ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Roadbook aplikace pro OS Android
Student:	Bc. Jakub Chlup
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Navrhněte a implementujte mobilní aplikaci pro OS Android, která bude sloužit pro rally a off-road navigaci pomocí roadbook. Roadbook bude v aplikaci možné též vytvořit.

Aplikace bude minimálně umožňovat:
- vytvoření a úpravy roadbook
- navigaci na trase pomocí roadbook
- měření projeté vzdálenosti pomocí připojení na OBD-II (On-Board Diagnostics) nebo GPS
- zaznamenávání statistik absolvovaných tras
- import a export roadbook

Postupujte v těchto krocích:
- proveďte rešerši existujících řešení
- navrhněte aplikaci včetně uživatelského rozhraní
- navrhněte a následně realizujte vhodné testování - minimálně akceptační, uživatelské a unit testy
- implementujte výslednou aplikaci, dle návrhu ji řádně otestujte
- aplikaci nasaďte na Google Play
- zhodnoťte vytvořenou aplikaci a navrhněte možná vylepšení, shrňte případné ohlasy z Google Play

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. října 2017

České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra softwarového inženýrství

Diplomová práca

Roadbook aplikace pro OS Android

Bc. Jakub Chlup

Vedúci práce: Ing. Jiří Hunka

8. januára 2018

Poďakovanie

Rád by som poďakoval vedúcemu práce Ing. Jiřímu Hunkovi za cenné rady počas písania práce. Tiež by som rád poďakoval autorovi nápadu Ing. Miroslavovi Jančovičovi za užitočnú spätnú väzbu a nadšenie pre myšlienku. Za finálnu korektúru tohto textu ďakujem Bc. Barbore Jančovičovej. A v neposlednej rade vďačím svojej rodine za motiváciu a podporu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 8. januára 2018 .....

Odkaz na túto prácu

Chlup, Jakub. *Roadbook aplikace pro OS Android.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Táto diplomová práca sa zaoberá vývojom aplikácie Roadbook pre OS Android. Aplikácia slúži na vytvorenie roadbook a jeho inštrukcií, pričom ponúka dve možnosti vytvárania a to online priamo v teréne, alebo offline. Vytvorený roadbook je možné použiť k navigácii, buď s použitím GPS, alebo zariadenia ELM327 pripojeným na riadiacu jednotku vozidla.

Klíčová slova Roadbook, navigácia, aplikácia, OS Andorid, OBDII, GPS, ELM327

Abstract

This final thesis is concerned with development of application Roadbook for Android OS. Application can be used for creating roadbook and list of its instructions. There are two options for creating roadbook – online creation in field or offline creation. Created roadbook can be used for navigation using either GPS or ELM327 device connected to engine control unit.

Keywords Roadbook, navigation, application, OS Android, OBDII, GPS, ELM327

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V tejto diplomovej práci su budem zaoberať vývojom softwerového riešenia pre roadbook. Nápad na vytvorenie takéhoto systému dostal môj známy, motorkár, spolu s ktorým som priebeh vývoja pravidelne konzultoval. Nápad dostal pri prezeraní portálu motoride.sk, kde veľa užívateľov rieši ako si vyrobiť mechanický či elektrický roadbook, tripmaster a iné potrebné zariadenia k navigácii podľa roadbook. Cena profesionálnych riešení a súprav zariadení sa pohybuje v stovkách eur, čo je pre nadšencov a hobby jazdcov jednoducho veľa. Preto sa na spomínanom portáli dajú nájsť návody ako si rolku papiera s inštrukciami pripevniť a zavrieť do obedovej nádoby, ako používať cyklopočítadlo, alebo ako si pomocou Arduina s displejom a magnetickým snímačom vytvoriť tripmaster na meranie prejdenej vzdialenosti.

Známy si vytvoril vlastný mechanický roadbook a pri prispôsobovaní motorky a následnom použití zistil, že použitie je komplikované, aj napriek tomu, že množstvo z ostatných motorkárov na toto riešenie nedá dopustiť. Väčšinu potrebných funkcií zvládne moderné mobilné zariadenie, ktorých ceny sa síce tiež pohybujú v stovkách eur. No na dané účely postačia aj tie lacnejšie modely, ktorých cena bude stále niekoľkokrát menšia ako cena špeciálnej výbavy a ponúknu viacero spôsobov využitia.

Vybraná bola natívna aplikácia pre operačný systém Andorid, nakoľko sa vývoju takýchto aplikácií venujem profesne už niekoľko rokov. Taktiež moja bakalárska práca bola aplikácia pre OS Android zaoberajúca sa získavaním dát z riadiacej jednotky vozidla, ktoré graficky spracováva spolu so vstupom z kamery zariadenia do videa.

Na úvod ešte ozrejmím čo je to vlastne roadbook. Zjednodušene povedané ide o zoznam inštrukcií, podľa ktorých sa dá navigovať v akomkoľvek teréne. Preto sa v rôznych formách používa pri športových podujatiach ako sú rally, off-road závody, či iné navigačné preteky. Môžme ho prirovnať k inštrukciám o ceste ktoré dostaneme keď nám niekto vysvetľuje cestu ako sa dostať na požadované miesto.

Cieľom práce je teda vývoj android aplikácie, ktorej hlavnými funkciami bude zostrojenie roadbook, jeho inštrukcií a navigácia na trase popísanej v ňom. Aplikácia bude poskytovať dva rôzne spôsoby merania prejdenej vzdialenosti, a to pomocou sledovania polohy zariadenia (GPS) alebo vďaka dátam získaným z riadiacej jednotky vozidla. Tým aplikácia nahradí zariadenia ako odometre a tripmastery. Dodatočnou funkciou aplikácie bude zaznamenávanie dát počas jazdy a zobrazenie štatistík prejdenej trasy. Medzi ďalšie funkcie patrí: čítanie hlasových povelov k inštrukciám, možnosť exportovania a importovania dát roadbook do a z XML súboru a kreslenie vlastných nákresov situácií.

Vymedzenie pojmov

Roadbook je itinerár, zjednodušene zoznam, tabuľka inštrukcií, pomocou ktorých je možná navigácia zo začiatočného bodu do cieľa. Najčastejšie sa používa pri automobilových a motocyklových terénnych športoch, kde je kľúčovým faktorom správna navigácia v teréne. V itinerári sú uvedené najmä jednotlivé zmeny na trase ako odbočky, zmeny smeru, upozornenia na prekážky, a iné upresnenia potrebné pre správnu orientáciu.

Inštrukcia roadbook, skrátene inštrukcia je jeden riadok roadbook, a teda jedna zmena na trase. Inštrukcia môže obsahovať celkovú a čiastočnú vzdialenosť, po ktorej bude nasledovať daná zmena. Zmena v navigácii sa najčastejšie zobrazuje nákresom križovatky alebo situácie, ktorá nasleduje. Ďalej môže byť doplnená textovým popisom alebo smerovaním na kompase.

Tulip nákres zmeny, križovatky, upozornenia pre inštrukciu v roadbook.

OBDII protokol slúžiaci na diagnostiku a prístup k jednotlivým systémom motorových vozidiel ako aj na získavanie real time dát počas jazdy. Protokol je definovaný normami ISO-9141, J1962, J1850 a ISO-15765. Tieto normy kladú požiadavky na každé vyrobené vozidlo za účelom digitálnej diagnostiky systémov vozidla, určujú rad štandardných chybových kódov, a taktiež zavádzajú jednotnú 16-pinovú diagnostickú zásuvku, ktorá musí byť v každom vozidle umiestnená a prístupná z miesta vodiča.

ELM327 je zariadenie, ktoré umožňuje komunikáciu s diagnostickou jednotkou vozidla cez OBDII protokol. Toto zariadenie sa zapája do diagnostickej zásuvky vo vozidle, je schopné komunikovať s externými zariadeniami pomocou bluetooth, alebo wifi pripojenia.

Odometer je vo všeobecnosti pomenovanie pre nástroj slúžiaci na meranie prejdenej vzdialenosti. Toto zariadenie môže byť elektrické, mechanické, prípadne ich kombinácia, tiež spôsob zobrazenia a použité jednotky môžu byť rôzne.

Tripmaster sú zariadenie podobné odometru, takže ich hlavným účelom je meranie prejdenej vzdialenosti, no často ponúkajú, zobrazujú aj ďalšie dáta ako napríklad aktuálnu rýchlosť, čas, či možnosť manuálne upraviť prejdenú vzdialenosť. Sú preto často používané pri navigácií podľa roadbook. Existujú rôzne metódy ako tieto zariadenia merajú prejdenú vzdialenosť, napríklad dvoma senzormi, kedy jeden je umiestnený na kolese a druhý na ráme na ktorý je koleso pripevnené, tak aby sa pri jednej otočke kolesa senzory raz stretli. Pri takomto spôsobe merania je ešte do tripmasteru nutné zadať obvod kolesa. Novšie modely ponúkajú meranie prejdenej vzdialenosti na základe sledovania GPS polohy.

WebView pod týmto pojmom sa v tejto práci rozumie UI komponent slúžiaci na zobrazenie webových stránok v rámci aplikácií.

Android SDK alebo skrátene iba SDK (Softwere development kit), je súbor nástrojov na vývoj pre operačný systém android.

Manifest – Android Manifest, je súbor poskytujúci základné informácie o aplikácii, ktoré systém potrebuje ešte pred tým, ako ju môže spustiť. Obsahuje údaje o názve balíku aj samotnej aplikácie, jej komponent, oprávnenia, ktoré požaduje, a vlastnosti zariadenia, ktoré pre svoj beh aplikácia požaduje.

Kotlin je programovací jazyk, ktorý je po Jave druhým oficiálne podporovaným jazykom pre vývoj android aplikácií.

Google I/O je každoročne sa konajúcim podujatím, konferenciou usporiadanou spoločnosťou Google určenou pre vývojárov.

Kapitola 1

Analýza

V tejto kapitole sa diplomová práca bude postupne zaoberať analýzou existujúcich riešení pre roadbook, ich výhodami, ale aj nedostatkami. Ďalej bude definovaná skupina užívateľov systému a pre lepšiu predstavu sa vytvoria aj dve persony. Nasledujúcim krokom bude spísanie zoznamu požiadaviek navrhovanej aplikácie a uvedenie jej prípadov použitia. Na koniec analýzy sa vytvoria analytické triedy, ktoré vzišli z predchádzajúcich krokov.

1.1 Analýza existujúcich riešení

Rešerše existujúcich riešení problému, ktorým sa zaoberá vyvíjaná aplikácia, pomôžu k zisteniu potrieb, nárokov a samotných požiadaviek na systém. Pre širší záber a hlavne predstavu o používaných formách roadbook budú uvedené aj iné ako softwerové riešenia. Cieľom je rozšírenie vedomostí o roadbook a navigácií pomocou takto spísanej trasy.

