České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Václav Legát**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **WheelTrip - multimodální mobilní navigace pro vozíčkáře**

Pokyny pro vypracování:

Navrhněte a implementujte mobilní navigační aplikaci určenou pro vozíčkáře. Při analýze vyjděte z předchozích prací studentů v rámci projektu NaviTerier. Zaměřte se především na ovládání aplikace pomocí joysticku a hlasového rozhraní. Dále se zaměřte na integraci navigační aplikace se senzory instalovanými na vozíku (jako například sensory otáčení kol, náklon vozíku, instalované kamery). Mobilní aplikace umožní vizualizaci dat ze senzorů a případné nastavování parametrů vozíku (u elektrické verze). Data ze senzorů využijte pro vlastní navigaci. Při návrhu a implementaci postupujte podle metodiky UCD a průběžně ověřujte řešení s uživateli.

Seznam odborné literatury:

[1] Martin Nuc: WheelGo - Plánovač tras pro vozíčkáře, diplomová práce, ČVUT v Praze, 2014.
[2] Kuniavsky, M. Observing the User Experience: A Practitioner's Guide to User Research. Morgan Kaufmann, 2003.
[3] Bill Buxton: Sketching User Experiences: Getting the Design Right and the Right Design (Interactive Technologies), 2013.

Vedoucí: doc. Zdeněk Míkovec Ing., Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

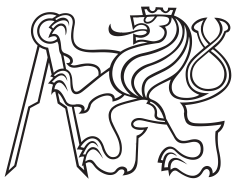
prof. Ing. Filip Železný, Ph.D.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2016

Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

WheelTrip – multimodální mobilní navigace pro vozíčkáře

Bc. Václav Legát

Květen 2016
Vedoucí práce: doc. Ing. Zdeněk Míkovec, Ph.D.

Poděkování / Prohlášení

Děkuji doc. Ing. Zdeňku Míkovcovi, Ph.D., za odborné vedení a věcné připomínky při psaní diplomové práce.

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Kladno 20. 5. 2016

.....

Abstrakt / Abstract

Diplomová práce se zabývá tvorbou mobilní navigační aplikace pro vozíčkáře. Navigace je určena pro využití při jízdě ve městě. K nalezení trasy a navigačních instrukcí aplikace využívá systém třetí strany. Aplikace informuje o problémových místech ve městě a také umožňuje vozíčkářům problémová místa nahlásit. Součástí aplikace jsou rovněž informace o bezbariérových místech, které jsou získány od samotných vozíčkářů. Důraz je kladen na využití multimodální interakce s aplikací. Kromě standardního ovládání dotykem je umožněno ovládat aplikaci také joystickem. Při interakci s aplikací je rovněž možno využít hlasového vstupu, například při vyhledávání míst. Hlasový výstup je použit v rámci navigace pro čtení instrukcí. Navigace je navržena tak, aby se při jízdě mohl vozíčkář maximálně soustředit na cestu.

Klíčová slova: navigační aplikace, mobilní aplikace, vozíčkáři, Android, joystick, rozpoznávání hlasu, převod textu na řeč

The master thesis deals with design and implementation of a mobile navigation application for wheelchair users. Navigation is intented for use while riding in the city. The application uses a third party system for routing process and navigation instructions. The application informs wheelchair users about difficulties on the route and provides users with an option to report other problems. Information about accessible places of interest is also part of the application. This information is verified by wheelchair users themselves. The application tries to be multimodal. Besides standard touch control, the application can be operated by a joystick. Voice input can be used to search places during interaction with the application. Text-to-speech service is used to read navigation instructions. Navigation is designed to be eyes free, so that wheelchair user can focus on the route as much as possible.

Keywords: navigation application, mobile application, wheelchair users, Android, joystick, speech recognition, text-to-speech

Title translation: WheelTrip – multimodal mobile navigation for wheelchair users

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

1.1 Motivace

Schopnost volného pohybu po městě je nedílnou součástí každodenního života. Pro většinu lidí je samozřejmostí, že si mohou dojít na nákup, do kina nebo se setkávat s přáteli. Ne všichni lidé však mají stejné možnosti bezproblémového pohybu po městě. Jsou omezeni různými překážkami, které jim mohou pohyb po městě komplikovat. Následkem těchto omezení dochází ke zhoršení kvality kulturního a společenského vyžití. U těchto lidí pak vzniká pocit izolovanosti, což může vést k rozvoji úzkosti či depresím [1]. Do této skupiny lidí patří osoby se sníženou schopností pohybu. V této práci bude tato skupina lidí reprezentována vozíčkáři.

V případě vozíčkářů existují mnohé překážky, které omezují volný pohyb na invalidním vozíku, v některých případech je volný pohyb dokonce znemožněn. Přestože při nových úpravách pozemních komunikací a veřejných prostranství – chodníky, přechody pro chodce, podchody – je zajištění bezbariérovosti jednou z hlavních podmínek, stále existují mnohá místa, která jsou pro vozíčkáře velmi špatně přístupná. Na jedné straně problémy vycházejí z vlastností povrchu komunikace – chodníku. Vozíčkáři potřebují pro bezpečnou jízdu rovný a pevný povrch. Na druhé straně jsou překážky, které blokují průjezd vozíku, například úzká komunikace nebo probíhající stavební práce. Avšak hlavním problémem pro vozíčkáře jsou schody. Obecně lze říci, že každé překonávání výškových rozdílů je pro vozíčkáře náročné. To, co je pro chodce pouze malý schůdek, je pro vozíčkáře často nepřekonatelný problém. Bez přítomnosti rampy či nájezdu je pro vozíčkáře výškový rozdíl hlavní překážkou volného pohybu. Často vozíčkáři potřebují asistenta, s jehož pomocí lze bezpečněji sjet schody nebo který jim pomůže do prudších nájezdů. Právě tato závislost na třetí osobě je pro vozíčkáře také omezující. Pokud by na druhé straně vozíčkáři věděli o omezeních, na která mohou na cestě narazit, a v lepším případě volit pouze cesty bez takovýchto omezení, snížili by tím závislost na pomoci třetí osoby a zvýšili tak pocit samostatnosti.

Známé aplikace pro vyhledávání tras, jako jsou Google Maps[1]) nebo Mapy.cz[2]), poskytují kromě standardního vyhledávání tras pro automobily také možnost vyhledávání tras pro chodce. Přestože se chodci a vozíčkáři pohybují po stejných typech komunikací, trasy pro chodce nelze v případě vozíčkářů využít. Platí, že ne všude, kam se dostane chodec, se dostane i vozíčkář. Hlavním důvodem je, že při vyhledávání tras pro chodce nejsou brány v potaz specifické požadavky uživatelů invalidního vozíku. S vysokou pravděpodobností se na trasách pro chodce mohou objevit schody či jiné překážky, které nepředstavují problém pro chodce, ale pro vozíčkáře jsou nepřekonatelné. Aplikace, která při hledání vhodné trasy zohledňuje omezení vozíčkářů, by mohla sloužit jako spolehlivý pomocník pro osoby upoutané na invalidní vozík.

[1]) `https://maps.google.com`
[2]) `https://mapy.cz`

1.2 Cíl práce

Cílem diplomové práce je navrhnout a implementovat první verzi navigační aplikace pro vozíčkáře. Aplikace bude implementována pro mobilní telefony s operačním systémem Android.

Práce bude zaměřena především na návrh uživatelského rozhraní aplikace, které tvoří stěžejní prvek mobilních aplikací. Uživatelské rozhraní musí být jednoduché na porozumění.

Velký důraz bude kladen na návrh uživatelského rozhraní, které umožní využívat externí ovladač k interakci s aplikací. To umožní využití aplikace pro širší okruh vozíčkářů, kteří mají další omezení, jež by jim zamezovala v interakci s aplikací pomocí dotykového ovládání.

Aplikace bude poskytovat informace o bezbariérových místech. Pro vozíčkáře je taková informace velmi důležitá. Vozíčkář se tak může dozvědět o problémech s přístupností spojených s daným místem dříve a může se tak vyhnout nepříjemným situacím, kdy až na místě zjistí, že nemá možnost místo navštívit.

Součástí aplikace budou také informace o problematických místech přímo v ulicích. Zároveň bude použit externí plánovač tras, který při plánování trasy tato omezení zohledňuje. Cílem plánovače je najít co nejoptimálnější trasu pro vozíčkáře.

1.3 Struktura

Diplomová práce je rozdělena na teoretickou a praktickou část. Teoretická část práce je reprezentována kapitolou 2 Analýza. Praktická část je rozdělena do tří kapitol: 3 Návrh, 4 Implementace a 5 Testování.

Kapitola 2 Analýza se zabývá popisem problematiky pohybu vozíčkářů ve městě. Kapitola obsahuje rozdělení vozíčkářů do dvou skupin podle vozíku, který používají a přehled omezení, s nimiž se obě skupiny setkávají při pohybu ve městě. Součástí kapitoly je také analýza technologií, které budou využity při implementaci aplikace. Kapitola zahrnuje i přehled existujících aplikací určených pro vozíčkáře. Na závěr kapitoly jsou definovány funkční a nefukční požadavky aplikace.

Hlavní tématem kapitoly 3 Návrh je princip návrhu uživatelského rozhraní aplikace. Kapitola obsahuje popis uživatelského rozhraní, funkčnosti jednotlivých obrazovek a systému ovládání aplikace. V kapitole se také nachází návrh struktury aplikace z pohledu implementace, a to rozdělení zdrojového kódu do balíčků nebo návrh nasazení aplikace.

V úvodu kapitoly 4 Implementace jsou představeny stěžejní programové závislosti aplikace. Poté následuje popis implementačních detailů základních prvků aplikace, kterou tvoří aktivity a fragmenty. V kapitole je také popsán způsob implementace ovládání, které umožní využití externího ovladače k interakci s aplikací. Na závěr kapitoly je představeno využití technologií popsaných v teoretické části práce.

První část kapitoly 5 Testování je věnována popisu využití automatizovaných testů k ověření správné funkčnosti aplikace. Druhá část kapitoly je věnována uživatelskému testování prototypu aplikace. Poslední část kapitoly poskytuje informace o uživatelském testování s cílovou skupinou uživatelů aplikace.

Kapitola 2
Analýza

V první podkapitole 2.1 této kapitoly bude popsána cílová skupina uživatelů aplikace. Rozdělíme uživatele do dvou skupin podle typu vozíku, který používají. Budou popsány výhody a nevýhody obou typů vozíků z pohledu jízdy ve městě. V podkapitole 2.3 budou označena problémová místa pro vozíčkáře. Budou popsána jednotlivá omezení, se kterými se mohou vozíčkáři setkat při jízdě po městě. V této podkapitole budou také uvedeny standardy, které by mělo splňovat veřejné prostranství tak, aby mohlo být označeno jako bezbariérové. V podkapitole 2.5 bude popsáno, jaké požadavky by měly splňovat navigační instrukce, aby byly pro uživatele srozumitelné. Dále bude v podkapitole 2.6 popsáno hlasové uživatelské rozhraní a systémy, které lze využít pro hlasem ovládané rozhraní. Důležitou částí této kapitoly je podkapitola 2.7, kde jsou popsány možnosti ovládání aplikace pomocí joysticku. Budou zde popsány způsoby zapojení externích ovladačů k mobilnímu zařízení a způsob návrhu uživatelského rozhraní tak, aby byla zajištěna jednoduchá integrace s ovladačem. V podkapitole 2.9 budou představeny existující aplikace určené pro vozíčkáře, jejich přínosy a nedostatky. V podkapitole 2.11 budou popsány požadavky získané z videokonference s konstruktérem elektrických vozíků. Na závěr kapitoly v podkapitole 2.12 budou představeny funkční a nefunkční požadavky aplikace.

2.1 Cílová skupina

Cílovou skupinu uživatelů aplikace tvoří vozíčkáři. Obecně lze za vozíčkáře označit všechny osoby, které k pohybu potřebují invalidní vozík. V České republice počet vozíčkářů není oficiálně evidován. V roce 2007 se počet vozíčkářů odhadoval na 30 000 s ročním přírůstkem okolo 300 osob [2]. Mezi vozíčkáře patří lidé s vrozenými vadami pohybového ústrojí, lidé po úrazech, při kterých došlo k poranění míchy, lidé omezeni v pohybu důsledkem nemoci nebo lidé po amputaci dolních končetin. Přehled klasifikace pohybových vad lze nalézt v [3].

Vzhledem k tomu, že v této skupině mohou existovat i lidé, kteří potřebují neustálou asistenci, musíme cílovou skupinu omezit pouze na uživatele, kteří jsou schopní ovládat mobilní aplikaci. To znamená na uživatele, kteří jsou schopni ovládat aplikaci pomocí dotykového ovládání nebo pomocí joysticku. Vybranou skupinu uživatelů můžeme dále rozdělit do dvou základních skupin, a to podle typu vozíku, který používají. První skupinu tvoří uživatelé mechanického vozíku. Tito uživatelé musí být fyzicky zdatní, protože potřebují k ovládání vozíku obě ruce. Jedná se především o lidi po úrazech, dopravních nehodách nebo amputacích. Mezi těmito vozíčkáři je velké procento aktivních lidí, bývalých sportovců, kteří chtějí zůstat aktivní i přesto, že zůstali upoutáni na vozík. Předpokládá se, že tito uživatelé budou používat aplikaci standardním způsobem. Budou využívat dotykovou obrazovku a interagovat s aplikací pomocí gest. Dále budou moci tito uživatelé k ovládání aplikace využívat také joystick, například z důvodu nemožnosti využít dotykové ovládání. To může být zapříčeněno například používáním rukavic při jízdě na vozíku.

Druhou skupinu tvoří uživatelé elektrického vozíku. Uživatelé elektrického vozíku mají zpravidla kromě omezení pohybu dolních končetin také další omezení, které jim neumožňuje používání mechanického vozíku. V těchto případech se jedná především o částečné omezení pohybu horních končetin. Uživatelé elektrického vozíku jsou zvyklí manipulovat s vozíkem pomocí ovladače. Při návrhu aplikace bude tato schopnost zohledněna a bude navrženo rozhraní, které bude možné ovládat pouze joystickem.

2.2 Typy vozíků

Invalidní vozíky výrazným způsobem zvyšují handicapovaným lidem životní komfort. Vozíčkáři nejsou odkázáni na pomoc druhých, a získávají tím větší samostatnost. V rámci projektu budeme rozlišovat vozíky podle typu pohonu, a to na vozíky mechanické, ovládané vozíčkářem pomocí obou rukou, a vozíky elektrické, ovládané jednou rukou pomocí joysticku.

Mechanický vozík. Mechanický vozík (obr. 2.1) je poháněn vlastní silou vozíčkáře. K ovládání je zapotřebí používat obě ruce. S tím souvisí i větší fyzická náročnost pohybu na mechanickém vozíku oproti elektrickému vozíku. Mechanický vozík je charakteristický většími zadními koly a menšími předními kolečky. Jeho výhodou je nízká hmotnost ve srovnání s elektrickými vozíky. Většina mechanických vozíků může být jednoduše složena, což je využíváno například při přepravě autem.

Obrázek 2.1 Ukázka mechanického vozíku[1])

Elektrický vozík. Elektrický vozík (obr. 2.2) je poháněn elektrickým motorem. Vozíčkáři nemusí být fyzicky zdatní, aby se mohli na elektrickém vozíku pohybovat. Vozík má menší a širší kola než mechanický vozík. Nejčastěji je ovládán jednou rukou pomocí joysticku (existují i hlasem ovládané vozíky nebo vozíky ovladatelné bradou a ústy). Vozíky jsou těžké a lze je přepravovat pouze ve speciálně uzpůsobených automobilech.

[1]) `http://safamedical.com/images/wheelchair/series/2.jpg`

Obrázek 2.2 Ukázka elektrického vozíku[1])

2.3 Omezení při pohybu ve městě

Vozíčkáři se mohou při pohybu po městě setkat s několika druhy omezení. Na jedné straně mohou tato omezení zcela zabránit pokračování v cestě, na straně druhé mohou výrazně ovlivnit komfort při jízdě na vozíku. Některá omezení vznikají už ze samotné konstrukce vozíku. Průjezd problémovými místy ovlivňuje šířka vozíku i velikost kol. V této části bude představen přehled omezení, na která mohou vozíčkáři narazit při pohybu po městě. Součástí přehledu budou i nároky na bezbariérovost veřejných komunikací, které by měly zamezit vzniku problémových míst. Standardy pro zajištění bezbariérovosti upravuje vyhláška Ministerstva pro místní rozvoj ČR č. 398/2009 Sb., o obecných technických požadavcích zabezpečujících bezbariérové užívání staveb [4]. Podrobněji zpracované požadavky na bezbariérové užívání staveb lze nalézt v knize *Projektujeme bez bariér* [5], která si klade za cíl sloužit jako příručka pro projektanty veřejných staveb.

Schody. Schody jsou pro vozíčkáře jedním z nejvíce kritických míst. Požadavky na bezbariérovost uvádějí jako maximální výškový rozdíl mezi dvěma různými povrchy 20 mm. Za schod proto budeme označovat všechna místa, která tento limit přesahují. Schodem můžeme označit i obrubník u přechodu pro chodce (obr. 2.3), pokud nesplňuje stanovený limit. Vyskytuje-li se na komunikaci větší výškový rozdíl, musí být zajištěn způsob pro snadné překonání tohoto rozdílu například pomocí nájezdu. Nájezdy by měly mít sklon maximálně 12,5 %, v ideálním případě by neměl sklon přesáhnout 8,33 %.

Při sjíždění ze schodu hrozí nebezpečí vyklopení z vozíku. V opačném případě, kdy vozíčkář najíždí do schodu, je tento způsob fyzicky náročný. Pro uživatele elektrického vozíku může být tento směr vzhledem ke konstrukci kol nepřekonatelný. V obou případech však záleží na velikosti schodu. V případě většího počtu schodů můžeme cestu označit za neprůjezdnou (obr. 2.4).

[1]) `http://safamedical.com/images/wheelchair/electricwheel/16.jpg`

Obrázek 2.3 Ukázka bariérového přechodu pro chodce[1])

Obrázek 2.4 Ukázka bariérových schodů[2])

Povrch. Na bezpečnou jízdu má vliv také povrch komunikace. Vozíčkáři využívají převážně chodníky. Povrch chodníku musí být rovný, pevný a upravený proti skluzu. Výškový rozdíl povrchu stejného typu musí být maximálně 5 mm. Při přechodu na jiný typ povrchu může být rozdíl maximálně 20 mm. Při jízdě na nezpevněném povrchu nebo rozbitém chodníku (obr. 2.5) hrozí zapadnutí kola. Problematická je i jízda po kamenné dlažbě. Ačkoliv tento typ problému nemusí být nepřekonatelnou překážkou, výrazně ovlivňuje komfort jízdy.

[1]) `www.chodcisobe.cz/praha/podnety/1263/barierovy-chodnik-u-vchodu-k-detskemu-lekari`
[2]) `www.chodcisobe.cz/praha/podnety/1447/barierove-schody`

Obrázek 2.5 Ukázka rozbitého chodníku[1])

Sklon. Výrazným způsobem ovlivňuje jízdu také sklon chodníku. Sklon je při jízdě směrem do kopce pro uživatele mechanického vozíku velice fyzicky náročný. Při jízdě z kopce naopak musí vozík vlastní silou brzdit. Uživatelé elektrického vozíku nejsou v tomto ohledu omezeni do takové míry jako uživatelé mechanických vozíků. V případě obou skupin vozíčkářů ale hrozí převrácení při jízdě na příliš prudkém kopci. Na problém sklonu se budeme dívat především z pohledu stoupání. Chodníky mohou mít podélný sklon maximálně 8,33 %, příčný sklon by neměl přesáhnout 2 %. Chodník se sklonem mezi 8,33 % a 12 % je určen pro zdatnější vozíčkáře, avšak chodník se sklonem přesahujícím 20 % lze pro vozíčkáře na mechanickém vozíku bez doprovodu označit za neprůjezdný [6]. Hodnota, při které se stane sklon bariérou, je pro každého vozíčkáře velice individuální. V případě stoupání záleží také na délce chodníku. Dlouhé stoupání je pro vozíčkáře vyčerpávající.

Šířka komunikace. Průjezdnost chodníku udává také jeho šířka. Bezbariérový chodník musí mít šířku alespoň 1 500 mm. Dostatečná šířka pro průjezd jednoho vozíku je 900 mm. Chodník s průjezdovou šířkou menší než 900 mm můžeme označit za neprůjezdný. Dalším faktorem průjezdnosti mohou být i překážky na ulici, které snižují celkovou šířku průjezdné části chodníku. Například špatně zaparkované automobily (obr. 2.6), které částečně zasahují do chodníku, nebo přímo na chodníku parkující automobily.

Průjezdnost ovlivňuje také výskyt stavebních prací, které zasahují do chodníku – částečně rozkopaný chodník nebo přítomnost lešení. Na jedné straně může lešení zmenšit šířku chodníku, na chodníku pak musí zůstat dostatečný prostor pro objetí. Na druhé straně může lešení zabrat celou šířku chodníku, v tomto případě musí být zajištěna dostatečná šířka podchodu pod lešením.

[1]) `www.chodcisobe.cz/praha/podnety/1508/dlouhodobe-spatny-stav-chodniku-v-ulici-na-zborenci`

Obrázek 2.6 Ukázka špatně zaparkovaných automobilů[1])

Dalším příkladem překážky, při které hraje vliv šířka chodníku, je například výskyt sloupků zamezujících průjezdu automobilů do parku nebo na pěší zónu. Jiným příkladem může být také zarůstání vegetace do prostoru chodníku.

2.4 Rozdělení omezení

Martin Nuc ve své diplomové práci [7] rozdělil překážky pro vozíčkáře do čtyř kategorií:

- nesjízdná místa – nesjízdné chodníky, schody, prudký sklon;
- technické závady – nefunkční plošiny, výtahy;
- stavební práce – rozkopaný chodník, zábor, lešení;
- překážky – obecná kategorie, například špatně parkující automobily.

Dále se zabýval problematikou překážek omezených časem. Překážky dále rozdělil podle délky trvání:

- řád hodin – špatně parkující automobily;
- řád dnů – nefunkční plošiny;
- řád týdnů/měsíců – opravy chodníků, lešení;
- trvalé.

Nedostatkem tohoto přístupu je, že nelze přesně odhadnout délku trvání překážek. Uživatelé by museli tipovat, jak dlouho budou tyto překážky představovat problém. V aplikaci by poté mohly být zobrazovány překážky, které už překážkami nejsou nebo naopak by se přestaly zobrazovat překážky, kterým už vypršel stanovený čas. Z obecného hlediska můžeme všechny překážky označit za dočasné. Špatný povrch chodníku může být opraven, schody mohou být doplněny bezbariérovými prvky. V naší aplikaci se zatím zaměříme pouze na základní rozdělení překážek.

Z kategorií dělení překážek využijeme označení stavebních prací. Stavební práce mohou omezit průjezd nebo průjezdu zcela zamezit. U tohoto typu překážky můžeme zaručeně říci, že v dohledné době přestane být překážkou. Pokud bude překážka označena jako stavební práce, můžeme se dotázat uživatele, který daným místem bude projíždět, zda toho omezení stále trvá.

Pro potřeby naší aplikace rozdělíme omezení do dvou kategorií. První kategorií budou částečná omezení. Do této kategorie budeme řadit taková omezení, která nezabraňují

[1]) `www.chodcisobe.cz/praha/podnety/495/parkovani-na-chodniku`

průjezdu vozíku, ale přesto snižují komfort jízdy. Druhou kategorií budou neprůjezdná místa. Do této kategorie budeme řadit místa, která zcela zamezují průjezdu vozíku. Přesné rozdělení do kategorií popisuje tab. 2.1. Stavební práce mohou ovlivnit povrch i šířku chodníku, přesto je označíme jako samostatný typ omezení.

Typ omezení	Částečné omezení	Neprůjezdné
Schody	1 schod (velikost schodu)	více schodů
Povrch	stále průjezdný	nesjízdný povrch
Sklon	od 8,33 % do 20 % (délka stoupání)	větší než 20 %
Šířka chodníku	minimálně 900 mm	méně než 900 mm
Stavební práce	stále průjezdné	neprůjezdné

Tabulka 2.1 Rozdělení omezení

2.5 Navigační instrukce

Navigační instrukce pomáhají vozíčkářům s orientací při jízdě ve městě. Takové instrukce by měly být dostatečně srozumitelné, aby vozíčkáři přesně věděli, jaký krok musí udělat. Zároveň by instrukce neměly být příliš dlouhé, aby vozíčkáři neměli problém s jejich zapamatováním. Z tohoto pohledu můžeme na navigační instrukce klást určité nároky.

