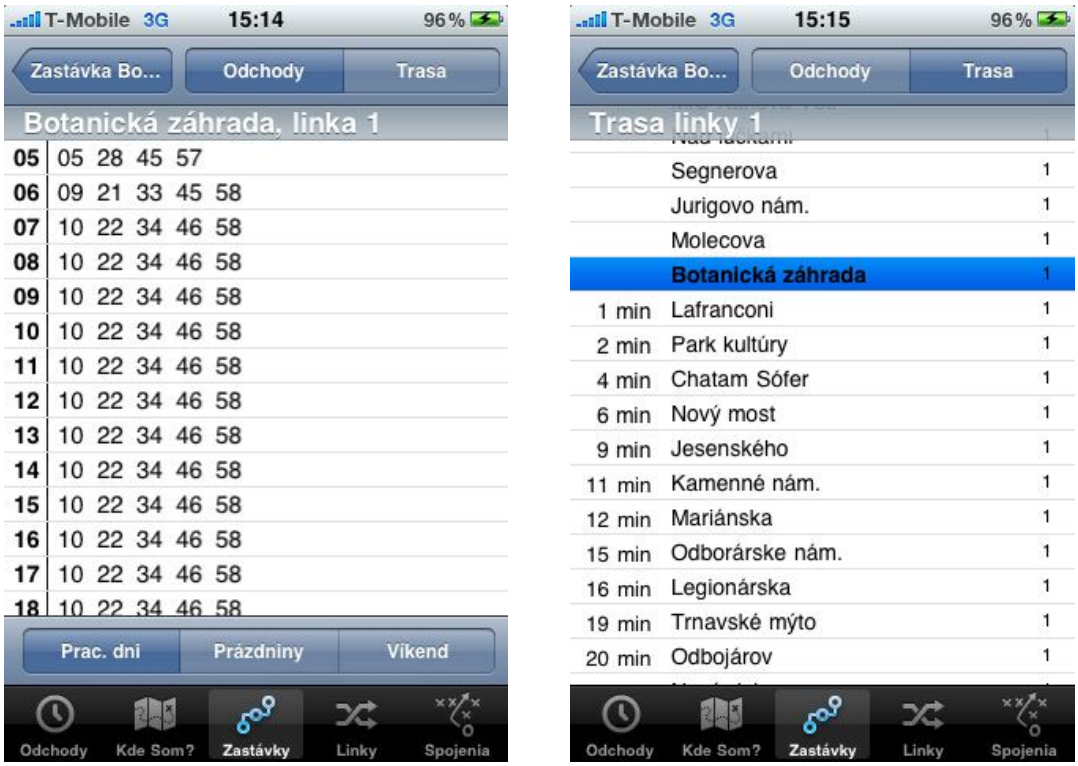
Po kliknutí na vybranú zastávku sa zobrazí nová obrazovka, ktorá slúži na výber linky. Je možné vybrať si zo zoznamu všetkých liniek prechádzajúcich danou zastávkou.

Obr. 45. Výber zastávky a linky.

Po výbere linky sa zobrazí nová obrazovka obsahujúca odchody zadanej linky zo zadanej zastávky počas pracovných dní, čo signalizuje v spodnej časti zvýraznené tlačidlo „Prac. dni“. Po kliknutí na jedno z ďalších dvoch tlačidiel „Prázdniny“ alebo „Víkend“ sa zobrazia odchody MHD v danom časovom obmedzení.

V hornej časti sa nachádza okrem zvýrazneného tlačidla „Odchody“ aj tlačidlo „Trasa“. Po jeho stlačení sa zobrazí trasa danej linky, to znamená, zobrazia sa zastávky na ktorých stojí linka, pričom zvolená zastávka je zvýraznená. Pri každej nasledujúcej zastávke je uvedený čas v minútach, ktorý označuje dĺžku cesty zo zvolenej zastávky a tarifné pásmo, v ktorom sa zastávka nachádza.

V hornej časti sa ešte nachádza tlačidlo „Zastávka“, pomocou ktorého sa je možné vrátiť na výber linky.



Obr. 46. Cestovné poriadky.

Linky

Podobnú kliknutí na tretie tlačidlo v spodnej časti sa zobrazí obrazovka pre výber linky pre zobrazenie cestovných poriadkov. Na zobrazenie konkrétneho cestovného poriadku je nutné zadať zastávku a linku.

Výber zastávky môže uľahčiť index s význačnými číslami liniek, ktorý sa nachádza na pravej strane obrazovky, prípadne použitie rýchleho vyhľadávania, ktoré sa nachádza v hornej časti. Linky sú usporiadané v skupinách podľa typu.

Po kliknutí na vybranú linku sa zobrazí nová obrazovka, ktorá slúži na výber zastávky. Je možné vybrať si zo zoznamu všetkých zastávok na trase danej linky.

Po výbere linky a zastávky sa zobrazí nová obrazovka obsahujúca odchody zadanej linky zo zadanej zastávky tak ako v Linky.

Obr. 47. Výber linky a zastávky.

Spojenia

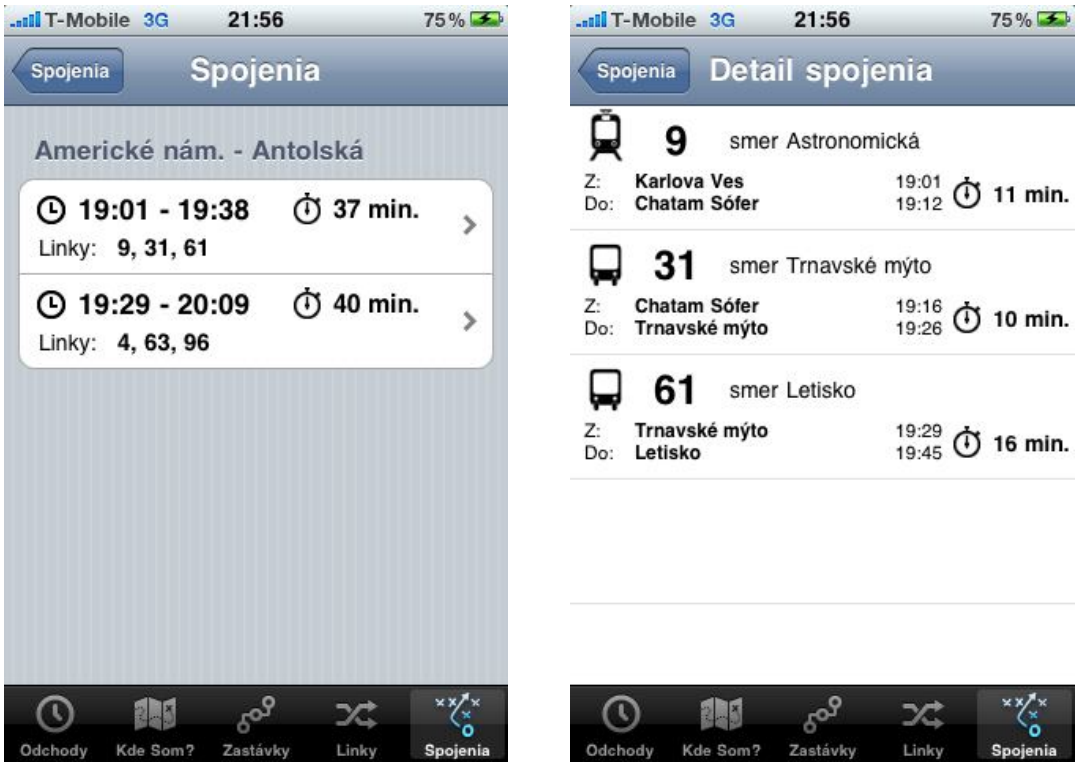
Táto funkcia poskytuje vyhľadanie spojenia medzi zastávkami. Čas odchodu (Odchod) je prednastavený na aktuálny.

Obr. 48. Vyhľadanie spojenia - zadávanie údajov

Zastávky (Štart, Cieľ) si je možné vybrať zo zoznamu všetkých zastávok. Po kliknutí na šípku, vyhľadáte zastávku podobne ako v „Zastávky“. Čas odchodu je tiež možné zmeniť kliknutím na šípku, objaví sa nová obrazovka na výber iného času. Stlačením tlačidla „Done“ v hornej časti je zvolený nový čas. Ak ste si voľbu nového času rozmysleli, dá sa jednoducho vrátiť späť stlačením „Spojenia“ v hornej časti obrazovky. Je možné nastaviť maximálny počet prestupov, prípadne priemernú rýchlosť chôdze.

Obr. 49. Výber času

Po vyplnení údajov možno vyhľadať spojenia kliknutím na „Vyhľadaj spojenia“. Zobrazia sa jednotlivé spojenia pozostávajúce z čísiel liniek, celkového času cesty a času odchodu. Po kliknutí na detail spojenia sa zobrazia k jednotlivým linkám zastávky nástupu a výstupu, smer a časy odchodu.



Obr. 50. Vyhľadané spojenia.

PRÍLOHA A : DOKUMENTÁCIA K JEDNOTLIVÝM SLUŽBÁM

getDepartures

Popis služby

Služba getDepartures poskytuje informácie o odchodoch spojov z lokality v špecifikovanom čase. Pre zadanú vstupnú polohu nájde všetky zastávky v definovanom okolí. Pre každú zastávku vyhľadá spoje obsluhujúce danú zastávku, ku každej dvojici spoj - zastávka bude nájdený najbližší odchod.