1.1.1 Papierová forma

Papierová podoba roadbook je stále jedna z najpoužívanejších foriem, v ktorých sú roadbook pri jazde terénom používané. Pri jazde so spolujazdcom je roadbook, teda jeho zoznam inštrukcií, vytlačený na papiere zvlášť, nakoľko spolujazdec má čas v nich listovať a sledovať presne, kde v zozname sa vozidlo práve nachádza. No pri jazde bez spolujazdca, alebo na motorke je táto podoba nepoužiteľná. Preto bol vytvorený takzvaný mechanický roadbook a neskôr aj elektrický.

1.1.1.1 Mechanický roadbook

Jedná sa o špeciálne zariadenie, ktoré sa najčastejšie používa na motorkách. Skladá za z dvoch osí a tela, do ktorého sú tieto osi pripevnené. Papierový roadbook je na začiatku celý namotaný na jednej z osí, to, či sa absolvované

zmeny na trase budú natáčať smerom dole alebo hore, záleží na usporiadaní inštrukcií. Na boku tela zariadenia presahujú osi tak, aby ich bolo možné manuálne otáčať a tým sa presúvať na ďalšie, alebo sa vrátiť na predošlé inštrukcie. Pri dlhých alebo náročných závodoch sa stáva, že papierový roadbook je tak dlhý, že ho nie je možné celý natočiť na os. V takom prípade je nutné osi s roadbook počas jazdy vymeniť. Výrobcov a teda aj prevedení mechanických roadbook je veľa, líšia sa spôsobom uchopenia, veľkosťou, odolnosťou prevedenia a použitých materiálov, niektoré ponúkajú aj podsvietenie. Výrobcovia často spolu so samotným zariadením predávajú aj papier – rolky, na ktoré sa roadbook vytlačia, a ktoré do daného zariadenia pasujú.

Nevýhodou takéhoto riešenia, okrem manuálneho – mechanického prepínania inštrukcií, je aj náročnosť vytvorenia a najmä vytlačenia roadbook na rolku papiera. Ďalším záporom môže byť cena zariadenia a jeho príslušenstva.

1.1.1.2 Elektrický roadbook

Elektrický roadbook je vylepšením toho mechanického. Tak ako názov napovedá, otáčanie osí už nie je ručné, ale zariadenie obsahuje elektromotory, ktoré otáčajú osi. Posunutie sa na ďalšiu/predošlú inštrukciu je možné tlačidlami umiestnenými buď na tele zariadenia alebo na externom ovládači, ktorý je možné pripojiť na riadidlá motorky.

Nevýhody sú v podstate totožné s mechanickým roadook, ďalej je na týchto zariadeniach náročné a veľmi dôležité nastavenie veľkosti posunutia tak, aby sa inštrukcie správne odvíjali.

1.1.2 Digitálny roadbook

Opäť ide o špeciálne zariadenie. Na rozdiel od mechanického a elektrického roadbook už neobsahuje rolku papiera a ani žiadne mechanické časti, či elektrické motory. Na zobrazenie roadbook slúži digitálny displej.

Výrobcov digitálnych roadbook je veľa, podobne ako pri predošlých spomínaných zariadeniach, líšia sa vo funkciách, použitých typoch displejov, ich veľkostiach, použitých materiáloch, odolnosti voči vode, či prachu. Preto budú ďalej popísané ich spoločné a najčastejšie sa vyskytujúce charakteristiky. Roadbook, teda zoznam jeho inštrukcií, sa do zariadenia ukladá cez dátový kábel z PC alebo na pamäťovej karte. Na vytvorenie samotného roadbook ponúkajú výrobcovia vlastný softwere, alebo webovú aplikáciu. Niektoré modely sú dodávané s operačným systémom Microsoft Windows CE a teda je na nich možné spustiť aj iné aplikácie, ako tie od dodávateľa zariadenia.

Ovládanie digitálnych roadbook je na každom zariadení odlišné, niektoré ponúkajú len niekoľko najnutnejších tlačidiel, iné obsahujú dotykový displej. Časté sú aj externé ovládače pripojené káblom, alebo pomocou bluetooth spojenia, ktorými je možné prepínanie inštrukcií, alebo úprava prejdenej

vzdialenosti. Pri novších modeloch je častá prítomnosť GPS technológie a aj mapových podkladov.

Nevýhodami týchto zariadení, okrem ich ceny, ktorá nebýva práve nízka, je uzavretosť systému, ťažká alebo nemožná úprava roadbook v teréne.

1.1.3 Softwerové roadbooky

Pod pojmom softwerový roadbook sá dá rozumieť akýkoľvek softwere na vytvorenie a navigáciu pomocou roadbook. Na vytváranie itinerárov existuje nespočetné množstvo aplikácií, najmä tých webových. Najčastejšie ponúkajú vytvorenie trasy na mapovom podklade, pridanie dát k jednotlivým bodom a následne export zoznamu inštrukcií do PDF, alebo iného formátu určeného na vytlačenie na papier.

Keďže vytváraná aplikácia bude určená na mobilné zariadenia, bližšie sa pozrieme na existujúce aplikácie určené na tieto zariadenia.

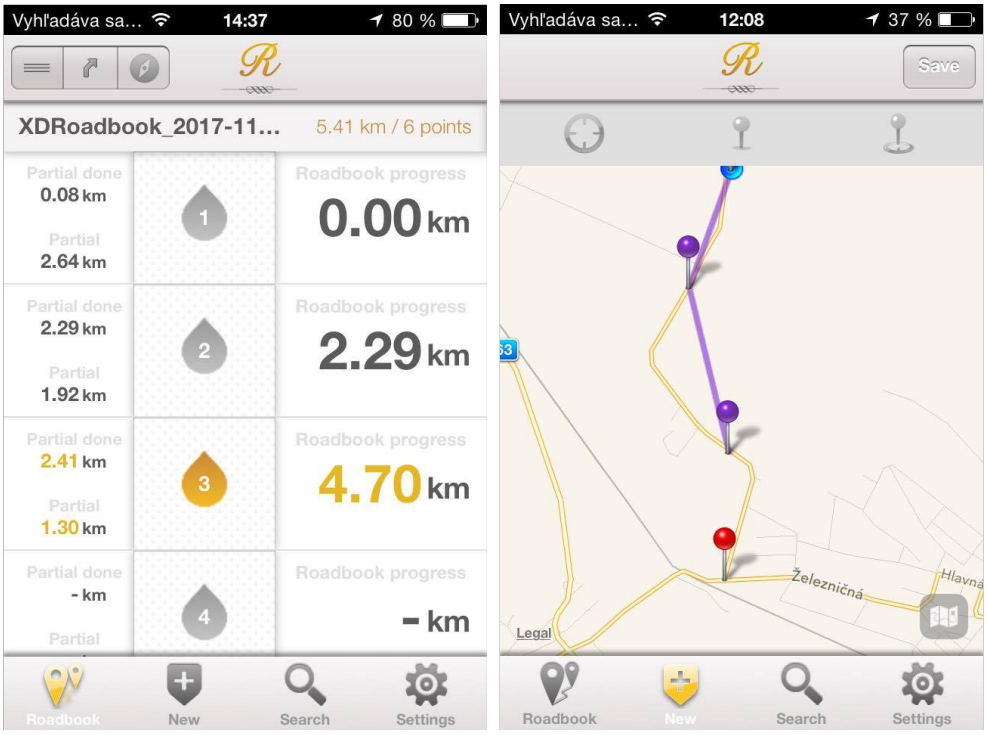
1.1.3.1 XD Roadbook

XD Roadbook je aplikácia pre iOS, v ktorej je možné vytvárať vlastné roadbook a potom sa podľa nich navigovať. Táto aplikácia nepodporuje pridávanie tulip k bodom, body je možné popísať iba textovo. Na hlavnej stránke užívateľ vidí zoznam vytvorených roadbook zoradených podľa času vytvorenia. V tomto zozname je možné vyhľadávať podľa niekoľkých kritérií.

V nastaveniach je možné určiť jednotkovú sústavu, presnosť GPS a vzdialenosť od bodu, v ktorej sa navigácia prepne na ďalší bod (10, 50 alebo 100 metrov). Je možné tiež vypnúť automatické zamykanie obrazovky a zobraziť informácie o aplikácii.

Vytváranie roadbook prebieha pomocou označovania bodov na mape. Aplikácia najprv nájde a zobrazí aktuálnu polohu užívateľa na mape, je možné zvoliť obyčajnú alebo satelitnú mapu. Na obrazovke pri vytváraní roadbook sa nachádzajú tri možnosti: nájsť polohu užívateľa, pridať bod a pridať bod na aktuálnej polohe. Roadbook je teda možné vytvárať prechádzaním po trase a postupným pridávaním bodov, alebo z jedného miesta posúvaním mapy a pridávaním bodov. Druhý spôsob je však o niečo komplikovanejší, keďže aplikácia vždy pridá bod doprostred obrazovky a jeho polohu je často nutné upraviť posunutím špendlíka na správne miesto. K bodu je možné pridať popis a komentár. Aplikácia ku každému bodu priradí zodpovedajúce GPS súradnice, vypočíta vzdialenosti medzi bodmi, celkovú dĺžku trasy a spojí body vzdušnou čiarou.

Pri navigácií pomocou vytvoreného roadbook aplikácia zobrazuje niekoľko informácií. Hlavná obrazovka pri navigácii zobrazuje kompas, ktorý užívateľa naviguje k nasledujúcemu bodu a taktiež ukazuje smerovanie od nasledujúceho bodu k bodu po ňom. Toto smerovanie aspoň čiastočne nahradzuje použitie tulip, užívateľ si ho však nemôže zvoliť, aplikácia ho pridá sama a zobrazí sa



Obr. 1.1: Ukážky aplikácie XD Roadbook.

až pri navigácii, nie pri vytváraní roadbook. V dolnej lište je možné prepínať medzi vzdialenosťou k ďalšiemu bodu, celkovou vzdialenosťou do cieľa, aktuálnou rýchlosťou, aktuálnou GPS lokáciou, súradnice aktuálneho bodu a popis a komentár k aktuálnemu bodu, ktoré zadal užívateľ. Taktiež je možné zobraziť prehľad všetkých bodov s prehľadom postupu. Užívateľ musí túto možnosť zvoliť v menu, ktoré sa zobrazí po kliknutí na obrazovku počas navigácie. Z tohto prehľadu je možné manuálne sa posunúť na ďalší bod, ak aplikácia z nejakého dôvodu nepreskočí na ďalší bod sama. Taktiež je možné zobraziť mapu s vyznačenými bodmi a aktuálnou polohou zariadenia.

Nevýhodou tejto aplikácie je, že nemá podporu pre najnovšiu verziu iOS 11. Ukážka aplikácie je na obrázku 1.1.

1.1.3.2 Rabbit roadbook

Rabbit roadbook je aplikácia pre android, ktorá je rozdelená na dve samostatné aplikácie. Rabbit roadbook designer a Rabbit roadbook. Užívateľ si najprv vytvorí svoj roadbook v prvej aplikácií, ktorá napokon vygeneruje súbor, ktorý je nutné poslať ďalej spoločnosti spolu s požiadavkou o vytvorenie roadbook pre druhú aplikáciu, ktorá bude potom schopná užívateľa navigovať.