Jakub Bokšanský se ve své diplomové práci [8] zabývá generováním navigačních instrukcí pro nevidomé. Stanovil několik požadavků, které by navigační instrukce měla splňovat. Tyto požadavky lze aplikovat i pro navigační instrukce pro vozíčkáře:

- přiměřená délka – obsahuje všechny důležité informace, ale zároveň si je lze snadno zapamatovat;
- přesné instrukce – dostatečně přesný popis kroku, který minimalizuje možnost udělání chyby;
- popis průchozích bodů – slouží k ujištění uživatele, že jede správnou cestou;
- informace o možném nebezpečí – popis problémových míst, na které lze při jízdě narazit;
- identifikace dokončení kroku – zpětná vazba vozíčkáři, že úspěšně dokončil krok.

2.6 Hlasové uživatelské rozhraní

Aplikace WheelTrip si klade za cíl poskytnout multimodální rozhraní k interakci s aplikací. To znamená, že kromě interakce prostřednictvím standardního grafického uživatelského rozhraní umožňuje interakci také pomocí jiného typu rozhraní. V tomto případě se jedná o hlasové rozhraní. S ohledem na to, že vozíčkáři při pohybu musí sledovat dění před sebou a zároveň pomocí rukou ovládat vozík, je využití hlasového rozhraní vhodným řešením. Aplikace se tak stává více hands-free a zároveň eyes-free.

Hlasové uživatelské rozhraní (VUI – Voice User Interface) je takové rozhraní, které umožňuje uživateli ovládání pomocí hlasu. Systémy s tímto rozhraním jsou založené na rozpoznávání lidského hlasu – vstupu, dokážou reagovat na jednotlivé hlasové instrukce a jako zpětnou vazbu uživateli poskytují hlasový výstup.

Rozhraní VUI je složeno z těchto částí [9]:

- povely – instrukce, které jsou při interakci rozpoznávány systémem nebo instrukce přehrávané uživateli;
- gramatika – množina instrukcí, které je systém schopen rozpoznat;
- logika dialogu – akce, které může systém vykonat na základě vstupu od uživatele.

Správně navržené hlasové uživatelské rozhraní minimalizuje možnost vzniku chyby při rozpoznávání hlasu. Na případné chyby dokáže systém dostatečně reagovat tak, aby uživatel mohl pokračovat v práci s rozhraním. V neposlední řadě lze takové rozhraní intuitivně ovládat pomocí vhodně zvolených hlasových instrukcí. Při návrhu rozhraní VUI je důležité definovat, co může uživatel říci a jak bude reagovat systém. Z pohledu uživatele je nutné zaměřit se na to, aby uživatel věděl, jaké instrukce může v danou chvíli použít. Z pohledu systému je nezbytné zajistit, že systém dokáže jednoduše rozpoznávat vstup od uživatele.

Zpravidla je interakce s rozhraním VUI rozdělena do čtyř fází:

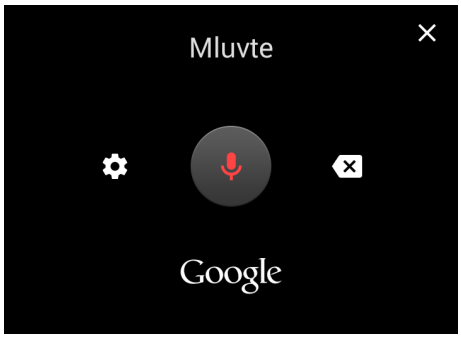
1. Zahájení – uživatel zahájí rozpoznávání řeči (stisk tlačítka, gesto).
2. Čekání na vstup – systém informuje uživatele, aby zadal hlasovou instrukci, a čeká na dokončení vstupu.
3. Zpracování vstupu – systém zahájí rozpoznávání hlasové instrukce.
4. Provedení akce – systém provede akci, která je spojena s danou instrukcí.

Další informace o návrhu hlasových uživatelských rozhraní lze nalézt v [10].

2.6.1 Android API pro rozhraní VUI

Operační systém Android disponuje dvěma API, která lze využít při implementaci rozhraní VUI. Prvním rozhraním je rozpoznávání hlasu (speech recognition). Druhým rozhraním je převod textu na řeč (text-to-speech). Obě aplikační rozhraní jsou popsána v následujících odstavcích.

Speech Recognition API. Rozhraní pro rozpoznávání hlasu je určené ke zpracování hlasového vstupu a zajišťuje převod takového vstupu do textové reprezentace. Nevýhodou je, že pro správnou funkčnost rozhraní je vyžadováno připojení k internetu, protože pro rozpoznání hlasového vstupu je využíván vzdálený server. Pro vkládání hlasového vstupu poskytuje Android standardní rozhraní ve formě dialogového okna (obr. 2.7). Výstup zpracovaného hlasového vstupu je reprezentován seznamem potenciálních převodů na text. Každá položka seznamu obsahuje speciální hodnotu, která udává, s jakou jistotou daný převod odpovídá původnímu vstupu.



Obrázek 2.7 Standardní dialogové okno pro rozpoznávání řeči

Text-to-Speech API. Rozhraní TTS převádí text na hlasový výstup. Ke své práci nevyžaduje připojení k internetu. Podporuje omezenou množinu jazyků, k nimž patří i čeština. Většina jazyků disponuje pouze jedním hlasem, který je využíván ke čtení textu. V současné době je systémový hlas pro češtinu reprezentován mužským hlasem. Nevýhodou tohoto hlasu je, že zní až příliš syntetizovaně, a neodpovídá tak přirozenému hlasu. Porozumění takovému hlasu se tak stává problematické. Alternativou může být využití aplikace Svox, která umožňuje dokoupit jazykový balíček pro češtinu[1]). Balíček češtiny využívá ženský hlas. Hlas je oproti tomu původnímu přirozenější a je mu lépe rozumět.

2.7 Možnosti ovládání telefonu

Manipulace s mobilním telefonem je při jízdě na vozíku velmi složitá. První překážkou je samotné umístění telefonu. Telefon musí být umístěn na vozíku tak, aby ho uživatel mohl bez větších problémů ovládat. Zároveň by umístění telefonu nemělo omezovat vozíčkáře při pohybu, to se týká především uživatelů mechanického vozíku, kteří musí mít zajištěn dostatečný prostor pro manipulaci s koly vozíku. Pro připevnění telefonu k vozíku slouží držáky na mobilní zařízení (obr. 2.8).

Obrázek 2.8 Ukázka držáku mobilního zařízení [2])

Druhou překážkou je manipulace s dotykovým displejem. Na jedné straně existují uživatelé, kteří mají omezenou motoriku do takové míry, že nezvládají používání standardních dotykových gest k interakci s telefonem. Na straně druhé existují uživatelé mechanického vozíku, kteří kvůli točení s koly nosí rukavice a pro něž by bylo velmi nepraktické kvůli telefonu neustále rukavice sundávat. Tyto problémy lze řešit využitím joysticku nebo jiného ovládacího zařízení, které dokáže minimalizovat potřebná gesta a snadno manipulovat s ovládacími prvky v aplikaci.

V následující části budou představeny dva způsoby bezdotykového ovládání mobilního telefonu. Budou popsána potřebná příslušenství i způsob propojení ovladačů s telefonem.

[1]) `https://play.google.com/store/apps/details?id=com.svox.classic.langpack.ces_cze_fem`
[2]) `http://wheelchairs.com/photos/accessories/cell_mount.jpg`

2.7.1 Technologie OTG

Prvním způsobem bezdotykového ovládání je využití technologie OTG (On-The-Go) [11]. Technologie OTG je rozšířením specifikace USB 2.0, která umožňuje propojení mobilních a periferních zařízení. Tato zařízení poté mohou společně komunikovat přímo bez nutnosti připojení k PC. V kontextu mobilních aplikací se nejčastěji používá k připojení externích ovladačů – myš, klávesnice, herní ovladače – k mobilnímu telefonu či tabletu. Pomocí technologie OTG tak může být aplikace ovládána přímo těmito externími ovladači.

Pro využití technologie OTG je nutné, aby mobilní zařízení tuto technologii podporovalo. Poté stačí externí ovladač připojit k mobilnímu zařízení přes kabel OTG. Další nastavení není potřeba. Výhodou tohoto způsobu ovládání je nízká pořizovací cena a také jednoduchost zapojení. Nevýhodou je, že stále velké množství zařízení neobsahuje potřebný hardware ani ovladače k zajištění podpory této technologie.

2.7.2 Tecla

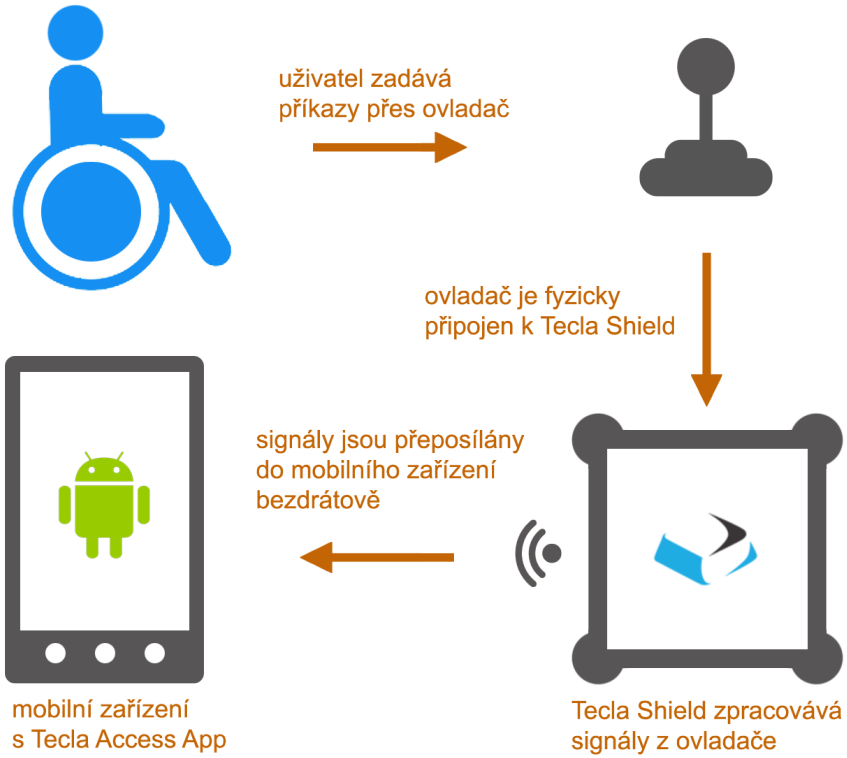
Tecla je systém umožňující ovládání telefonu, tabletu i počítače pomocí ovladače elektrického vozíku či jiných externích ovladačů. Může se jednat nejen o joystick, ale i o další typy ovladačů, které jsou například ovládány jazykem, dechem nebo mrkáním.

Systém Tecla je tvořen zařízením Tecla Shield a aplikací Tecla Access App. Tecla Access App v sobě dále obsahuje klávesnici Tecla.

Tecla Shield. Tecla Shield (obr. 2.9) je bezdrátové hardwarové zařízení, které umožňuje připojení externího ovladače. Tecla Shield zpracovává vstupní instrukce z ovladače a posílá je dále do mobilního zařízení. Komunikace s mobilním zařízením probíhá bezdrátově a je realizována pomocí standardu Bluetooth.

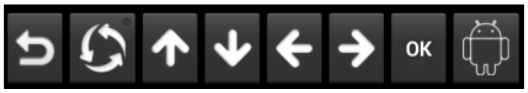
Obrázek 2.9 Tecla Shield

Tecla Access App. Tecla Access App je aplikace běžící na straně mobilního zařízení. Úzce spolupracuje s operačním systémem, a zprostředkovává tak většinu jeho funkcí.



Obrázek 2.10 Diagram zapojení systému Tecla

Klávesnice Tecla. Pro zpřístupnění klávesnice Tecla je nutné ji po instalaci aplikace Tecla Access App zvolit jako hlavní vstupní metodu v nastavení mobilního zařízení. Po nastavení klávesnice se v dolní části zařízení zobrazí navigační klávesnice (obr. 2.11). Zcela vlevo je umístěno tlačítko zpět, následuje tlačítko pro opakování zvolené instrukce. Dále se zde nacházejí směrová tlačítka, tlačítko pro potvrzování a zcela vpravo tlačítko pro změnu kontextu klávesnice. V novém kontextu lze zobrazit alfanumerickou klávesnici, měnit hlasitost zařízení, zadávat hlasové instrukce nebo vyhledávat.



Obrázek 2.11 Navigační klávesnice Tecla

Mobilní zařízení lze ovládat ve dvou režimech. V prvním režimu jsou jednotlivé instrukce prováděny přímo, to znamená, že například pohybem joysticku dolů se vybere další položka v seznamu. Druhý režim je full-screen, ten redukuje interakci s telefonem pouze na jeden ovládací prvek, a výrazně tím zvyšuje přístupnost pro uživatele s různými typy postižení. V tomto režimu se postupně podsvěcují jednotlivé instrukce na navigační klávesnici (obr. 2.12). Celá klávesnice se takto cyklicky skenuje a uživatel vybírá aktuálně podsvícenou instrukci kliknutím na potvrzovací tlačítko, což v případě různých typů ovladačů může být centrální tlačítko, pohnutí jazykem, vydechnutí nebo mrknutí.

Obrázek 2.12 Skenování navigační klávesnice

V případě alfanumerické klávesnice (obr. 2.13) probíhá skenování řádků a sloupců. Uživatel nejprve vybere řádek, na kterém se nachází požadovaný znak, a až poté samotný znak. Rychlost skenování klávesnice lze upravit v nastavení aplikace Tecla Access App. Další informace o způsobu připojení systému Tecla a popis rozšířeného nastavení lze nalézt v uživatelské příručce [12].

Obrázek 2.13 Alfanumerická klávesnice Tecla – skenování řádků a sloupců

2.8 Naviterier

Jednou z důležitých součástí aplikace je schopnost nalézt takové trasy, které pro vozíčkáře nebudou představovat problém. To znamená trasy bez zbytečných překážek, jako jsou schody či vysoké obrubníky. Pro potřeby nalezení tras a získání navigačních instrukcí bude aplikace využívat systém Naviterier [13].

Projekt Naviterier se zabývá vývojem navigačního systému pro navigaci chodců se sníženou schopností orientace a pohybu, jako jsou nevidomí, vozíčkáři nebo senioři. Naviterier se vyznačuje tím, že využívá efektivní způsob navigace chodců pomocí speciálně vybraných bodů v terénu a vlastností prostředí. V současné podobě je systém orientován především na nevidomé uživatele a běží ve zkušebním režimu. Některé funkčnosti, jako je správa databáze překážek ve městě, zatím nejsou podporovány. S ohledem na možná omezení bude v aplikaci pro nepodporované funkčnosti připraveno pouze uživatelské rozhraní pro budoucí integraci a nepodporované funkčnosti budou simulovány.

Dalším omezením je rozsah oblasti pro plánování tras. Naviterier zatím pokrývá oblast o velikosti 1 km^2 v centru Prahy ohraničené ulicemi: Rašínovo nábřeží, Na Moráni, Karlovo náměstí, Ječná, náměstí I. P. Pavlova, Sokolská, Mezibranská, Washingtonova, Opletalova, Václavské náměstí, 28. října, Národní, Spálená a Myslíkova.

Mezi podporované funkčnosti patří:

- nalezení trasy – lze vyhledat trasy mezi dvěma zadanými body specifikovanými třemi způsoby:
  - název adresy;
  - ID adresy;
  - ID bodu zájmu;
- nalezení zdrojových dat pro danou oblast – vrací zdrojová data (adresy, body zájmu, segmenty tras) pro specifikovanou oblast. Oblast je specifikována souřadnicemi a maximální vzdáleností od dané souřadnice;
- nalezení všech adres – vrací všechny podporované adresy;
- nalezení všech bodů zájmu – vrací všechny podporované body zájmu.

2.9 Existující aplikace

V oblasti mobilních aplikací existuje několik aplikací zaměřených na vozíčkáře. V následující části budou představeny dvě mobilní aplikace pro vozíčkáře.

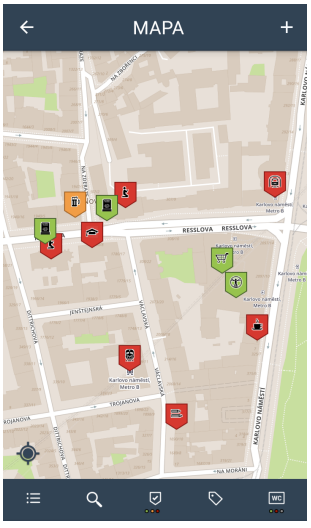
Wheelmap. Projekt Wheelmap[1]) vznikl z iniciativy německé neziskové organizace SOZIALHELDEN e.V.[2]) Projekt je založen na principu shromažďování dat od komunity uživatelů. Wheelmap je přístupný prostřednictvím webové a mobilní aplikace.

Wheelmap využívá informace o místech zájmu získaných z dat OpenStreetMap[3]). Tato data jsou doplněna o informaci týkající se typu přístupnosti.

Na mapě jsou místa zájmu rozlišena barvou podle typu přístupnosti. Zelenou barvou jsou označena přístupná místa, oranžovou barvou jsou označena částečně přístupná místa a červenou barvou jsou označena nepřístupná místa. Místa, u nichž informace o přístupnosti není dostupná, jsou označena šedou barvou. V aplikaci lze zvolit, která místa jsou zobrazována. Zobrazením pouze míst, u nichž existuje informace o přístupnosti, lze zvýšit přehlednost mapy, protože většina míst nenese žádnou informaci o přístupnosti.

U každého místa lze jednoduše označit typ přístupnosti a dostupnost toalet. Místu lze přiřadit adresu, komentář a také fotografii. Nevýhodou však je, že nové informace může k místům přiřadit kdokoliv, jejich správnost není kontrolována, a tak může dojít k nepřesnému označení z pohledu přístupnosti.

Aplikace nedisponuje vlastním vyhledávačem tras a navigací. Pro vyhledávání tras a navigaci využívá aplikace třetích stran, v tomto případě aplikaci Waze[4]) nebo Google Maps[5]). Hlavním nedostatkem je, že při vyhledávání tras není brán ohled na omezení vozíčkářů. Žádná z těchto aplikací prozatím neposkytuje nalezení trasy vhodné pro vozíčkáře. Lze využít pouze trasy pro chodce, což je nejbližší možné upřesnění. Avšak na trase se mohou objevit například schody či jiné překážky, které nejsou problémem pro chodce, ale pro vozíčkáře mohou být nepřekonatelné.



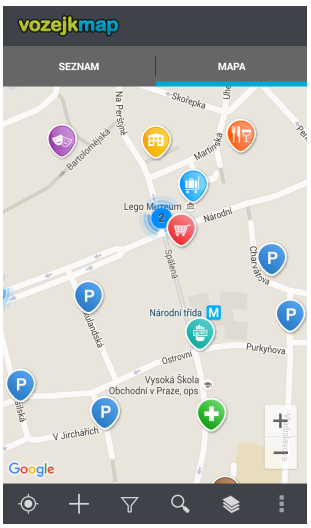
Obrázek 2.14 Ukázka aplikace Wheelmap

[1]) http://wheelmap.org/
[2]) http://sozialhelden.de/
[3]) https://www.openstreetmap.org
[4]) https://www.waze.com
[5]) https://maps.google.com/

VozejkMap. Druhým příkladem mobilní aplikace pro vozíčkáře je VozejkMap[1]). Tuto aplikaci provozuje Česká asociace paraplegiků. Aplikace nabízí přehled přístupných míst. Místa jsou rozdělena do několika kategorií, kterými jsou například kultura, instituce, ubytování nebo doprava. Informace o přístupnosti míst jsou shromažďovány od uživatelů aplikace. Velkou výhodou je, že informace jsou ověřovány administrátorem aplikace – vozíčkářem, v aplikaci se tak objevují pouze přístupná místa.

Místa nesou informace o typu přístupnosti, zda je v místě přítomna rampa či nájezd nebo zda se v místě nenachází žádné schody. Další informací je možnost bezbariérového parkování v blízkosti místa a také přítomnost bezbariérové toalety. Všechna tato data jsou poskytována zdarma pod licencí Creative Commons CC BY-NC-SA 4.0[2]). Podrobný popis struktury dat je předmětem podkapitoly 2.10.

Stejně jako Wheelmap využívá aplikace VozejkMap pro nalezení trasy a navigaci aplikaci třetí strany. V tomto případě pouze Google Maps. Stejně jako Wheelmap disponuje VozejkMap i webovým rozhraním.



Obrázek 2.15 Ukázka aplikace VozejkMap

Hodnocení existujících aplikací. Obě představené aplikace slouží především jako přehled bezbariérových míst. Uživatelé těchto aplikací se mohou podílet na jejich obsahu. V případě Wheelmap mohou označovat bezbariérovost míst. V případě VozejkMap mohou uživatelé přidávat nová bezbariérová místa. Vyhledávání tras je umožněno pouze prostřednictvím externích aplikací, které nepodporují vyhledávání tras pro vozíčkáře. Stejným zaměřením disponuje většina aplikací určených pro vozíčkáře, například On Wheels[3]) nebo WheelMate[4]). Žádná z těchto aplikací neřeší problematiku nalezení vhodné trasy pro vozíčkáře. Velkou nevýhodou těchto aplikací je také fakt, že kromě dotykového ovládání neumožňují alternativní způsob interakce.

[1]) `http://www.vozejkmap.cz/`
[2]) `https://creativecommons.org/licenses/by-nc-sa/4.0/`
[3]) `https://play.google.com/store/apps/details?id=com.onwheelsapp.onwheels&hl=cs`
[4]) `https://play.google.com/store/apps/details?id=com.novasa.wheelmate&hl=cs`

2.10 Data VozejkMap

VozejkMap disponuje rozsáhlou databází bezbariérových míst. Obsahuje přes 5 000 míst. U těchto míst je bezbariérovost ověřena samotnými vozíčkáři. Databáze těchto míst je volně přístupná pod licencí Creative Commons CC BY-NC-SA 4.0. Databáze je distribuována v podobě jednoho souboru ve formátu JSON. Tento soubor je přístupný na adrese www.vozejkmap.cz/opendata/locations.json a je aktualizován jednou denně. Pro přístup k datům je nutné požádat správce o zaslání unikátního přístupového klíče. Struktura dat obsažených v souboru locations.json je popsána v tabulkách 2.2, 2.3 a 2.4.

atribut	typ hodnoty	popis
TITLE	string	Název místa
LOCATION_TYPE	int	ID typu místa
DESCRIPTION	string	Popis místa
STREET	string	Ulice
STREET_NUMBER	string	Číslo popisné
CITY	string	Město
ZIP	string	PSČ
PHONE	string	Telefon
MAIL	string	E-mail
LINK	string	Web
OPENING_HOURS	string	Otevírací hodiny
LAT	float	Zeměpisná šířka místa
LNG	float	Zeměpisná délka místa
ATTR1	int	ID typu bezbariérového přístupu
ATTR2	string	Bezbariérové WC
ATTR3	string	Parkoviště pro vozíčkáře
AUTHOR_NAME	string	Autor

Tabulka 2.2 Struktura dat VozejkMap

ID	Typ
1	Kultura
2	Sport
3	Instituce
4	Jídlo a pití
5	Ubytování
6	Lékaři, lékárny
7	Jiné
8	Doprava
9	Veřejné WC
10	Benzínka
11	Obchod
12	Banka, bankomat
13	Parkoviště
14	Prodejní a servisní místa Škoda Auto

Tabulka 2.3 Typy bezbariérových míst

ID	Typ
1	Bez schodů
2	Nájezd či rampa
3	Výtah
4	Schodišťová plošina
5	Zdviž
6	Nájezd či rampa + výtah
7	Nájezd či rampa + schodišťová plošina
8	Nájezd či rampa + zdviž
9	Zdviž + schodišťová plošina
10	Zdviž + výtah
11	Výtah + schodišťová plošina

Tabulka 2.4 ID typu bezbariérového přístupu

2.11 Ovládání elektrického vozíku pomocí chytrého telefonu

Dne 21. 4. 2016 proběhla videokonference se zástupcem občanského sdružení Život bez bariér[1]) Josefem Fučíkem, který se zabývá konstrukcí elektrických vozíků. Cílem konference bylo získat další funkční požadavky aplikace a také zjistit možnosti integrace vozíku s aplikací. Tento rozhovor pomohl určit směr, jakým by se měl budoucí vývoj aplikace ubírat. Během rozhovoru bylo definovánno několik požadavků, z nichž byly vybrány tři požadavky, které budou dále zapracovány do aplikace. Těmito požadavky jsou:

- Nastavení základních parametrů vozíku – uživatel bude mít možnost nastavit maximální rychlost vozíku a zrychlení. Vozíky lze půjčovat krátkodobě na 1 den až 3 měsíce nebo dlouhodobě na 1 až 2 roky. Toto nastavení by tak mělo být intuitivní, aby i v případě krátkodobé výpůjčky mohl být vozík snadno nastaven.
- Odesílání polohy – pro nouzové případy bude moci uživatel odeslat svoji polohu, aby mu mohla být zajištěna případná pomoc.
- Zobrazení stavu baterie – uživatel bude informován o stavu baterie vozíku, aby mohl baterii dobít v případě, kdy je stav nabití baterie mezi 60 % až 70 %.