Input

Endpoint URI: http://tp.ktokoho.info/api/getDepartures.api

Vstupné parametre:

- lat - GPS súradnica - zem. šírka – povinný
- lon - GPS súradnica - zem. dĺžka – povinný
- time - čas klienta v tvare hh:mm - povinný
- offset - v minútach, časový limit pre odchod spojení (time+offset < každý z odchodov), v prípade nevyplnenia automaticky nastavený na 60 – nepovinný
- radius - V metroch, max. vzdialenosť zastávky od používateľa, aby bola zahrnutá do výsledkov, v prípade nevyplnenia automaticky nastavená na 500 - nepovinný

Output

Example request:

http://tp.ktokoho.info/api/getDepartures.api?lat=48.154&lon=17.0754&time=16:50&offset=60&radius=500

Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<?xml version="1.0"?>
<p1:root xmlns:p1="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
      <p1:departureData>
            <p1:line>
                  <p1:lineId>string</p1:lineId>
                  <p1:type>NIGHTBUS</p1:type>
                  <p1:number>string</p1:number>
                  <p1:terminus>string</p1:terminus>
                  <p1:departure>
                        <p1:time>string</p1:time>
                        <p1:station>
                              <p1:stationId>string</p1:stationId>
                              <p1:name>string</p1:name>
                              <p1:position>
                                    <p1:lat>-1E4</p1:lat>
                                    <p1:lon>-1E4</p1:lon>
                              </p1:position>
```

```
                    </p1:station>
                </p1:departure>
            </p1:line>
        </p1:departureData>
</p1:root>
```

Error:

```
Code: 1 Message: Some parameter value is missing.
Code: 2 Message: Parameter lon has bad type. Maybe, it is not float number.
Code: 3 Message: Parameter lat has bad type. Maybe, it is not float number.
Code: 4 Message: Parameter time is bad.
Code: 5 Message: Parameter offset is bad.
Code: 6 Message: Parameter radius is bad.
Code: 7 Message: Cannot connect to DB.
```

getPoi

Popis

Služba getPoi poskytuje GPS súradnice vybraného typu poi (points of interests). Pre zadanú vstupnú polohu nájde všetky poi (zadaného typu) v definovanom rádiuse.

Input

Endpoint URI: http://api.itransit.sk/dev/getPOI.api?lat=48.154&lon=17.075&type=ALL&radius=200

Vstupne parametre:

- lat - GPS súradnica - zem. šírka - povinný
- lon - GPS súradnica - zem. dĺžka - povinný
- type - druh POI - (STOP, TRANSIT, FUN, HISTORY, ALL) v prípade nevyplnenia automaticky nastavený na ALL - nepovinný
- radius - v metroch, max. vzdialenosť POI od používateľa, v prípade nevyplnenia automaticky nastavená na 500 - nepovinný
- debug - ak nastavený na 1, tak radius sa nastaví na 500km - nepovinný

Output

Example request: Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
        <poiData>
                <poiId>2</poiId>
                <type>STOP</type>
                <name>ZOO</name>
                <position>
```

```
                        <lat>48.154000</lat>
                        <lon>17.075400</lon>
                </position>
        </poiData>
        <poiData>
                <poiId>35</poiId>
                <type>STOP</type>
                <name>Slávičie údolie</name>
                <position>
                        <lat>48.154700</lat>
                        <lon>17.074500</lon>
                </position>
                <description>linky: 31, 39, N31</description>
        </poiData>
</root>
```

V prípade, že pri spracovaní nastala chyba, tak výstupom je správa v nasledovnom formáte:

```
<?xml version="1.0"?>
<p1:root xmlns:p1="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
 <p1:errorData>
        <p1:errorCode>string by default</p1:errorCode>
        <p1:message>string by default</p1:message>
 </p1:errorData>
</p1:root>
```

Error :

```
Code: 1 Message: Some parameter value is missing.
Code: 2 Message: Parameter lon has bad type. Maybe, it is not float number.
Code: 3 Message: Parameter lat has bad type. Maybe, it is not float number.
Code: 6 Message: Parameter radius is bad.
Code: 7 Message: Cannot connect to DB.
Code: 8 Message: Parameter type is bad.
Code: 9 Message: XML create action failed.
Code: 10 Message: Parameter debug is bad.
```

getJourneyPlan

Popis

Služba getJourneyPlan poskytuje informácie o pláne cesty zo zadanej nástupnej zastávky do zadanej výstupnej zastávky v definovanom čase. Plán pozostáva z čísla linky, jej najbližšieho odchodu z nástupnej zastávky, času príchodu na výstupnú zastávku, celkového času cesty, prípadne prestupných zastávok - názov zastávky, čas príchodu, číslo novej linky a čas jej odchodu.

Input

Endpoint URI: http://api.itransit.sk/dev/getJourneyPlan.api?from=1&to=2&time=18:00

Vstupné parametre:

- from - id zastávky, z ktorej sa vychádza - povinný
- to - id zastávky, do ktorej sa chce dostať - povinný
- time - čas klienta/zadaný čas v tvare hh:mm - povinný
- change - maximálny počet prestupov, v prípade nevyplnenia nastavený na 3 - nepovinný

Output

Example request:

Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<it:root xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel
datamodel.xsd" xmlns:it="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<it:journeyData>
                <it:part>
                        <it:line>
                                <it:lineId>1</it:lineId>
                                <it:type>BUS</it:type>
                                <it:number>39</it:number>
                                <it:terminus>Súhvezdná</it:terminus>
                                <it:departure>
                                        <it:time>10:35</it:time>
                                        <it:station>
                                            <it:stationId>1</it:stationId>
                                            <it:name>ZOO</it:name>
                                        </it:station>
                                </it:departure>
                        </it:line>
                        <it:endStation>
                                <it:stationId>2</it:stationId>
                                <it:name>Zochova</it:name>
                        </it:endStation>
                        <it:time>12</it:time>
                         <it:comment>prestup</it:comment>
                </it:part>
                <it:part>
                        <it:line>
                                <it:lineId>3</it:lineId>
                                <it:type>BUS</it:type>
                                <it:number>93</it:number>
                                <it:terminus>Hlavná stanica</it:terminus>
                                <it:departure>
                                        <it:time>10:47</it:time>
                                        <it:station>
                                            <it:stationId>2</it:stationId>
                                            <it:name>Zochova</it:name>
                                        </it:station>
                                </it:departure>
                        </it:line>
                        <it:endStation>
                                <it:stationId>3</it:stationId>
                                <it:name>Hlavná stanica</it:name>
                        </it:endStation>
                        <it:time>7</it:time>
                </it:part>
```

```
                <it:totalTime>25</it:totalTime>
        </it:journeyData>
</it:root>
```

Error:

```
Code: 1 Message: Some parameter value is missing.
Code: 2 Message: Parameter "from" has bad type. Expecting integer value.
Code: 3 Message: Parameter "to" has bad type. Expecting integer value.
Code: 4 Message: Parameter time is bad.
Code: 5 Message: Cannot connect to DB.
```

getStations

Popis

Služba getStations poskytuje zoznam všetkých zastávok s ich ID a názvami. Ak je zadaný parameter lineId, zobrazia sa len zastávky, cez ktoré prechádza zvolený spoj.

Input

Endpoint URI: http://api.itransit.sk/api/getStations.api

Vstupné parametre:

- lineId - id spoja - nepovinný

Output

Example request:

Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
  <stationData>
        <station>
                <stationId>1</stationId>
                <name>Zochova</name>
        </station>
        <station>
                <stationId>2</stationId>
                <name>ZOO</name>
        </station>
  </stationData>
</root>
```

V prípade, že pri spracovaní nastala chyba, tak výstupom je správa v nasledovnom formáte:

```
<?xml version="1.0"?>
```

```
<p1:root xmlns:p1="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
 <p1:errorData>
        <p1:errorCode>string by default</p1:errorCode>
        <p1:message>string by default</p1:message>
 </p1:errorData>
</p1:root>
```

Error

```
Code: 7 Message: Cannot connect to DB.
```

getLineTimetable

Popis

Služba getLineTimetable poskytuje informácie o odchodoch linky na konkrétnej zastávke. Pre zadanú linku a zastávku vráti všetky časy odchodov danej linky z danej zastávky, všetky zastávky patriace k danej linke a časový interval medzi nimi.

Input

Endpoint URI: http://api.itransit.sk/tst/getLineTimetable.api?parStation=4015&line=1

http://api.itransit.sk/tst/getLineTimetable.api?station=30829&line=1

Vstupné parametre:

- station - id zastávky (stationposition) - povinne voliteľné
- parStation - id zastávky (station) - povinne voliteľné
- line - id linky - povinné

Output

Example request:

Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<it:root xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel
datamodel.xsd" xmlns:it="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <it:lineInfoData>
                <it:stationSequence>
                        <it:station>
                                <it:stationId>13848</it:stationId>
                                  <it:parentId>145</it:parentId>
                                <it:name>Hlavná stanica</it:name>
                        </it:station>
                        <it:duration>0</it:duration>
                        <it:station>
                                <it:stationId>38643</it:stationId>
                                  <it:parentId>12</it:parentId>
                                <it:name>SAV</it:name>
```

```
                        </it:station>
                        <it:duration>2</it:duration>
                </it:stationSequence>
                <it:departureTime>
                         <it:validity>SCHOOL</it:validity>
                        <it:time>10:45</it:time>
                        <it:time>10:55</it:time>
                        <it:time>11:05</it:time>
                        <it:time>11:15</it:time>
                </it:departureTime>
                 <it:departureTime>
                        <it:validity>WEEKEND</it:validity>
                        <it:time>10:00</it:time>
                        <it:time>10:30</it:time>
                </it:departureTime>
         </it:lineInfoData>
</it:root>
```

Error

```
Code: 1 Message: Some parameter value is missing.
Code: 2 Message: Parameter "station" or "parStation" has bad type.
Expecting integer value.
Code: 3 Message: Parameter "line" has bad type. Expecting integer value.
Code: 4 Message: Cannot connect to DB.
```

getStationInformation

Popis

Služba getStationInformation poskytuje informácie o zastávke. Pre zadanú zastávku vráti všetky čísla liniek, ktoré z nej odchádzajú.

Input

Endpoint URI: http://api.itransit.sk/dev/getStationInformation.api

Vstupné parametre:

- stop - id zastávky - povinné

Output

Example request: http://api.itransit.sk/dev/getStationInformation.api?stop=1

Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<it:root xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel
datamodel.xsd" xmlns:it="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <it:stationData>
                <it:station>
                        <it:stationId>1</it:stationId>
                        <it:name>ZOO</it:name>
```

```
                        <it:line>
                                <it:lineId>1</it:lineId>
                                <it:type>BUS</it:type>
                                <it:number>39</it:number>
                                <it:terminus>Súhvezdná</it:terminus>
                        </it:line>
                        <it:line>
                                <it:lineId>2</it:lineId>
                                <it:type>BUS</it:type>
                                <it:number>31</it:number>
                                <it:terminus>Trnavské mýto</it:terminus>
                        </it:line>
                </it:station>
        </it:stationData>
</it:root>
```

getLines

Popis

Služba getLines poskytuje zoznam všetkých liniek s ich ID, názvami, typom a cieľovej stanice. Návratovú množinu zastávok je možné obmedziť nepovinným parametrom stationId. V prípade, že je špecifikovaný, sú vrátené len spoje, ktoré prechádzajú danou zastávkou. Ak nie je špecifikovaný, služba vracia všetky spoje.