Táto služba je samozrejme spoplatnená, výška poplatku závisí od dĺžky trasy.

Pri návrhu vlastného roadbook užívateľ postupne vytvára trasu počas toho, ako ňou skutočne prechádza. Iný spôsob vytvorenia nie je možný. Užívateľ zvolí pridanie nového bodu a aplikácia zaznamená jeho aktuálnu GPS polohu, užívateľ si potom môže vybrať z množstva tulipov, ktoré chce k aktuálnemu bodu pridať, vrátane ďalších dodatočných ikon, značiek, či doplnkového textu. Užívateľ tiež môže nakresliť vlastný tulip, odfotiť ho, nahrať ako obrázok, pridať hlasový povel alebo skombinovať viac možností. Ku každému bodu je tiež možné pridať dodatočnú informáciu, možnosti sú podobné ako pri pridávaní prvého tulip. K bodom je možné pridať priemernú rýchlosť na úseku alebo časový údaj, za ktorý je nutné daný úsek prejsť, aby bol dosiahnutý ideálny čas. Aplikácia pre navigáciu potom tieto údaje spracováva a zobrazuje užívateľovi, či má zrýchliť alebo spomaliť, aby daný čas dosiahol.

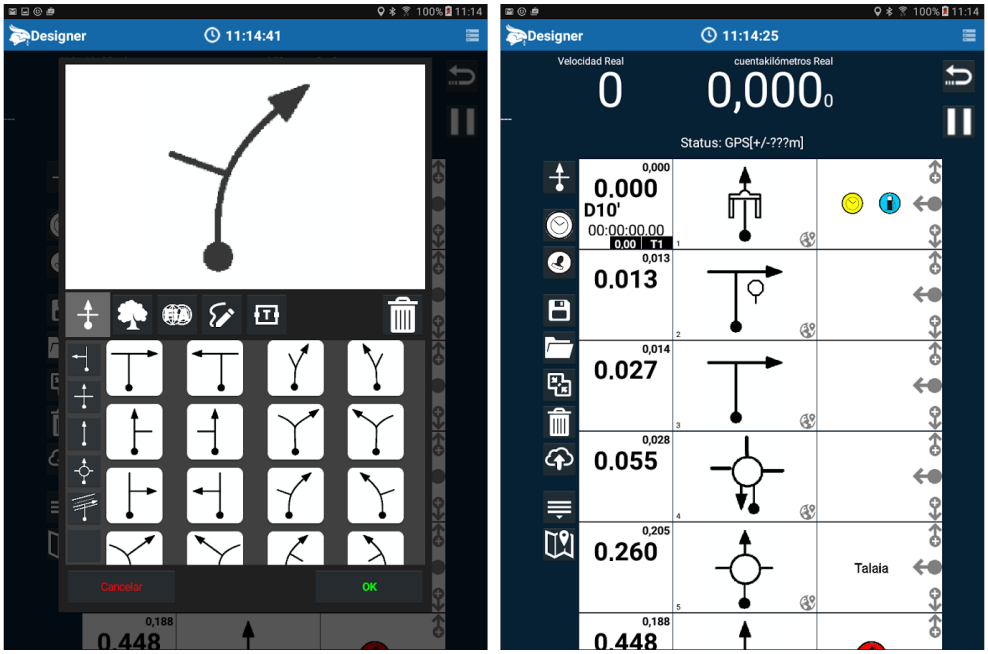
Aplikácia sama počíta vzdialenosti medzi jednotlivými bodmi počas ich vytvárania na trase. Skutočnosť, že GPS poloha bola zaznamenaná, aplikácia oznámi zvukovým signálom, tak vodič vie, že môže pokračovať v ceste k ďalšiemu bodu.

Vytvorenie štartovacieho bodu už v priebehu trasy je taktiež možné, aplikácia umožňuje vynulovať odometer a od istého bodu začne vzdialenosti počítať odznova. Umožňuje taktiež vytvoriť checkpoint, užívateľ zastaví vozidlo, zvolí v menu ikonu pre vytvorenie checkpoint a počká na zaznamenanie polohy. Aplikácia vytvorí checkpoint a zaznamená vzdialenosť a GPS lokáciu. Ukážku aplikácie je možné vidieť na obrázku 1.2.

Aplikácia pre navigáciu zobrazuje údaje, ktoré sú pripravené v roadbook, ale aj ďalšie aktuálne údaje ako čas užívateľa na trase, aktuálna rýchlosť a časový údaj o koľko sa jeho čas líši od ideálneho času. Užívateľ môže počas jazdy roadbook prispôsobovať, napríklad posunutím odometra o nejaký číselný údaj. Aplikácia potom prepočíta priemerné rýchlosti alebo časy na úsekoch, ktoré boli do roadbook zadané počas vytvárania, tak, aby užívateľ dosiahol daný čas na trase. Vzhľadom na to, že tieto informácie sa upravujú za jazdy, spoločnosť ponúka zariadenie, ktoré je možné pripojiť k aplikácii, a zjednodušiť si prepínanie stláčaním tlačidiel namiesto klikania na obrazovku. Toto zariadenie sa nazýva rabbit box.

1.2 Užívatelia

Medzi koncových užívateľov, pre ktorých bude aplikácia navrhnutá patria všetci nadšenci off-road jazdenia, pretože práve pri tomto štýle sa najčastejšie používajú roadbooky. Aplikácia je určená pre autá rovnako tak aj pre motorky, aj keď tie nie vždy obsahujú ODBII port, čo ale nie je prekážkou, nakoľko je možné použiť GPS, alebo externý merač vzdialenosti a inštrukcie prepínať manuálne.



Obr. 1.2: Ukážky aplikácie Rabbit Designer.

1.2.1 Persony

Persona je archetypom užívateľa aplikácie, ide o snahu vytvoriť konkrétnu osobu, čo možno najpodobnejšiu skutočnému užívateľovi, ktorá reprezentuje cieľovú skupinu užívateľov. Často sa vytvárajú viaceré persony, ktorým je priradená priorita, čo napomáha pri ohodnotení jednotlivých funkcií systému a ich dôležitosti pre samotných užívateľov. Persony sú taktiež dôležité pri návrhu užívateľského rozhrania a jeho testovaní pomocou kognitívneho priechodu, kedy vykonávateľovi testovania pomôžu vcítiť sa do role užívateľa aplikácie.

Každá persona by mala obsahovať fiktívne meno, to napomáha reálnejšej predstave o osobe, taktiež pohlavie a vek na lepšie zaradenie. Ďalej zamestnanie a demografické údaje ako sú rodinný stav, vzdelanie, či počet detí. Dôležitým je tiež definovanie potrieb a cieľov persony. Vhodné je aj pridanie fotografie, ktorá pomôže vnímať personu ako skutočného človeka a celkovo zlepší predstavu o užívateľovi. Ďalším častým atribútom je citát, ktorý zhrňuje postoj danej persony. [1]

Forma akou má byť persona spísaná nie je presne definovaná, no informácie by mali byť v ľahko čitateľnom formáte a logicky usporiadané. [2] Pre persony k vytváranej aplikácií bola zvolená forma tabuľky, vytvorené boli dve, primárna a sekundárna persona.

1.2.1.1 Primárna persona – Jozef Adamec

Persona	Motorkár nadšenec – dobrodruh
Fotka	
Meno	Jozef Adamec
Práca	Fotograf pre auto-moto časopis
Demografia	45 rokov ženatý 2 deti stredoškolské vzdelanie
O Jozefovi	Jozef miluje dlhé jazdy po neznámych a málo zmapovaných cestách. Rád objavuje krásy tohto sveta a so svojou dvojkolesovou láskou navštívil nespočetne veľa krajín.
Ciele	Svoje cesty a objavené miesta by si chcel uchovať a zaznamenať, aby ich mohol znova absolvovať a svoje trasy aj jednoducho rozšíriť medzi svojich kamarátov.
Citát	“Pôjdem na cestu okolo sveta”

1.2.1.2 Sekundárna persona – Stanislav Vŕba

Persona	Mladý off-road závodník
Fotka	
Meno	Stanislav Vŕba
Práca	študent
Demografia	21 rokov slobodný bezdetný študent vysokej školy technickej
O Stanislavovi	Stano sa už od malička venuje rôznym športom, no najviac mu k srdcu prirástlo off-road závodenie. Zúčastnil sa už na niekoľkých lokálnych podujatiach, zatiaľ bez väčšieho úspechu, ale stále na sebe pracuje a nevzdáva sa.
Ciele	Vo svojom okolí má Stano hneď niekoľko off-road dráh na ktoré chodí cvičiť a pretekať sa s priateľmi. Keďže sa chce zlepšovať a byť najrýchlejším, chce si svoje výsledky porovnávať a uchovávať. Taktiež by chcel usporiadať menší závod na ním vytýčenej lesnej ceste, kde by sa závodníci museli riadiť podľa roadbook.
Citát	“Dáme závod?”

1.3 Definícia požiadaviek

Určenie požiadaviek a teda funkcií systému je základným krokom vývoja akéhokoľvek softvéru. Cieľom špecifikácie požiadaviek je jasné definovanie na čo má daný systém slúžiť a aké problémy riešiť. Požiadavky sa obecne delia na dve skupiny a síce funkčné a nefunkčné.

Požiadavky na systém boli vytvorené na základe rešerší existujúcich riešení a v spolupráci s autorom nápadu.

1.3.1 Funkčné požiadavky

Funkčné požiadavky popisujú čo by mal systém robiť, čo by s ním malo byť možné dosiahnuť, ale už nepopisujú, ako by tieto požiadavky mali byť realizované.

- **Vytváranie roadbook:** aplikácia umožní užívateľom vytváranie a úpravu roadbook, kde je možné pridávanie, odoberanie, úprava inštrukcií a ich dát. Inštrukcie obsahujú celkovú a čiastočnú vzdialenosť, tulip, smerovanie na kompase, textový popis, dodatočné ikony, GPS súradnice a text hlasového povelu pre inštrukciu. Pre pridanie inštrukcie je potrebné zadať aspoň jeden zo spomenutých atribútov.

- **Navigácia pomocou roadbook:** aplikácia bude schopná užívateľa navigovať pomocou inštrukcií definovaných v roadbook. Pričom bude merať prejdenú vzdialenosť buď s použitím GPS, alebo ELM237 zariadenia pripojeného do vozidla a komunikujúceho cez bluetooth rozhranie. Počas navigácie bude aplikácia čítať hlasové povely zadané k inštrukciám, alebo pripravené príkazy k tulip, taktiež bude upozorňovať na prejdenú vzdialenosť, respektíve na blížiacu sa inštrukciu.