Další požadavky:

- Využití systému kamer – přední kamera s mikrofonem (pro případnou vzdálenou asistenci), kamera na couvání (zobrazení obrazu na displej telefonu) a kamera snímající povrch komunikace (zjištění rychlosti vozíku podle změny obrazu).
- Nastavení optimálního příčného a podélného náklonu vozíku, polohy sedadla.
- Nastavení intenzity světel vozíku.
- Ovládání více vozíků najednou – ovládá se první vozík, další vozíky jsou v řadě za sebou a každý sleduje vozík před sebou. Vozík lze ovládat vzdáleně.
- Telefon slouží k ověření identity vozíčkáře. Bez ověření nelze vozík ovládat.

Technické provedení integrace mobilního zařízení s vozíkem zatím nebylo definováno.

[1]) `http://www.zbb.cz/`

2.12 Požadavky aplikace

V této části budou popsány funkční (FRQ) a nefunkční (NRQ) požadavky aplikace. Některé z těchto požadavků jsou omezeny v závislosti na systému Naviterier. Pro požadavky, které by měly využívat tento systém, ale zatím nemají dostatečnou podporu od systému Naviterier, bude v aplikaci vytvořeno uživatelské rozhraní. Funkčnost bude u těchto požadavků simulována.

2.12.1 Funkční požadavky

- FRQ 01: Aplikace bude umožňovat vyhledávání míst.
  - Místa bude možné vyhledat pomocí textového nebo hlasového vstupu.
  - Aplikace bude umožňovat zvolit hledané místo označením na mapě.
  - Data VozejkMap budou sloužit jako primární zdroj pro vyhledávání míst.
  - Sekundární zdroj dat bude zajištěn využitím dat získaných prostřednictvím Google Places API.
  - Mezi těmito dvěma zdroji dat se bude možné přepnout v nastavení aplikace.
- FRQ 02: Aplikace bude zobrazovat detailní informace o vyhledaném místu.
  - Detail místa bude dostupný po vyhledání místa, po kliknutí na ikonu místa na mapě a po zvolení místa na mapě.
  - V případě vyhledání místa z dat VozejkMap bude na detailu místa zobrazen název místa, typ místa, druh bezbariérového přístupu, dostupnost bezbariérové toalety a bezbariérového parkování.
  - V případě místa vyhledaného prostřednictvím Google Places API bude zobrazen pouze název a adresa místa.
  - V případě vyhledání místa na mapě bude zobrazena pouze adresa místa.
  - Ve všech třech případech vyhledávání místa bude možné prohlédnout místo pomocí aplikace Google StreetView nebo spustit vyhledávání trasy k danému místu.
- FRQ 03: Aplikace bude umožňovat vyhledání trasy.
  - Trasy budou vyhledávány pouze na základě zadaného počátečního a koncového bodu trasy.
  - Pokud nedojde ke změně, bude jako počáteční bod trasy nastavena aktuální poloha uživatele.
  - Aplikace bude umožňovat vyhledat trasu z obrazovky pro zadání počátečního a koncového bodu trasy.
  - Aplikace bude umožňovat vyhledat trasu k místu z dialogového okna s podrobnými informacemi o místu.
- FRQ 04: Aplikace bude sledovat aktuální polohu uživatele.
  - Při spuštění aplikace bude nalezena aktuální poloha uživatele. Tato poloha bude zakreslena na mapě.
  - Pro nalezení polohy bude nutné zapnout sledování polohy v nastavení telefonu. V případě vypnutého sledování polohy nebude aktuální poloha zakreslena na mapě a uživatel bude upozorněn formou dialogového okna o nutnosti povolení sledování polohy.
- FRQ 05: Aplikace bude zobrazovat informace o bezbariérových místech.

  - K zobrazení informací o bezbariérovosti míst bude využito dat z aplikace VozejkMap.
  - Tato místa budou zakreslena do mapy. Místa budou rozlišena graficky v závislosti na typu místa.
- FRQ 06: Aplikace bude umožňovat filtrovat místa podle jejich typu.
  - Aplikace bude umožňovat vybrat jednotlivé typy míst nebo vybrat/zrušit všechny typy míst, které se budou zobrazovat na mapě.
  - Při vyhledávání míst z dat VozejkMap budou zobrazena k výběru pouze místa s povoleným typem.
- FRQ 07: Aplikace bude zobrazovat informace o překážkách na trase.
  - V případě výskytu překážky na trase bude tato skutečnost zakreslena na mapě formou ikony.
- FRQ 08: Aplikace bude zobrazovat navigační instrukce.
  - V průběhu navigace bude aplikace zobrazovat navigační instrukce formou textové zprávy a ikonou určující přibližný směr jízdy.
  - Aplikace bude umožňovat přepínání mezi jednotlivými instrukcemi.
  - Aplikace bude využívat systém pro převod textu na řeč k předčítání instrukcí.
- FRQ 09: Aplikace bude umožňovat přeplánovat trasu.
  - V případě, že uživatel při jízdě narazí na překážku, bude moci trasu přeplánovat.
  - Pro tuto funkčnost bude vytvořeno pouze uživatelské rozhraní. Samotná logika přeplánování není prostřednictvím systému Naviterier dostupná.
- FRQ 10: Aplikace bude umožňovat označení neprůjezdného místa.
  - V případě, že uživatel při jízdě narazí na překážku, bude moci označit neprůjezdné místo.
- FRQ 11: Aplikace bude informovat uživatele v průběhu navigace o případných problémech či jiných událostech.
  - Aplikace přehraje zvukový tón a text informující o dané události.
  - Události budou typu: „dostavte se na start“, „jste v cíli“, „vyjeli jste mimo trasu“, „nízký stav nabití baterie“, „nebezpečný náklon vozíku“.
- FRQ 12: Aplikace umožní nastavit základní parametry vozíku.
  - Základní parametry vozíku budou maximální rychlost a zrychlení.
- FRQ 13: Aplikace umožní uživateli odeslat svou polohu pro nouzové případy.

2.12.2 Nefunkční požadavky

- NRQ 01: Aplikace bude určena pro mobilní telefony s operačním systémem Android.
  - Nejnižší podporovaná verze bude Android Jelly Bean (API level 16).
- NRQ 02: Aplikaci bude možno ovládat pomocí joysticku.
  - Kromě standardního ovládání dotykem bude aplikace umožňovat ovládání pomocí joysticku. K tomu bude možné využít technologii OTG.

  - Aplikace bude podporovat systém ovládání Tecla. Uživatelské rozhraní aplikace bude připraveno pro jednoduchou integraci s tímto systémem.

- NRQ 03: Aplikace bude využívat mapové podklady Google Maps.
- NRQ 04: Aplikace bude využívat k rozpoznávání hlasu Speech Recognition API od společnosti Google.
- NRQ 05: Aplikace bude využívat k převodu textu na řeč Text-To-Speech API od společnosti Google.
- NRQ 06: Pro vyhledávání tras a získání navigačních instrukcí bude využit systém Naviterier.

Kapitola 3
Návrh

Tato kapitola je věnována návrhu aplikace. Kapitola je rozdělena do dvou hlavních podkapitol.

První podkapitola se zabývá návrhem uživatelského rozhraní. V této podkapitole bude představen základní koncept systému ovládání, který umožní ovládat aplikaci jednoduše pouze pomocí joysticku. Pro aplikaci WheelTrip je tento způsob ovládání klíčový, protože umožní využívat aplikaci širšímu okruhu uživatelů. Ovládání bude založeno na principu menu a rychlých akcí.

V této podkapitole bude také představeno grafické uživatelské rozhraní aplikace. V rámci této části bude představen diagram interakce mezi obrazovkami aplikace, kde bude zakresleno propojení obrazovek aplikace a budou zde popsány akce, které slouží k přechodu mezi obrazovkami.

Návrh rozhraní obrazovek prošel několika fázemi, od prvních skic, prvního lo-fi prototypu [14] až po konečnou verzi návrhu rozhraní implementovanou přímo pro platformu Android. Součástí podkapitoly budou ukázky uživatelského rozhraní aplikace. Bude popsáno složení jednotlivých obrazovek aplikace a také využití navrženého systému ovládání na daných obrazovkách.

Druhá podkapitola bude věnována popisu architektury aplikace. Součástí této podkapitoly bude popis základní balíčkové struktury aplikace. Bude zde představen doménový model aplikace. Závěr podkapitoly bude věnován popisu návrhu nasazení aplikace a komunikace aplikace s dalšími systémy. V této podkapitole budou pro popis logického uspořádání dílčích částí aplikace použity diagramy UML [15].

3.1 Návrh uživatelského rozhraní

Tato podkapitola je rozdělena do dvou částí. V první části bude představen základní koncept a ovládání aplikace. V části druhé budou představena grafická uživatelská rozhraní hlavních obrazovek aplikace.

3.1.1 Základní koncept a ovládání

Vzhledem k tomu, že systém navigace mezi ovládacími prvky, který je určený k zajištění přístupnosti aplikace a který je definován platformou Android, nemusí být vždy efektivní, bude navržen odlišný způsob navigace. Implicitně je navigace mezi ovládacími prvky řešena tak, že se lze mezi jednotlivými prvky posouvat, a spuštění akce pro daný prvek je mapováno na potvrzovací tlačítko (centrální tlačítko na joysticku, klávesa Enter na klávesnici). Aplikace WheelTrip bude umožňovat spouštět akce přímo i směrovými tlačítky.

Základní koncept. Základní koncept ovládání aplikace je postaven na využití systému rychlých akcí a menu. Tento systém bude složen ze dvou skupin ovládací prvků. První skupinu budou tvořit tlačítka pro spuštění rychlých akcí. Po aktivaci těchto tlačítek bude namapovaná akce provedena ihned.

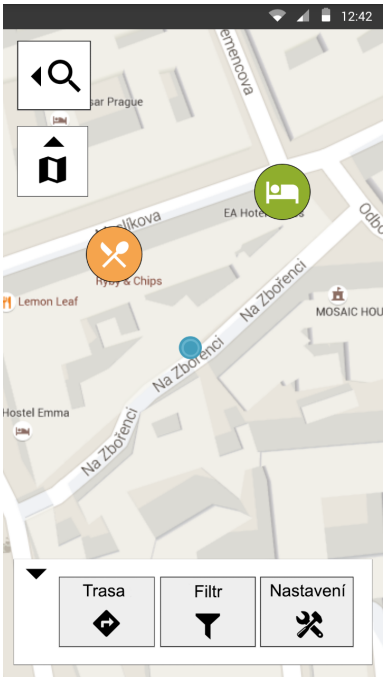
Obrázek 3.1 Vyhledávací tlačítko s ikonou směru

Druhou skupinou budou tlačítka obsažená v menu. V menu budou umístěna tlačítka pro akce, u nichž se předpokládá méně frekventované využití. Pro aktivaci těchto tlačítek bude nutné menu nejdříve otevřít. Menu bude umístěno v dolní části obrazovky.

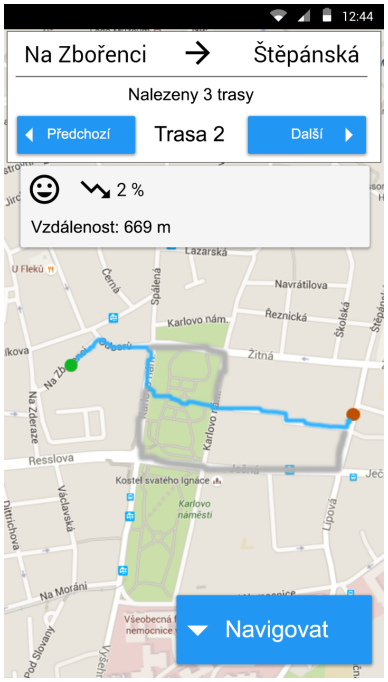
Obrázek 3.2 Menu s ikonou směru

Menu a rychlé akce budou ovládané přímo směrovými tlačítky, proto budou pro rychlé akce vyhrazena tři směrová tlačítka. Čtvrté tlačítko bude vyhrazeno pro přístup k menu. V menu se opět mohou nacházet maximálně tři další položky. Na obrazovkách, které neobsahují menu, bude i čtvrté tlačítko použito pro spuštění rychlé akce.

U tlačítek rychlých akcí bude graficky znázorněný požadovaný směr formou malé šipky, který je nutný ke spuštění akce (obr. 3.1). Stejně bude označeno i menu (obr. 3.2). Na obr. 3.3 je zobrazen lo-fi prototyp hlavní obrazovky aplikace, kde je využit základní koncept ovládání s menu. Příkladem obrazovky prototypu, na které se nevyužívá menu, je obrazovka trasy (obr. 3.4).



Obrázek 3.3 Ukázka použití konceptu ovládání s menu



Obrázek 3.4 Ukázka použití konceptu ovládání bez menu

Ovládání. Na způsob ovládání aplikace je kladen velký důraz. Aplikace bude podporovat standardní dotykové ovládání. Kromě dotykového ovládání bude umožněno celou aplikaci ovládat pomocí externího ovladače. Cílem je navrhnout ovládání aplikace tak, aby bylo snadné aplikaci integrovat se systémem Tecla. Systém Tecla využívá vlastní softwarovou klávesnici, která mezi základními ovládacími prvky obsahuje čtyři směrová tlačítka, tlačítko zpět a potvrzovací tlačítko. Tento rozsah dostupných akcí lze aplikovat na většinu joysticků či jiných externích ovladačů. Pokud bude pro aplikaci vytvořeno rozhraní podporující tyto ovládací prvky, měla by být aplikace dostupná většině vozíčkářů, kteří jsou zvyklí ovládat vozík pomocí ovladače.

Při návrhu ovládacího rozhraní budeme pracovat s abstraktní verzí ovladače, který bude podporovat výše zmíněné ovládací prvky. Požadavky na tvorbu aplikace kompatibilní se systémem Tecla jsou uvedeny v [16]. Tyto požadavky lze shrnout do dvou hlavních změn, které je nutné udělat v aplikaci, aby mohla být ovládána systémem Tecla.

První změnou je grafické odlišení aktuálně vybraného prvku na obrazovce. Prvky se mohou nacházet v několika stavech, kde by měl být každý stav zvýrazněn jiným způsobem. Při kliknutí na tento prvek dochází ke změně jeho vzhledu, uživatel tak má zpětnou vazbu, že byl tento prvek aktivován. V jiném případě může být prvek pouze zvýrazněn při výběru, což je využíváno například při procházení seznamů, kdy je zvýrazněn aktuálně vybraný řádek seznamu. Při využití standardních prvků Android je toto chování podporováno implicitně. Možnosti platfomy Android jsou ale veliké, a tak lze vytvořit vlastní ovládací prvky. Proto je nutné nezapomenout pro aplikaci vytvořit i potřebné grafické zdroje, aby aplikace zůstala přístupná.

Obrázek 3.5 Ukázka stavů tlačítka[1])

Druhou změnou je zajištění reakce na vstupy z ovladače. Signál zaslaný z ovladače by měl být zpracován a aktivovat funkci definovanou pro tento signál.

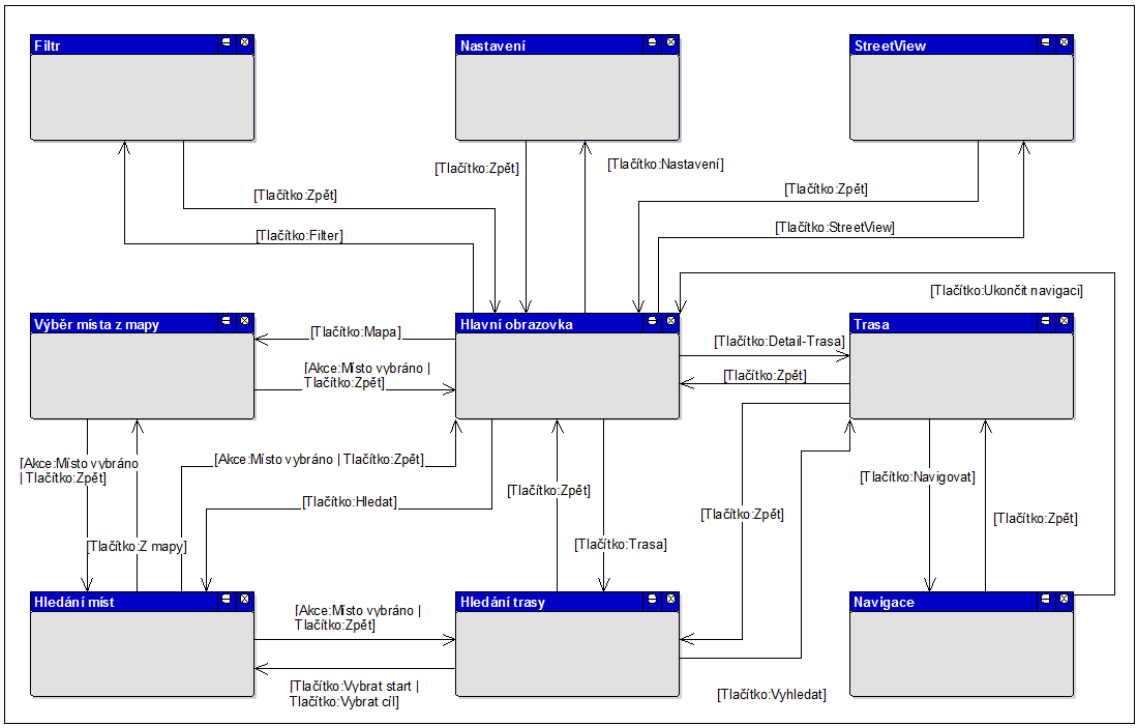
Android definuje další postupy, jak vytvořit přístupnou aplikaci [17]. Jedním z hlavních postupů je vytvoření tzv. focus navigace. To znamená, že jednotlivé prvky na obrazovce jsou provázány a při použití směrových tlačítek se lze mezi těmito prvky navigovat. Tento způsob navigace může být neefektivní pro obrazovky s větším počtem ovládacích prvků, kdy je nutné se nejdříve k požadovanému prvku doklikat. Focus navigace bude využita na jednodušších obrazovkách, jako jsou například obrazovky obsahující seznam. Pro komplexnější obrazovky využijeme způsob ovládání popsaný v části 3.1.1.

3.1.2 Návrh obrazovek

V následujících částech bude popsán návrh grafického uživatelského rozhraní aplikace. Při návrhu rozhraní obrazovek byl brán v potaz požadavek na ovladatelnost joystickem, proto byl do rozhraní obrazovek zahrnut systém menu a rychlých akcí popsaný v části 3.1.1. Pro ilustraci vzhledu rozhraní zde budou použity ukázky obrazovek přímo z aplikace implementované pro platformu Android. Původní návrhy lo-fi prototypu jsou součástí přílohy C.

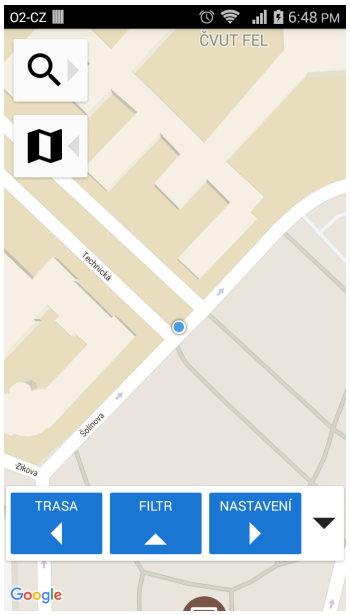
Návrh interakce obrazovek. Diagram interakce obrazovek 3.6 zobrazuje propojení jednotlivých obrazovek aplikace. V diagramu jsou zakresleny akce sloužící k přechodům mezi obrazovkami. Tento diagram je využit při testování, kdy je možné na základě tohoto diagramu vytvořit testovací cesty pro průchod aplikací. Testování průchodu aplikací je popsáno v kapitole 5 v podkapitole 5.1.2.

[1]) `https://www.google.com/design/spec/components/buttons.html#`



Obrázek 3.6 Diagram interakce

Hlavní obrazovka aplikace. Hlavní obrazovka aplikace (obr. 3.7) tvoří hlavní rozcestník funkčností aplikace. Tato obrazovka je zobrazena při spuštění aplikace jako první. Podklad obrazovky je tvořen mapou, která vyplňuje celý prostor obrazovky. Nad touto základní vrstvou jsou umístěné ovládací prvky. Pozice mapy je zarovnána tak, aby byla aktuální pozice uživatele umístěna ve středu mapy. Na mapě jsou také zobrazena nejbližší bezbariérová místa.



Obrázek 3.7 Ukázka hlavní obrazovky aplikace

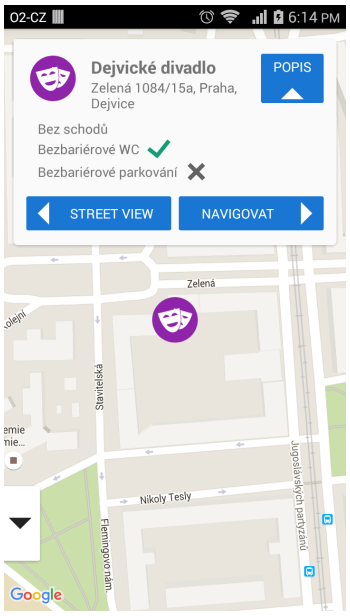
Na této obrazovce lze poprvé zaznamenat použití systému menu a rychlých akcí. V horním pravém rohu se vyskytují rychlé akce „vyhledat“ a „mapa“. Do rychlých akcí lze také zařadit akci pro otevření menu. První akce „vyhledat“ slouží k zobrazení obrazovky pro vyhledávání míst. Druhou rychlou akcí na hlavní obrazovce je režim mapy.

Po vyhledání místa je zobrazen detail bezbariérového místa. Detail obsahuje základní informace o místu, jako je název místa a jeho adresa. Detail rovněž obsahuje informace týkající se bezbariérového přístupu. První z těchto informací je typ bezbariérového přístupu do objektu. Dále následují informace o možnosti bezbariérového parkování a přítomnosti bezbariérových toalet.

Detail obsahuje také tři rychlé akce. První akce, která je umístěna v pravém horním rohu detailu, slouží k zobrazení detailního popisu místa. Tato akce je velmi důležitá, protože popis místa může obsahovat podrobný komentář o tom, jak se lze do objektu dostat. Příklad takového popisu je součástí podkapitoly 4.6.

Rychlá akce umístěná v dolním levém rohu detailu slouží k otevření mapy v režimu *StreetView*. V tomto režimu se lze po ulici pohybovat přímo bez nutnosti speciálního nastavení joysticku. Režim *StreetView* je z pohledu vozíčkáře také velmi důležitý. Vozíčkář se může podívat, jak vypadá okolí místa ve skutečnosti, a může sám identifikovat případné problémy s přístupností.

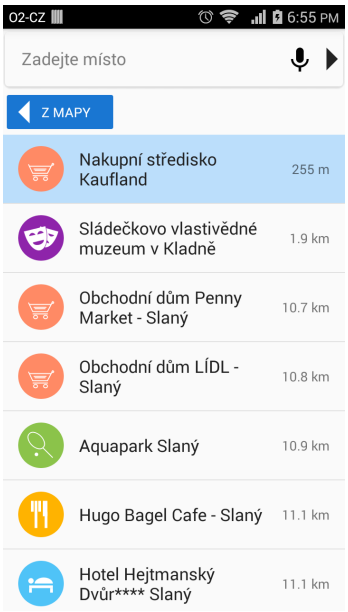
Poslední rychlou akcí je možnost vyhledat trasu přímo bez nutnosti zadávání počátečního a koncového bodu trasy. V takovém případě se jako počáteční bod trasy použije aktuální pozice uživatele. Koncovým bodem trasy bude aktuálně v detailu zobrazené místo.