Input

Endpoint URI: http://api.itransit.sk/dev/getLines.api?stationId=...

Vstupné parametre:

- stationId - nepovinný, zastávka, cez ktorú spoj musí prechádzať. (id je mapované na tabuľku stationposition)
- parentId - nepovinný, zastávka, cez ktorú spoj musí prechádzať. (id je mapované na tabuľku station)

Output

Example request: Výstupom je správa v nasledovnom formáte (v prípade, že pri spracovaní nenastala chyba):

```
<it:root xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel
datamodel.xsd" xmlns:it="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <it:lineData>
                <it:line>
                        <it:lineId>1</it:lineId>
                        <it:type>BUS</it:type>
                        <it:number>39</it:number>
                        <it:terminus>Súhvezdná</it:terminus>
                </it:line>
                <it:line>
                        <it:lineId>2</it:lineId>
                        <it:type>BUS</it:type>
```

```
                        <it:number>31</it:number>
                        <it:terminus>Trnavské mýto</it:terminus>
                </it:line>
        </it:lineData>
</it:root>
```

Error

```
Code: 1 Message: Cannot connect to DB.
```

PRÍLOHA B : TESTOVACIE SCENÁRE

Roly

Analytik:

- Vytvorí testovacie scenáre
- Navrhne spôsob testovania

Tester:

- Vykoná testy
- Zapíše výsledky testov

Vývojár:

- Zúčastňuje sa testovania
- Odstráni nájdené nedostatky

Zákazník:

- Akceptuje výsledky testov

Popis prostredia

Testované riešenie sa konceptuálne skladá z dvoch častí. Prvou časťou je aplikácia nasadená na aplikačnom serveri. Funkcionalita, na ktorú sa vzťahuje tento dokument je zahrnutá v jednej REST službe na preddefinovanom endpointe. Táto služba poskytuje údaje druhej časti implementovaného riešenia. Klientská časť pozostáva s aplikácie, ktorá komunikuje so serverom pomocou protokolu HTTP. Po zavolaní služby a predaní parametrov dostáva odpoveď vo formáte XML. Dáta sú potom prezentované v grafickej forme používateľovi.

Testovacie scenáre pre 2. šprint

TS01: Služba getDepartures – úspešné prevzatie správy

Služba poskytuje dáta klientskej aplikácii. Je preto potrebné otestovať správanie služby a výstupné dáta bez pripojeného klienta.

Vstupné požiadavky:

Ako vstup slúži URI služby spolu s testovacími parametrami. Na serveri musí bežať testovaná služba. Databáza pre službu musí bežať a byť správne nakonfigurovaná. Databáza musí obsahovať testovacie dáta (súčasť dokumentu ako SQL skript, príloha 1). Tester musí mať prístup pre HTTP protokol na testovací server.

Testovacia URI:

http://tp.ktokoho.info/api-dev/getDepartures.api?lat=48.154&lon=17.0754&time=16:50&offset=60&radius=500

Očakávaný výstup:

XML správa obsahujúce požadované hodnoty. Správa je zároveň valídny dokument podľa XSD schémy (príloha 2).

Očakávaná XML správa:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
    <departureData>
        <line>
            <lineId>3</lineId>
            <type>BUS</type>
            <number>32</number>
            <terminus>Hlavna</terminus>
            <departure>
                <time>16:55</time>
                <time>17:07</time>
                <time>17:19</time>
                <time>17:31</time>
                <time>17:43</time>
                <station>
                    <stationId>2</stationId>
                    <name>ZOO</name>
                    <position>
                        <lat>48.154000</lat>
                        <lon>17.075400</lon>
                    </position>
                </station>
            </departure>
        </line>
        <line>
            <lineId>1</lineId>
            <type>BUS</type>
            <number>39</number>
            <terminus>Suhvezdna</terminus>
            <departure>
                <time>17:01</time>
                <time>17:21</time>
                <time>17:41</time>
                <station>
                    <stationId>27</stationId>
                    <name>Televizia</name>
                    <position>
                        <lat>48.156700</lat>
                        <lon>17.071000</lon>
                    </position>
```

```
            </station>
        </departure>
        <departure>
            <time>17:02</time>
            <time>17:22</time>
            <time>17:42</time>
            <time>17:02</time>
            <time>17:22</time>
            <time>17:42</time>
            <station>
                <stationId>2</stationId>
                <name>ZOO</name>
                <position>
                    <lat>48.154000</lat>
                    <lon>17.075400</lon>
                </position>
            </station>
        </departure>
    </line>
    <line>
        <lineId>2</lineId>
        <type>BUS</type>
        <number>31</number>
        <terminus>TrnavskeMyto</terminus>
        <departure>
            <time>16:56</time>
            <time>17:08</time>
            <time>17:16</time>
            <time>17:28</time>
            <time>17:36</time>
            <time>17:48</time>
            <station>
                <stationId>27</stationId>
                <name>Televizia</name>
                <position>
                    <lat>48.156700</lat>
                    <lon>17.071000</lon>
                </position>
            </station>
        </departure>
        <departure>
            <time>16:57</time>
            <time>17:09</time>
            <time>17:17</time>
            <time>17:29</time>
            <time>17:37</time>
            <time>17:49</time>
            <time>16:57</time>
            <time>17:09</time>
            <time>17:17</time>
            <time>17:29</time>
```

```
                <time>17:37</time>
                <time>17:49</time>
                <station>
                    <stationId>2</stationId>
                    <name>ZOO</name>
                    <position>
                        <lat>48.154000</lat>
                        <lon>17.075400</lon>
                    </position>
                </station>
            </departure>
        </line>
    </departureData>
</root>
```

Priebeh testu:

- Spustenie internetového prehliadača
- Vloženie URI –
- http://tp.ktokoho.info/api-dev/getDepartures.api?lat=48.154&lon=17.0754&time=16:50&offset=60&radius=500
- Prijatie výslednej požiadavky – xml správy
- Validácia oproti XSD schéme
- Porovnanie navrátených údajov so vzorovou správou uvedenou v tomto dokumente

TS02: Služba getDepartures – chýbajúce parametre

Účel scenára je otestovať správanie služby v prípade, že nie sú zadané všetky parametre

Vstupné požiadavky:

Ako vstup slúži URI služby spolu s testovacími parametrami. Jeden z povinných parametrov chýba. Na serveri musí bežať testovaná služba. Databáza pre službu musí bežať a byť správne nakonfigurovaná. Databáza musí obsahovať testovacie dáta (súčasť dokumentu ako SQL skript, príloha 1). Tester musí mať prístup pre HTTP protokol na testovací server.

Testovacia URI: http://tp.ktokoho.info/api-dev/getDepartures.api

Očakávaný výstup:

Služba vygeneruje chybovú správu s chybovým kódom 1. Podľa tabuľky chybových kódov (príloha 3) je chyba označená ako Some parameter value is missing.

Očakávaná XML správa:

```
<?xml version="1.0" encoding="UTF-8"?>
<p1:root xmlns:p1="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
    <p1:errorData>
        <p1:errorCode>1</p1:errorCode>
```

```
        <p1:message>Some parameter value is missing.</p1:message>
    </p1:errorData>
</p1:root>
```

Priebeh testu:

- Spustenie internetového prehliadača
- Vloženie URI – http://aplikacny-server/api/getDepartures.api
- Prijatie výslednej požiadavky – xml správy s informáciami o chybe
- Validácia oproti XSD schéme
- Výstupná správa musí obsahovať chybový kód podľa špecifikácie testu

TS03: Služba getDepartures – chybné parametre

Účel scenára je otestovať správanie služby v prípade, že nie sú parametre zadané v správnom formáte.

Vstupné požiadavky:

Ako vstup slúži URI služby spolu s testovacími parametrami. Jeden z povinných parametrov je v nesprávnom formáte. Na serveri musí bežať testovaná služba. Databáza pre službu musí bežať a byť správne nakonfigurovaná. Databáza musí obsahovať testovacie dáta (súčasť dokumentu ako SQL skript, príloha 1). Tester musí mať prístup pre HTTP protokol na testovací server.

Testovacia URI:

http://tp.ktokoho.info/api-dev/getDepartures.api?lat=8154&lon=17.0754&time=16:50&offset=60&radius=500

Očakávaný výstup:

Služba vygeneruje chybovú správu s chybovým kódom 3. Podľa tabuľky chybových kódov (príloha 3) je chyba označená ako Parameter lat has bad type. Maybe, it is not float number.

Očakávaná XML správa:

```
<?xml version="1.0" encoding="UTF-8"?>
<p1:root xmlns:p1="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
    <p1:errorData>
        <p1:errorCode>3</p1:errorCode>
        <p1:message>Parameter lat has bad type. Maybe, it is not float
number</p1:message>
    </p1:errorData>
</p1:root>
```

Priebeh testu:

- Spustenie internetového prehliadača
- Vloženie URI – http://aplikacny-server/api/getDepartures.api
- Prijatie výslednej požiadavky – xml správy s informáciami o chybe

- Validácia oproti XSD schéme
- Výstupná správa musí obsahovať chybový kód podľa špecifikácie testu

TS04: Služba getDepartures – chyba spracovania

Účel scenára je otestovať správanie služby v prípade, že nastane chyba pri spracovaní požiadavky

Vstupné požiadavky:

Ako vstup slúži URI služby spolu s testovacími parametrami. Na serveri musí bežať testovaná služba. Služba má nesprávne nakonfigurovaný prístup k databázovému serveru. Tester musí mať prístup pre HTTP protokol na testovací server.

Testovacia URI:

http://tp.ktokoho.info/api-dev/getDepartures.api?lat=48.154&lon=17.0754&time=16:50&offset=60&radius=500

Očakávaný výstup:

Služba vygeneruje chybovú správu s chybovým kódom 7**.** Podľa tabuľky chybových kódov (príloha 3) je chyba označená ako Cannot connect to DB**.**

Očakávaná XML správa:

```
<?xml version="1.0" encoding="UTF-8"?>
<p1:root xmlns:p1="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
    <p1:errorData>
        <p1:errorCode>7</p1:errorCode>
        <p1:message>Cannot connect to DB.</p1:message>
    </p1:errorData>
</p1:root>
```

Priebeh testu:

- Spustenie internetového prehliadača
- Vloženie URI – http://aplikacny-server/api/getDepartures.api
- Prijatie výslednej požiadavy – xml správy s informáciami o chybe
- Validácia oproti XSD schéme
- Výstupná správa musí obsahovať chybový kód podľa špecifikácie testu

TS05: klientská časť – používateľské rozhranie

Tento testovací scenár má za úlohu otestovať chyby v používateľskom rozhraní. Je preto potrebné eliminovať ostatné premenné (ako spracovanie správ, prípadne komunikáciu so serverom)

Vstupné požiadavky:

Aplikácia bežiaca v prostredí iPhone simulátora XCode. Pre naplnenie používateľského prostredia je použitý MOCK objekt obsahujúci predvyplnené údaje o spojoch.

Očakávaný výstup:

Používateľské rozhranie zobrazí všetky údaje z MOCK objektu bez viditeľných chýb v zobrazovaní.

Priebeh testu:

- Nastavenie MOCK objektu pri testovaní rozhrania
- Inicializácia objektu hodnotami
- Spustenie aplikácie v simulátore
- Vizuálna inšpekcia používateľského rozhrania

TS06: klientská časť – rozhranie klient/server

Tento testovací scenár má za úlohu otestovať rozhranie klientskej aplikácie cez ktorú komunikuje so serverovou časťou. Zadaním požiadavky na získanie údajov do servera sa musí vyslať požiadavka, prijať výsledok a naplniť dátami príslušné dátové entity. Ako zdroj dát je potrebné použiť MOCK službu vracajúcu statické dáta – XML správu (príloha 3).

Vstupné požiadavky:

Aplikácia bežiaca v prostredí iPhone simulátora XCode. Serverová časť emulovaná MOCK službou, vracajúca statickú XML správu. Simulátor má prístup na MOCK službu.

Očakávaný výstup:

Rozhranie vykoná dopyt, získa dáta a naplní príslušné dátové entity.

Priebeh testu:

- Nastavenie MOCK služby
- Vykonanie požiadavky na rozhranie klient/server pomocou Unit Testu
- Overenie prijatia výsledku
- Kontrola dátových objektov UnitTestom

TS07: komunikácia klient-server

Tento testovací scenár má za úlohu otestovať integráciu obidvoch súčastí do funkčného celku.

Vstupné požiadavky:

Aplikácia bežiaca v prostredí iPhone simulátora XCode. Služba bežiaca na aplikačnom serveri. Simulátor má prístup na aplikačný server. Databáza pre službu musí bežať a byť správne nakonfigurovaná. Databáza musí obsahovať testovacie dáta (súčasť dokumentu ako SQL skript, príloha Tester musí mať prístup pre HTTP protokol na testovací server. GPS súradnice posielané ako parameter do služby sú emulované MOCK objektom zastupujúcim GPS senzor.

Súradnice sú nastavené na: 48.154,17.076.

Očakávaný výstup:

Používateľské rozhranie zobrazí odchody spojov pre nasledovné zastávky:

- ZOO
- Televízia

Priebeh testu:

- Nastavenie a spustenie služby
- Nastavenie MOCK objektu ako GPS senzora
- Spustenie aplikácie v simulátore
- Vizuálna inšpekcia používateľského rozhrania
- Kontrola výpočtu časov do odchodu. Každý riadok obsahujúci čas odchodu obsahuje v zátvorke zostávajúci čas, Výpočert AKTUÁLNY ČAS – ODCHOD musí byť rovný tomuto údaju.

Výsledky testov

Táto časť dokumentu obsahuje tabuľku s výsledkami testovacích scenárov po ich vykonaní.

Tab. B-1. Výsledky testovacích scenárov.

Číslo	Názov	Tester	Výsledok
TS01	Služba getServices – úspešné prevzatie správy	Martin Jačala	OK
TS02	Služba getServices – chýbajúce parametre	Marek Brandobúr	OK
TS03	Služba getServices – chybné parameter	Michal Macko	OK
TS04	Služba getServices – chyba spracovania	Michal Macko	OK
TS05	klientská časť – používateľské rozhranie	Hana Časnochová	OK
TS06	klientská časť – rozhranie klient/server	-	netestované
TS07	komunikácia klient-server	Hana Časnochová	OK

Testovacie scenáre pre 3. šprint

TS01: Test simulátora GPS pozície

Objekt reprezentujúci simulátor poskytuje aplikácií v mobilnom telefóne simulované údaje GPS pozície telefónu pre potreby testovaní iných funkcionalít.

Vstupná požiadavka:

Vstupom je žiadosť aplikácie o GPS súradnice. Simulátor dokáže vygenerovať dvojicu GPS súradníc z požadovaného intervalu. Simulátor dokáže ponechať vygenerované súradnice po určitý časový interval (1 min). Po tomto intervale generuje novú náhodnú pozíciu alebo od poslednej súradnice generujú súradnice tak, ako by simuloval pohyb telefónu (pešia chôdza, bus, ...).

Požadovaný interval pre Bratislavu je

- lat 48° 05 48° 12
- lon 16° 50 17° 15

Povolený interval simulátora (pre Slovensko)

- lat 47° 40 49° 35
- lon 16° 50 22° 35

Očakávaný výstup:

Výstupom je dvojica GPS súradníc iba z povoleného intervalu.

Priebeh testovania:

- Tester spustí simulátor GPS
- Zapíše výsledok a overí, či lon a lan sú intervalu

Príklad lat = 48 0712 lon = 16 7821

TS02: Zobrazenie pozície na mape – úspešné

Aplikácia vykreslí príslušný úsek mapy Bratislavy na základe GPS súradníc mobilu s tým, že v strede zobranej mapy pozíciu vyznačí pozíciu na mape.

Vstupné požiadavky:

Vstupom je dvojica GPS súradníc (zatiaľ poskytnutá objektom simulovania GPS, neskôr nahradené mobilom), ktorá je iba povoleného intervalu. Ďalším vstupom je obrázok mapy danej lokality s príslušnou štvoricou dvojíc (lat, lon) GPS súradníc hraničných rožných bodoch na mape. Vstupné GPS súradnice musia reprezentovať bod na mape.

Očakávaný výstup:

Výstupom aplikácie je vykreslenie časť mapy na obrazovke mobilného telefónu s vyznačením aktuálnej pozície.

Priebeh testu:

- Spustenie aplikácie na mobile/simulačnom prostredí
- Spustenie „kde som“
- Zobrazenie mapy na displeji mobilu
- Spustenie google maps a zadanie GPS pozície
- Porovnanie oboch máp a polôh daných GPS

TS03: Zobrazenie pozície na mape – neúspešne GPS mimo rozsah mapy

Aplikácia vykreslí príslušný úsek mapy Bratislavy na základe GPS súradníc mobilu s tým, že v strede zobranej mapy pozíciu vyznačí pozíciu na mape.

Vstupné požiadavky:

Ako vstupné dáta GPS označujúce bod mimo mapy. Prípadne chýbajúca mapa.

Očakávaný výstup:

Namiesto vykreslenia časti mapy aplikácia sa zobrazí v mobile/simulátore hlášku o tom, že *„Miesto, kde sa nachádzate ,nie je možné zakresliť na mapu, lebo je mimo mapy“*.

Priebeh testu:

- Spustí aplikáciu na mobile/simulátore
- Nastavím objekt simulujúci GPS pozíciu na generovanie GPS súradníc mimo rozsahu mapy
- Zvolím možnosť „kde som“
- Zobrazenie na displeji hlášky „*Miesto*“

TS04: Zobrazenie pozície na mape – neúspešné – chýbajúca mapa, časť mapy

Aplikácia nedokáže zobraziť polohu na mape, pretože sa nevie k mape dostať. Buď to nie je nahraná v mobile.

Vstupné požiadavka:

Súradnice GPS aktuálnej pozície sa nachádzajú v mape. Mapa nie je lokalizovaná v mobile. Prípadne časť mapy server nepošle.

Očakávaný výstup:

Na mobile/simulátore sa nevykreslí časť mapy. Namiesto toho sa zobrazí hláška.

Priebeh testovania:

- Spustenie aplikácie na mobile/simulátore

- Vymažem mapu z mobilu, zmením jej lokalizáciu
- Zvolenie si možnosť kde som
- Zobrazenie hlášky.

TS05: Serverová časť – getPOI - úspešné vybranie informácií

Testovací scenár slúži na otestovanie getPOI na aplikačnom serveri. Klient sa pripojí na server prostredníctvom URL adresy a vyžiada sa informácie o POI vzhľadom na jeho polohu. Server odpovie vytvorením XML súboru s informáciami o POI v lokalite.

Vstupné požiadavka:

Vstupom je požiadavka pre server z klientskej strany o informácie o POI z databázy. V rámci požiadavky budú uvedené dodatočné informácie (GPS polohy mobilu, rozsah, typ). Musí existovať databáza pre POI a mala byť naplnená nejakými POI s informáciami. Server musí vedieť sa spojiť s databázou a extrahovať informácie z nej pomocou SQL príkazov.

Testovacia URI: http://api.itransit.sk/dev/getPOI.api?lat=48.154&lon=17.075&type=ALL&radius=200

Očakávaný výstup:

Výstupom je XML obsahujúce informácie o POI v danej lokalite.

Priebeh testu:

- Tester spojí sa na URL
- Server spracuje požiadavku, pripojí sa na databázu
- Server vykoná výpočty, vytvorí sql príkaz pre extrakciu informácií o POI
- Server vytvorí XML súbor s určenou schémou, do ktorého vloží vyextrahované informácie o POI.

Sem príde štruktúra XML súbor pre požiadavku a pre vloženie informácií o POI

```
<?xml version="1.0" encoding="UTF-8" ?>
<root xmlns="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
      <poiData>
            <poiId>2</poiId>
            <type>STOP</type>
            <name>ZOO</name>
            <position>
                  <lat>48.154000</lat>
                  <lon>17.075400</lon>
            </position>
      </poiData>
      <poiData>
            <poiId>35</poiId>
            <type>STOP</type>
            <name>Slávičie údolie</name>
            <position>
```