- **Zobrazenie mapy:** počas vytvárania trasy poskytne aplikácia možnosť určiť bod na mape a tak zadať GPS súradnice inštrukcie. Pri navigácií dokáže zobraziť mapu s aktuálnou polohou, vykreslenou prejdenou trasou a polohami inštrukcií, ak majú zadané GPS súradnice.

- **Zaznamenávanie a zobrazenie štatistík prejdených trás:** systém bude počas navigácie ukladať údaje potrebné na neskoršie zobrazenie štatistík absolvovaných jázd, medzi tieto údaje patria čas a vzdialenosť absolvovania trasy a jej jednotlivých inštrukcií. Tieto dáta a porovnanie rôznych absolvovaných jázd bude zobrazené prehľadnou a zrozumiteľnou formou.

- **Export roadbook:** aplikácia bude poskytovať možnosť užívateľom vytvorený roadbook exportovať do súboru a importovať do aplikácie v inom zariadení.

- **Import roadbook:** aplikácia bude poskytovať možnosť importovať roadbook z predtým exportovaného súboru.

- **Import GPX súboru:** pri vytváraní roadbook poskytne aplikácia užívateľom možnosť zadať GPX súbor, ktorý načíta a vytvorí zoznam inštrukcií s GPS súradnicami definovanými v danom súbore.

1.3.2 Nefunkčné požiadavky

Nefunkčné požiadavky, naopak od tých funkčných, neurčujú čo má systém robiť, ale upresňujú akým spôsobom má byť realizovaný, respektíve určujú požadovaný výkon systému. Preto sa často delia na požiadavky na vývoj a požiadavky na výkon.

- **Aplikácia pre Android OS:** vytvorená aplikácia bude určená pre zariadenia s operačným systémom android, a to od API verzie 16, teda Androidu 4.1 Jelly Bean.
- **Spoľahlivosť systému:** navrhovaná aplikácia by mala byť čo najviac spoľahlivá, najmä čo sa týka procesu navigácie, pretože práve pri ňom budú akékoľvek zlyhania a pády pre užívateľov nanajvýš nepríjemné.
- **Responzívnosť:** dizajn a jednotlivé UI prvky aplikácie by mali byť responzívne a schopné prispôsobiť sa rôznym veľkostiam a rozlíšeniam displejov zariadení.
- **Viditeľný stav systému:** aplikácia by nemala zamŕzať a neodpovedať na užívateľove vstupy, v prípade dlhších činností musí zobrazovať, že pracuje.
- **Údržba a rozvoj:** kód aplikácie a samotná zvolená architektúra by mala umožňovať čo najjednoduchšiu údržbu, opravu chýb a zároveň ponúkať príležitosť na rozšírenia aplikácie.
- **Jazykové variácie:** systém bude vytvorený s podporou dvoch jazykov a to slovenčina a angličtina. Jazyk aplikácie bude automaticky zvolený na základe nastavenia v zariadení, teda v prípade, že je jazyk zariadenia slovenčina, použije sa tá, inak angličtina.

1.4 Prípady použitia

Po definovaní požiadaviek na aplikáciu je ďalším krokom analýzy vytvorenie prípadov jej použitia, takzvaných usecases. Prípady použitia sú dôležitou súčasťou návrhu designu orientovaného na koncového užívateľa, pretože sa zameriavajú práve na jeho interakciu s vyvíjanou aplikáciou, čím poskytujú pohľad na to, ako bude výsledný systém reálne používaný.

Jeden prípad použitia zobrazuje teda postupnosť krokov užívateľa a s nimi spojených reakcií aplikácie za účelom úspešného dokončenia akcie aj s prípadnými alternatívami, či chybnými krokmi, na ktoré systém musí správne reagovať. Cieľom vytvorenia je tak získanie detailného pohľadu na priebeh použitia systému a jeho konkrétnych častí. [3]

Vytvorené boli štyri scenáre použitia, jeden pre vytvorenie trasy, dva pre navigáciu s rôznymi nastaveniami aplikácie a jeden pre zobrazenie štatistík

absolvovaných jázd. Spísané sú formou akcie užívateľa a očakávanej reakcie systému.

1.4.1 Prípad použitia – Vytvorenie roadbook

Popis: Užívateľ si chce vytvoriť vlastný roadbook, zvolí preto na úvodnej obrazovke možnosť "Vytvoriť roadbook".

Cieľ: Umožniť užívateľovi čo možno najjednoduchšie a intuitívne vytvorenie trasy a jej priradených inštrukcií.

Scenár:

1. Užívateľ zadá meno, názov pre roadbook.
2. Systém skontroluje, či sa zadaný názov už nenachádza v systéme, ak nie systém presunie užívateľa na zoznam inštrukcií, ktorý je zatiaľ prázdny.
3. Užívateľ zvolí možnosť "plus", pridať inštrukciu.
4. Systém zobrazí obrazovku na zadávanie údajov inštrukcie.
   K inštrukcií je možné zadať nasledovné údaje:
   - celkovú vzdialenosť
   - čiastočnú vzdialenosť
   - smerovanie – stupne na kompase
   - hlavný tulip – obrázok, nákres križovatky
   - textový popis
   - tri dodatočné obrázky
   - GPS súradnice
   - text hlasového povelu
5. Užívateľ zadá celkovú vzdialenosť.
6. Systém skontroluje či zadaná hodnota nie je menšia ako celková vzdialenosť predošlej inštrukcie. Dopočíta hodnotu čiastočnej vzdialenosti a vyplní príslušné pole obrazovky.
7. Užívateľ vyberie z galérie tulip.
8. Systém vykreslí zvolený tulip na príslušné miesto vrámci dát vytváranej inštrukcie.
9. Užívateľ zvolí akciu "uložiť inštrukciu".
10. Aplikácia skontroluje, či je zadaná aspoň jedna z položiek, ak áno inštrukciu vloží do databázy. Zobrazí zoznam všetkých inštrukcií trasy a správu o úspešnom uložení inštrukcie.
11. Užívateľ vyberie akciu "uložiť roadbook".

12. Systém overí, či bola pridaná minimálne jedna inštrukcia, ak áno uloží roadbook do databázy, zobrazí úvodnú stránku spolu so správou o úspešnom uložení trasy.

Alternatívy:

2. a)

1. Systém zistí zhodu zadaného mena s iným názvom v databáze a zobrazí chybovú správu.
2. Užívateľ zmení zadaný názov.

5. a)

1. Aplikácia zistí, že zadaná hodnota je menšia ako hodnota predošlej inštrukcie, informuje o tom užívateľa chybovou správou a zvýrazní príslušné dátové pole.
2. Užívateľ zmení zadanú hodnotu celkovej vzdialenosti.

9. a)

1. Užívateľ zvolí možnosť pridať GPS súradnice k inštrukcii.
2. Systém otvorí okno s mapou a značkou zobrazujúcou aktuálnu polohou, alebo miestom pri predošlej inštrukcií, ktorá má zadané GPS súradnice.
3. Užívateľ mapu priblíži a kliknutím na požadované miesto presunie značku označujúcu polohu inštrukcie a svoj výber potvrdí.
4. Systém zatvorí okno s mapou a vyplní do príslušného pola GPS súradnice zo zadanej polohy.

9. b)

1. Užívateľ zadá smerovanie – stupne na kompase.
2. Systém skontroluje, či je zadaná hodnota v rozpätí 1 – 360, ak nie zobrazí chybovú správu.

9. c)

1. Užívateľ zvolí akciu “úprava tulip”.
2. Systém zobrazí obrazovku kreslenia a vykreslí predtým zvolený tulip.
   Obrazovka kreslenia ponúka na výber osem farieb, kreslenie voľnou rukou, kreslenie čiary, elipsy, obdĺžnika, zmenu hrúbky pera a gumu.
3. Užívateľ do obrázku dokreslí detailnejší nákres križovatky. A zvolí akciu uložiť.
4. Aplikácia zavrie okno kreslenia, zobrazí upravený tulip, zároveň ho uloží do pamäte zariadenia a do databázy zapíše nový záznam obrázku s typom “Vlastné”.

9. d)

1. Užívateľ zvolí akciu "odstrániť inštrukciu".
2. Systém zobrazí okno s otázkou, či si užívateľ naozaj želá odstrániť inštrukciu.
3. Užívateľ potvrdí odstránenie.
4. Systém odstráni záznam z databázy, a ak po práve vymazanej inštrukcií nasledujú ďalšie, upraví ich poradové čísla. Zobrazí zoznam inštrukcií a správu o úspešnom vymazaní.
5. Užívateľ pokračuje krokom 11.

12. a)

1. Systém zistí, že roadbook neobsahuje žiadnu inštrukciu, zobrazí teda chybovú správu s možnosťami na zrušenie akcie, alebo odstránenie trasy.
2. Užívateľ vyberie možnosť "odstrániť roadbook".
3. Systém vymaže roadbook z databázy, zobrazí úvodnú obrazovku a správu o úspešnom odstránení trasy.

1.4.2 Prípad použitia – Navigácia podľa roadbook

Popis: Užívateľ chce absolvovať trasu popísanú v predtým vytvorenom roadbook s použitím merania vzdialenosti pomocou bluetooth zariadenia ELM327.

Cieľ: Navigovať užívateľa postupne po jednotlivých inštrukciách až do cieľa.

Scenár:

1. Užívateľ zvolí na úvodnej obrazovke možnosť "Roadbook".
2. Systém zobrazí zoznam uložených trás.
3. Užívateľ si zo zoznamu vyberie trasu – roadbook, ktorú chce absolvovať.
4. Systém zobrazí okno navigácie spolu s informáciami o GPS a ELM327. Okno informuje užívateľa o priebehu pripájania sa na bluetooth zariadenie, ktoré bolo zvolené v nastaveniach.
5. Užívateľ počká na úspešné pripojenie na zariadenie a spustí navigáciu.
6. Systém získava z riadiacej jednotky cez OBDII protokol prejdenú vzdialenosť, ktorú zobrazuje užívateľovi a podľa jeho nastavení číta hlasové povely k danej inštrukcii. Keď sa prejdená vzdialenosť rovná vzdialenosti aktuálnej inštrukcie a užívateľ má nastavený automatický režim presúvania inštrukcií, systém sa presunie na ďalšiu, táto zmena je užívateľovi zobrazená.
7. Užívateľ pokračuje podľa inštrukcií.

8. Systém sa presunie na poslednú inštrukciu. Keď dosiahne jej vzdialenosť upozorní užívateľa na koniec navigácie a zobrazí jej čas.

Alternatívy:

4. a)

1. Systém zobrazí okno navigácie spolu s informáciami o GPS a ELM327. Okno informuje užívateľa o priebehu pripájania sa na bluetooth zariadenie, pripojenie sa nepodarí, komunikácia so zariadením nie je možná.
2. Užívateľ zvolí akciu "nastavenia".
3. Systém zobrazí okno nastavení aplikácie.
4. Užívateľ vyberie položku "Zvoliť bluetooth zariadenie".
5. Systém zobrazí zoznam spárovaných zariadení.
6. Užívateľ vyberie iné zariadenie zo zoznamu.