Obrázek 3.8 Ukázka detailu místa

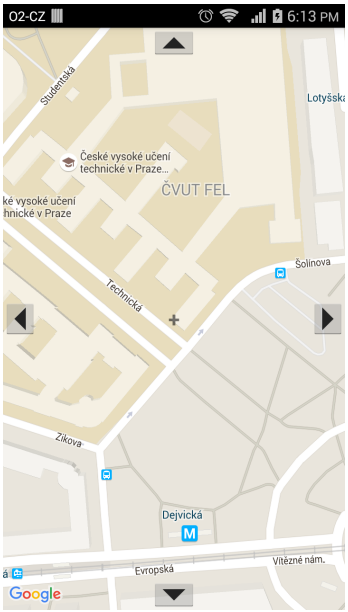
V dolní části hlavní obrazovky se nachází menu. Použitím rychlé akce pro otevření nebo zavření menu lze určit kontext, v němž se budou vyhodnocovat použité směrové šipky. V případě otevřeného menu nelze například spustit vyhledávání míst nebo režim mapy. Menu obsahuje akce pro zadání nové trasy, filtr pro volbu zobrazovaných míst na mapě a v neposlední řadě možnost pro úpravu nastavení aplikace.

Vyhledávací obrazovka. Na vyhledávací obrazovce (obr. 3.9) je v horní části umístěno textové pole a tlačítko pro hlasový vstup. Po otevření vyhledávací obrazovky je pod textovým polem zobrazen seznam nejbližších bezbariérových míst. Zadáním názvu hledaného místa ať už textově, nebo hlasovým vstupem dojde k filtrování seznamu a zobrazí se pouze místa odpovídající hledanému řetězci. Místo lze vyhledat také z mapy. Do mapy se lze přepnout tlačítkem „z mapy“ umístěném vlevo pod textovým polem.



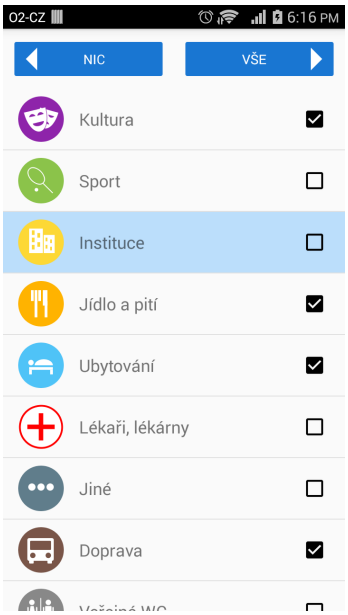
Obrázek 3.9 Ukázka vyhledávací obrazovky

Obrazovka režimu mapy. Tento režim je použit z důvodu omezení ovladatelnosti joystickem, kdy jsou v základním stavu směrová tlačítka použita pro spouštění akcí. V tomto režimu může uživatel pohybovat mapou pomocí joysticku. Ve středu mapy je zobrazen bod, který určuje aktuálně vybrané místo. Po kliknutí na potvrzovací tlačítko se jako hledaný bod vybere právě místo ležící pod tímto bodem. Režim mapy se ukončí po stisku zpětného tlačítka nebo po výběru místa.



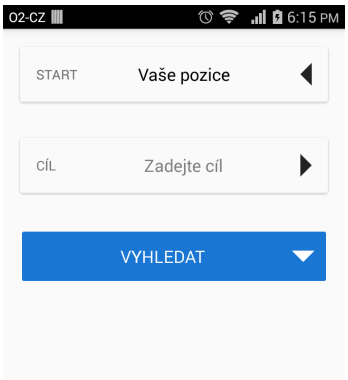
Obrázek 3.10 Ukázka hlavní obrazovky v režimu mapy

Filtr míst. VozejkMap poskytuje data s různými bezbariérovými místy. Tato místa lze rozdělit podle typu do několika kategorií. Vozíčkáře zpravidla nemusí zajímat všechny typy míst, například nepotřebují vědět, kde lze bezproblémově zaparkovat, pokud nepoužívají automobil. Vzhledem k tomu, že se místa zakreslují na mapu, lze výběrem pouze některých typů míst zvýšit přehlednost mapy. Zároveň při hledání míst na vyhledávací obrazovce se v seznamu nabízejí pouze místa, která jsou na obrazovce filtru označena. Na obrazovce filtru lze volit jednotlivé typy míst nebo lze hromadně vybrat či zrušit všechny typy míst.



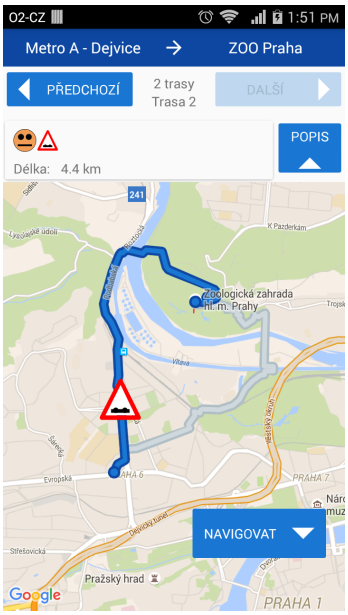
Obrázek 3.11 Ukázka obrazovky filtru míst

Obrazovka pro vyhledání trasy. Zadání nové trasy probíhá standardní cestou, kdy je možné zvolit počáteční a koncový bod trasy. Počáteční bod je v základním nastavení určen aktuální pozicí uživatele. Oba body lze nastavit směrovými šipkami doleva a doprava, kdy se spustí vyhledávací obrazovka. Poté lze jako body trasy nastavit konkrétní místa, případně zvolit místa na mapě. V případě, že bylo vybráno konkrétní místo, je v polích, která označují počáteční a koncový bod trasy, zobrazen název místa. V případě obecného místa vybraného z mapy se na stejném místě zobrazuje adresa místa. Na závěr je zde umístěno tlačítko pro vyhledávání trasy mezi těmito dvěma body.



Obrázek 3.12 Ukázka obrazovky pro vyhledání trasy – formulář

Obrazovka trasy. Obrazovka trasy (obr. 3.13) slouží k poskytnutí obecného přehledu o trase. V horní části obrazovky je zobrazeno označení počátečního a koncového bodu trasy. Pro případy, kdy je nalezeno více rozdílných tras, je zde zobrazena komponenta na přepínání tras. Dále se zde nachází počet nalezených tras a také číslo aktuálně vybrané trasy.

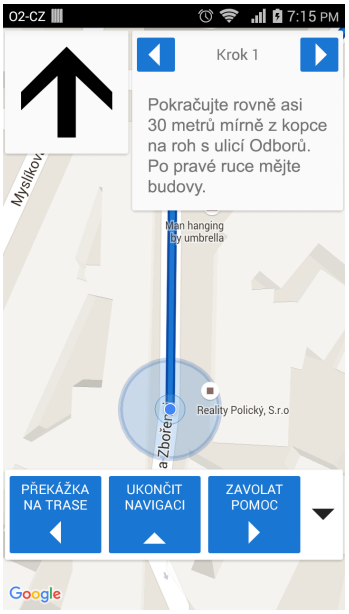


Obrázek 3.13 Ukázka obrazovky trasy

Pod touto komponentou jsou zobrazeny údaje o obtížnosti trasy. Hlavním identifikátorem obtížnosti trasy je smajlík, který rychle a jednoznačně určuje, jak je trasa náročná. Dále jsou zde zobrazeny ikony případných omezení na trase. Tato omezení by neměla zásadně ovlivňovat průjezdnost trasy. Napravo od sekce s informacemi o obtížnosti mapy je umístěno tlačítko pro zobrazení obecného popisu trasy. Tento popis bude obsahovat informace, které poskytuje systém Naviterier. Dále bude obsahovat informace týkající se případných problémů na trase.

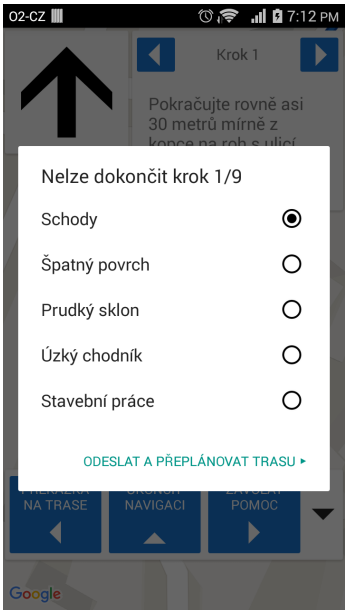
Vetší část obrazovky je zaplněna mapou, na které jsou nalezené trasy zakresleny. Aktuálně vybraná trasa je vždy barevně zvýrazněna. Zároveň jsou na mapě v odpovídajícím úseku trasy zakresleny překážky. V dolní části lze nalézt tlačítko pro spuštění navigace.

Navigační obrazovka. V režimu navigace je neustále sledována pozice uživatele s největší možnou přesností. Proto je nutné, aby bylo na telefonu povoleno sledování pomocí GPS. Navigační instrukce jsou zobrazeny v horní části obrazovky. Pokud je dostupná nová instrukce, zobrazí se v horní části obrazovky notifikace a zazní zvukový signál, aby uživatel věděl, že došlo ke změně stavu aplikace. Následně se zobrazí navigační instrukce, které budou zároveň odeslány na hlasový výstup. Přehrávání instrukcí dovolí uživateli maximálně se věnovat jízdě a nebude muset neustále sledovat obrazovku aplikace.



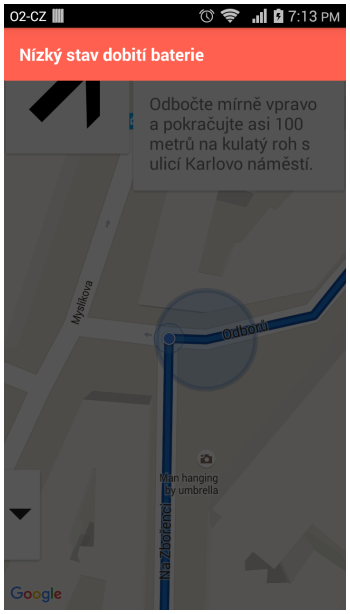
Obrázek 3.14 Ukázka navigační obrazovky

Na této obrazovce je opět použito menu, které obsahuje tři akce. V případě, že vozíčkář narazí na trase na nečekanou překážku, která zcela zabrání pokračování v cestě, bude mít možnost trasu přeplánovat. K tomu bude sloužit první akce „krok nelze dokončit“. Po spuštění této akce bude vozíčkář vyzván k volbě typu překážky a následně může trasu přeplánovat. Druhou akcí je „ukončení navigace“, která zavede uživatele opět na hlavní obrazovku aplikace. Třetí akcí je možnost „odeslat polohu“. Tato akce je určena pro využití v případech, kdy uživatel bude potřebovat asistenci.



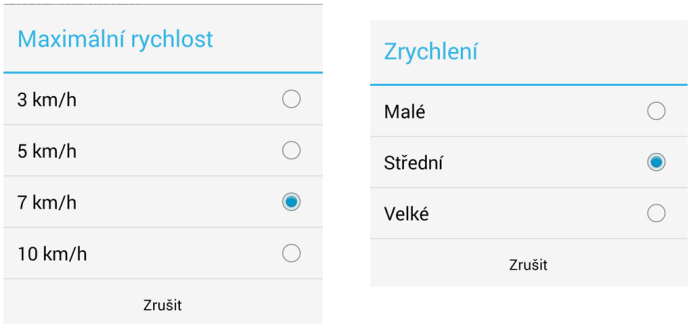
Obrázek 3.15 Ukázka dialogového okna pro přeplánování trasy

Při jízdě může být nezbytné informovat vozíčkáře o jiných problémech či událostech. Vozíčkář může například sjet z naplánované trasy. Pro tyto případy bude využíván systém notifikací. Při vzniku dané události se na obrazovce zobrazí notifikace a přehraje se zvukové upozornění, poté následuje přehrání popisu události.



Obrázek 3.16 Ukázka notifikace

Obrazovka nastavení. Obrazovka nastavení má standardní formu nastavení definovanou platformou Android. Na této obrazovce se budou nastavovat globální parametry aplikace. Jedním z takových parametrů bude například nastavení zdroje dat pro vyhledávání míst. Jako zdroj dat bude možné nastavit VozejkMap, Naviterier nebo Google Places API. Významnou částí nastavení bude sekce s parametry vozíku, kde bude uživatel moci nastavit maximální rychlost a zrychlení elektrického vozíku (obr. 3.17).



Obrázek 3.17 Ukázka dialogových oken pro nastavení parametrů vozíku

3.2 Návrh struktury, modelu a nasazení aplikace

Tato podkapitola obsahuje návrh aplikace z pohledu architektury. V první části je popsáno rozdělení aplikace do balíčků. V druhé části jsou popsány hlavní objekty aplikace. Závěrečná část se věnuje popisu nasazení aplikace a komunikace se systémy třetích stran.

3.2.1 Balíčky

Struktura aplikace bude definována balíčky, které budou seskupovat jednotlivé hlavní související části aplikace. Na nejvyšší úrovni rozdělíme aplikaci do osmi základních balíčků. Základní balíčky budou na dalších úrovních strukturovaněji děleny v závislosti na specifické funkčnosti aplikace.

Bus. Tento balíček bude obsahovat základní třídu pro správu událostí v aplikaci. Tato třída bude dědit od třídy `Bus` z knihovny Otto. Součástí balíčku budou také třídy reprezentující události, které bude možné zasílat prostřednictvím *event bus* do různých částí aplikace.

Controller. Tento balíček bude obsahovat třídy zabývajícími se tvorbou a modifikací informací, které se zobrazují prostřednictvím uživatelského rozhraní. Tyto třídy budou obsahovat například logiku pro práci s mapou, volání vzdálených API a zpracování zaslaných odpovědí.

Db. Tento balíček bude obsahovat kontrakty definující konstanty pro usnadnění práce s tabulkami databáze. Dále bude součástí tohoto balíčku pomocná třída spravující tvorbu a verzování databáze a příslušných tabulek. Tato pomocná třída dědí od třídy `SQLiteOpenHelper`[1]). Hlavní třídou bude třída pro přístup k datům v databázi dědící od třídy `ContentProvider`[2]).

Model. Tento balíček bude obsahovat definici objektů POJO. Některé objekty budou využívány při deserializaci zpráv obdržených při volání vzdálených API. Další objekty POJO budou sloužit jako pomocné kontejnery pro přenos dat.

Receiver. Tento balíček bude obsahovat třídy dědící od třídy `BroadcastReceiver`[3]). Tyto třídy umí zachytávat události vzniklé v systému, jako jsou například změny dostupnosti připojení k internetu nebo změny v dostupnosti GPS.

Service. Tento balíček bude obsahovat služby, které dokážou zpracovávat změny senzorů nebo zpřístupňovat API poskytované přímo mobilním zařízením.

UI. Tento balíček bude obsahovat komponenty uživatelského rozhraní. Komponenty budou dále rozděleny do podbalíčku na aktivity, fragmenty a adaptéry.

Util. Tento balíček bude obsahovat pomocné třídy. Takové třídy budou sloužit například k přístupu k uloženým uživatelským předvolbám a k přístupu k nastavení aplikace. Dále zde budou například třídy pro práci s obrázky a textovými řetězci.

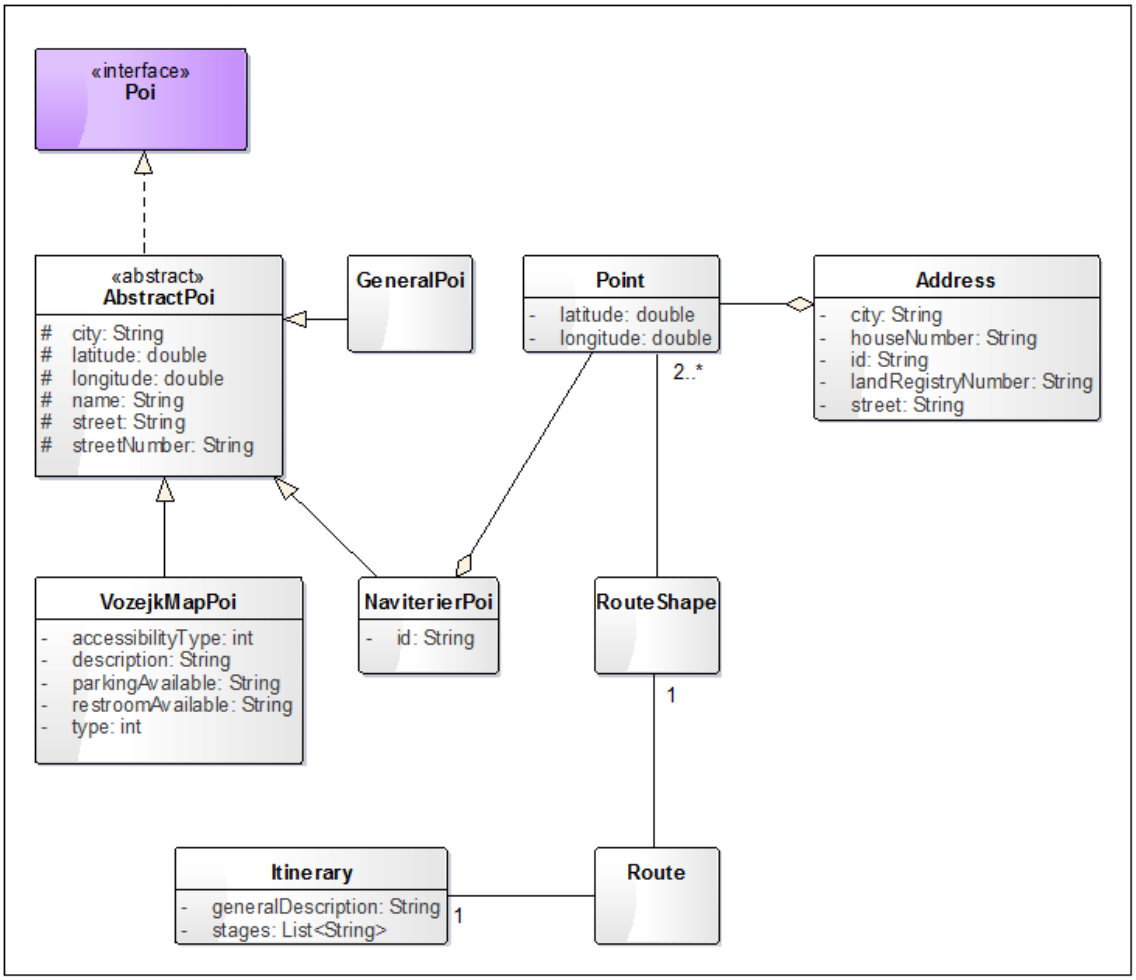
3.2.2 Model

Diagram doménového modelu 3.18 zobrazuje návrh objektů a jejich vzájemných vztahů použitý v aplikaci WheelTrip. Tyto objekty budou využity k deserializaci zpráv obdržených při stahování dat VozejkMap a také při zpracování odpovědí při volání Naviterier API.

Základní entitou celé aplikace je místo, které budeme označovat jako POI (place of interest). V aplikaci existují tři typy míst. Prvním typem místa je místo označující bezbariérový objekt, který se nachází v datech VozejkMap. Druhým typem místa je místo, které se nachází v databázi systému Naviterier. Třetím typem místa je obecné místo. Obecné místo vznikne například výběrem místa z mapy, kdy je známa pouze adresa místa a další informace jsou nedostupné. Všechna tato místa dědí ze společného předka `AbstractPoi`, který obsahuje všechny společné atributy.

Další entity, jako jsou například `Address`, `RouteShape`, `Route` nebo `Itinerary`, jsou využity při sestavování trasy a navigačních instrukcí. Tyto entity jsou definovány systémem Naviterier.

[1]) `http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html`
[2]) `https://developer.android.com/reference/android/content/ContentProvider.html`
[3]) `https://developer.android.com/reference/android/content/BroadcastReceiver.html`



Obrázek 3.18 Diagram modelu

3.2.3 Nasazení

Diagram nasazení 3.19 ilustruje způsob, jakým spolu komunikují jednotlivé fyzické elementy. Aplikace WheelTrip instalovaná na mobilním zařízení bude komunikovat celkem se třemi servery a bude využívat dvě služby dostupné přímo v mobilním zařízení.

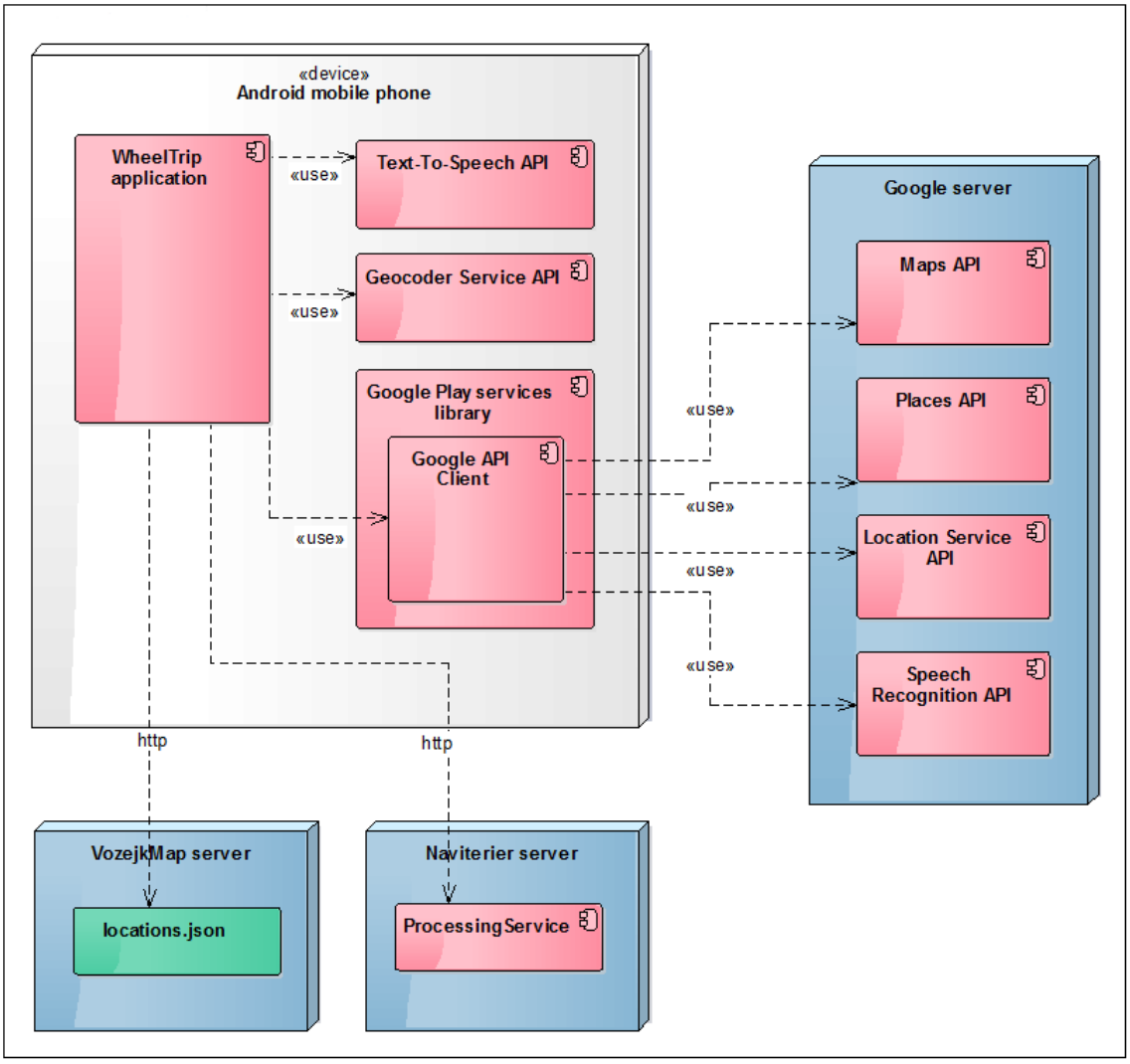
Server VozejkMap. Tento server poskytuje soubor `locations.json`. Tento soubor obsahuje informace o bezbariérových místech z databáze aplikace VozejkMap. Komunikace se serverem probíhá pomocí protokolu HTTP.

Server Naviterier. Tento server poskytuje přístup ke čtyřem službám popsaným v podkapitole 2.8. Komunikace probíhá pomocí protokolu HTTP. V požadavku zaslaném na server lze specifikovat formát dat, který bude použit pro odpověď. Podporované formáty jsou XML, JSON, JSV a CSV.

Server Google. Tento server poskytuje velké množství různých API. Pro potřeby aplikace bude využito čtyř API – Maps API, Location Services API, Places API, Speech Recognition API. Komunikace s tímto serverem neprobíhá přímo, ale prostřednictvím komponenty Google API Client, která se stará o síťové připojení mezi mobilním zařízením a službami poskytovanými serverem Google. Google API Client je součástí knihovny Google Play services.

Text-to-Speech API. Toto API umožňuje přístup k enginu pro převod textu na řeč. Pro využití API je nutné, aby byl engine dostupný na mobilním zařízení.