```
                <lat>48.154700</lat>
                <lon>17.074500</lon>
            </position>
            <description>linky: 31, 39, N31</description>
    </poiData>
    <poiData>
            <poiId>36</poiId>
            <type>STOP</type>
            <name>Slávičie údolie</name>
            <position>
                <lat>48.154000</lat>
                <lon>17.075400</lon>
            </position>
            <description>linky: 31, 39, N31</description>
    </poiData>
    <poiData>
            <poiId>38</poiId>
            <type>STOP</type>
            <name>Slávičie údolie</name>
            <position>
                <lat>48.154400</lat>
                <lon>17.076900</lon>
            </position>
    <description>linky: 30, 32, 37, 92, N29</description>
    </poiData>
</root>
```

TS06: Serverová časť – getPOI - chýbajúce parametre

Testovací scenár slúži na otestovanie getPOI na aplikačnom serveri. Klient sa pripojí na server prostredníctvom URL adresy a vyžiada sa informácie o POI vzhľadom na jeho polohu. Server odpovie vytvorením XML súboru s informáciami o POI v lokalite. Preto je dobré otestovať správane nie simulovaní chyby.

Vstupné požiadavky:

Nastavenie ľubovoľnej vlastností. Odpojenie databázy s POI od servera. Zmena testovacieho URL s vynechaním povinných parametrov.

Testovacia URI: http://api.itransit.sk/dev/getPOI.api?type=ALL&radius=200

Očakávaný výstup:

XML súbor s chybovou správou.

Priebeh testu:

- pripojenie sa na testovacie URL
- Prijatie XML správy o chybe

TS07: Klientská časť Zobrazenie najbližších zastávok – úspešné

Testovací scenár na otestovanie aplikačnej časti na iPhone v simulátore XCODE. Aplikácia zobrazuje najbližšie zastávky na mape.

Vstupné požiadavky:

Mapa sa nachádza v mobile, GPS súradnice polohy nie sú mimo mapy. Aplikácia v rámci User Story „kde som“ sa podarilo zobraziť časť mapy a lokalizáciu polohy na mape. Klientská časť sa pripojil na serveru a vyžiadal si zoznam zastávok v danom časti mapy. Prijaté údaje vo formáte XML o zástavkách zo servera spracovala a zobrazila na mape najbližšie zastávky.

Očakávaný výstup:

Na displeji v rámci mapy zobrazí okrem aktuálnej pozície aj ikonky najbližších zastávok (definícia najbližšie ich vzdialenosť od pozície je menej ale rovná *BlizkeZastávky*). Na ikonku zástavky sa dá kliknúť. Nad každou ikonkou je zobrazení meno zastávky.

Priebeh testovania:

- Spustenie aplikácie na iPhone
- Zvolenie možnosti „kde som“
- Aplikácia overí, že sa nachádzam v okolí Bratislavy
- Aplikácia pošle žiadosť o informácie o najbližších zastávok k mojej pozícií
- Po prijatí informácií o najbližších zástavkách, zobrazí ich na mape ako ikonky

TS08: Klientská časť Zobrazenie najbližších zastávok – neúspech, žiadne zastávky v blízkosti mojej pozície

Testovací scenár na otestovanie aplikačnej časti na iPhone v simulátore XCODE.

Vstupné požiadavka:

Mapa sa nachádza v mobile, GPS súradnice polohy nie sú mimo mapy. Aplikácia v rámci User Story „kde som“ sa podarilo zobraziť časť mapy a lokalizáciu aktuálnej polohy na mape. Klientská časť sa pripojila serveru vyžiadala si zoznam zastávok v danom časti mapy. Prijaté údaje vo forme XML súboru zo servera spracovala a zobrazila najbližšie zastávky

Očakávaný výstup:

Vykreslená mapa bez zastávok.

Priebeh testovania:

- Spustenie aplikácie na mobile
- Nastavenie GPS pozície na lokalitu Bratislavy bez zastávok
- Zvolenie možnosti „kde som“
- Aplikácia overí, že sa nachádzam v okolí Bratislavy

- Aplikácia pošle žiadosť o informácie o najbližších zastávok k mojej pozícií
- Aplikácia zobrazí iba mapu bez zastávok

TS09: Klientská časť showPOI– úspešné

Testovací scenár na otestovanie aplikačnej časti na iPhone v simulátore XCODE. Zobrazenie POI na danej časti mapy.

Vstupná požiadavka:

Klient má komponentu pre „google map“ a prijal a spracoval dáta od servera o POI pre daný úsek google map. Dáta obsahujú aspoň jednu POI. Klientské prostredie má implementovanú komponentu pre google map. Dokáže komunikovať so serverovou časťou a vyžiadať si od nej POI pre danú lokalitu v rámci aktuálnej časti google mapy. Po prijatí dát o POIs od servera vykreslí na google mapu prijaté POI.

Očakávaný výstup:

Vykreslená google mapa na obrazovke mobilu so zobrazením POI pre danú lokalitu.

Priebeh testu:

- Tester nastaví objekt simulácie GPS súradníc na lokalitu mapy s množstvom POI
- Tester zvolí možnosť zobrazenia POI na google mape
- Klient zobrazí google mapu
- Klient pošle požiadavku serveru o POI
- Server pošle klientovi dáta o POI
- Klient príjme, spracuje dáta o POI a zobrazí ich na mape

TS10: Klientská služba showPOI– úspech, žiadne POI

Testovací scenár na otestovanie aplikačnej časti na iPhone v simulátore XCODE. Testuje sa zobrazenie POI na google v prípade, že server neposlal žiadne dáta.

Vstupná požiadavka:

Klientské prostredie má implementovanú komponentu pre google map. Dokáže komunikovať so serverovou časťou a vyžiadať si od nej POI pre danú lokalitu v rámci aktuálnej časti pre POI. Po prijatí dát o POIs od servera vykreslí na google mapu prijaté POI. Prijaté dáta neobsahujú žiadne POI pre danú lokalitu.

Očakávaný výstup:

Vykreslená google mapa so zobrazením žiadneho POI.

Priebeh testu:

- Tester nastaví objekt simulácie GPS súradníc na časť, kde sa nenachádzajú žiadne POIs

- Tester zvolí možnosť pre zobrazenie POI na mape
- Klient vyšle požiadavku pre server o POI
- Server príjme, identifikuje a obslúži požiadavku
- Server vytvorí XML dáta o POI pre klienta a pošle mu ho
- Klient príjme dáta o POI obsahujúce žiadne POI
- Klient zobrazí iba google mapu pre danú lokalitu bez POI

TS11: Klientská služba showPOI - neúspešné, žiadne spojenie so serverom

Testovací scenár na otestovanie aplikačnej časti na iPhone v simulátore XCODE. Testuje sa zobrazenie POI na mape v prípade, že klient sa nedokáže pripojiť na server a vyžiadať si zoznam POI a informácie o nich.

Vstupná podmienka:

Klientská časť nedostane odpoveď na požiadavku o zaslanie dát o POI pre danú lokalitu google mapy. Klientská časť predpokladá, že server nevie odpovedať, je vypnutú alebo neexistuje spojenie. Zobrazí iba mapu pre danú lokalitu a hlášku o nedostupnosti servera pre daný okamih. Server je odpojený.

Očakávaný výstup:

Klient zobrazí google mapu, no v hornej časti zobrazí správu o nedostupnosti server v tomto okamihu.

Priebeh testovania:

- Správca server odpojí server
- Tester zvolí možnosť pre zobrazenia POI na mape
- Klient pošle požiadavku pre server o POI a čaká na odpoveď
- Klient zobrazí google mapu
- Klient po neprijatí odpovede od servera po určitom časom intervali zobrazí správu o nedostupnosti servera.

Výsledky testov

Táto časť dokumentu obsahuje tabuľku s výsledkami testovacích scenárov po ich vykonaní.

Tab. B-2. Výsledky testovacích scenárov.

Číslo	Názov	Tester	Výsledok
TS01	Simulovanie GPS suradnic – úspešné	Martin Jačala	OK
TS03	Zobrazenie pozície na mape – neúspešne GPS mimo rozsah mapy		
TS04	Zobrazenie pozície na mape – neúspešné – chýbajúca mapa, časť mapy		
TS05	Serverová časť - getPOI -úspešné vybranie informácií	Martin Blažko	OK
TS06	Serverová časť - getPOI -chýbajúce parametre	Martin Blažko	OK
TS07	Klientská časť - Zobrazenie najbližších zastávok - úspešné		
TS08	Klientská časť - Zobrazenie najbližších zástavok - neúspech, žiadne zastávky v blízkosti mojej pozície		

TS09	Klientská časť showPOI– úspešné
TS10	Klientská služba showPOI– úspech, žiadne POI
TS11	Klientská služba showPOI . neúspešné, žiadne spojenie so serverom

Testovacie scenáre pre 4. šprint

TS01: Serverová časť getStations - úspešné

Testovací scenár pre otestovanie serverovej služby getStations. Testovanie poskytnutia všetkých zastávok z databázy na serveri.

Vstupná podmienka:

Databáza je naplnená dátami o zástavkách a musí byť správne nakonfigurovaná. Aplikačný server beží a má implementovanú službu getStation.api. Služba poskytne názvy a id všetkých zastávok nachádzajúcich sa v databáze. Výsledok zobrazí vo formáte XML schémy.

Testovacia URI: http://api.itransit.sk/dev/getStations.api

Očakávaný výstup

Výstupom bude spáva v nasledujúcom formáte, skrátená ukážka výstupu pre dve zastávky ZOO a Zochová.