6. a)

1. Systém získava z riadiacej jednotky cez OBDII protokol prejdenú vzdialenosť, ktorú zobrazuje užívateľovi a podľa jeho nastavení číta hlasové povely k danej inštrukcii. Keď užívateľ nemá nastavený automatický režim posúvania inštrukcií, systém ho zvukovým a vizuálnym efektom upozorní na skutočnosť, že sa prejdená vzdialenosť rovná vzdialenosti aktuálnej inštrukcie.
2. Užívateľ potvrdí presun na ďalšiu inštrukciu kliknutím na ľubovolné miesto obrazovky, ktoré neobsahuje inú akciu.
3. Postup sa opakuje až po poslednú inštrukciu trasy, nasleduje krok 8.

7. a)

1. Užívateľ zvolí akciu "znížiť prejdenú vzdialenosť", pretože si uvedomil, že zle odbočil.
2. Systém zníži prejdenú vzdialenosť o hodnotu nastavenú užívateľom v nastaveniach aplikácie.
3. Užívateľ pokračuje v navigácií.

7. b)

1. Užívateľ spraví gesto potiahnutie smerom dole.
2. Systém posunie zoznam inštrukcií smerom dole a zobrazí priebežné údaje o jazde, ako prejdenú vzdialenosť bez korekcií, priemernú rýchlosť, čas jazdy, dĺžku trasy, a tlačidlo na zobrazenie mapy.
3. Užívateľ zvolí akciu "mapa".

4. Systém zobrazí namiesto zoznamu inštrukcií mapu spolu s prejdenou trasou a aktuálnou polohou. Taktiež zobrazí pozície inštrukcií, ktoré majú určenú polohu.
5. Užívateľ sa vráti na zoznam inštrukcií tlačidlom späť a pokračuje krokom 7.

7. c)

1. Užívateľ zvolí akciu "presun na ďalšiu inštrukciu".
2. Systém posunie zoznam inštrukcií a zvýrazní nasledujúcu zmenu na trase, rovnako tak vynuluje čiastočnú vzdialenosť. Pokračuje v počítaní vzdialenosti.
3. Užívateľ pokračuje krokom 7.

1.4.3 Prípad použitia – Navigácia podľa roadbook 2

Popis: Užívateľ chce absolvovať trasu popísanú v predtým vytvorenom roadbook s použitím merania vzdialenosti pomocou GPS a zapnutým ručným prepínaním inštrukcií.

Cieľ: Navigovať užívateľa po jednotlivých inštrukciách až do cieľa.

Scenár:

1. Užívateľ v bočnom menu zvolí možnosť "nastavenia".
2. Systém zobrazí nastavenia aplikácie.
3. Užívateľ zvolí ako metódu merania vzdialenosti GPS, a vypne automatické prepínanie inštrukcií.
4. Tlačidlom späť prejde na úvodnú stránku a zvolí možnosť "Roadbook".
5. Systém zobrazí zoznam uložených trás.
6. Užívateľ si zo zoznamu vyberie trasu – roadbook, ktorú chce absolvovať.
7. Systém zobrazí okno navigácie spolu s informáciami o GPS a správou, že bluetooth komunikácia so zariadením ELM327, je vypnutá.
8. Užívateľ spustí navigáciu.
9. Aplikácia aktualizuje polohu zariadenia, z ktorej vypočítava prejdenú vzdialenosť, ktorú zobrazuje užívateľovi a podľa jeho nastavení číta hlasové povely k danej inštrukcií. Keď sa prejdená vzdialenosť rovná vzdialenosti aktuálnej inštrukcie notifikuje o tejto skutočnosti užívateľa zvukovým a vizuálnym efektom.
10. Užívateľ potvrdí presun na ďalšiu inštrukciu kliknutím na ľubovolné miesto obrazovky, ktoré neobsahuje inú akciu.
11. Kroky 9. a 10. sa opakujú až kým užívateľ nepotvrdí presun za poslednú inštrukciu, čiže dosiahnutie konca trasy a cieľa.

1.4.4 Prípad použitia – Zobrazenie štatistík

Popis: Užívateľ si chce pozrieť štatistiky svojich absolvovaných trás.

Cieľ: Prehľadnou formou zobraziť dáta zozbierané počas navigácie, tak aby užívateľovi priniesli prehľad o absolvovaných jazdách.

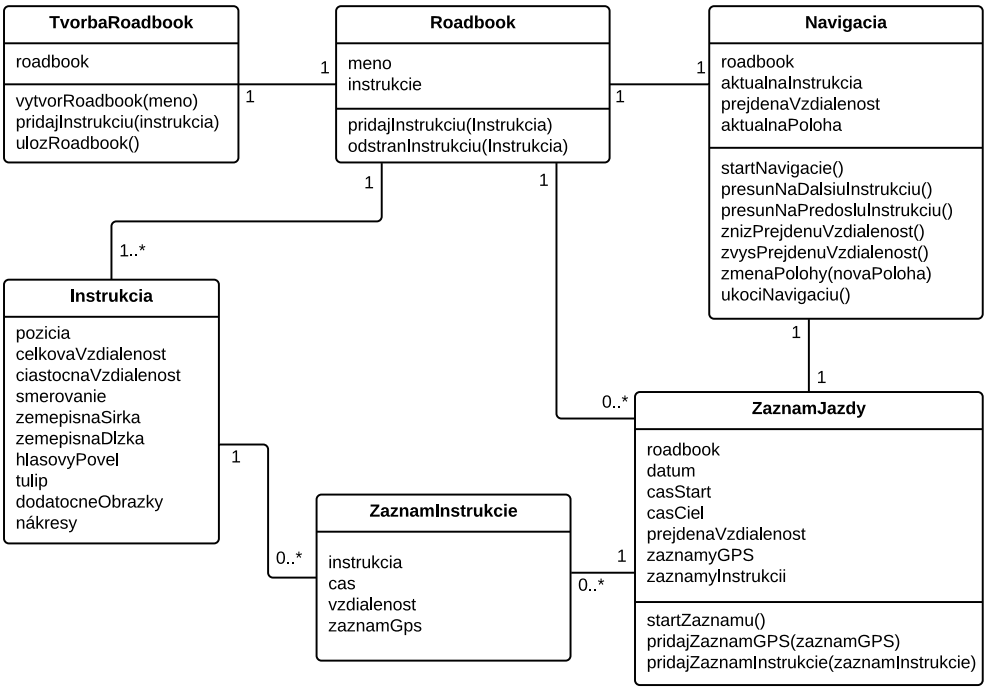
Scenár:

1. Užívateľ zvolí na úvodnej stránke akciu "Štatistiky".
2. Systém zobrazí zoznam trás nachádzajúcich sa v databáze.
3. Užívateľ vyberie požadovaný roadbook kliknutím na položku zoznamu predstavujúcu jednu trasu.
4. Systém načíta údaje o absolvovaných navigáciach na zvolenej trase.
   Zobrazuje sa:
   - graf celkových časov jednotlivých jázd
   - zoznam jázd zoradený podľa času, ktorý ukazuje čas a prejdenú vzdialenosť porovnanú voči priemeru
   - zoznam inštrukcií s najhorším a najlepším celkovým a čiastočným nameraným časom.
5. Užívateľ klikne na inštrukciu v zozname.
6. Systém otvorí okno s detailom štatistiky zvolenej inštrukcie.
   Zobrazuje sa:
   - podoba inštrukcie
   - graf celkového času absolvovania inštrukcie v jednotlivých zaznamenaných jazdách
   - graf čiastočného času, čiže času nameraného od predchádzajúcej inštrukcie
   - zoznam absolvovaní inštrukcie s celkovým a čiastočným časom porovnaným voči priemeru

Alternatívy

1. a)

1. Užívateľ zvolí na úvodnej stránke akciu "Roadbook".
2. Systém zobrazí zoznam uložených trás.
3. Užívateľ zvolí akciu menu pri položke zoznamu predstavujúcu jednu trasu.
4. Systém zobrazí menu, ktoré obsahuje akcie detail, štatistiky, vymazanie, kopírovanie, zdieľanie.
5. Užívateľ zvolí akciu "Štatistiky".



Obr. 1.3: Analytický model tried.

1.5 Analytické triedy

Cieľom vytvorenia analytických tried je určenie hlavných entít, ich atribútov a identifikácia ich vzájomných prepojení. Tie vyplývajú zo spísaných požiadaviek, rešerší a prípadov použitia systému. Analytický model tried je zobrazený na obrázku 1.3. Vytvorené triedy budú ďalej použité pri návrhu databázovej vrstvy aplikácie a vytvorení triedneho diagramu entít.

1.5.1 Trieda Roadbook

Trieda reprezentujúca jednu trasu popísanú inštrukciami.

Atribúty:

- meno – názov trasy
- instrukcie – zoznam zmien na trase

Metódy:

- pridajInstrukciu(instrukcia)
- odstranInstrukciu(instrukcia)

1.5.2 Trieda Instrukcia

Trieda predstavujúca jeden riadok – jednu zmenu na trase.
Atribúty:

- pozicia – poradové číslo vrámci roadbook
- celkovaVzdialenost – vzdialenosť od štartu
- ciastocnaVzdialenost – vzdialenosť od predchádzajúcej inštrukcie
- smerovanie – stupne na kompase
- zemepisnaSirka
- zemepisnaDlzka
- hlasovyPovel – text pre hlasový príkaz
- tulip – hlavný nákres situácie
- dodatocneObrazky – dodatočné ikony, nákresy

1.5.3 Trieda TvorbaRoadbook

Trieda slúžiaca na vytváranie roadbook.
Atribúty:

- roadbook

Metódy:

- vytvorRoadbook(meno)
- pridajInstrukciu(instrukcia)
- ulozRoadbook

1.5.4 Trieda Navigacia

Trieda slúžiaca na navigáciu po trase opísanej v roadbook.
Atribúty:

- roadbook
- aktualnaInstrukcia
- prejdenaVzdialenost
- aktualnaPoloha

Metódy:

- startNavigacie()
- presunNaDalsiuInstrukciu()
- presunNaPredosluInstrukciu()
- znizPrejdenuVzdialenost()
- zvysPrejdenuVzdialenost()
- zmenaPolohy(novaPoloha)
- ukonciNavigaciu()

1.5.5 Trieda ZaznamJazdy

Trieda obsahujúca dáta o prejdenej trase.
Atribúty:

- roadbook – trasa, o ktorej záznam ide
- datum – dátum absolvovania trasy
- casStart – čas začatia navigácie
- casCiel – čas ukončenia navigácie
- prejdenaVzdialenost – celková prejdená vzdialenosť bez korekcií počas navigácie
- zaznamyGPS – zoznam zaznamenaných GPS súradníc počas jazdy
- zaznamyInstrukcii – zoznam dát zaznamenaných pre každú inštrukciu

Metódy:

- startZaznamu()
- pridajZaznamGPS(zaznamGPS)
- pridajZaznamInstrukcie(zaznamInstrukcie)
- ukonciZaznam()

1.5.6 Trieda ZaznamInstrukcie

Trieda reprezentujúca zaznamenané dáta k absolvovaniu jednej inštrukcie počas navigácie.