Geocoder Service API. Tato služba poskytuje třídu `Geocoder`, která umožňuje využít geocoding i reverzní geocoding. Geocoding slouží k nalezení souřadnic pro adresu. Reverzní geocoding pracuje na opačném principu. Aplikace WheelTrip bude využívat reverzní geocoding, a to v případech, kdy bude uživatel vybírat místo z mapy. Pro souřadnice místa se vyhledá jeho adresa, se kterou se bude dále pracovat.



Obrázek 3.19 Diagram nasazení

Kapitola 4
Implementace

V této kapitole budou popsány implementační detaily aplikace. V první podkapitole 4.1 bude popsán framework, na němž bude aplikace postavena, a také knihovny poskytující hlavní funkce aplikace. V dalších podkapitolách 4.2 a 4.3 bude popsána hierarchie aktivit a hierarchie fragmentů, které tvoří uživatelské rozhraní aplikace. V podkapitole 4.4 je popsán systém zpracování vstupních událostí z joysticku a reakce na tyto události. V podkapitole 4.5 je popsán způsob zpracování signálu z GPS. Podkapitola 4.6 se věnuje využití knihovny Retrofit pro získání databáze míst ze serveru VozejkMap. Podkapitola 4.7 se věnuje problematice převodu hlasového vstupu na text. V podkapitole 4.8 je popsána implementace služby pro převod textu na řeč využívající Text-to-Speech API.

4.1 Framework a knihovny

V této podkapitole bude popsán framework, na němž je aplikace WheelTrip postavena. V dalších částech této podkapitoly budou představeny hlavní knihovny a další programové závislosti. Přehled všech závislostí je součástí přílohy.

4.1.1 Framework

Aplikace WheelTrip je postavena na frameworku RoboGuice [18]. RoboGuice je *dependency injection* framework pro aplikace programované pro platformu Android. Hlavním cílem frameworku je minimalizovat počet řádek zdrojového kódu nutný k inicializaci závislostí typu *view*, *service* nebo *resource*.

Ukázka inicializace komponent standardní cestou – všechny proměnné se musí nejprve inicializovat (řádky 11–14):

```
 1 class OldActivity extends Activity {
 2
 3       TextView name;
 4       ImageView icon;
 5       Drawable image;
 6       String text;
 7
 8       public void onCreate(Bundle savedInstanceState) {
 9           super.onCreate(savedInstanceState);
10           setContentView(R.layout.main);
11           name = (TextView) findViewById(R.id.name);
12           icon = (ImageView) findViewById(R.id.icon);
13           image = getResources().getDrawable(R.drawable.image);
14           text = getString(R.string.text);
15       }
16 }
```

Ukázka inicializace stejných komponent při použití frameworku RoboGuice – všechny proměnné jsou automaticky inicializovány (řádky 4–7) při spuštění aktivity:

```
 1 @ContentView(R.layout.main)
 2 class NewActivity extends RoboActivity {
 3
 4       @InjectView(R.id.name) TextView name;
 5       @InjectView(R.id.icon) ImageView icon;
 6       @InjectResource(R.drawable.image) Drawable image;
 7       @InjectResource(R.string.text) String text;
 8
 9      public void onCreate(Bundle savedInstanceState) {
10            super.onCreate(savedInstanceState);
11        }
12 }
```

4.1.2 Crashlytics

Crashlytics Android SDK [19] slouží k reportování pádů aplikace. Pokud aplikace vyhodí výjimku, Crashlytics zaznamená *stack trace* a stav mobilního zařízení, na němž došlo k pádu aplikace, a odešle je na server pro zpracování. Přehledné reporty pro aplikaci jsou přístupné přes webové rozhraní. Ukázku reportu lze nalézt v příloze. Ve výchozím nastavení Crashlytics zpracovává pouze neošetřené výjimky, toto chování lze upravit v kódu tak, aby se zpracovávaly všechny typy výjimek.

V aplikaci se Crashlytics aktivuje v metodě `onCreate` potomka třídy `Application`.

```
public class WheelTripApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        Fabric.with(this, new Crashlytics());
    }
}
```

4.1.3 Otto

Otto [20] je *event bus*, který slouží k zasílání událostí v systému. Tato knihovna podporuje *low coupling*, to znamená, že od sebe umožňuje oddělit různé části aplikace, které tak na sobě nemusí být závislé. Přesto, že jsou od sebe různé části aplikace odděleny, tato knihovna umožňuje jejich vzájemnou komunikaci. Knihovna pracuje na principu *Publish/Subscribe*.

Využití *event bus* je následující.

Nejprve je nutné získat referenci na objekt typu `Bus` například vložením závislosti (*injection*):

```
@Inject
Bus bus;
```

Události se zasílají synchronně ve formě instance objektu, událost může být jakýkoliv objekt. Událost je poté doručena všem (*subsribers*), kteří se pro danou událost registrovali. Odeslání události vyžaduje zavolání metody `post`:

```
bus.post(new NumberEvent(91));
```

Subscriber musí být označen anotací `@Subscribe` a jako jediný vstupní parametr musí obsahovat pouze typ události:

```
@Subscribe
public void numberSelected(NumberEvent event) {
    System.out.println(event.getNumber());
}
```

Aby mohl *subscriber* obdržet registrovanou událost, musí být navíc třída obsahující tento *subscriber* registrována u objektu `bus`:

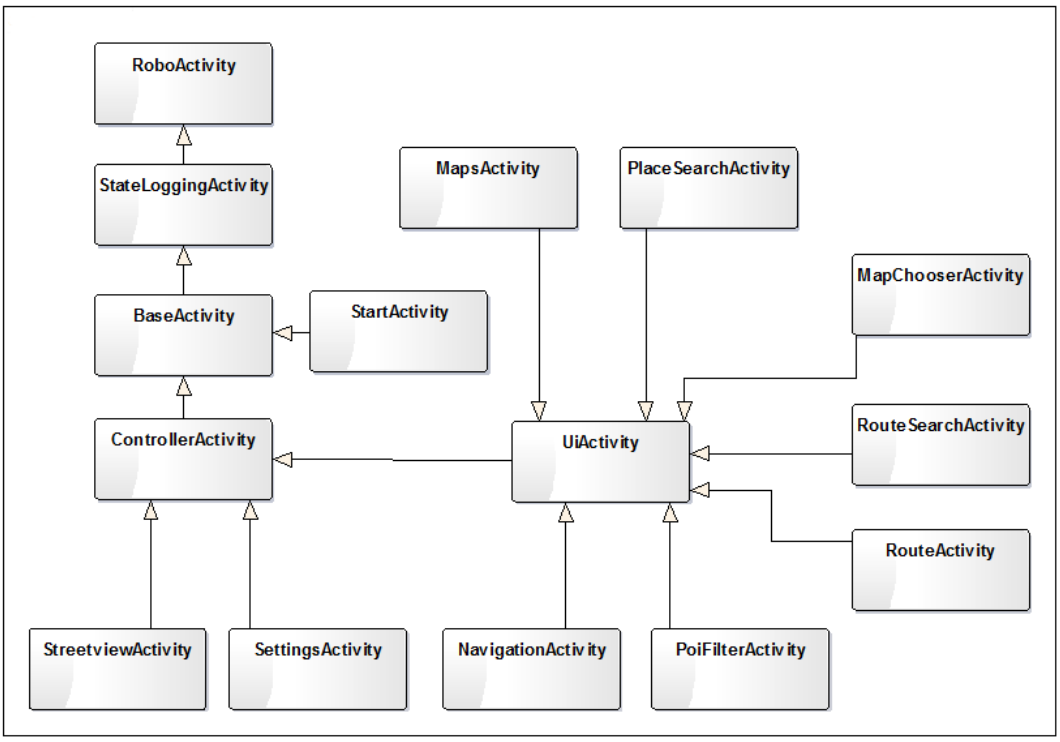
```
bus.register(this);
```

4.1.4 Retrofit

Retrofit [21] je typově bezpečný klient HTTP pro Android a Javu, který převádí HTTP API na javovské rozhraní. Umožňuje zasílat synchronní i asynchronní požadavky HTTP na vzdálený server. Ukázka použití klienta je součástí podkapitoly 4.6.

4.2 Aktivity

Aktivita[1]) je jednou ze základních komponent systému Android. Aktivita zpravidla reprezentuje jednu obrazovku aplikace, obsahuje tedy uživatelské rozhraní. V aplikaci WheelTrip budou aktivity sloužit jako kontejnery pro fragmenty. Uživatelské rozhraní bude pro většinu aktivit implementováno až ve fragmentech. Popis použitých fragmentů se nachází v podkapitole 4.3. Přehled hlavních aktivit aplikace je uveden v následující části. Aktivity jsou rozděleny podle specifické funkčnosti. Funkčnosti potřebné ve více třídách jsou do těchto tříd dodány prostřednictvím dědičnosti.



Obrázek 4.1 Diagram dědičnosti aktivit

[1]) `http://developer.android.com/guide/components/activities.html`

RoboActivity. `RoboActivity` je v hierarchii dědičnosti aktivit na nejvyšší pozici. Jedná se o třídu poskytovanou frameworkem RoboGuice, která umožňuje jednoduše vkládat závislosti přímo do třídy bez nutnosti složité inicializace.

StateLoggingActivity. `StateLoggingActivity` slouží k logování životního cyklu aktivity. Jednotlivé stavy aktivity jsou zaznamenány. Tyto informace lze využít jak při debugování aplikace, tak při prohlížení reportů o pádu aplikace. To se děje v případě, že je aplikace testována na mobilním zařízení mimo vývojové prostředí, a nelze tak využít debugovací konzoli. V obou případech tato aktivita zlepšuje schopnost porozumět vzniklým chybám.

BaseActivity. `BaseActivity` registruje *event bus* pro zasílání událostí.

```
public class BaseActivity extends StateLoggingActivity {

    @Inject protected AppBus mAppBus;

    @Override
    protected void onStart() {
        super.onStart();
        mAppBus.register(this);
    }

    @Override
    protected void onStop() {
        mAppBus.unregister(this);
        super.onStop();
    }
}
```

StartActivity. `StartActivity` je spuštěna při startu aplikace jako první. Tato aktivita se stará o to, aby byla v aplikaci vždy dostupná aktuální data, kontroluje aktuálnost dat a zahajuje proces stahování dat. Neobsahuje uživatelské rozhraní. Po zahájení aktualizace dat předává řízení `MapsActivity`.

ControllerActivity. `ControllerActivity` je stěžejní aktivita aplikace, protože umožňuje využití joysticku k ovládání aplikace. Zpracovává události zaslané z ovladače a volá odpovídající metody definované pro jednotlivá tlačítka v rozhraní aplikace. Popis implementace procesu ovládání joystickem je součástí podkapitoly 4.4.

StreetViewActivity. `StreetViewActivity` slouží k prohlížení reálného prostředí. Pro vozíčkáře je tato funkce velmi důležitá. Mohou si prohlédnout, jak ve skutečnosti vypadá místo, které chtějí navštívit, a sami se tak mohou ujistit, zda se na místě nenacházejí překážky. `StreetViewActivity` využívá Street View Panorama API[1]), které poskytuje fragment pro zobrazení reálného prostředí. Protože načtení tohoto fragmentu probíhá asynchronně, musí aktivita implementovat rozhraní `OnStreetViewPanoramaReadyCallback`. U fragmentu lze nastavit místo počátečního pohledu kamery. Aktivita může být spuštěna pouze z detailu místa. Pozice tohoto místa je zaslána do fragmentu k nastavení počátečního pohledu kamery. Ovládání `StreetViewPanoramaFragment` je nastaveno implicitně. Po ulicích se lze pohybovat pomocí směrových tlačítek. Centrální tlačítko slouží k přepínání mezi úrovněmi přiblížení, které se cyklicky opakují.

[1]) `https://developers.google.com/maps/documentation/android-api/streetview`

Ukázka třídy `StreetViewActivity`:

```
@ContentView(R.layout.activity_streetview)
public class StreetViewActivity extends ControllerActivity
        implements OnStreetViewPanoramaReadyCallback {

    private LatLng position;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        position = getIntent().getParcelableExtra(STREETVIEW_POSITION);

        StreetViewPanoramaFragment streetViewPanoramaFragment =
                (StreetViewPanoramaFragment) getFragmentManager()
                        .findFragmentById(R.id.streetviewpanorama);
        streetViewPanoramaFragment.getStreetViewPanoramaAsync(this);
    }

    @Override
    public void onStreetViewPanoramaReady(
            StreetViewPanorama streetViewPanorama) {
        streetViewPanorama.setPosition(position);
    }
}
```

UiActivity. `UiActivity` reprezentuje aktivity, které obsahují fragment s implementovaným uživatelským rozhraním. Ten je předáván v definici konkrétní aktivity. Dále se stará o zpracování události kliknutí na tlačítko zpět.

Ukázka `UiActivity`:

```
@ContentView(R.layout.activity_base)
public class UiActivity<F extends BaseFragment>
        extends ControllerActivity {

    protected BaseFragment fragment;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        fragment = createFragmet();
        fragment.setArguments(getIntent().getExtras());
        getFragmentManager().beginTransaction().replace(
                R.id.activity_frame, fragment, fragment.getTag())
                .commit();
    }

    protected F createFragmet() {
        ParameterizedType superClass =
                (ParameterizedType) getClass().getGenericSuperclass();
        Class<F> type =
                (Class<F>) superClass.getActualTypeArguments()[0];
        try {
            return type.newInstance();
        } catch (Exception e) {
```

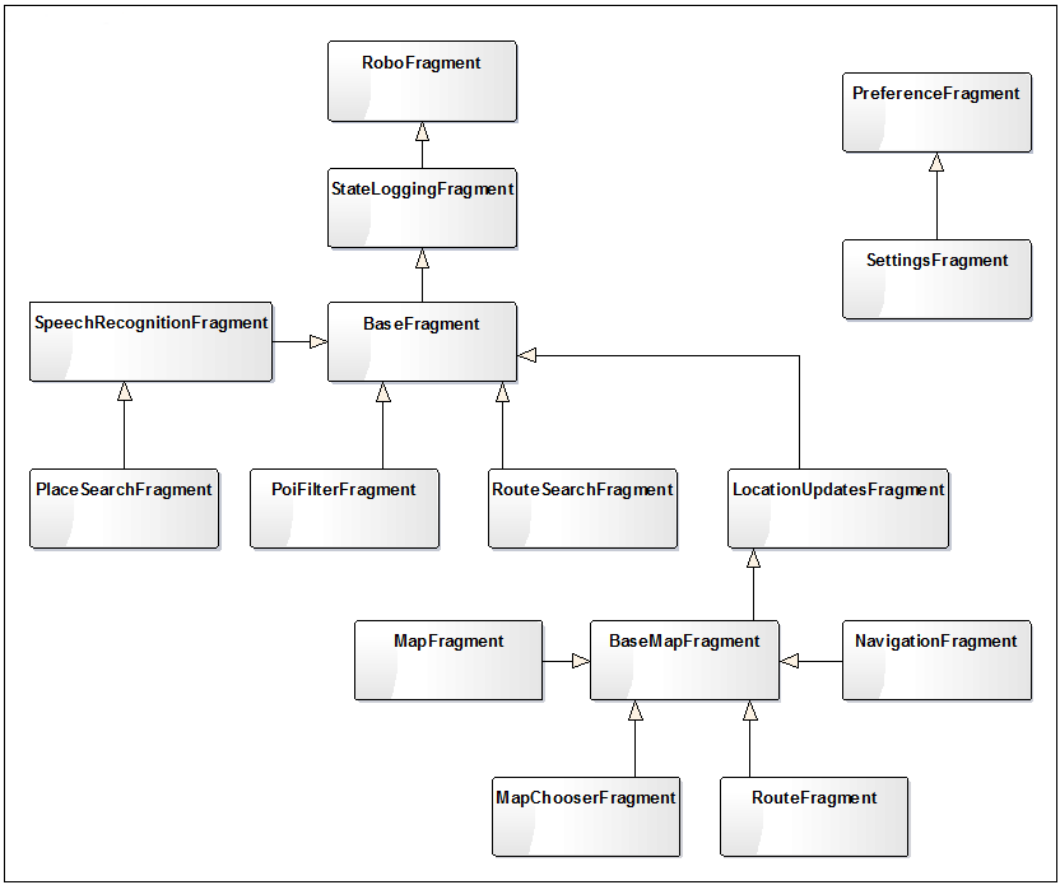
```
            Crashlytics.logException(e);
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onBackPressed() {
        if (fragment != null && fragment.onBackPressed()) {
            return;
        }
        super.onBackPressed();
    }
}
```

Ukázka použití `UiActivity`:

```
public final class MapsActivity extends UiActivity<MapFragment>
```

4.3 Fragmenty

V aplikaci WheelTrip obsahuje většina fragmentů uživatelské rozhraní. V této podkapitole budou popsány hlavní fragmenty aplikace. Podrobnější popis implementace funkcí je součástí následujících podkapitol.



Obrázek 4.2 Diagram dědičnosti fragmentů

`RoboFragment`, `StateLoggingFragment` a `BaseFragment` odpovídají svojí funkčností stejnojmenným aktivitám s tím rozdílem, že jsou implementovány pro využití jako fragmenty.

SpeechRecognitionFragment. `SpeechRecognitionFragment` obsahuje logiku pro zpracování hlasového vstupu. Popis využití Speech Recognition API je součástí podkapitoly 4.7.

PlaceSearchFragment. `PlaceSearchFragment` slouží k vyhledávání míst. Místa lze vyhledávat pomocí textového i hlasováho vstupu. Pro hlasové vyhledávání je použit postup popsaný u `SpeechRecognitionFragment`.

PoiFilterFragment. `PoiFilterFragment` slouží k výběru typu míst, která se budou zobrazovat na mapě a také která se budou zobrazovat při vyhledávání míst, pokud je zvoleno vyhledávání míst z dat VozejkMap.

RouteSearchFragment. `RouteSearchFragment` slouží k vyhledání trasy. Pro nalezení trasy je nutné zadat koncové místo trasy. Počáteční bod trasy je implicitně nastaven na aktuální polohu uživatele, případně lze polohu změnit.

LocationUpdatesFragment. `LocationUpdatesFragment` slouží k určení polohy uživatele. V tomto fragmentu dochází k tvorbě `GoogleApiClient`, který se připojuje na vzdálený server společnosti Google, jenž poskytuje Location Service API. Další informace o získávání polohy jsou uvedeny v podkapitole 4.5.

BaseMapFragment. `BaseMapFragment` se stará o životní cyklus fragmentů, které ve svém rozhraní pracují s mapou.

MapChooserFragment. `MapChooserFragment` slouží k výběru míst z mapy.

MapFragment. `MapFragment` obsahuje rozhraní s mapou, které je zobrazeno po spuštění aplikace. Zároveň také slouží jako rozcestník funkcí aplikace.

RouteFragment. `RouteFragment` slouží k zobrazení vyhledané trasy. Pokud je nalezeno více tras, lze mezi nimi přepínat.

NavigationFragment. `NavigationFragment` je využit při navigaci. Zobrazuje navigační instrukce a pozici uživatele na mapě. Zároveň jsou v tomto fragmentu zobrazeny případné notifikace.

SettingsFragment. `SettingsFragment` jako jediný fragment není členem hlavní hierarchie fragmentů, protože musí dědit od `PreferenceFragment`. Ten umožňuje využití standardního nastavení předvoleb aplikace.

4.4 Ovládání pomocí joysticku

Hlavním cílem aplikace WheelTrip je umožnit uživatelům ovládat aplikaci pomocí joysticku. O zpracování vstupu nejen z joysticku, ale i jiného typu ovladače se stará aktivita `ControllerActivity`. Ta využívá rozhraní `JoystickController`, které specifikuje povolené operace ovladače:

```
public interface JoystickController {

    void onConfirmEvent();
    void onBackEvent();
    void onUpEvent();
    void onDownEvent();
    void onLeftEvent();
    void onRightEvent();

}
```

Rozhraní `JoystickController` je využito v aktivitě `ControllerActivity` jako zpětné volání:

```
protected JoystickController mCallback;

public void setCallback(JoystickController callback) {
        mCallback = callback;
}
```

Samotné zpracování událostí zasílaných při stisknutí tlačítek probíhá v metodě `dispatchKeyEvent` třídy `Activity`, kterou je nutné přepsat, aby bylo možné na tyto události reagovat vlastními akcemi. Pokud se jedná o stisknutí tlačítka reprezentovaného událostí `KeyEvent.ACTION_DOWN`, dochází ke zpracování jednotlivých kódů tlačítek. Pro každý podporovaný kód tlačítka je zavolána odpovídající metoda zpětného volání.

Ukázka přepsané metody `dispatchKeyEvent`:

```
@Override
public boolean dispatchKeyEvent(KeyEvent event) {

        if (mCallback == null) {
            return super.dispatchKeyEvent(event);
        }

        if (event.getAction() == KeyEvent.ACTION_DOWN) {

            switch (event.getKeyCode()) {
                case KeyEvent.KEYCODE_ENTER:
                case KeyEvent.KEYCODE_DPAD_CENTER:
                    mCallback.onConfirmEvent();
                    break;
                case KeyEvent.KEYCODE_BACK:
                    mCallback.onBackEvent();
                    break;
                case KeyEvent.KEYCODE_DPAD_UP:
                    mCallback.onUpEvent();
                    break;
                case KeyEvent.KEYCODE_DPAD_DOWN:
                    mCallback.onDownEvent();
                    break;
                case KeyEvent.KEYCODE_DPAD_LEFT:
                    mCallback.onLeftEvent();
                    break;
                case KeyEvent.KEYCODE_DPAD_RIGHT:
                    mCallback.onRightEvent();
                    break;
            }
        }

        return super.dispatchKeyEvent(event);
}
```

Zpětné volání je využito ve fragmentech, které obsahují uživatelské rozhraní. Nejprve je nutné vytvořit objekt zpětného volání, který bude obsahovat logiku pro zpracování událostí rozhraní `JoystickController`.

Ukázka vytvoření objektu zpětného volání ve fragmentu `PoiFilterFragment`:

```
private JoystickController mCallback = new JoystickController() {

        @Override
        public void onConfirmEvent() {
            clickItem(mAdapter.getSelected());
        }

        @Override
        public void onBackEvent() {
            getActivity().onBackPressed();
        }

        @Override
        public void onUpEvent() {
            mAdapter.setSelected(-1, mRecyclerView);
        }

        @Override
        public void onDownEvent() {
            mAdapter.setSelected(1,mRecyclerView);
        }

        @Override
        public void onLeftEvent() {
            unselectAll.performClick();
        }

        @Override
        public void onRightEvent() {
            selectAll.performClick();
        }
    };
```

V této ukázce například pohyb tlačítka doprava nebo stisknutí pravého směrového tlačítka spustí akci kliknutí na tlačítko pro výběr všech hodnot v seznamu míst filtru.

Aby mohlo být zpětné volání použito v aktivitě `ControllerActivity`, musí být nastaveno pomocí metody `setCallback`. To lze provést v metodě fragmentu `onActivityCreated`, která reprezentuje stav v životním cyklu fragmentu, kdy je uživatelské rozhraní inicializováno.

```
@Override
public void onActivityCreated(Bundle savedInstanceState) {
   ...
   ((ControllerActivity) getActivity()).setCallback(mCallback);
   ...
}
```

4.5 Location Service API

Location Service API slouží k získání aktuální polohy uživatele. Připojení k tomuto API probíhá prostřednictvím `GoogleApiClient`[1]). Je nutné vytvořit objekt `GoogleApiClient` a nastavit mu API, ke kterému se má připojit. Připojení ke klientovi probíhá ve fragmentu `LocationUpdatesFragment`. Aktualizace polohy se spustí metodou `requestLocationUpdates`:

```
LocationServices.FusedLocationApi.requestLocationUpdates(
                    mGoogleApiClient, mLocationRequest, this);
```

První parametrem metody je objekt `GoogleApiClient`. Druhým parametrem je objekt `LocationRequest`[2]), ten obsahuje informace o tom, v jakém intervalu se bude poloha aktualizovat a také zdroj aktualizací polohy, který je dán specifikovanou prioritou. V aplikaci bude využita priorita `LocationRequest.PRIORITY_HIGH_ACCURACY`, to znamená, že jako zdroj polohy bude sloužit především GPS. Posledním parametrem je `LocationListener`[3]), což je rozhraní poskytující metodu `onLocationChanged`, do které jako parametr vstupuje aktuálně získaná poloha. V aplikaci WheelTrip je poloha zaslána pomocí *event bus* všem objektům, které si registrovaly událost `LocationFoundEvent`:

```
@Override
    public void onLocationChanged(Location location) {
        LatLng currentLocation =
         new LatLng(location.getLatitude(), location.getLongitude());
        mAppBus.post(new LocationFoundEvent(currentLocation));
    }
```

4.6 Data VozejkMap

Data VozejkMap jsou využita při vyhledávání míst a také k zobrazování míst na mapě. Na mapě jsou místa zobrazena pomocí ikony identifikující kategorii, do níž místo spadá. Data jsou volně přístupná na adrese www.vozejkmap.cz/opendata/locations.json. Přístup k datům je podmíněn získáním API klíče, o který je třeba zažádat.