```
<it:root xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel
datamodel.xsd" xmlns:it="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <it:stationData>
                <it:station>http:
                        <it:stationId>1</it:stationId>
                        <it:name>Zochova</it:name>
                </it:station>
                <it:station>
                        <it:stationId>2</it:stationId>
                        <it:name>ZOO</it:name>
                </it:station>
        </it:stationData>
</it:root>
```

Priebeh testu:

- Tester vloží do priehliadača testovaciu URL http://api.itransit.sk/api/getStations.api
- Pripojí sa na databázu a získanie zoznamu zastávok z databázy
- Porovná oba zoznamy prvý získaný z testovacej URL s druhým získaním priamo z databázy

TS02: getJorneyPlanApi.api - úspešné

Služba getJourneyPlan poskytuje informácie o pláne cesty zo zadanej nástupnej zastávky do zadanej výstupnej zastávky v definovanom čase. Plán pozostáva z čísla linky, jej najbližšieho odchodu z nástupnej zastávky, času príchodu na výstupnú zastávku, celkového času cesty, prípadne prestupných zastávok - názov zastávky, čas príchodu, číslo novej linky a čas jej odchodu.

Vstupná požiadavka:

Databáza je naplnená údajmi, musí byť zadaná nástupná a výstupná zastávka.

Testovacia URI: http://api.itransit.sk/api/getJorneyPlanApi.api&from=2&to=4&time=07:34

Očakávaný výstup:

V prípade bez objavenia chyby, výsledkom bude XML súbor obsahujúci plán cesty zo zadanej zastávky do zadanej výstupnej zastávky.

Priebehu testu:

- Tester sa pripojí na EndPoint URI :
- Nastaví parameter nástupnej a výstupnej zastávky
- Vytiahne informácie pre
- Pripojí sa na stránku www.imhd.zoznam.sk/ba/index.php?w=3825242e29ef2f3033ef2f30253ea a vloží obe zástavky
- Porovná výsledky www.imhd.sk a služby

TS03: getJorneyPlanApi.api - neúspešné, chýbajúce parametre

Služba na serveri poskytuje dáta o pláne cesty zo zvolenej nastupujúcej zastávky na zvolenú vystupujúcu zastávku. Preto je treba otestovať, či správne funguje pri nezadaní povinných parametrov. Výsledkom by malo byť zobrazenie chybovej hlášky.

Vstupná požiadavka

Server má implementované službu getJorneyPlan. Pri vložení testovacej URI vynechanie povinných vstupných parametrov pre nástupnú, výstupnú zastávku, alebo time.

Testovacia URI: http://kto.koho.info/tp/api/getJorneyPlanApi.api

http://kto.koho.info/tp/api/getJorneyPlanApi.api&to=3&time=10:49

http://kto.koho.info/tp/api/getJorneyPlanApi.api&to=3

Očakávaný výstup:

Vrátenie chybovej správy „Code 1: Message: Some parameter value is missing.“

Priebeh testu:

- Tester sa pripojí na EndPoint URI
- Vynechá aspoň jeden povinný vstupný parameter.
- Overenie, či v každom z uvedených prípadov sa nastala chyba

TS04: getJorneyPlanApi.api - chybné parametre

Služba na serveri poskytuje dáta o pláne cesty zo zvolenej nástupnej zastávky na zvolenú výstupnú zastávku. Preto je potrebné otestovať, či správne funguje pri zadaní chybných parametrov.

Vstupná požiadavka:

Nastavenie vstupného parametra iného formátu než je definované.

Očakávaný výstup:

Vrátenie jednej z nasledovných chybovej správy „Code 3: Message: “ alebo „Code 4: Message: “ alebo „Code 5: Message: “

TS05: Klientská časť - parsovanie zoznamu zástavok z XML

Aplikácia na klientovi po prijatí zoznamu zastávok v formáte XML súboru, vytiahne informácie o zástavkách a uschová si ich.

Vstupná požiadavka:

Aplikácia bežiaca v prostredí iPhone simulátora XCODE. Služba „getStations.api“ na aplikačnom serveri. Simulátor má prístup na aplikačný server. Databáza obsahuje tabuľku o zástavkách a musí byť správna nakonfigurovaná. Tabuľka obsahuje testovacie dáta.

Očakávaný výstup:

Zoznam zastávok získaný z aplikačného servera.

Priebeh testu:

- Vytvorenie Unit Testu pre otestovanie parsovania na klientovi.
- Vytvorenie fiktívneho XML súboru zo zastávkami zo servera
- Spustenie testu
- Overenie zoznamu zastávok s fiktívnym XML súborom
- Kontrola dátových objektov UnitTestom

TS07: Klientská časť - zobrazenie zoznamu zastávok v dialógu

Testovací scenár slúžiaci na otestovanie funkcionality zobrazenie dialógu zastávok. Po pripojení na server na EndPoint URL http://api.itransit.sk/dev/getStations.api a získaní XML pre zoznam všetkých zastávok klientom, spracovaním informácií o zástavkách aplikácia zobrazí dialóg obsahujúci ???.

Vstupné požiadavky:

Úspešné vytvorení získaného zoznamu od serveru a vytvorenie reprezentácie zastávok

Očakávaný výstup:

Dialógové okno pre zadávanie nástupnej zastávky a výstupnej zastávky plus prípadne iné parametre.(maximálny počet prestupov).

Priebeh testu:

- Tester zvolí možnosť Plánovanie cesty
- Aplikácia sa pripojí na URL a vyžiada si zoznam zastávok
- Server odpovie XML súborom
- Klientská aplikácia spracuje súbor, vytiahne informácie o zástavkách
- Klient zobrazí dialóg

TS08: Klientská časť - zobrazenie plánu na základe trasy - úspech

Testovací scenár testuje v rámci aplikácie v iPhone funkcionalitu zobrazenie výsledku cesty na displeji iPhone v simulátore.

Vstupné požiadavky:

Fiktívna XML správa o pláne cesty.

Očakávaný výstup:

Zobrazenie okna s informáciami o pláne cesty informáciami o odchode spoja zo nástupnej zastávky, číslom spoja, príchodom do cieľovej stanice, počtom prestupov a

Priebeh testu:

- Tester si pripraví XML súbor o pláne cesty
- Dá ho ako vstup pre parsovanie informácií
- Vyextrahované dáta nechá zobraziť na displej iPhonu.

TS09: Klientská časť - Integračný test čiastkových testov

Testovací scenár testuje výsledný spojenie jednotlivých testov do výslednej User Story na klientovi. Server a klient pracujú nad reálnymi dátami.

Vstupné požiadavky:

Aplikačný server musí mať implementované služby getJorneyPlan.api a getStations.api. Server je pripojený. Aplikácia dokáže sa pripojiť na obe URI a vyžiadať od servera. Musia byť zadané všetky povinné vstupné parametre.

Očakávaný výstup:

Postupnosť obrazoviek, ktorých výsledkom bude vyhľadanie a zobrazenie výsledného spojenia zo zadaných dvoch zastávok a to nástupnej a výstupnej. Od získania zoznamu všetkých zastávok pomocou getStations.api , zobrazenie dialógu pre voľbu nástupnej a výstupnej zástavky, pripojenie na server a službou getJourneyPlan.api poslanie plánu a zobrazenie výsledku na displeji mobilu.

Výsledky testov

Táto časť dokumentu obsahuje tabuľku s výsledkami testovacích scenárov po ich vykonaní.

Tab. B-3. Výsledky testovacích scenárov.

Číslo	Názov	Tester	Výsledok
TS01	Serverová časť getAllStations	úspešné	
TS02	getJorneyPlanApi.api	úspešné	
TS03	getJorneyPlanApi.api parametre	neúspešné, chýbajúce	
TS04	getJorneyPlanApi.api	chybné parametre	

TS05	Klientská časť parsovanie zoznamu zástavok z XML
TS06	
TS07	Klientská časť: Zobrazenie zoznamu zastávok v dialógu
TS08	Klientská časť: Zobrazenie plánu na základe trasy - úspech
TS09	Klientská časť: Integračný test čiastkových testov

Testovacie scenáre pre 5. šprint

TS01: getLineTimetable - úspešné

Testovací scenár otestuje službu getLineTimetable. Snaha otestovať, či služba vracia správny výsledok, ak jej zadáme všetky parameter pre daný spoj.

Testovacia URI: http://api.transit.sk/api/getLineTimetable.api?stop=4&line=1

Vstupná požiadavka:

Server má implementovanú službu getLineTimetable, server sa vie pripojiť k databáze.

Očakávaný výstup:

XML súbor, ktorý obsahuje informácie o danom spoji na konkrétnej zastávke

Priebeh testu:

- tester zadá endpoint uri http://api.transit.sk/api/getLineTimetable.api?
- mení postupne parametre a výsledky zaznamená
- pripojí sa na stránku www.imhd.sk a vyhľadá informácie nad spojmi, ktoré testoval
- porovná výsledky získané službou s výsledkami vyhľadanými na stránke

TS02: getLineTimetable - neúspešné, chýbajúce parametre

Testovací scenár testuje správanie služby getLineTimetable v prípade, že neboli zadanie všetky potrebné vstupné parametre.

Testovacia URI: http://api.transit.sk/api/getLineTimetable.api

Požadovaný vstup:

Server má implementovanú službu getLineTimetable, povinné vstupné parametre vynecháme. Databáza je prístupná pre server.

Očakávaný výstup

XML súbor s chybovou správou „Some parameters is missing“.

Priebeh testu:

- tester zadá endpoint url http://api.transit.sk/api/getLineTimetable.api bez uvedenia povinných parametrov
- Overí, či služba vráti XML súbor s chybovou hláškou „Some parameters is missing“

TS03: getStationInformation.api - úspešné

Testovací scenár má otestovať službu getStationInformation.api, či správne funguje a dáva relevantné dáta. T. j. Spoje odchádzajúce z danej zastávky.

Testovacia URI: http://api.itransit.sk/dev/getStationInformation.api&stop=125

Požadovaný vstup:

Sever má implementovanú službu getStationInformation.api, má pripojenie k databáze, povinné vstupný parameter je zadaný

Očakávaný výstup:

XML súbor so spojmi odchádzajúcimi z danej zastávky

Priebeh testu:

- tester vyberie zástavku, ktoré chce testovať
- zadá testovacie url s tým, že ako parameter stop zadá id zastávky z databázy
- výsledné XML uloží
- pripojí sa na stránku www.imhd.sk a vyhľadá si zastávku, ktorej id zadal
- výsledky porovná a vyhodnotí

TS04: getStationInformation.api - neúspešné, chýbajúci parameter

Testovací scenár má otestovať službu getStationInformation, či správne funguje pri chýbajúci parameter. Výsledkom je zobrazenie chybovej hlášky.

Testovacia URI: http://api.itransit.sk/api/getStationInformation.api

Vstupná požiadavka:

Server má implementovanú službu getStationInformation.api, má pripojenie k databáze, povinný parameter nie je zadaný

Očakávaný výstup

XML súbor s chybovou správou „Some parameter is missing“