- instrukcia – inštrukcia ku ktorej záznam patrí
- cas – čas absolvovania inštrukcie, čiže čas uplynutý od predchádzajúcej po nasledovnú inštrukciu
- vzdialenost – prejdená vzdialenosť od začiatku trasy po moment prechodu na ďalšiu inštrukciu
- zaznamGPS – zaznamenaná poloha v momente prechodu na ďalšiu inštrukciu

Kapitola 2

Návrh užívateľského rozhrania

Pre vytvorenie užívateľsky prívetivej aplikácie je dôležité dobre navrhnuté užívateľské prostredie. Užívateľským prostredím sa rozumie spôsob, môže ich byť aj viac, akým užívateľ so systémom komunikuje. Preto je hlavnou úlohou jeho návrhu zabezpečiť, aby sa systém dobre ovládal, vyžadoval čo najmenej interakcií na dosiahnutie požadovaného cieľa, bol jednoduchý na použitie aj pre začiatočníkov, znižoval potrebu pamätania si postupov a krokov, predchádzal chybám a celkovo bol príjemný na použitie.

Existuje niekoľko bodov, princípov, na ktoré je dobré pri návrhu užívateľského prostredia myslieť. Prvým je zameranie sa na potreby užívateľa, poskytnúť mu prostredie na rýchle a jednoduché vykonanie úloh, ktoré od systému požaduje a zároveň toto prostredie minimalizovať, zobrazovať iba užitočné informácie. Ďalšími princípmi sú spätná väzba na akciu, tolerancia chýb – ponúknuť možnosť opravy, vrátenia sa o krok späť, konzistentnosť jazyka – názvov, použitých ikon či rozmiestnenia prvkov. Pri návrhu je ďalej potrebné zohľadniť špecifiká platformy pre ktorú je aplikácia určená, aké sú bežné postupy a zvyky jej užívateľov. [4]

Prvým krokom návrhu bude spísanie task listu – zoznamu všetkých možných úloh z pohľadu užívateľa systému a vytvorenie task modelu, ktorý tieto úlohy zobrazuje v prehľadnom grafe. Následne bude vytvorený prototyp – mock-up systému, navrhnuté budú jednotlivé obrazovky, hlavne rozmiestnenie informácií a ovládacích prvkov.

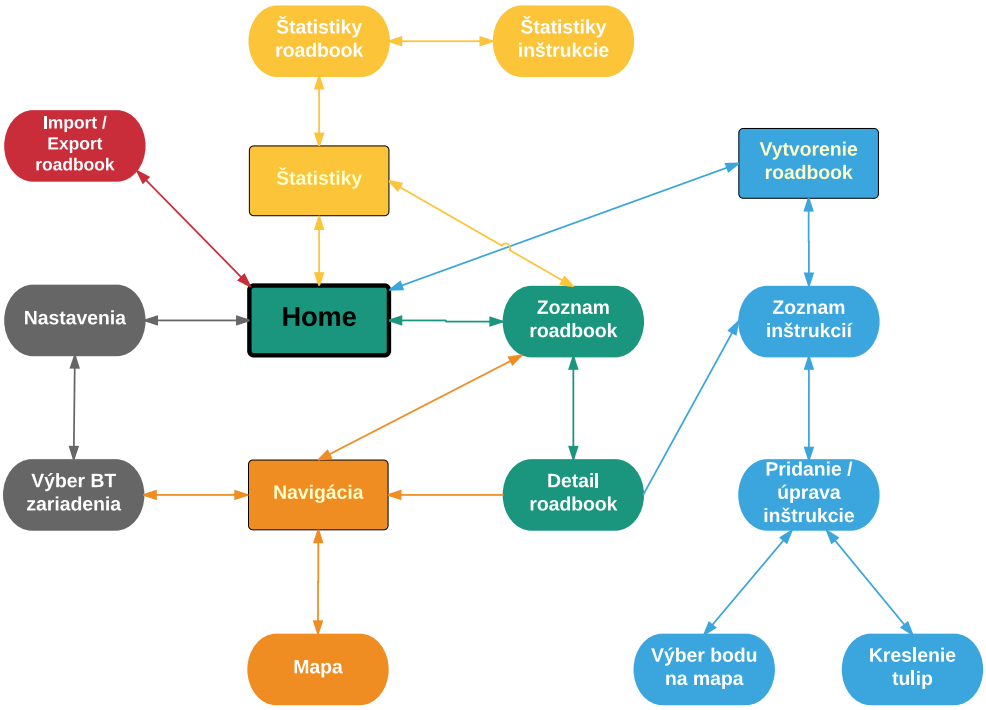
Pri návrhu užívateľského prostredia budú využité informácie zozbierané v analýze, najmä funkčné požiadavky, persony a prípady použitia aplikácie. Na vytvorenie návrhu užívateľského prostredia – wireframe bol použitý nástroj Balsamiq Cloud, pomocou ktorého bol následne vytvorený interaktívny mock-up aplikácie v podobe PDF. Vzniknutý prototyp bude otestovaný pomocou kognitívneho priechodu a nájdené nedostatky budú upravené.

2.1 Task list

Task list pomáha pri prvotnom návrhu užívateľského prostredia, jeho vstupom, podkladom, sú prípady použitia. Cieľom je spísanie všetkých, aj triviálnych úloh, ktoré systém užívateľovi ponúka.

Task list:

- Vytvorenie trasy
- Zadanie mena trasy
- Pridanie inštrukcie
- Odstránenie inštrukcie
- Úprava inštrukcie
- Zadanie údajov inštrukcie
- Výber polohy inštrukcie na mape
- Nakreslenie tulip
- Úprava tulip
- Pridanie tulip k inštrukcii
- Označenie tulip ako obľúbeného
- Uloženie trasy
- Úprava trasy
- Zobrazenie zoznamu trás
- Zobrazenie zoznamu inštrukcií jednej trasy
- Spustenie navigácie
- Zmena aktuálnej inštrukcie
- Úprava prejdenej vzdialenosti
- Ukončenie navigácie
- Zobrazenie mapy, prejdenej trasy a aktuálnej polohy
- Zobrazenie nastavení systému
- Výber bluetooth zariadenia EML327
- Zobrazenie štatistík trasy



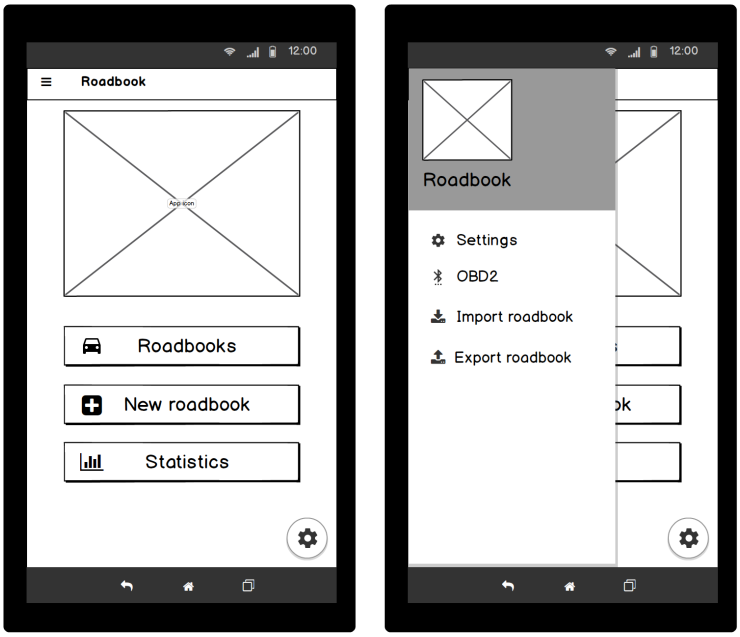
Obr. 2.1: Task graf systému.

- Zobrazenie štatistík jednej inštrukcie
- Importovanie roadbook
- Exportovanie roadbook

2.2 Task model

Task model pomáha predstave o náväznosti a vzťahoch jednotlivých úloh – taskov. Tento model nemá presne definovanú formu, môže ísť o strom, flowchart, alebo graf, kde sú zobrazené hlavné úlohy systému, ktoré môžu byť ďalej rozdelené na podúlohy.

Vstupom pre vytvorenia task modelu je task list, no pre lepší prehľad sú zobrazené iba dôležitejšie úlohy. Vybraný bol model formou grafu, ktorý je zobrazený na obrázku 2.1. Vstupná – domovská obrazovka je zobrazená v hrubom čiernom ráme, hlavné úlohy sú obdĺžnikového tvaru a ich nadväzujúce podúlohy sú oválového tvaru. Jednotlivé úlohy sú farebne odlíšené podľa hlavných taskov, prechody medzi nimi sú znázornené čiarami zakončenými šípkami pre znázornenie smeru prechodu.



Obr. 2.2: Home – úvodná obrazovka aplikácie vľavo a s vysunutým bočným menu vpravo.

2.3 Obrazovky

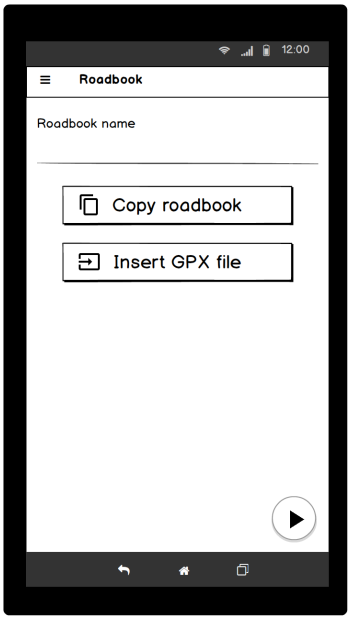
2.3.1 Home – úvodná obrazovka

Úvodná obrazovka sa užívateľovi zobrazí po otvorení aplikácie, obsahuje odkazy na hlavné funkcie systému, ďalšie dodatočné možnosti sú prístupné z vysúvacieho bočného menu, wireframy sú zobrazené na obrázku 2.2.

2.3.2 Vytvorenie roadbook

Vytváranie roadbook sa skladá hneď z niekoľkých obrazoviek. Prvou je stránka na zadanie mena, jej ukážka je na obrázku 2.3. Obrazovka obsahuje možnosť na skopírovanie iného, predtým vytvoreného, roadbook, alebo na vloženie GPX súboru podľa ktorého sa vytvorí zoznam inštrukcií s definovanými GPS súradnicami.

Ďalšou je zoznam už pridaných inštrukcií aj s ich atribútmi. Na tejto obrazovke zobrazenej na obrázku 2.4 užívateľ nájde možnosti na odstránenie a úpravu každej jednej inštrukcie, tiež akcie pridania ďalšieho bodu a uloženia trasy. Gestom posunutia smerom dole (ak je zoznam na prvej inštrukcii), alebo cez tlačidlo “i” sa zobrazia informácie o trase ako jej názov, počet inštrukcií a celková dĺžka.