Ukázka dat ve formátu JSON:

```
{
   "title":"Divadlo Rokoko",
   "location_type":1,
   "description":"Pokladna divadla v˜Pasáži Rokoko je přímo přístupná.
   Od pokladny do divadla vozíčkáře doprovází obsluha (objednat předem).
   Do divadla se jede výtahem hotelu Rokoko. Za výtahem je chodba 135 cm
   široká, vede do divadla (šatny). Od šatny je přístup k˜sálu pomocí
   plošiny. K sálu vede 11 schodů (vysoké 15 cm). Vozíčkář používá
   plošinu (pomocí tlačítka, ale obsluhu objednat předem). Plošina je
   92 cm široká, 127 cm dlouhá). V˜sále jsou 3 místa pro vozíčkáře,
   vlevo za sebou, každé je 80 cm široké a˜100 cm dlouhé.",
   "street":"Václavské náměstí",
```

[1]) https://developers.google.com/android/reference/com/google/android/gms/common/api/GoogleApiClient

[2]) https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest

[3]) https://developers.google.com/android/reference/com/google/android/gms/location/LocationListener

```
    "street_number":"794/38",
    "city":"Praha",
    "zip":"",
    "phone":"+420 222 996 185",
    "mail":"rezervace@m-d-p.cz",
    "link":"www.mestskadivadlaprazska.cz",
    "opening_hours":"",
    "lat":50.0816423,
    "lng":14.4261593,
    "attr1":11,
    "attr2":"yes",
    "attr3":"no",
    "author_name":"standa"
},
```

V této ukázce se nachází velmi důležité informace pro vozíčkáře v části `description`, které podrobně popisují přístup do divadla. Hodnota `attr1` odpovídá bezbariérovému typu „Výtah + schodišťová plošina“. Hodnota `attr2` říká, že je v objektu bezbariérové WC, naopak hodnota `attr3` říká, že se u objektu nenachází bezbariérové parkování.

K získání dat je využit klient HTTP Retrofit popsaný v podkapitole 4.1.4. Přístup k datům je umožněn využitím javovského rozhraní. Rozhraní definuje volání HTTP API – metoda `getVozejkMapPOIs` je označena anotací `@GET`; ta říká, že volání bude uskutečněno pomocí metody HTTP GET. Hodnota použitá v anotaci udává relativní adresu URL.

```
public interface VozejkMapServiceInterface {

    @GET("/opendata/locations.json?key=API_KEY")
    Call<List<VozejkMapPOI>> getVozejkMapPOIs();
}
```

Toto rozhraní je využito ve třídě `VozejkMapController`, která obsahuje logiku pro práci s daty VozejkMap. Ukázka použití `VozejkMapServiceInterface`:

```
Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();

   VozejkMapServiceInterface service =
    retrofit.create(VozejkMapServiceInterface.class);
```

Třída `Retrofit` vytváří implementaci rozhraní `VozejkMapServiceInterface`. Při tvorbě instance třídy `Retrofit` se předává parametr `BASE_URL`, ten označuje základní část adresy URL, v případě dat VozejkMap je to `http://www.vozejkmap.cz`. Druhým parametrem je *converter*. Ten udává, jakým způsobem se budou serializovat požadavky na server a deserializovat odpovědi ze serveru.

Výsledná služba je poté zavolána asynchronně. V případě bezchybné odpovědi je zpracování této odpovědi provedeno v metodě `onResponse`. Dojde-li k jakékoliv chybě, je zavolána metoda `onFailure`.

```
Call<List<VozejkMapPOI>> call = service.getVozejkMapPOIs();

call.enqueue(new Callback<List<VozejkMapPOI>>() {
   @Override
```

```
    public void onResponse(
        Response<List<VozejkMapPOI>> response, Retrofit retrofit) {
         // zpracování bezchybné odpovědi
    }

    @Override
    public void onFailure(Throwable t) {
        // zpracování chyby
    }
});
```

4.7 Převod hlasového vstupu na text

Hlasový vstup je v aplikaci využit v případech, kdy je nutné získat od uživatele specifický vstup, který nelze namapovat na využití joysticku. Tento specifický vstup je využit například pro vyhledávání míst, kdy je možné v aplikaci nadiktovat název hledaného místa nebo adresy. K tomu aplikace využívá Speech Recognition API. Toto API se používá ve fragmentu `SpeechRecognitionFragment`. Hlavní metodou fragmentu je promtSpeechInput, která se stará o spuštění hlasového rozpoznávání. K tomu je potřeba vytvořit *Intent*[1]):

```
protected void promptSpeechInput(int code) {
        Intent intent =
         new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                 RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(
         RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Mluvte");
        startActivityForResult(intent, code);
}
```

Intent je využit při volání metody `startActivityForResult`, která zahájí samotné rozpoznávání. Parametr `code` v tomto případě slouží k rozlišení akce, pro kterou bylo rozpoznávání spuštěno. Výsledek rozpoznávání lze získat v metodě `onActivityResult`. Pokud například bylo rozpoznávání spuštěno s parametrem `int code = 1`, slovo, které nejvíce odpovídá hlasovému vstupu, lze získat takto:

```
@Override
public void onActivityResult(
int requestCode, int resultCode, Intent data) {
      if (requestCode == 1 && resultCode == RESULT\_OK) {
            ArrayList<String> results =
             data.getStringArrayListExtra(
                    RecognizerIntent.EXTRA_RESULTS);
            String result = results.get(0);
     }
}
```

[1]) `http://developer.android.com/reference/android/content/Intent.html`

4.8 Převod textu na hlasový výstup

V aplikaci WheelTrip je pro převod textu na řeč určena třída `TextToSpeechService`. Aby mohlo být Text-to-Speech API využito, musí úspěšně proběhnout inicializace enginu TTS.

Pro převod textu se používá metoda `speak` enginu TTS, ta má na vstupu tři parametry – textová zpráva pro převod, typ fronty a seznam parametrů.

```
tts.speak(text, TextToSpeech.QUEUE_ADD, params);
```

Typ fronty udává, zda se má zpráva zařadit do fronty (`TextToSpeech.QUEUE_ADD`), nebo zda se má přehrát ihned (`TextToSpeech.QUEUE_FLUSH`); v takovém případě budou předchozí zprávy odebrány z fronty. Jako volitelný parametr bude použit parametr `TextToSpeech.Engine.KEY_PARAM_STREAM`, který určuje, jaký stream se má použít pro přehrání převedeného textu. V závislosti na nastaveném streamu se pro přehrání použije nastavení hlasitosti daného streamu. V aplikaci WheelTrip použijeme pro přehrávání stream určený pro budík.

V aplikaci bude využito také přehrávání zvukových upozornění. Engine TTS pro tento případ využívá *earcon*, což je krátký zvukový signál. Aby mohl být signál přehrán, je nutné *earcon* nejprve vložit do enginu.

```
tts.addEarcon(EARCON_NAME, "cz.cvut.fel.dcgi.wheeltrip", R.raw.tethys);
```

Pro tento případ je třeba vložit do složky `res/raw` soubor se zvukovým signálem. Poté lze *earcon* přehrát podobně jako standardní zprávu.

```
tts.playEarcon(EARCON_NAME, TextToSpeech.QUEUE_ADD, null);
```

`TextToSpeechService` poskytuje tři metody, které určují, jak má být notifikace přehrána.

- `speakMessage(String message)` – přehrání zprávy
- `speakMessageWithNotification(String message)` – přehrání zprávy s notifikací, pauza mezi zvukovým signálem a zprávou je nastavena na 500 ms
- `speakMessageWithNotification(String message, int pause)` – přehrání zprávy s notifikací, pauzu mezi zvukovým signálem a zprávou lze nastavit podle potřeby

Kapitola 5
Testování

V této kapitole bude popsáno testování aplikace WheelTrip. Součástí kapitoly jsou testovací scénáře, které byly vytvořeny na základě diagramu interakce. Pro tyto scénáře budou sestaveny automatizované testy, které budou simulovat interakci reálného uživatele s aplikací. V další části bude popsáno uživatelské testování lo-fi prototypu. Závěr kapitoly je věnován uživatelskému testování implementované aplikace přímo na mobilním telefonu.

5.1 Automatizované testování UI

U mobilních aplikací tvoří uživatelské rozhraní klíčovou roli, proto je nutné zajistit, aby uživatel při interakci s aplikací nenarazil na problém. Správnou funkčnost aplikace lze zajistit automatizovanými testy uživatelského rozhraní. Při těchto testech je ověřován průchod aplikací.

V aplikacích pro Android se využívají k testování uživatelského rozhraní instrumentované unit testy. Tyto testy běží přímo na mobilním zařízení nebo emulátoru. Sestavení testu je podmíněno vytvořením třídy s využitím `ActivityInstrumentationTestCase2` a dodáním potřebné aktivity, která má být v rámci testu spuštěna.

```
public class UiInteractionTest
        extends ActivityInstrumentationTestCase2<MapsActivity> {

    public UiInteractionTest() {
        super(MapsActivity.class);
    }
    ...
```

Poté je nutné vložit potřebnou instrumentaci v metodě označené anotací `@Before`, v rámci níž lze přistupovat ke kontextu aplikace.

```
@Before
public void setUp() throws Exception {
        super.setUp();
        injectInstrumentation(
                InstrumentationRegistry.getInstrumentation());
}
```

Samotné testovací případy jsou obsažené v metodách označených anotací `@Test`.

U aplikací určených pro systém Android lze využít testovací framework Espresso [22]. Tento framework umožňuje psát testy, které simulují interakci uživatele s aplikací. Pro potřeby návrhu testů využijeme diagram interakce mezi obrazovkami (obr. 3.6). Pro účely testování nebudeme uvažovat některé zpětné přechody.

V rámci návrhu testovacích scénářů bude zvolen průchod aplikací tak, aby byly pokryty všechny stavy – obrazovky diagramu.

5.1.1 Využití frameworku Espresso

Framework Espresso umožňuje napsat testy, které maximálně odpovídají reálné interakci uživatele s aplikací. Espresso využívá tři základní konstrukty:

- `ViewMatcher` – slouží k nalezení komponenty UI prostřednictvím funkce `onView`.
- `ViewAction` – slouží k provedení akce prostřednictvím funkce `perform`.
- `ViewAssertion` – slouží k ověření, zda je splněna určitá podmínka prostřednictvím funkce `check`.

Pro potřeby testování budeme využívat pouze `ViewMatcher` sloužící k nalezení požadované komponenty UI a `ViewAssertion` určený k vyhodnocení testované podmínky. Ke spuštění akcí nebudeme využívat `ViewAction`, místo toho využijeme metodu `sendKeys` třídy `ActivityInstrumentationTestCase2`. Tento přístup bude využit proto, aby se testy co nejvíce přiblížily reálnému ovládání aplikace pomocí joysticku. Obrazovku filtru lze například otevřít zasláním události kliknutí na směrovou šipku nahoru.

```
sendKeys(KeyEvent.KEYCODE_DPAD_UP);
```

5.1.2 Testování průchodu aplikací

Průchod aplikací bude vždy začínat na hlavní obrazovce. Testovací cesty se budou lišit podle toho, zda je například na obrazovce otevřené menu nebo je zobrazen detail hledaného místa.

Pro otestování průchodu aplikací byly zvoleny tyto cesty:

- 1. cesta (Otevřené menu): Mapa – Nastavení – Mapa – Filtr – Mapa – Hledání trasy – Trasa – Navigace – Mapa
- 2. cesta (Zavřené menu): Mapa – Výběr místa z mapy – Mapa – StreetView – Mapa
- 3. cesta (Zavřené menu): Mapa – Hledání místa – Mapa – Trasa

Na základě těchto cest jsou vytvořeny testovací scénáře, jež slouží k implementaci automatizovaných testů. Scénáře obsahují popis akce, kterou je nutné provést, za ní následuje popis očekávaného výsledku.

Testovací scénář 1:

1) Spustit aplikaci. – Zobrazí se hlavní obrazovka, menu je otevřeno.
2) Kliknout na tlačítko „NASTAVENÍ“ v dolním menu. – Zobrazí se nastavení aplikace.
3) Kliknout na zpětné tlačítko. – Zobrazí se hlavní obrazovka aplikace.
4) Kliknout na tlačítko „FILTR“ v dolním menu. – Zobrazí filtr míst.
5) Kliknout na zpětné tlačítko. – Zobrazí se hlavní obrazovka aplikace.
6) Kliknout na tlačítko „TRASA“ v dolním menu. – Zobrazí se obrazovka pro vyhledání trasy.
7) Kliknout na tlačítko „CÍL“. – Zobrazí se obrazovka pro vyhledávání míst.
8) Do vyhledávacího pole zadat řetězec „retro“. – Zobrazí se záznamy odpovídající zadanému textu.
9) Kliknout na první nalezenou položku seznamu. – Zobrazí se obrazovka pro vyhledání trasy s koncovým bodem trasy nastaveným na vybrané místo.
10) Kliknout na tlačítko „VYHLEDAT“. – Zobrazí se nalezená trasa.
11) Kliknout na tlačítko „NAVIGOVAT“. – Zobrazí se navigační obrazovka.
12) Kliknout na tlačítko „UKONČIT NAVIGACI“. – Zobrazí se hlavní obrazovka aplikace.

Testovací scénář 2:

1) Spustit aplikaci. – Zobrazí se hlavní obrazovka, menu je otevřeno.
2) Kliknout na tlačítko pro uzavření menu. – Dojde k zavření menu.
3) Kliknout na tlačítko „MAPA“ v horní části obrazovky. – Zobrazí se obrazovka pro vyhledávání míst z mapy.
4) Kliknout na potvrzovací tlačítko. – Zobrazí se hlavní obrazovka aplikace s otevřeným detailem vybraného místa.
5) Kliknout na tlačítko „STREETVIEW“. – Zobrazí se pohled na okolí vybraného místa.
6) Kliknout na zpětné tlačítko. – Zobrazí se hlavní obrazovka aplikace.

Testovací scénář 3:

1) Spustit aplikaci. – Zobrazí se hlavní obrazovka, menu je otevřeno.
2) Kliknout na tlačítko menu. – Dojde k zavření menu.
3) Kliknout na tlačítko „HLEDAT“ v horní části obrazovky. – Zobrazí se obrazovka pro vyhledávání míst.
4) Do vyhledávacího pole zadat řetězec „retro“. – Zobrazí se záznamy odpovídající zadanému textu.
5) Kliknout na první nalezenou položku seznamu. – Hlavní obrazovka s otevřeným detailem vybraného místa.
6) Kliknout na tlačítko „TRASA“ v detailu místa. – Zobrazí se nalezená trasa.

Ukázka implementace testovacího scénáře 3:

```
@Test
public void test3ApplicationWalkthrough() throws InterruptedException {
        // zavřít menu
        sendKeys(KeyEvent.KEYCODE_DPAD_DOWN);
        // čekání na dokončení animace zavření menu
        Thread.sleep(3000);
        // kontrola zobrazení tlačítka hledat
        onView(withId(R.id.search_button))
         .check(matches(isDisplayed()));
        // kliknutí na tlačítko hledat
        sendKeys(KeyEvent.KEYCODE_DPAD_RIGHT);
        // čekání na načtení dat
        Thread.sleep(3000);
        // zadání vyhledávacího řetězce
        onView(withId(R.id.edittext_places)).perform(typeText("r"));
        onView(withId(R.id.edittext_places)).perform(typeText("e"));
        onView(withId(R.id.edittext_places)).perform(typeText("t"));
        onView(withId(R.id.edittext_places)).perform(typeText("r"));
        onView(withId(R.id.edittext_places)).perform(typeText("o"));
        //čekání na vyhledání dat
        Thread.sleep(2000);
        // vybrání první položky nalezených míst
        sendKeys(KeyEvent.KEYCODE_ENTER);
        // kontrola zobrazení tlačítka pro nalezení trasy
        // na detailu místa
        onView(withId(R.id.navigate)).check(matches(isDisplayed()));
        // kliknutí na tlačítko pro vyhledání trasy
        sendKeys(KeyEvent.KEYCODE_DPAD_RIGHT);
        // kontrola zobrazení tlačítka pro spuštění navigace
```

```
            onView(withId(R.id.navigate_button))
             .check(matches(isDisplayed()));
    }
```

5.1.3 Pokrytí zdrojového kódu

V rámci testování automatizovanými testy lze vytvořit dokumentaci pokrytí zdrojového kódu. Tato dokumentace slouží k lepšímu pochopení aplikace z pohledu dalšího možného testování. Lze zjistit jaké instrukce a také jaké logické větve programu jsou pokryty testy. Dále lze v dokumentaci nalézt části kódu, které nebyly testy pokryty, a případně připravit další testovací scénáře. Výsledek pokrytí kódu navrženými testy je ilustrován na obrázku 5.1.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
cz.cvut.fel.dcgi.wheeltrip.ui.fragment		81%		53%	170	433	244	1 009	69	296
cz.cvut.fel.dcgi.wheeltrip.util		66%		57%	43	90	58	154	15	38
cz.cvut.fel.dcgi.wheeltrip.model.naviterier		23%		n/a	21	37	34	50	21	37
cz.cvut.fel.dcgi.wheeltrip.controller.vozejkmap		81%		75%	7	33	23	119	2	19
cz.cvut.fel.dcgi.wheeltrip.model.route		83%		59%	33	111	46	162	24	95
cz.cvut.fel.dcgi.wheeltrip.ui.adapter		86%		62%	31	78	34	188	8	39
cz.cvut.fel.dcgi.wheeltrip.db		76%		50%	13	25	28	83	5	14
cz.cvut.fel.dcgi.wheeltrip.service.tts		29%		33%	8	13	22	37	5	10
cz.cvut.fel.dcgi.wheeltrip.receiver		18%		7%	22	24	27	34	4	6
cz.cvut.fel.dcgi.wheeltrip.service.activityrecognition		44%		33%	11	15	15	33	1	4
cz.cvut.fel.dcgi.wheeltrip.ui.activity		86%		76%	10	55	13	126	4	40
cz.cvut.fel.dcgi.wheeltrip.bus		29%		50%	6	9	9	17	5	8
cz.cvut.fel.dcgi.wheeltrip.controller.map		93%		87%	5	40	5	97	1	21
cz.cvut.fel.dcgi.wheeltrip.model.directions		77%		n/a	8	29	13	41	8	29
cz.cvut.fel.dcgi.wheeltrip.model		69%		n/a	8	29	8	36	8	29
cz.cvut.fel.dcgi.wheeltrip.controller.naviterier		93%		100%	2	19	8	62	2	17
cz.cvut.fel.dcgi.wheeltrip.db.naviterier		79%		n/a	8	12	8	14	8	12
cz.cvut.fel.dcgi.wheeltrip.controller.directions		96%		92%	3	18	7	86	1	6
cz.cvut.fel.dcgi.wheeltrip.service.compass		93%		75%	5	17	4	50	1	9
cz.cvut.fel.dcgi.wheeltrip.bus.event		65%		n/a	4	12	8	22	4	12
cz.cvut.fel.dcgi.wheeltrip		86%		100%	5	29	7	48	5	28
cz.cvut.fel.dcgi.wheeltrip.db.vozejkmap		84%		n/a	4	6	4	7	4	6
default		100%		50%	4	10	0	183	0	6
cz.cvut.fel.dcgi.wheeltrip.model.vozejkmap		100%		n/a	0	11	0	16	0	11
cz.cvut.fel.dcgi.wheeltrip.model.filter		100%		n/a	0	7	0	10	0	7
Total	2 481 of 12 724	81%	275 of 656	58%	431	1 162	625	2 684	205	799

Obrázek 5.1 Pokrytí zdrojového kódu aplikace

5.2 Uživatelské testování lo-fi prototypu

Prototyp byl navržen ve webovém rozhraní aplikace proto.io[1]). Proto.io umožňuje spustit prototyp přímo na cílovém zařízení prostřednictvím vlastní mobilní aplikace. Ukázky lo-fi prototypu jsou součástí přílohy C.

5.2.1 Participanti

Testování se zúčastnilo 5 participantů. Vzhledem k tomu, že se jednalo pouze o ověření srozumitelnosti rozhraní, byli participanti vybráni z řad studentů ČVUT. Nejednalo se o vozíčkáře.

5.2.2 Průběh testování

Participanti byli na začátku testování upozorněni, že mohou aplikaci ovládat pouze joystickem (4 polohy + 2 tlačítka), při interakci s prototypem pouze oznamovali, jakou polohu nebo tlačítko chtějí použít.

[1]) `https://proto.io/`

Participanti plnili následující úkoly:

- Zjistěte informace o nejbližší restauraci.
- Zajistěte, že se na mapě nebudou zobrazovat hotely.
- Nalezněte trasu z ulice Na Zbořenci do ulice Štěpánská (přesná pozice je na rohu ulice Štěpánská a Na Rybníčku).
- Zjistěte, zda je trasa náročná.
- Spusťte navigaci.
- Nacházíte se v ulici Odborů, ale nemůžete pokračovat dál kvůli probíhající stavbě, vyřešte tuto situaci.

Participanti byli požádáni, aby každý svůj krok komentovali.

5.2.3 Výsledky testování

Testování probíhalo bez větších problémů. Participanti si rychle zvykli na způsob ovládání. Kladně bylo hodnoceno zobrazení ikon pozice ovladače přímo u ovládacích prvků. Přesto byly v průběhu testování zjištěny dva nálezy.

- Ovládání joystickem na hlavní obrazovce
  - Ve čtyřech případech, kdy měli participanti zjistit informace o nejbližší restauraci, chtěli rovnou pomocí joysticku označit ikonu restaurace (pohybovat mapou), toho lze dosáhnout až v režimu mapy (poloha nahoru).
  - Možné řešení: při prvním startu aplikace zobrazit tutoriál ve formě částečně průhledné obrazovky, kde bude popsáno, že pohyb mapou je možný pouze po přepnutí do režimu mapy, a tato možnost bude zvýrazněna; po stisknutí potvrzovacího tlačítka se tutoriál skryje.
- Upřesnění pozice při hledání podle názvu ulice
  - Při hledání podle názvu ulice je výběr ze tří míst nedostatečný (pro dlouhé ulice). Participanti se dotazovali, podle jakého klíče jsou tato tři místa vybrána. Pokud by vybírali místo blízko bodu A, které označuje levou část ulice, nemají možnost, jak určit, že chtějí na pravou část ulice, kromě výběru vlastního místa v režimu mapy.
  - Možné řešení: při vyhledávání místa podle názvu ulice rovnou zobrazit režim mapy, výběr z nabízených míst ponechat pouze pro případ, kdy backend potřebuje upřesnit polohu uživatele (i v tomto případě by mohla být nabízena rovnou možnost výběru v režimu mapy). Dalším možným řešením je nabízet uživateli pouze konkrétní místa tak, aby bylo místo jednoznačně určeno svojí polohou. Uživatel by tak nebyl nucen upřesňovat polohu a musel by se rozhodovat pouze mezi dostupnými místy.

5.3 Uživatelské testování finální aplikace

V této podkapitole bude popsáno testování aplikace WheelTrip s cílovou skupinou uživatelů.

5.3.1 Cíle testování

Hlavním cílem testování je ověřit použitelnost aplikace. To znamená především nalézt problémy v rozhraní, které by mohly nějakým způsobem omezovat uživatele v interakci s aplikací. Zároveň bude vedlejším cílem testování srovnání dvou způsobů ovládání aplikace. Na jedné straně bude k ovládání aplikace využit joystick, na straně druhé bude testováno ovládání aplikace pomocí klávesnice Tecla. Proto bude mít testování formu komparativního testování [23].

5.3.2 Participanti

Pro uživatelské testování byla vybrána skupina participantů reprezentující cílovou skupinu uživatelů aplikace. Byla oslovena specifická skupina vozíčkářů, která pravidelně provozuje různé sportovní aktivity. Tato skupina sestávala pouze z vozíčkářů používajících mechanický vozík. Celkem se testování zúčastnilo 5 participantů.

5.3.3 Příprava testování

Pro potřeby testování byl vytvořen seznam úvodních otázek, jehož cílem bylo seznámit se s jednotlivými participanty a s jejich pohledem na využívání mobilního telefonu při jízdě na vozíku.

Dále byl vytvořen seznam úkolů (příloha F), které budou participanti vykonávat v průběhu testování. Úkoly byly zvoleny tak, aby participanti prošli hlavní funkce aplikace. Zároveň byly úkoly zvoleny tak, aby participanti vykonávali reálné scénáře použití aplikace.