Priebeh testu:

- tester zadá endpoint url http://api.itransit.sk/api/getStationInformation.api
- vynechanie parametera from
- Skontroluje, či dostal XML súbor s chybou hláškou „some paramter si missing“

TS05: getLineTimeTable.api - úspešné

Testovací scenár má otestovať správnu funkcionalitu serverovej služby **getLineTimeTable**, či sa správa adekvátne, ak dostane všetky potrebné parametre a vracia zodpovedajúci výsledok.

Testovacia URI: http://api.transit.sk/dev/getLineTimeTable.api?line=1&station=38029

Vstupná požiadavka:

Server má implementovanú službu getLineTimeTable, databáza je online a je prístupná pre server. Všetky povinné vstupné parametre sú zadané

Očakávaný výstup:

XML súbor, pozostávajúci zo zoznamu zastávok, a tabuľka pre jednotlivé typy odchody (WEEK, WEEKEND, WORK, VACATION, HOLIDAYS)

Priebeh testovania:

- tester vyhľadá z databázy zástavku a spoj
- tester pripojí sa na testovacie URL a parametre nastaví zodpovedajúce zástavke a spoju, pre ktoré to testuje
- výsledné XML od servera uloží
- pripojí sa na www.imhd.sk a vyhľadá zástavku a spoj, ktorú testoval
- porovná výsledky z XML súboru s informáciami na stránke imhd
- test opakuje pre ďalšie zástavky a výsledky zaznamenáva

TS06: getLineTimeTable.api - neúspešné chýbajúci parameter

Testovací scenár má otestovať správnu funkcionalitu serverovej služby **getLineTimeTable**, či sa správa adekvátne, ak služba nedostane všetky povinné vstupné parametre.

Testovacia URI: http://api.transit.sk/dev/getLineTimeTable.api?line=1

http://api.transit.sk/dev/getLineTimeTable.api?station=5

Vstupná požiadavka:

Server má implementovanú službu getLineTimeTable, databáza je online a je prístupná pre server, aspoň jeden povinný vstupný parameter nie je zadaný.

Očakávaný výstup:

XML súbor s chybovou správou „Some parameter is missing“

Priebeh testu:

- tester vyberie parameter pre spoj, zastávku, pričom úmyselne vynechá aspoň jeden povinný parameter.
- pripojí sa endpoint url http://api.transit.sk/api/getLineTimeTable.api
- Skontroluje XML, či obsahuje iba chybovú správu „Some pamater is missing“.

TS07: getLineTimeTable.api - neúspešné, neexistujúca zástavka, spoj

Testovací scenár má za úlohu otestovať funkcionalitu serverovej služby getLineTimeTable.api, či sa správa správne, ak dostane ako parameter neexistujúci spoj alebo zastávku. (napr. v prípade neúspešnej aktualizácie zoznamu spojov, zástavok).

Vstupná požiadavka:

Server má implementovanú službu getLineTimeTable, má pripojenie k databáze, vstupné parametre pre službu sú zadané, parameter spoj alebo zastávka reprezentujú neexistujúci spoj.

Očakávaný výstup:

Zobrazenie XML neobsahujúci žiadne informácie.

Priebeh testu:

- tester vyberie parameter pre spoj, zastávku, ktorú nie sú v databáze
- pripojí sa endpoint url http://api.transit.sk/api/getLineTimeTable.api?
- Skontroluje, či XML súbor vrátený službou, je prázdny

T08: Klientská časť - Zobrazenie okna s výsledkom služby getLineTimetable

Testovací scenár má otestovať funkcionalitu klientskej aplikácie po získaní informácie od serverovej služby getLineTimetable.api, vyparsovaní dát, a nakoniec zobrazí okno s informáciami s informáciami.

Vstupná požiadavka

Klient má prístup k internetu, server je online, databáza je online, server má naimplementovanú službu getLineTimetable.api.

Očakávaný výstup

Aplikácia zobrazí obrazovku s informáciami získanými od služby o spoji na danej zastávke

Priebeh testu:

- tester spustí simulátor
- vyberie možnosť pre informáciu o linke
- postupne volí čísla liniek a zobrazené obrázovky si ukladá vo formáte obrázkov
- Nakoniec výsledky porovná s informáciami o linkách na stránke www.imhd.sk.

TS09: Klientská časť - Zobrazenie informácie o danej zastávke - úspešné

Testovací scenár má otestovať funkcionalitu aplikácie pre zobrazenie informácií o zastávke. Testuje celý priebeh správania klientskej aplikácie pri voľbe „Informácie o zastávke“.

Vstupné požiadavky

Server má implementovanú služby getAllStations.api a getStationInformation.api, server sa dokáže spojiť s databázou, klient sa vie pripojiť na end point URLs:

http://api.itransit.sk/api/getStations.api

http://api.itransit.sk/api.getStationInformation.api

Očakávaný výstup

Aplikácia zobrazí obrazovku s informáciami získanými od služby o informácie o zastávke a aké linky z nej odchádzajú.

Priebeh testu

- tester spustí simulátor pre iPhone
- zvolí možnosť informácie o zastávke
- zvolí zastávku zo zoznamu zastávok , o ktorej chce získať
- výslednú obrazovku s informáciami zaznamená v podobe obrázka
- vyhľadá zastávku na www.imhd.sk
- porovná výsledky zo simulátora iPhonu a zo stránky www.imhd.sk

TS10: Klientská časť - Informácie o odchodoch zo zastávky pre daný spoj

Testovací scenár má otestovať klientskú aplikáciu na iPhone pre zobrazenie Informácie o odchodoch zo zástavky pre daný spoj.

Vstupné požiadavky:

Server má naimplementované služby getLines.api, getStations.api, getStationInformation.api a getLineTimeTable.api. Databáza je prístupná pre server.

Očakávaný výstup:

Obrazovka na iPhone s informáciami o nasledovných zástavka, s časovými rozdielmi medzi zástavkami a pre danú zástavku a spoj zobrazenie v tabuľkách odchody rozdelené podľa typov do tabuliek(VACCATION, WEEK, WEEKEND, WORK, SCHOOL).

Priebeh testu:

- tester spustí simulátor pre iPhone
- vyberie možnosť informácie o odchodoch pre linku z danej zastávky
- zvolí linku a zastávku
- iPhone zobrazí odchody
- pripojí sa na stránku www.imhd.sk a vyhľadá informácie, ktoré zadal ako vstupné paramtre
- porovná oba výsledky a zapíše výsledkov testu
- test opakuje pre iné kombinácie zástavok a spojov

Výsledky testov

Číslo	Názov	Tester	Výsledok
TS01	TS01: getLineTimetable - úspešné	Martin Blažko	netestované
TS02	TS02: getLineTimetable - neúspešné, chýbajúce parametre	Martin Blažko	netestované
TS03	TS03: getStationInformation.api - úspešné	Martin Blažko	netestované
TS04	TS04: getStationInformation.api - neúspešné, chýbajúci parameter	Martin Blažko	netestované
TS05	TS05: getLineTimeTable.api - úspešné	Martin Blažko	netestované

TS06	TS06: getLineTimeTable.api - neúspešné chýbajúci parameter	Martin Blažko	netestované
TS07	TS07: getLineTimeTable.api - neúspešné, neexistujúca zástavka, spoj	Martin Blažko	netestované
TS08	T08: Klientská časť Zobrazenie okna s výsledok od služby getLineTimetable.api		netestované
TS09	TS09: Klientská časť Zobrazenie informácie o danej zastávke - úspešné		netestované
TS10	TS10: Klientská časť Informácie o odchodoch zo zastávky pre daný spoj		netestované

Testovacie Scenáre pre 7. šprint.

User Story #141 Zobrazenie najbližších odchodov v obrazovke s informáciami o zastávke

Implementácia:		Testovanie:		
Vlastník US:	mce	Návrh testov:	mhr, mja,	Dátum
Implementácia:	mhr	Vykonanie testov:	mja	Dátum

Popis User Story

Používateľ klikol na mape na zastávku (alebo skôr podzastávku) a zobrazil sa mu zoznam liniek odchádzajúcich z tejto podzastávky. Pri každej linke sa tiež zobrazia dva najbližšie odchody.

Súvisiace úlohy

ID	Popis	Vlastník
#194	Implementovať zobrazenie najbližších odchodov do kde som - klik na zastávku.	mhr

Testovacie prípady

TC141-01 Zobrazenie najbližších odchodov v obrazovke s informáciami o zastávke

Do pohľadu sa doťahujú dáta z dvoch služieb - getDepartures a getLines. Test pre zastávku ZOO (smer Cintorín slávičie údolie).

Priebeh testu:

Krok	Popis	Príloha
1	Nastaviť vstup zo služby getLines aby pohľad získal priložené data.	A1
2	Nastaviť vstup zo služby getDepartures aby pohľad získal priložené data.	A2
3	Spustenie aplikácie, výber “Kde Som”, zastávka ZOO smer cint. Slávičie údolie	
4	Klik na disclosure button.	
5	Vizuálna inšpekcia podľa očakávaného výstupu	

Očakávaný výstup:

Zobrazené časy najbližších odchodov pre linky, ktoré odchádzajú z danej zastávky.

ID	Výstup	Stav
1	Dva najbližšie časy odchodov sú zobrazené iba pri linke 39	OK
2	Jeden najbližší odchod je zobrazený pri linke 31	OK
3	Žiadne najbližšie odchody nie sú zobrazené pri linke N31, názov konečnej zastávky je vertikálne centrovaný	OK

TC141-01 Zobrazenie najbližších odchodov v obrazovke s informáciami o zastávke – žiadne odchody

Do pohľadu sa doťahujú dáta z dvoch služieb - getDepartures a getLines. Test pre zastávku ZOO (smer Cintorín slávičie údolie).

Priebeh testu:

Krok	Popis	Príloha
1	Nastaviť vstup zo služby getLines aby pohľad získal priložené data.	A1
2	Nastaviť vstup zo služby getDepartures aby pohľad získal prázdne, priložené data.	A3
3	Spustenie aplikácie, výber "Kde Som", zastávka ZOO smer cint. Slávičie údolie	
4	Klik na disclosure button.	
5	Vizuálna inšpekcia podľa očakávaného výstupu	

Očakávaný výstup:

Žiadne zobrazené časy najbližších odchodov pre linky, ktoré odchádzajú z danej zastávky.

ID	Výstup	Stav
1	ku zobrazeným linkám nie sú priradené žiadne časy odchodov, všetky názvy konečných zastávok jednotlivých liniek sú vertikálne centrované.	

Odhalené nedostatky:

ID	Popis
-	

A1:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">
    <lineData>
        <line>
            <lineId>31</lineId>
            <type>BUS</type>
            <number>31</number>
            <terminus>Cintorín Slávičie údolie</terminus>
        </line>
        <line>
            <lineId>10039</lineId>
            <type>BUS</type>
            <number>39</number>
            <terminus>Cintorín Slávičie údolie</terminus>
        </line>
        <line>
            <lineId>431</lineId>
            <type>NIGHTBUS</type>
```

```
                    <number>N31</number>
                    <terminus>Cintorín Slávičie údolie</terminus>
                </line>
        </lineData>
</root>
```