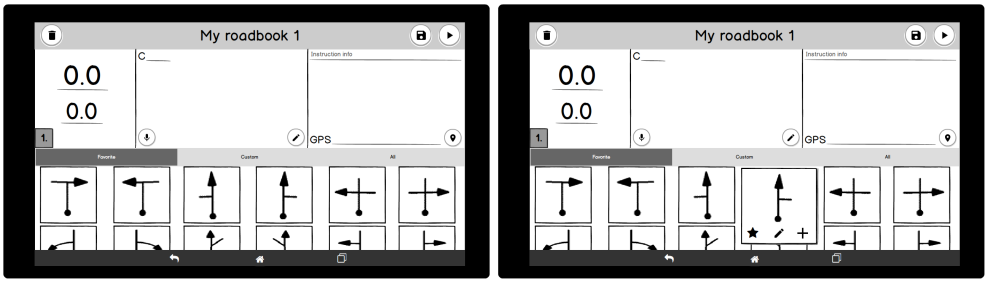
Obr. 2.3: Prvá obrazovka vytvárania roadbook.

Obr. 2.4: Zoznam už vytvorených inštrukcií.

2.3.2.1 Vytvorenie inštrukcie

Táto obrazovka ponúka možnosť zadať údaje k jednej zmene na trase. Prvky inštrukcie sú rozmiestnené rovnako ako sú aj pri navigácii a ako sú zaužívané pri väčšine roadbook, vytváranie je preto rýchle a intuitívne. Možné je zadať celkovú vzdialenosť, podľa ktorej sa dopočíta čiastočná, alebo naopak. Ďalej je tu na výber z galérie pripravených tulip a tiež možnosť upraviť ich, alebo nakresliť vlastný. K ďalším atribútom, ktoré je možné zadať patria smerovanie, textový popis, tri dodatočné ikony, hlasový povel pre inštrukciu (akcia "mikrofón"), GPS súradnice.

Pri aktivácii jedného z textových polí sa zobrazí klávesnica tak, aby boli prvky inštrukcie stále viditeľné, inak je viditeľná galéria tulip zobrazená na



Obr. 2.5: Wireframy vytváranie inštrukcie, vpravo s otvorenou ponukou pre tulip.

obrázku 2.5 vľavo, ktorá ponúka obrázky rozdelené do troch oddielov a to obľúbené, vytvorené užívateľom a všetky. Kliknutie na vybraný tulip ho vloží do inštrukcie ako hlavný nákres situácie, dlhé podržanie otvorí ponuku ďalších možností, ktorá umožňuje daný nákres označiť ako obľúbený, respektíve zrušiť toto označenie, otvoriť jeho úpravu a pridať ho ako vedľajší – dodatočný nákres. Tieto možnosti sú zobrazené na obrázku 2.5 vpravo.

2.3.2.2 Kreslenie tulip

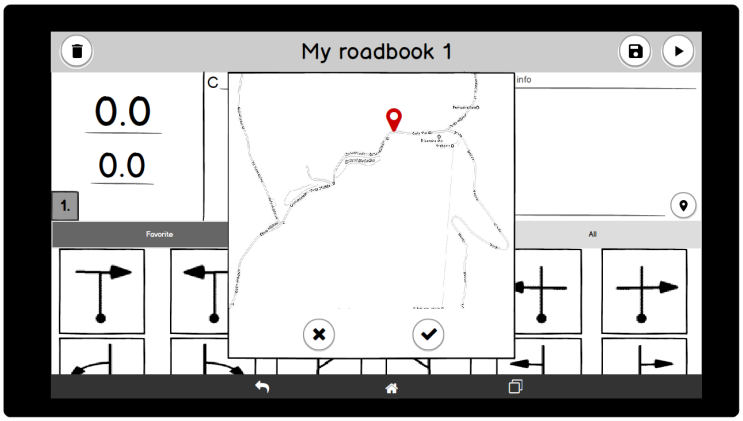
Táto stránka je prístupná z obrazovky vytváranie inštrukcie, ponúka základné kresliace nástroje a síce osem farieb, kreslenie voľnou rukou, kreslenie čiary, oválu, obdĺžnika, gumu a nastavenie hrúbky pera. Možná je aj úprava pripravených alebo už nakreslených a uložených obrázkov.

2.3.2.3 Zadanie polohy inštrukcie z mapy

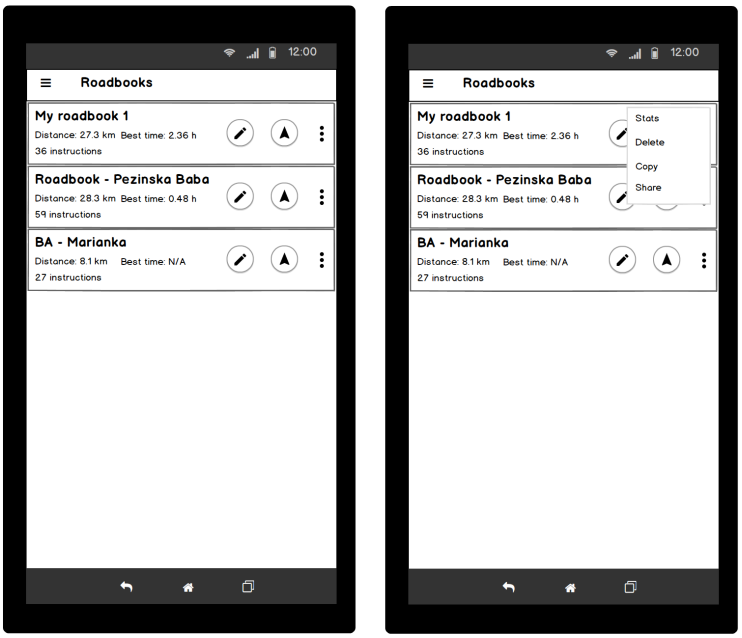
Aplikácia ponúka možnosť zadania polohy priamo z mapového podkladu, táto akcia je prístupná z obrazovky vytvárania inštrukcie. Ako je možné vidieť na obrázku 2.6 na mape sa zobrazuje značka, ktorá sa nachádza na aktuálnej polohe, prípadne pri pozícii predošlej inštrukcie, ak tá má zadané GPS súradnice. Značku je možné presunúť posunom, alebo kliknutím na želané miesto. Mapu je možné posúvať, približovať a oddľalovať. Po vybratí bodu na mape sa do patričného poľa zapíšu jeho GPS súradnice.

2.3.3 Roadbook

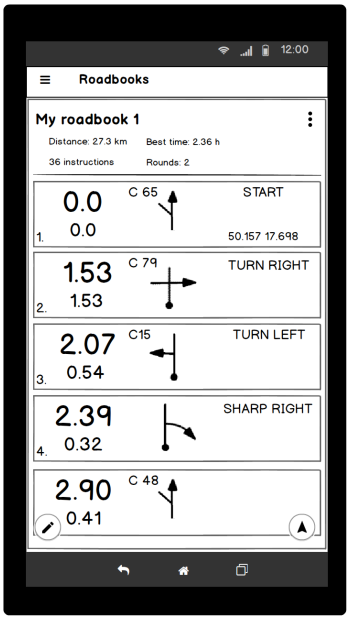
Obrazovka obsahuje zoznam užívateľových trás, ku každej ponúka dve hlavné akcie priamo – úpravu a navigáciu, ďalšie možnosti sú prístupné cez rozbaľovacie menu. Nákresy týchto obrazoviek sú na obrázku 2.7, menu ponúka zobrazenie štatistík, vymazanie, kopírovanie a zdieľanie roadbook. Ku každej trase v zozname sú informácie o počte inštrukcií, dĺžke trasy a prípadne najlepšom čase dosiahnutom na danej trase.



Obr. 2.6: Vytváranie inštrukcie trasy – výber bodu z mapy.



Obr. 2.7: Zoznam užívateľových roadbook vľavo a s otvoreným menu jednej trasy vpravo.



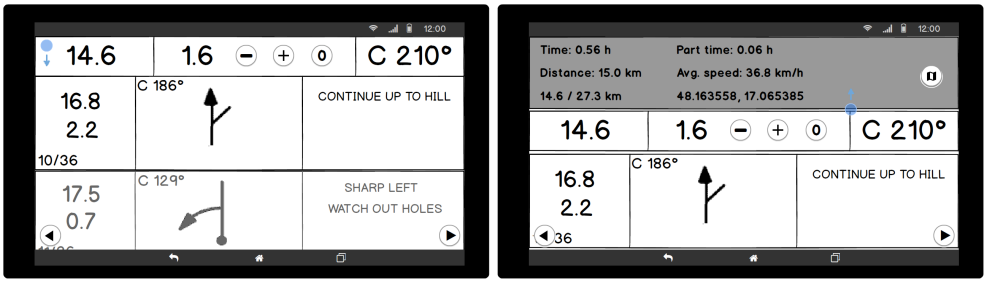
Obr. 2.8: Detail roadbooku so zoznamom inštrukcií.

2.3.3.1 Detail roadbook

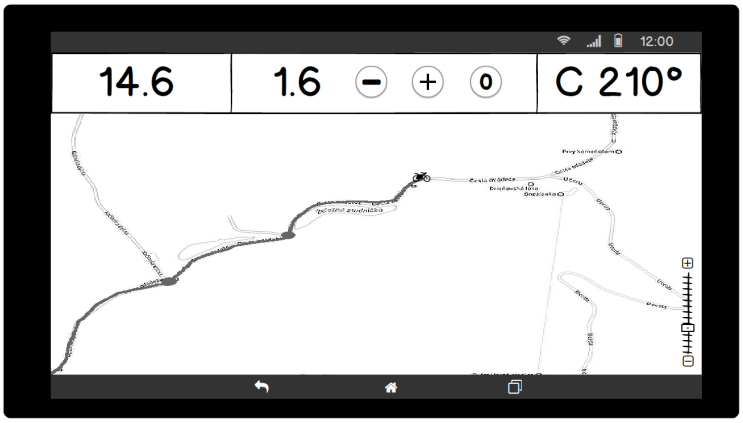
Detail trasy je prístupný zo zoznamu roadbook. Ponúka rovnaké akcie ako položka v zozname trás, navyše však obsahuje list inštrukcií, ako je možné vidieť na obrázku 2.8.