Z důvodu otestování ovládání aplikace pomocí joysticku byl pro testování zvolen telefon Xiaomi Mi3, který podporuje technologii OTG. K telefonu byl připojen gamepad Saitek PS2700 Rumble Pad, jehož ovládací páčka simulovala použití joystickové páčky. Zároveň vzhledem k problematickému připevnění mobilního telefonu spolu s joystickem k vozíku nebylo možné provést testování v terénu. Participanti plnily úkoly u stolu, mobilní telefon a joystick byly umístěny na stole v přibližné vzdálenosti, ve které se předpokládá i využití v terénu, a na straně odpovídající tomu, zda byl participant levák nebo pravák. Průjezd trasou byl simulován pomocí skrytého tlačítka, kterým se bylo možné přepínat na další bod trasy. Pro převod textu na řeč byl použit modul SVOX Classic TTS, který byl nastaven na čtení češtiny pomocí ženského hlasu s názvem Iveta.

5.3.4 Průběh testování

Testovací sezení bylo rozděleno do několika částí. Nejprve byl v první části participant seznámen s průběhem testování. V druhé části proběhlo krátké interview 5.3.5. Participanti byli obeznámeni s principem ovládání aplikace. Hlavní část sezení se týkala plnění úkolů z přílohy F. Participanti byli požádáni, aby každý svůj krok komentovali. Každý participant plnil úkoly dvakrát pro oba způsoby ovládání – joystickem a klávesnicí Tecla. Na závěr byli participanti požádáni o srovnání obou způsobů ovládání. Participanti 1, 3 a 5 začínali plnit úkoly za pomoci joysticku. Participanti 2 a 4 začínali s klávesnicí Tecla.

5.3.5 Zápis z úvodního interview

V této části budou obsaženy odpovědi na otázky z úvodního interview. Je zde poskytnuto shrnutí odpovědí vytvořené ze zápisků z jednotlivých sezení.

Participant 1

- Jaké máte zkušenosti s používáním mobilního telefonu při jízdě na vozíku?

  Pokud jede participant na vozíku, snaží se telefon nepoužívat. Je to pro něj nepraktické, protože nosí telefon v kapse a musí ho vyndavat.

- Pokud musíte jet na neznámé místo, jak si zjišťujete kudy jet?

  Participant jezdí pravidelně stejné trasy, o nichž ví, že jsou pro něj bezproblémové. Zpravidla nejezdí na neznámá místa, zná okolí, kde se často pohybuje. Když už jede na neznámé místo, zjistí si, zda se nachází místo v blízkosti pro něj známých bodů, a orientuje se podle nich.

- Jakou aplikaci pro vyhledávání tras nyní využíváte?

  Participant nepoužívá žádnou aplikaci na plánování tras. Zná dobře své okolí.

- Co se vám na aplikaci líbí?

  –

- Co vám v aplikaci schází?

  –

- Kolikrát týdně byste použili aplikaci pro vyhledávání tras s navigací pro vozíčkáře?

  Participant by aplikaci nevyužíval, nepotřebuje si plánovat trasy.

Participant 2

- Jaké máte zkušenosti s používáním mobilního telefonu při jízdě na vozíku?

  Participant nepoužívá telefon při jízdě na vozíku.

- Pokud musíte jet na neznámé místo, jak si zjišťujete kudy jet?

  Krátké trasy jezdí zpaměti. Pro delší trasy využívá dopravu automobilem. Pokud jede na neznámé místo, využívá ke zjištění podrobností o místu internet.

- Jakou aplikaci pro vyhledávání tras nyní využíváte?

  Používá webovou aplikaci Mapy.cz, ale plánování tras ani tak nevyužívá. Spíše se kouká pouze na mapu.

- Co se vám na aplikaci líbí?

  Participantovi se líbí možnost prohlížení panorama, kterou často využívá, aby si mohl prohlédnout celé ulice.

- Co vám v aplikaci schází?

  Aplikaci považuje pro své potřeby za dostatečnou, přidal by informace o bezbariérovosti míst. Pro tyto informace využívá webovou aplikaci VozejkMap, chtěl by mít vše na jednom místě.

- Kolikrát týdně byste použili aplikaci pro vyhledávání tras s navigací pro vozíčkáře?

  Participantovi postačuje aplikace Mapy.cz, kterou používá a je s ní spokojen. Myslí si, že by další aplikaci nepotřeboval.

Participant 3

- Jaké máte zkušenosti s používáním mobilního telefonu při jízdě na vozíku?

  Participant nepoužívá telefon při jízdě na vozíku.

- Pokud musíte jet na neznámé místo, jak si zjišťujete kudy jet?

  Na neznámá místa na vozíku nejezdí. Pro dopravu využívá především automobil.

- Jakou aplikaci pro vyhledávání tras nyní využíváte?

  Používá pouze navigaci do auta. Mobilní aplikaci na vyhledávání tras nepoužívá. Důležitá je pro něj pouze možnost bezbariérového parkování.

- Co se vám na aplikaci líbí?

  Navigace mu nalezne trasu a přehrává mu navigační instrukce. Protože využívá automobil a nemusí delší trasy jezdit na vozíku, je pro něj tato navigace ideální.

- Co vám v aplikaci schází?

  Informace o bezbariérovém parkování.

- Kolikrát týdně byste použili aplikaci pro vyhledávání tras s navigací pro vozíčkáře?

  Při jízdě na vozíku by ji nevyužíval. Myslí si, že by mu vadilo přehrávání instrukcí na ulici, protože by musely být instrukce přehrávány velmi nahlas, aby je slyšel, a to by mu bylo nepříjemné. V autě mu to nevadí, protože je v uzavřeném prostoru.

Participant 4

- Jaké máte zkušenosti s používáním mobilního telefonu při jízdě na vozíku?

  Při jízdě na vozíku má telefon u sebe, občas telefonuje nebo píše textové zprávy. Telefon má v pouzdře připevněném na vozíku.

- Pokud musíte jet na neznámé místo, jak si zjišťujete kudy jet?

  Podívá se do mapy.

- Jakou aplikaci pro vyhledávání tras nyní využíváte?

  Participant používá Google Maps, ale ne na vyhledávání trasy. Dokáže se zorientovat na mapě a trasu si vytvořit sám. Označí si místo, kam má jet, a pokud právě neví kudy pokračovat, podívá se znovu do mapy.

- Co se vám na aplikaci líbí?

  Hezká a jednoduchá mapa. Může si označit oblíbená místa. Může se podívat na místo pomocí StreetView.

- Co vám v aplikaci schází?

  Nedokáže z mapy poznat nebezpečná místa. Na mapě nejsou označeny přechody.

- Kolikrát týdně byste použili aplikaci pro vyhledávání tras s navigací pro vozíčkáře?

  Většinou jezdí pravidelné trasy. Občas se jede podívat i jinam, zhruba jednou týdně. Aplikaci by využíval pro nalezení trasy, hlasové instrukce by nevyužíval, pouze by se občas podíval, kde se nachází a kudy má jet dál.

Participant 5

- Jaké máte zkušenosti s používáním mobilního telefonu při jízdě na vozíku?

  Při jízdě na vozíku nepoužívá mobilní telefon. Nevlastní chytrý telefon s dotykovým displejem.

- Pokud musíte jet na neznámé místo, jak si zjišťujete kudy jet?

  Podívá se na internet, vyhledává trasy pro chodce. Jinak používá hromadnou dopravu.

- Jakou aplikaci pro vyhledávání tras nyní využíváte?

  Mapy.cz.

- Co se vám na aplikaci líbí?

  Hezká barevná mapa. Obsahuje spoustu informací, na mapě jsou označeny body zájmu (restaurace, obchody a další).

- Co vám v aplikaci schází?

  Nedokáže naplánovat trasu pro vozíčkáře. Musí plánovat trasy pro chodce.

- Kolikrát týdně byste použili aplikaci pro vyhledávání tras s navigací pro vozíčkáře?

  Aplikaci by využil jednou až dvakrát do týdne, ale musela by být dostupná přes internet. Na telefonu ji nemůže využít, nový telefon si kupovat nechce.

5.3.6 Zápis z průběhu testování

V této části bude souhrnně popsáno s jakými problémy se participanti setkali při plnění konkrétních úkolů. Většinu úkolů dokázali participanti dokončit. Zde budou popsány pouze problémy s úkoly, které byly pro participanty nejobtížnější.

- Úkol 1: Participant nejprve zkoušel spustit vyhledávání s otevřeným menu, místo vyhledávání se mu spustilo nastavení. Poté si všiml, že při zavření menu se zvýraznily šipky u tlačítek v horní části obrazovky a vyhledávání tak spustil. Při vyhledávání místa chtěl participant použít klávesnici, ta ale s joystickem nelze použít, rozhodl se proto pro hlasové vyhledávání.

  Další participant chtěl využít pro hledání restaurace filtr, protože si nevšiml ikony lupy v horním rohu obrazovky.
- Úkol 4: Participant se rozhodl vyhledat trasu přes obrazovku pro vyhledávání trasy. Poté ale zjistil, že nemůže použít vyhledávání z dat VozejkMap pro nalezení zastávky Náměstí Míru. Proto přepnul v nastavení aplikace zdroj dat pro vyhledávání.

  Další participant se snažil využít režim mapy pro označení zastávky Náměstí Míru. Stěžoval si, že nelze přesně kurzorem uprostřed obrazovky označit cílové místo. To je způsobeno Google Maps API, které podporuje pouze přesun mezi dvěma místy. Mapu tak lze vždy posunout jen o konstantní vzdálenost.

- Úkol 5: Participant ihned označil první trasu za nejkratší, protože se domníval, že zelený smajlík označuje nejlepší, a tudíž nejkratší trasu. To ale nemusí být vždy pravda.

Srovnání způsobů ovládání. Po splnění úkolů s pomocí obou způsobů ovládání byli participanti požádáni o jejich srovnání. Z výsledků srovnání je patrné, že ovládání pomocí joysticku je mnohem lepší než ovládání pomocí klávesnice Tecla.

U joysticku oceňovali participanti především rychlost ovládání. Jako jediný problém s joystickem se jeví nemožnost zadávání vstupu z klávesnice. To je ale vyváženo možností hlasového vstupu.

Klávesnice Tecla byla označena jako příliš složitá a pomalá na ovládání. Participanti si stěžovali, že když nestihnou stisknout potvrzovací tlačítko ve chvíli, kdy je zvýrazněn potřebný směr, musí čekat, dokud se nedokončí skenování klávesnice a dokud nezačne skenování od začátku. Klávesnice Tecla na jednu stranu poskytuje možnost vstupu z klávesnice, ale z pohledu participantů je tento způsob zadávání příliš pomalý, což se projeví především při zadávání delších řetězců. Nevýhodou participanti také označili, že je nutné sledovat skenování klávesnice, aby mohli označit danou akci, což u joysticku není nutné.

5.3.7 Vyhodnocení testování

Cíl ověření použitelnosti aplikace byl splněn. Při testování aplikace nebyly nalezeny žádné závažné problémy, které by zásadním způsobem ovlivňovaly použitelnost aplikace. Participanti neměli větší problém s plněním zadaných úkolů. U některých úkolů se někteří participanti setkali s menšími problémy, avšak po podrobnějším prostudování rozhraní dokázali tyto problémy sami vyřešit.

Kladně bylo hodnoceno označení ovládacích prvků aplikace směrovými šipkami a také dostatečná čitelnost textu. Participantům se také líbila grafika použitá pro označení míst z dat VozejkMap. Participanti rovněž ocenili srozumitelnost čtení instrukcí. To lze přikládat tomu, že byla použita ke čtení instrukcí komerční aplikace místo systémového nastavení.

Ikona označující akci pro spuštění režimu mapy nebyla některým participantům srozumitelná, proto by bylo vhodné dodat popisek i k této akci. Dále si participanti stěžovali na nepřesnost kurzoru v režimu mapy. Lepší řešení zatím není možné, protože Google Maps API nepodporuje plynulý pohyb mapy, a proto se lze na mapě posouvat pouze o konstantní vzdálenost.

Vedlejší cíl testování, kterým bylo srovnání ovládání pomocí joysticku a pomocí klávesnice Tecla, byl také splněn. Participanti zvolili ovládání pomocí joysticku jako celkově lepší, jeho hlavní výhodou oproti klávesnici Tecla je jeho rychlost. To lze připsat volbě testovacího vzorku, kdy participanti nebyli omezeni v pohybu horních končetin. Lze předpokládat, že pro některá omezení vozíčkářů, jež by zamezovala v plynulém pohybu joysticku, by byla klávesnice Tecla lepší alternativou, protože k jejímu ovládání je potřeba pouze jedno tlačítko.

Kapitola 6
Závěr

Cílem této diplomové práce bylo navrhnout a implementovat navigační aplikaci pro vozíčkáře. Vozíčkáři tvoří velmi specifickou skupinu uživatelů mobilních aplikací. Při jízdě na vozíku jsou do určité míry omezeni v možné interakci s mobilním zařízením, protože musí ruce používat k ovládání vozíku. U mechanických vozíků musí vozíčkáři používat obě ruce k otáčení kol, u elektrických vozíků používají vozíčkáři pouze jednu ruku k ovládání vozíku joystickem. V diplomové práci byla snaha tato omezení zohlednit při návrhu uživatelského rozhraní a systému ovládání aplikace.

Důraz byl v diplomové práci kladen na návrh způsobu ovládání, který by umožňoval využití externího ovladače k interakci s aplikací. Pro tento případ byl navržen systém menu a rychlých akcí. Rychlé akce reprezentují často používané funkce aplikace a mohou být aktivovány ihned. Menu slouží jako kontejner pro méně frekventované akce, k nimž lze přistupovat pouze v případě, že je menu otevřeno. Samotné ovládání je řešeno využitím šesti typů vstupu. Čtyři vstupy reprezentují čtyři směry joysticku, každý směr je mapován na jednotlivé ovládací prvky aplikace. Pátý vstup slouží pro potvrzování akcí, jako je například výběr položky seznamu. Poslední vstup slouží jako náhrada zpětného tlačítka, které se nachází na většině mobilních zařízení ať už ve formě hardwarového, nebo softwarového tlačítka. Za pomoci pouze těchto šesti vstupů lze kompletně ovládat celou aplikaci.

Z pohledu ovládání aplikace byly v práci analyzovány dvě technologie. První techologií je technologie OTG, která umožňuje připojení joysticku k mobilnímu telefonu. Signály z ovladače jsou tak posílány přímo do telefonu a lze na ně ihned reagovat. Druhou technologií je Tecla. Tato technologie je založena na využití speciální klávesnice. Ta je pravidelně skenována a aktuálně vybraný vstup klávesnice je zvýrazněn. Spuštění akce související se zvýrazněným vstupem je mapováno pouze na potvrzovací tlačítko. Protože základní rozložení klávesnice obsahuje mimo jiné i šest vstupů definovaných při návrhu ovládacího systému, může být systém Tecla jednoduše integrován do aplikace.

Aplikace byla zaměřena také na poskytování informací o bezbariérových místech. K tomu byla využita databáze bezbariérových míst, kterou poskytuje Česká asociace paraplegiků prostřednictvím projektu s názvem VozejkMap. Databáze obsahuje pouze místa, o nichž lze s jistotou říci, že jsou bezbariérová, protože tyto informace jsou ověřovány samotnými vozíčkáři. To je velká výhoda oproti jiným aplikacím pro vozíčkáře, které jsou založené na komunitním sbírání dat, a některá místa mohou být nepřesně označena jako bezbariérová.

Pro nalezení trasy a navigačních instrukcí bylo prozkoumáno využití systému Naviterier. Ten poskytuje navigační instrukce pro nevidomé. Instrukce jsou velice podrobné a mají formu přirozeného jazyka. Toho lze využít i pro případ navigace pro vozíčkáře, kdy lze tyto instrukce předčítat vozíčkáři, a ten tak nemusí sledovat obrazovku mobilního zařízení. Systém Naviterier je stále v procesu vývoje, a tak je jeho hlavní nevýhodou omezená oblast, ve níž lze trasy vyhledávat. Navíc nelze vyhledávat trasy mezi dvěma souřadnicemi, ale pouze mezi místy, která obsahuje databáze systému Naviterier. Proto

bylo jako alternativa implemetováno využití Google Maps Directions API. To je využito k nalezení trasy mezi dvěma místy, které jsou definovány svými souřadnicemi.

S aplikací lze interagovat více způsoby. To je zajištěno kromě využití joysticku, také využitím dalších služeb. První službou je rozpoznávání hlasu, které se používá například pro zadávání vyhledávacího řetězce při hledání míst. Druhou službou je převod textu na řeč. Tato služba je využita především při navigaci. Vozíčkář slyší navigační instrukce, a nemusí tak sledovat mapu, aby určil kudy má jet. Převod textu na řeč je využit také pro přehrávání notifikací, které slouží jako krátké informační zprávy v průběhu navigace. Vozíčkář může být upozorněn, pokud vyjede mimo trasu nebo pokud se jeho vozík dostává do nebezpečného náklonu, který by mohl skončit pádem.

V diplomové práci se povedlo navrhnout použitelné rozhraní. Použitelnost byla ověřena cílovou skupinou uživatelů v rámci testování aplikace. Při testování nebyly nalezeny žádné závažné nedostatky, které by bránily používání aplikace. V rámci testování také došlo na srovnání ovládání pomocí joysticku a pomocí klávesnice Tecla. Podle hodnocení participantů byl joystick označen jako rychlejší a celkově lepší způsob ovládání oproti klávesnici Tecla. Skenování klávesnice se participantům zdálo příliš pomalé.

Vzhledem k tomu, že některé funkčnosti aplikace nemají dostatečnou podporu ať už jde o využití technologií, či systémů třetích stran, jsou tyto funkčnosti v aplikaci pouze simulovány. V budoucím rozšíření aplikace by bylo dobré tyto nedostatky odstranit.

Vývoj aplikace by se měl zaměřit na využití infomací o problémech a překážkách na trase. V aplikaci je pro tuto funkčnost vytvořeno uživatelské rozhraní. Problematický je sběr těchto problémů. V tomto ohledu se plánuje zařazení databáze problémů do systému Naviterier tak, aby mohl systém při plánování tras zohledňovat také tyto nové problémy a poskytovat pouze bezproblémové trasy nebo trasy s překážkami, které pouze snižují komfort jízdy, ale nezamezí pokračování v jízdě. Alternativou by mohlo být využití databáze portálu Chodci sobě, který obsahuje řadu problematických míst pro chodce. Tyto informace jsou relevantní i pro vozíčkáře. V současnosti ale portál nenabízí veřejné API pro přístup k databázi problémů.

Aplikace by také mohla umožňovat přidávání nových bezbariérových míst. K tomu by mohlo být využito rozhraní aplikace VozejkMap, které zadávání nových bezbariérových míst umožňuje. Problémem by byla úprava rozhraní aplikace VozejkMap, aby mohla být ovládána pomocí joysticku. Dále by bylo nutné domluvit s projektem VozejkMap poskytnutí veřejného rozhraní pro přidávání míst. Místa by tak mohla být přidávána přímo z aplikace WheelTrip.

Na základě videokonference s konstruktérem elektrických vozíků byly definovány další požadavky na využití senzorů vozíku v aplikaci. V současné době ale není dostupná technologie pro integraci vozíku s mobilním zařízením ani rozhraní, jakým by spolu mohl telefon a vozík komunikovat. Předpokládá se využití technologie Bluetooth k propojení telefonu s vozíkem, protože by bylo nutné připojit i několik externích zařízení (kamery, mikrofon, pohybové senzory). Technologie Bluetooth umožňuje propojení několika zařízení současně.

Literatura

[1] Iezzoni, Lisa I., Ellen P. McCarthy, Roger B. Davis a Hilary Siebens. *Mobility Difficulties Are Not Only a Problem of Old Age*. Dostupné z: http://dx.doi.org/10.1046/j.1525-1497.2001.016004235.x.

[2] Velinská, Pavla. *Analýza přístupnosti objektů osobám se zdravotním postižením v městské části Brno-Bystrc*. Brno: Masarykova univerzita, Pedagogická fakulta, Brno, 2007. Bakalářská práce.

[3] Pidrová, Olga. *Integrace mladých lidí upoutaných úrazem na vozík zpět do společnosti*. Brno: Masarykova univerzita, Pedagogická fakulta, Brno, 2013. Bakalářská práce.

[4] Vyhláška č. 398/2009 Sb., o obecných technických požadavcích zabezpečujících bezbariérové užívání staveb. In: *Sbírka zákonů*. Praha: Tiskárna Ministerstva vnitra, p. o, 2009. částka 129.

[5] Filipiová, Daniela. *Projektujeme bez bariér*. 1. vydání. Praha: Ministerstvo práce a sociálních věcí, 2002. ISBN 80-865-5218-7.

[6] Kudláček, Martin a Ondřej Ješina. *Integrovaná tělesná výchova, rekreace a sport*. 1. vydání. Olomouc: Univerzita Palackého v Olomouci, 2014. ISBN 978-80-244-4374-4. Dostupné z: https://publi.cz/books/156/Cover.html.

[7] Nuc, Martin. *WheelGo – Plánovač tras pro vozíčkáře*. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická. Diplomová práce.

[8] Bokšanský, Jakub. *Automatic generation of route description for visually impaired users*. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická. Diplomová práce.

[9] Loukotová, Klára. *Současná uživatelská rozhraní dialogových informačních systémů*. Praha: Univerzita Karlova v Praze, Filozofická fakulta, Praha, 2006. Diplomová práce.

[10] Milette, Greg a Adam Stroud. *Professional Android Sensor Programming*. 1. vydání. Indianapolis: John Wiley & Sons, Inc., 2012. ISBN 978-1-118-18348-9.

[11] *USB On-The-Go and Embedded Host* [online]. [cit. 2016-03-16]. Dostupné z: http://www.usb.org/developers/onthego/.

[12] *Tecla Access for Android – User Guide v0.4* [online]. [cit. 2016-03-16]. Dostupné z: http://www.komodoopenlab.com/pub/media/pdfs/.

[13] *Naviterier* [online]. [cit. 2016-03-27]. Dostupné z: http://www.naviterier.cz/.

[14] Buxton, William. *Sketching user experiences*. Amsterdam: Morgan Kaufmann Publishers, 2007. ISBN 978-0-12-374037-3.

[15] Arlow, Jim a Ila Neustadt. *UML 2 a unifikovaný proces vývoje aplikací*. 2. aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[16] *Tecla Access* [online]. [cit. 2016-03-28]. Dostupné z: http://mobile-accessibility.idrc.ocadu.ca/projects/tekla/developers/tecla-for-android-developers.

[17] *Making Applications Accessible* [online]. [cit. 2016-03-28]. Dostupné z: http://developer.android.com/guide/topics/ui/accessibility/apps.html.

[18] *Google Guice on Android, version 3.0* [online]. [cit. 2016-03-30]. Dostupné z: https://github.com/roboguice/roboguice.

[19] *Crashlytics* [online]. [cit. 2016-03-30]. Dostupné z: http://try.crashlytics.com/sdk-android/.

[20] *Otto* [online]. [cit. 2016-03-30]. Dostupné z: http://square.github.io/otto/.

[21] *Retrofit* [online]. [cit. 2016-03-30]. Dostupné z: http://square.github.io/retrofit/.

[22] *Espresso* [online]. [cit. 2016-04-17]. Dostupné z: https://google.github.io/android-testing-support-library/docs/espresso/.

[23] Goodman, Elizabeth, Mike Kuniavsky a Andrea Moed. *Observing the user experience*. 2. vydání. Waltham: Morgan Kaufmann, c2012. ISBN 978-0-12-384869-7.

[24] Annuzzi, Joseph, Lauren Darcey a Shane Conder. *Advanced android application development*. 4. vydání. Upper Saddle River, NJ: Addison-Wesley, 2015. ISBN 978-013-3892-383.