A2:

http://api.itransit.sk/testDepartures.xml

A3:

```
<?xml version="1.0" encoding="UTF-8"?>

<root xmlns="http://www.itransit.sk/xsd/datamodel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://www.itransit.sk/xsd/datamodel/datamodel.xsd">

        <departureData>

        </departureData>

</root>
```

PRÍLOHA C : DIZAJN

Obr. C- 1. Dizajn stránky produktu.

Obr. C- 2. Zmenený dizajn stránky produktu.

Obr. C- 3. Plagát na podporu stránky produktu.

Obr. C- 4. Ukážky z reklamného prúžku.

PRÍLOHA D : PLÁNOVANIE SPOJENÍ

Použité štruktúry

Reprezentuje jednotlivé spoje/linky

```
struct {
        int linkNumber;         Číslo linky
        int time;               Čas odchodu v minútach od 00:00
        int duration;           Trvanie v minútach
        int flag;               Príznak odchodu spoja: 0-Nic, 1-celý týždeň, 2-víkend, 4-dni
                                školského  vyučovania, 8-sviatky
        int fromStationId;      ID zastávky, z ktorej vychádza
        int toStationId;        ID zastávky, do ktorej smeruje
} LINK;
```

Reprezentuje zoznam spojov/liniek

```
struct {
        int linksCount;         Počet spojov
        int indexPointer;       Ukazovateľ na spoj
        int overMidnight;       Príznak: 1-ide cez polnoc, 0-nejde cez polnoc
        int toStationId;        ID zastávky, do ktorej smerujú spoje tohto zoznamu
        LINK *links;            Jednotlivé spoje
} LINKLIST;
```

Mapovanie informácii o zastávke na indexy riadkov zoznamu priamych spojov v tabuľke zoznamov priamych spojov

```
struct {
        int index;              Index riadka
        int stationId;          ID zastávky
        int parrentId;          ID rodičovskej zastávky
        float latPos;           Lat GPS pozícia
        float lonPos;           Lon GPS pozícia
        char name[40];          Názov zastávky
} IDMAP;
```

Informácie o zastávke z pohľadu linky a poradia v trase linky

```
struct {
        int stationId;          ID zastávky
        int timeToNextStation;  Čas na ďalšiu zastávku
        int seqNum;             Poradové číslo zastávky
} STATIONSEQELEMENT;
```

Informácie o linke

```
struct {
        int lineId;                     ID linky
        char lineNum[5];                Číslo linky
        char type[10];                  Typ linky
        char terminus[30];              Koncová zastávka
        int stationsCount;              Počet zastávok
        STATIONSEQELEMENT *stationSeq;  Poradie zastávok
} LINE;
```

Kalendár

```
struct {
        int year;           Rok
        int month;          Mesiac
        int day;            Deň
        int type;           Typ dňa: 0 - Nič, 1 - Deň školského vyučovania, 2 - Víkend, 4 - Pracovný deň,
                            8 - Sviatok
} CALENDARDAY;
```

Informácie o spojoch, zastávkach a ich prepojeniach

```
struct {
        LINKLIST **zps;                 Tabuľka zoznamov priamych spojení
        int **distances;                Tabuľka vzdialenosti
        IDMAP *ids;                     Mapovanie informácii o zastávke
        int fromStationsCount;          Počet zastávok, z ktorých idú spoje
        int *toStationsCounts;          Počet zastávok, do ktorých spoje smerujú
        LINE *lines;                    Linky
        int linesCount;                 Počet liniek
        CALENDARDAY *calendarDays;      Kalendár
        int calendarDaysCount;          Počet dní v kalendári
} DIRECTLINKLISTTABLE;
```

Výsledný zoznam kombinácie spojov

```
struct {
        LINKLIST *zs;           Zoznamy spojov
        int count;              Počet zoznamov spojov
        int unused;             Príznak predošlého použitia
        int stationId;          ID zastávky
        int walkStation;        Príznak informujúci o použití pešieho presunu na danú zastávku
        LINK walkLink;          Informácia o pesom presune
} RESULTLINKLIST;
```

Algoritmu vyhľadávania kombinácie dopravných spojov

```
char *findShortestWay(int fromStationId, int toStationId, int maxMoveCount, int maxMoveTime,
int time, int day, int month, int year)
{
        DIRECTLINKLISTTABLE tzps;
        RESULTLINKLIST zvs;
        INTARRAY fromStationIds, toStationIds;
        char *result;
        int i;

        // nacitaj data zo suboru
        tzps = readDataFromFile("<nazov suboru>");

        // najdi podzastavky hlavnej zastavky s ID cislom fromStationId
        fromStationIds.items = (int*) malloc(tzps.fromStationsCount * sizeof(int));
        fromStationIds.count = 0;
        for (i = 0; i < tzps.fromStationsCount; i++)
        {
```

```
            if (tzps.ids[i].parrentId == fromStationId)
            {
                fromStationIds.items[fromStationIds.count] = tzps.ids[i].stationId;
                fromStationIds.count += 1;
            }
        }

        // najdi podzastavky hlavnej zastavky s ID cislom toStationId
        toStationIds.items = (int*) malloc(tzps.fromStationsCount * sizeof(int));
        toStationIds.count = 0;
        for (i = 0; i < tzps.fromStationsCount; i++)
        {
            if (tzps.ids[i].parrentId == toStationId)
            {
                toStationIds.items[toStationIds.count] = tzps.ids[i].stationId;
                toStationIds.count += 1;
            }
        }

        // najdi kombinacie dopravnych spojov
        zvs = findLinksCombination(tzps, fromStationIds, toStationIds, time, maxMoveCount,
        maxMoveTime, day, month, year);

        // vytvor XML odpoved
        result = createXmlResponse(zvs, tzps);

        return result;
}
```

RESULTLINKLIST findLinksCombination (DIRECTLINKLISTTABLE tzps, INTARRAY fromStationIds, INTARRAY toStationIds, int time, int maxMoveCount, int maxMoveTime, int day, int month, int year)

```
{
        RESULTLINKLIST **routes;
        RESULTLINKLIST result;
        INTARRAY toAnotherStationIds;
        int contains, i, j, index;

        // Alokacia jednorozmernych poli
        routes = (RESULTLINKLIST**) malloc((maxMoveCount + 1) * sizeof(RESULTLINKLIST*));
        for (i = 0; i <= maxMoveCount; i++)
        {
            routes[i] = (RESULTLINKLIST*) malloc(tzps.fromStationsCount *
            sizeof(RESULTLINKLIST));
        }

        // Inicializacia jednorozmernych poli
        for (i = 0; i <= maxMoveCount; i++)
        {
            for (j = 0; j < tzps.fromStationsCount; j++)
            {
                routes[i][j].unused = 1;
```

```
                routes[i][j].walkStation = 0;
                routes[i][j].count = 0;
        }
}

// Vkladanie hodnot do inicializacneho jednorozmerneho pola
routes[0] = insertFromStations(routes[0], tzps.ids, tzps.fromStationsCount, fromStationIds);

// Vytvaranie pola indexov necielovych zastavok
toAnotherStationIds.items = (int*) malloc(tzps.fromStationsCount * sizeof(int));
toAnotherStationIds.count = 0;
for (i = 0; i < tzps.fromStationsCount; i++)
{
        contains = 0;

        for (j = 0; j < toStationIds.count; j++)
        {
                if (tzps.ids[i].stationId == toStationIds.items[j])
                {
                        contains = 1;
                        break;
                }
        }

        if (contains == 0)
        {
                toAnotherStationIds.items[toAnotherStationIds.count] = tzps.ids[i].stationId;
                toAnotherStationIds.count += 1;
        }
}

// Vyhladavanie kombinacii dopravnych spojov
for (i = 0; i <= maxMoveCount; i++)
{
        // Zistenie zastavok s moznych pesim presunom
        routes[i] = insertWalkStations(routes[i], tzps.ids, tzps.distances,
        tzps.fromStationsCount, maxMoveTime);

        // Vyhladavanie priamych spojov do cielovych zastavok
        if ((maxMoveCount - i) > 0)
        {
                routes[i + 1] = findDirectLinks(routes[i], routes[i + 1], tzps, time, toStationIds,
                toStationIds, day, month, year, maxMoveCount);
        }

        // Vyhladanie priamych spojov do ostatnych zastavok
        if ((maxMoveCount - i) > 1)
        {
                routes[i + 1] = findDirectLinks(routes[i], routes[i + 1], tzps, time,
                toAnotherStationIds, toStationIds, day, month, year, maxMoveCount);
        }
}
```

```
	result.count = 0;
	// Inicializacia struktury reprezentujucej vysledok
	result.zs = (LINKLIST*) malloc(relevantLinkCount * sizeof(LINKLIST));

	// Vytvorenie struktury reprezentujucej vysledok
	for (i = 1; i <= maxMoveCount; i++)
	{
		for (j = 0; j < toStationIds.count; j++)
		{
			index = getRowIndex(tzps.ids, tzps.fromStationsCount, toStationIds.items[j]);
			result = insertZvsToZvs(result, routes[i][index]);
		}
	}

	return result;
}
```

RESULTLINKLIST *findDirectLinks(const RESULTLINKLIST *beforeRoutes, RESULTLINKLIST *afterRoutes, const DIRECTLINKLISTTABLE tzps, const int time, const INTARRAY toStationIds, const INTARRAY finalStationIds, const int day, const int month, const int year, const int maxMoveCount)

```
{
	int contains, toStationId, toStationIndex, i, j, k;

	// Hladanie priameho dopravneho spojenia medzi zastavkami
	for (i = 0; i < tzps.fromStationsCount; i++)
	{
		if (beforeRoutes[i].unused == 1 || tzps.toStationsCounts[i] <= 0)
			continue;

		contains = 0;
		for (j = 0; j < finalStationIds.count; j++)
		{
			if (beforeRoutes[i].stationId == finalStationIds.items[j])
			{
				contains = 1;
				break;
			}
		}

		if (contains == 1)
			continue;

		for (j = 0; j < tzps.toStationsCounts[i]; j++)
		{
			contains = 0;

			for (k = 0; k < toStationIds.count; k++)
			{
				if (tzps.zps[i][j].toStationId == toStationIds.items[k])
				{
					contains = 1;
```

```
                                break;
                        }
                }

                if (contains == 0)
                        continue;

                toStationId = tzps.zps[i][j].toStationId;
                toStationIndex = getRowIndex(tzps.ids, tzps.fromStationsCount, toStationId);

                // Najdenie najlepsieho priameho spojenia
                afterRoutes[toStationIndex] = findBestLinks(afterRoutes[toStationIndex],
                beforeRoutes[i], tzps.zps[i][j], time, tzps.calendarDaysCount,
                tzps.calendarDays, day, month, year, maxMoveCount);
        }
    }

    return afterRoutes;
}
```