Příloha A
Zkratky

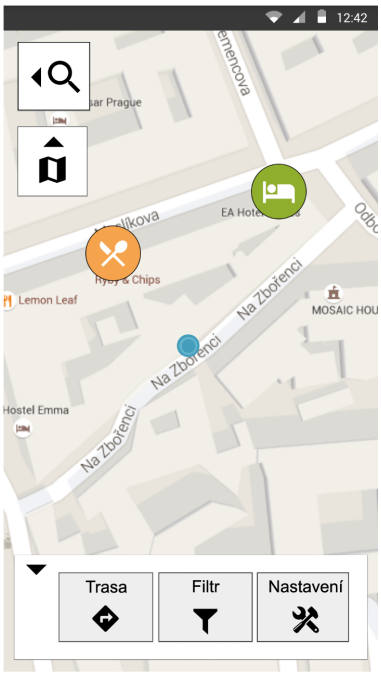
API	Application Programming Interface
CSV	Comma-Separated Values
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JSV	Java Structure Viewer
POI	Place of Interest
SDK	Software Development Kit
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

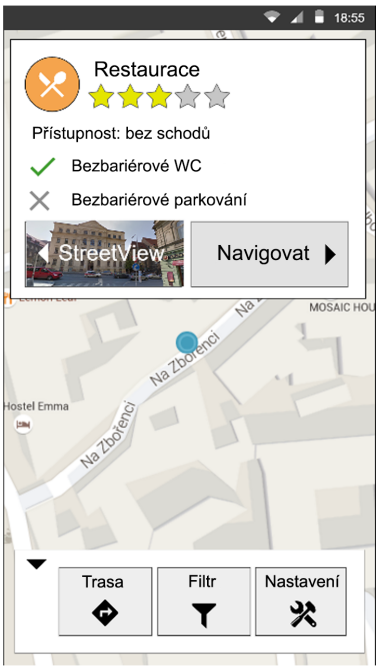
Příloha B
Obsah přiloženého CD

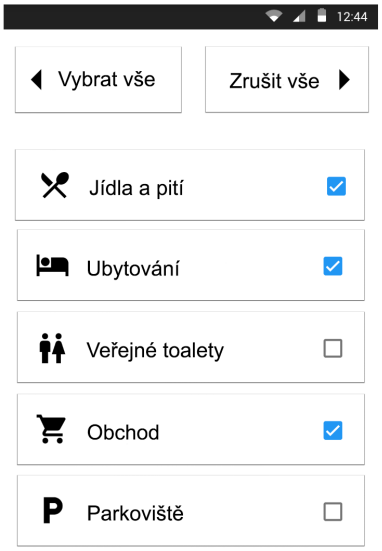
Na přiloženém CD lze nalézt čtyři složky:

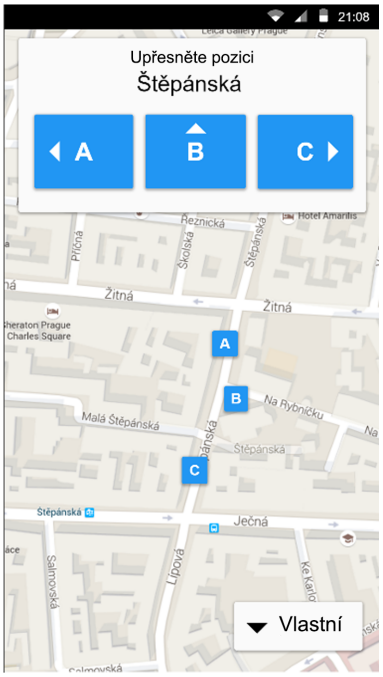
- `wheeltrip` – obsahuje zdrojové kódy aplikace
- `apk` – obsahuje instalační soubor `wheeltrip.apk`
- `dp` – obsahuje soubor PDF této diplomové práce
- `dp-src` – obsahuje zdrojové kódy a obrázky pro sazbu diplomové práce

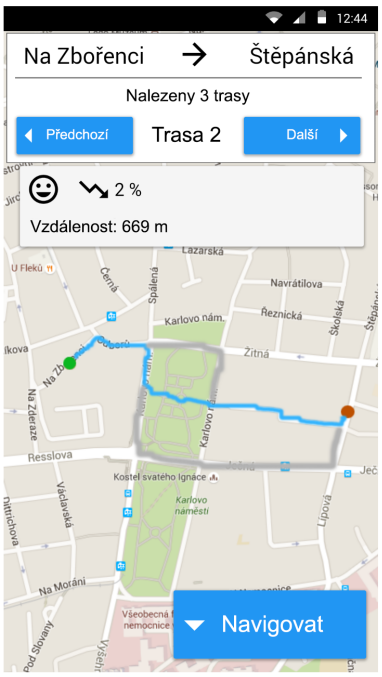
Příloha C
Lo-fi prototyp

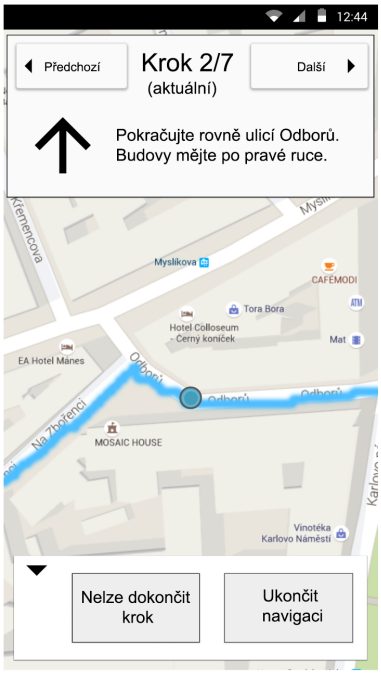


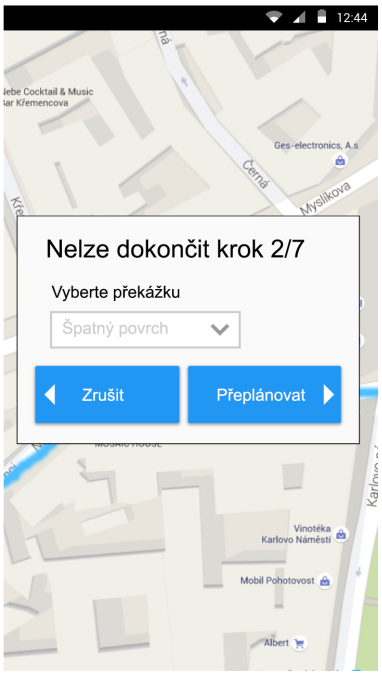


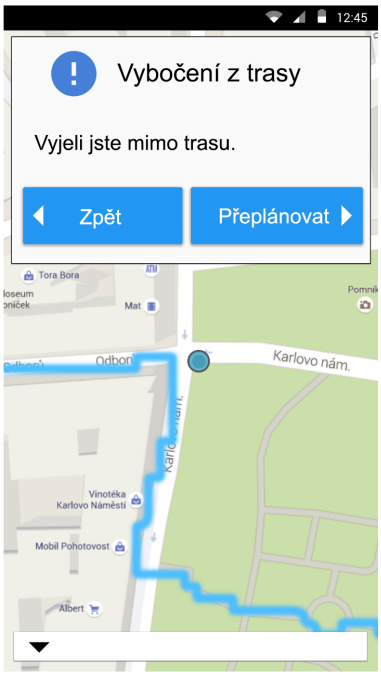





12:44
Na Zbořenci
Štěpánská
Nalezeny 3 trasy
Předchozí
Trasa 2
Další
2 %
Vzdálenost: 669 m
Navigovat


12:44
Předchozí
Krok 2/7
(aktuální)
Další
Pokračujte rovně ulicí Odborů.
Budovy mějte po pravé ruce.
Nelze dokončit krok
Ukončit navigaci

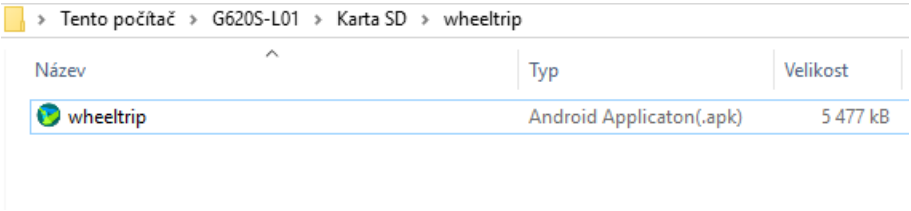

12:44
Nelze dokončit krok 2/7
Vyberte překážku
Špatný povrch
Zrušit
Přeplánovat


12:45
Vybočení z trasy
Vyjeli jste mimo trasu.
Zpět
Přeplánovat

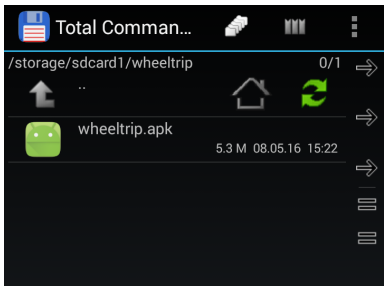
Příloha D
Instalační příručka

Instalace aplikace WheelTrip je velmi jednoduchá. K instalaci je nutné mít v mobilním zařízení nainstalovanou aplikaci Total Commander nebo jiný prohlížeč souborů.

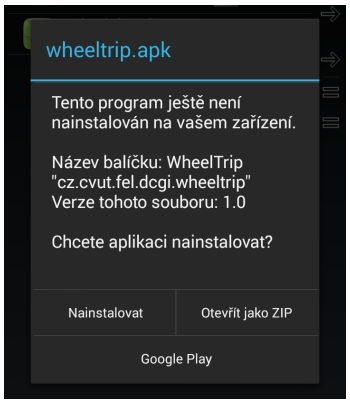
1. Připojte mobilní zařízení k PC. Vytvořte na disku ve svém mobilním zařízení složku `wheeltrip` a zkopírujte do ní soubor `wheeltrip.apk`.



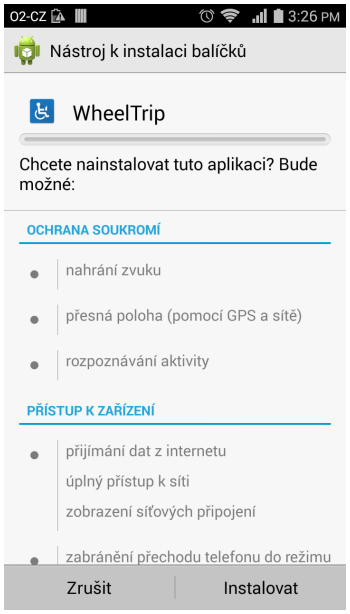
2. V prohlížeči souborů vyhledejte složku `wheeltrip` a klepněte na soubor `wheeltrip.apk`.



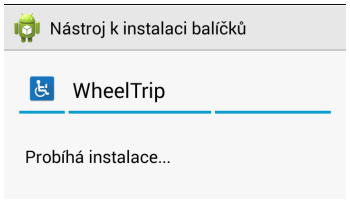
3. Zobrazí se dialogové okno. Klepněte na tlačítko „Nainstalovat“.



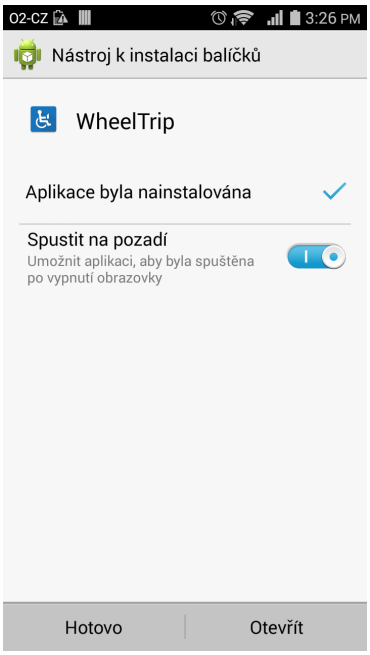
4. Zobrazí se obrazovka s informacemi o oprávnění. Klepněte na tlačítko „Instalovat“.



5. Vyčkejte na dokončení instalace.



6. Po dokončení instalace můžete aplikaci spustit. Klepněte na tlačítko „Otevřít“.



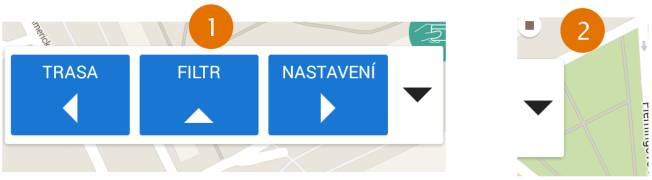
7. Pro správnou funkčnost aplikace je potřeba v telefonu povolit sledování polohy a mít připojení k internetu. Při prvním startu aplikace je nutné připojení přes Wi-Fi, aby mohla být stažena data VozejkMap a Naviterier.

Příloha E
Uživatelská příručka

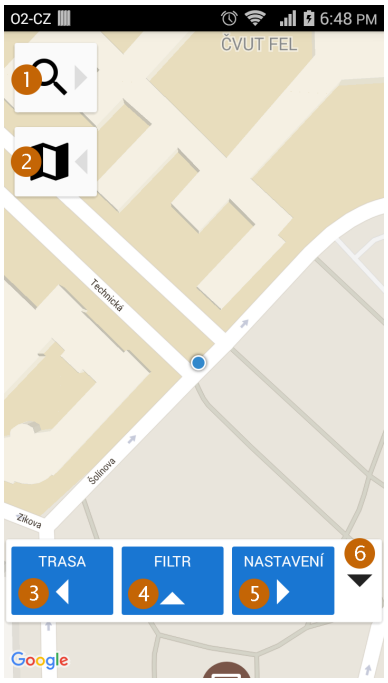
Aplikace WheelTrip je navržena tak, aby ji bylo možné ovládat pomocí joysticku. Aplikace dokáže reagovat na šest možných vstupů. Čtyři vstupy odpovídají čtyřem pozicím joysticku – nahoru, dolů, doprava, doleva. Zbývající dva vstupy jsou jsou rezervovány pro zpětné tlačítko (to odpovídá zpětnému tlačítku na dotykových telefonech) a pro potvrzovací tlačítko.

V aplikaci má každý ovládací prvek v uživatelském rozhraní u sebe zobrazenou směrovou šipku. Šipka udává do jakého směru je nutné pohnout joystickem, aby došlo ke spuštění odpovídající akce.

Aplikace využívá systém menu a rychlých akcí. Rychlé akce jsou spuštěny ihned při změně směru joysticku. Menu obsahuje další rychlé akce, avšak aby mohly být tyto akce spuštěny, je nutné, aby bylo menu otevřené (1). Zavřené menu (2) umožní aktivaci rychlých akcí ve zbylé části obrazovky.

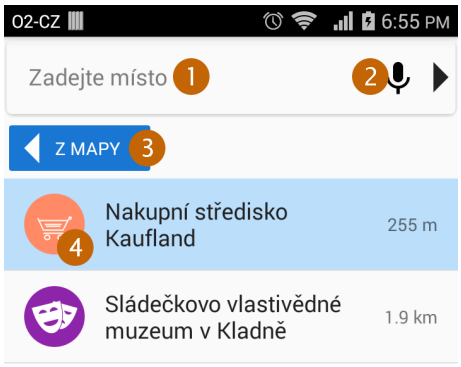


Hlavní obrazovka obsahuje rychlé akce pro vyhledávání míst (1) a režim mapy (2). V menu jsou poté dostupné další akce – pro vyhledání trasy (3), filtr míst (4) a nastavení aplikace (5). Menu se otevírá i zavírá pohybem joysticku směrem dolů (6).

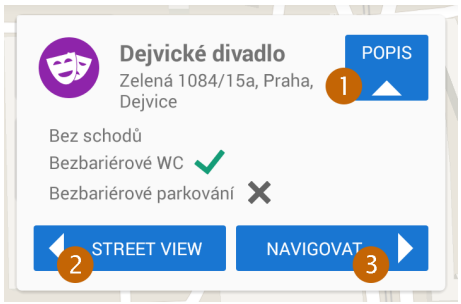


Režim mapy slouží k výběru místa na mapě nebo k prohlížení mapy. Na mapě se lze pohybovat do čtyř směrů pomocí joysticku. K výběru místa slouží potvrzovací tlačítko. Režim mapy lze ukončit stisknutím zpětného tlačítka.

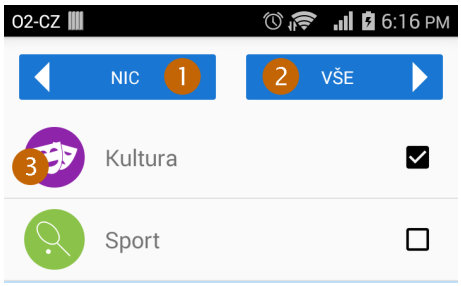
Na vyhledávací obrazovce lze zadat místo textově (1), hlasově (2) nebo z mapy (3). Nalezená položka (4) obsahuje ikonu typu místa, název místa a vzdálenost k místu z vaší aktuální pozice. Pohybem joysticku směrem nahoru a dolů se lze v seznamu posouvat. K výběru místa slouží potvrzovací tlačítko.



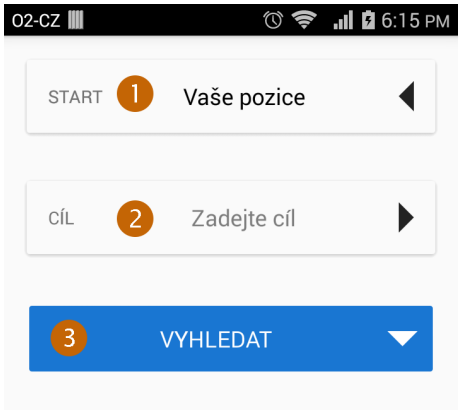
Detail nalezeného místa poskytuje informace o místu. Lze zobrazit podrobnější popis místa (1). Místo lze zobrazit také v režimu StreetView (2) nebo je možné nalézt trasu k místu z vaší aktuální pozice (3).



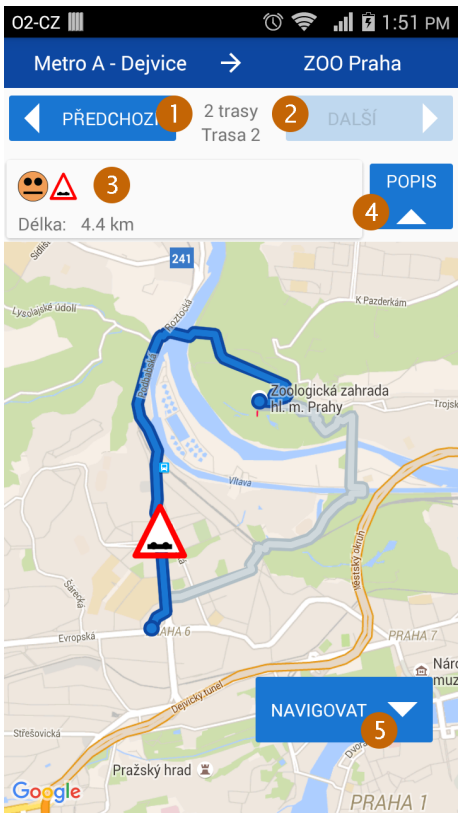
Filtr míst slouží k označení míst, která budou nabízena při vyhledávání míst a která budou zobrazena na mapě. Na obrazovce filtru míst lze zrušit (1) nebo vybrat (2) všechna místa. Jednotlivé položky seznamu (3) obsahují ikonu typu místa, název místa a zaškrtávací políčko. Pokud je políčko zaškrtnuto, znamená to, že dané místo bude použito. V seznamu se lze posouvat pohybem joysticku směrem nahoru a dolů, ke změně výběru jednotlivých položek použijte potvrzovací tlačítko.



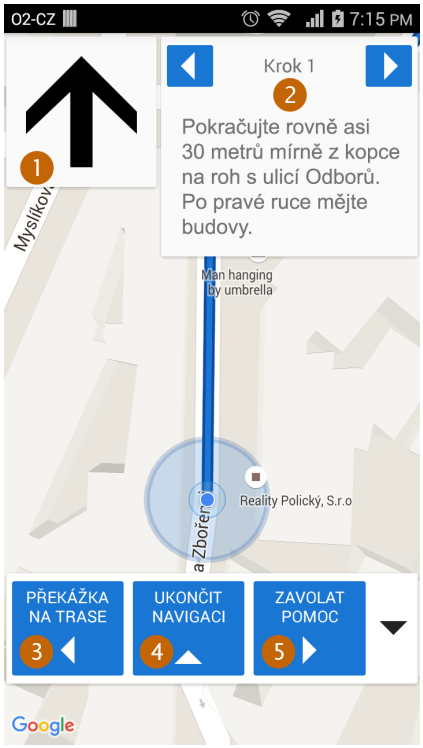
Obrazovka pro hledání trasy slouží především v případech, kdy chcete zadat vlastní počáteční bod trasy (1). Poté stačí zadat koncový bod trasy (2) a spustit plánování trasy (3).



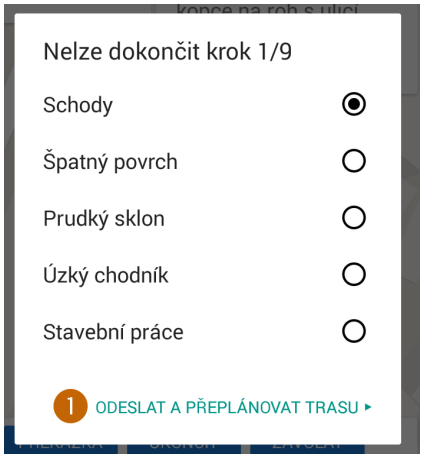
Nalezená trasa je zobrazena na mapě. Pokud je nalezeno více tras, lze se přepnout na předchozí (1) nebo následující (2) trasu. Aktuálně vybraná trasa je na mapě zvýrazněna. Na obrazovce jsou zobrazeny základní informace o trase (3) – celková obtížnost trasy reprezentována smajlíkem, ikony problémů na trase a délka trasy. Dále lze zobrazit podrobnější informace o trase (4). Akce *Navigovat* (5) slouží ke spuštění navigace.



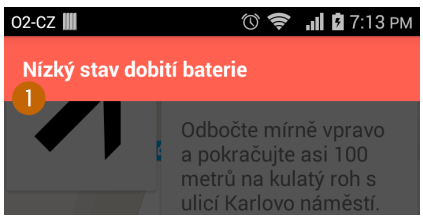
Na navigační obrazovce je zobrazena ikona směru aktuální instrukce (1) a také text samotné instrukce (2). Menu obsahuje akce pro označení nepřekonatelné překážky (3), pro ukončení navigace (4) a pro odeslání vaší pozice v případě problémů (5).



K označení překážky slouží dialogové okno se seznamem možných překážek. V seznamu se lze pohybovat nahoru a dolů, k označení problému slouží potvrzovací tlačítko. Po označení problému lze trasu přeplánovat (1).



V rámci navigace může nastat nečekaná událost. Na tuto událost vás upozorní notifikace zobrazená v horní části obrazovky (1). Tato notifikace má pouze informační charakter, pro zavření notifikace použijte potvrzovací tlačítko.



Příloha F
Uživatelské testování – úkoly

1. Jdete s kamarádem do restaurace Retro. Nalezněte restauraci a zjistěte o ní nějaké podrobnosti.
2. Zjistěte, co se nachází napravo od restaurace.
3. Na mapě se zobrazuje spousta parkovišť, vy ale nevlastníte automobil, zajistěte, aby se na mapě parkoviště nezobrazovala.
4. S kamarádem máte sraz u výstupu z metra Náměstí Míru, který je blízko restaurace. Nalezněte trasu z vaší aktuální pozice k výstupu z metra.
5. Vyberte si nejkratší trasu. Jaká je její obtížnost?
6. Nyní se můžete vydat na cestu.
7. Pokud narazíte na překážku, označte toto problémové místo.
8. Postupujte podle instrukcí a dojeďte až do cíle.

Příloha G
Použité závislosti

Všechny závislosti aplikace WheelTrip jsou definovány v části `dependencies` souboru `wheeltrip/app/build.gradle`.

Název	Verze	Popis
Android Support Library v4	23.0.1	Zpětná kompatibilita pro API od verze 4.
Android Support Library v7	23.0.1	Zpětná kompatibilita pro API od verze 7.
Apace Commons Lang 3	3.0	Knihovna pro práci se standardními třídami Javy.
CardView v7	23.0.1	Kartové widgety.
Crashlytics Android SDK	2.5.5	Hlášení chyb při pádu aplikace.
Espresso	2.2.1	Testovací framework pro Android.
Google Maps utility library	0.3.4	Rozšiřující knihovna pro Google Maps API.
Google Play Services	8.4.0	Klient pro komunikaci se službami Google.
Gson	2.5	Knihovna pro převod objektů do/z formátu JSON.
JUnit	4.12	Framework pro tvorbu unit testů.
OkHttp	2.7.0	Klient HTTP a HTTP/2 pro Android.
Otto	1.3.8	Event bus pro Android.
RecyclerView v7	23.0.1	Tvorba seznamů s velkým objemem dat.
FlexibleDivider v7	1.2.6	Vizuální oddělení položek seznamu.
Retrofit	2.0.0-beta2	Typově bezpečný klient HTTP pro Android.
Retrofit Gson Converter	2.0.0-beta2	Konvertor pro Retrofit.
RoboBlender	3.0.1	Anotační procesor.
RoboGuice	3.0.1	Dependency injection framework pro Android.
Support Annotations	23.0.1	Podpora anotací pro Android.

Tabulka G.1 Přehled závislostí aplikace WheelTrip