2.3.4 Navigácia

Hlavná obrazovka navigácie je zobrazená vľavo na obrázku 2.9, obsahuje prejdenú celkovú a čiastočnú vzdialenosť s možnosťami jej úpravy. Ďalej zoznam inštrukcií so zvýraznenou nasledujúcou zmenou na trase a akciami pre presun na ďalšiu, alebo na predchádzajúcu inštrukciu. Jednotlivé položky zoznamu, teda zmeny na trase majú rovnaké rozloženie prvkov ako je pri vytváraní inštrukcií. Akcie na presúvanie sa medzi inštrukciami sú zámerne umiestnené v dolnej časti obrazovky, aby boli čo najbližšie. V navigácii je možné zobraziť dodatočné informácie o priebehu jazdy, rovnako ako pri zozname inštrukcií vo vytváraní trasy gestom potiahnutia zoznamu smerom dole, wireframe obrazovky na ľavej strane obrázka 2.9. Navigácia ponúka ešte jedno zobrazenie a tým je mapa s aktuálnou polohou, prejdenou trasou a vyznačenými tými inštrukciami, ktoré obsahujú GPS polohu. Táto obrazovka je na obrázku 2.10.



Obr. 2.9: Obrazovka navigácie vľavo a s otvorenými dodatočnými informáciami o aktuálnej jazde vpravo.



Obr. 2.10: Obrazovka navigácie so zobrazenou mapou.

2.3.5 Štatistiky

Počas navigácie sa zaznamenávajú prejdené časy a vzdialenosti celej trasy, ale aj jednotlivých inštrukcií. Aplikácia vie následne tieto zozbierané informácie zobraziť v podobe rôznych štatistík.

2.3.5.1 Štatistiky roadbook

K vybranému roadbook je zobrazený graf časov jázd, zoznam jázd s porovnaním cieľového času a vzdialenosti voči priemeru, zoznam inštrukcií s ich najhorším, najlepším celkovým a čiastkovým časom absolvovania danej inštrukcie.

2.3.5.2 Štatistiky inštrukcie

Ku každej inštrukcii vrámci jednej trasy sú zobrazené dva grafy, a to graf celkového a čiastočného času absolvovania danej zmeny na trase. Ďalej v zo-

zname si užívateľ môže pozrieť dátumy absolvovania, jej celkový a čiastkový čas porovnaný s príslušným priemerným časom.

2.4 Kognitívny priechod

Kognitívny priechod je metóda testovania bez reálnych užívateľov, testovanie vykonávajú experti – dizajnéri aplikácie a snažia sa vcítiť do potrieb a očakávaní užívateľov. Priechod prebieha za použitia mockup systému, prípadne na výslednej aplikácii, prechádza sa vybranou skupinou úloh odzrkadľujúcich jeden prípad použitia systému. Pri každom kroku priechodu aplikáciou si vykonávateľ testovania kladie štyri otázky, ktoré sú zamerané na skutočnosť či je systém pochopiteľný a tiež ako ľahko sa s ním nový, alebo nepravidelný užívateľ naučí pracovať. [5]

Otázky sú nasledovné:

1. Pokúsi sa užívateľ dosiahnuť účinok akcie? – Táto otázka sa zameriava na skutočnosť, či užívateľ bude chcieť danú akcia vykonať, respektíve, či ju v danom kroku očakáva.

2. Je akcia viditeľná? – Otázka upriamuje pozornosť na samotný ovládací prvok, ktorým má byť požadovaná akcia vyvolaná, najmä teda, či je na obrazovke viditeľný a kde je umiestnený.

3. Je akcia zrozumiteľná a rozpoznateľná? Otázka sa opäť upriamuje na ovládací prvok akcie, tento krát ale na jeho zrozumiteľnosť, čím sa myslí napríklad, či názov, alebo použitá ikona akcie zodpovedá skutočnej funkcii.

4. Je reakcia na akcie zrozumiteľná? Táto otázka slúži na overenie reakcie aplikácie spôsobenej danou akciou, či je očakávaná, požadovaná, teda či dostane užívateľ správnu spätnú väzbu.

Hlavnou nevýhodou kognitívneho priechodu je, že hodnota výsledkov, zistení pri priechode, úzko súvisí so skúsenosťami, schopnosťami samotného hodnotiteľa, vykonávateľa priechodov a jeho pohľadu na systém.

Kognitívny priechod bol vykonaný na interaktívnom mockupe systému vytvorenom pomocou nástroja Balsamiq, v ktorom je možné definovať kliknutia na akcie, ktoré spôsobia prechod na inú obrazovku. Nástroj vie následne vygenerovať PDF súbor, kde tieto kliknutia znamenajú presun na inú stránku súboru. Pre testovanie boli zvolené dve hlavné funkcie aplikácie a to vytváranie trasy a navigácia. Pre každú z funkcií boli ďalej zvolené dve rôzne postupnosti krokov.

2.4.1 Priechod 1 – Vytvorenie roadbook

Akcie:

1. užívateľ zvolí vytvoriť roadbook – “Create roadbook”
2. užívateľ zadá názov roadbook
3. užívateľ potvrdí meno tlačidlom “ďalej”
4. užívateľ pridá novú inštrukciu tlačidlom “+”
5. užívateľ pridá k inštrukcii tulip – obrázok inštrukcie
6. užívateľ uloží inštrukciu tlačidlom “uložiť”
7. užívateľ zobrazí detailné informácie o trase
8. užívateľ uloží vytvorenú trasu

Kognitívny priechod 1 je v tabuľke 2.1.

2.4.2 Priechod 2 – Zobrazenie a úprava existujúcej trasy

Akcie:

1. užívateľ zvolí akciu “Roadbooks”
2. užívateľ zvolí konkrétnu trasu – zobrazí zoznam inštrukcií
3. užívateľ zvolí akciu upraviť roadbook
4. užívateľ zvolí akciu upraviť inštrukciu
5. užívateľ zmení GPS súradnice inštrukcie pomocou mapy
6. užívateľ uloží zmenenú inštrukciu
7. užívateľ uloží zmenený roadbook

Kognitívny priechod 2 je v tabuľke 2.2. .

2.4.3 Priechod 3 – Navigácia

Akcie:

1. užívateľ zvolí akciu “Roadbooks”
2. užívateľ zvolí akciu navigovať
3. užívateľ zobrazí detailné informácie o aktuálnom stave navigácie

Tabuľka 2.1: Kognitívny priechod 1

Akcia č.	**Pokúsi sa užívateľ dosiahnuť účinok akcie?**	**Je akcia viditeľná?**	**Je akcia zrozumiteľná a rozpoznateľná?**	**Je reakcia na akciu zrozumiteľná?**
1.	áno	áno	áno	áno
2.	áno, inak sa neposunie ďalej	nie úplne, lepšie bude zobraziť rovno aj klávesnicu	áno	áno, zadané meno je ďalej zobrazené
3.	áno	áno, tlačidlo je viditeľné aj keď je aktívna klávesnica	áno	áno
4.	áno	áno	áno	áno, zobrazí sa obrazovka vytvorenia inštrukcie
5.	áno	áno	áno	áno, zvolený tulip sa zobrazí v inštrukcii
6.	áno	áno	áno	nie, chýba správa o uložení
7.	áno	pre menej skúsených užívateľov nie	pre menej skúsených užívateľov nie, nevedia prečo sa informácia zobrazuje potiahnutím zoznamu smerom dole	áno
8.	áno	áno	áno	nie, chýba správa o uložení

Tabuľka 2.2: Kognitívny priechod 2

Akcia č.	**Pokúsi sa užívateľ dosiahnuť účinok akcie?**	**Je akcia viditeľná?**	**Je akcia zrozumiteľná a rozpoznateľná?**	**Je reakcia na akciu zrozumiteľná?**
1.	áno	áno	áno	áno
2.	áno	pre menej skúsených užívateľov nie (pridať tlačidlo Detail do rozbaľovacieho menu)	áno	áno
3.	áno	áno	áno	áno
4.	áno	áno	áno	áno
5.	áno	áno	áno	áno
6.	áno	áno	áno	nie, chýba správa o uložení
7.	áno	áno	áno	nie, chýba správa o uložení

4. užívateľ zobrazí mapu
5. užívateľ opustí mapu
6. užívateľ preskočí na ďalšiu inštrukciu
7. užívateľ sa vráti na predošlú inštrukciu
8. užívateľ ukončí navigáciu

Kognitívny priechod 3 je v tabuľke 2.3.

2.4.4 Priechod 4 – Navigácia – úprava prejdenej vzdialenosti

Akcie:

Tabuľka 2.3: Kognitívny priechod 3

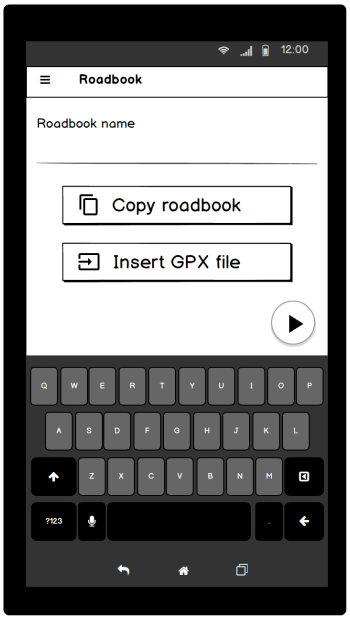
Akcia č.	**Pokúsi sa užívateľ dosiahnuť účinok akcie?**	**Je akcia viditeľná?**	**Je akcia zrozumiteľná a rozpoznateľná?**	**Je reakcia na akciu zrozumiteľná?**
1.	áno	áno	áno	áno
2.	áno	áno	áno	nie, treba pridať úvodnú obrazovku a tlačidlo na štart
3.	pokročilejší užívateľ možnosť nájde, ale inak nie	nie	áno	áno
4.	pokročilejší užívateľ možnosť nájde, ale inak nie	áno	áno	áno
5.	áno	nie, dá sa iba cez HW tlačidlo späť	nie	áno
6.	áno	áno	áno	áno
7.	áno	áno	áno	áno
8.	áno	áno	áno	áno

Tabuľka 2.4: Kognitívny priechod 4

Akcia č.	**Pokúsi sa užívateľ dosiahnuť účinok akcie?**	**Je akcia viditeľná?**	**Je akcia zrozumiteľná a rozpoznateľná?**	**Je reakcia na akciu zrozumiteľná?**
1.	áno	áno	áno	áno
2.	áno	áno	áno	nie, treba pridať úvodnú obrazovku a tlačidlo na štart
3.	závisí od skúseností užívateľa s používaním roadbook	áno	áno	áno
4.	závisí od skúseností užívateľa s používaním roadbook	áno	áno	áno
5.	áno	áno	áno	áno
6.	áno	áno	áno	áno

1. užívateľ zvolí akciu “Roadbooks”
2. užívateľ zvolí akciu navigovať
3. užívateľ zvýši prejdenú vzdialenosť k ďalšej inštrukcii
4. užívateľ zníži prejdenú vzdialenosť k ďalšej inštrukcii
5. užívateľ vynuluje prejdenú vzdialenosť k ďalšej inštrukcii
6. užívateľ ukončí navigáciu
7. užívateľ ukončí navigáciu

Kognitívny priechod 4 je v tabuľke 2.2.



Obr. 2.11: Prvá obrazovka vytvárania roadbook – zadanie mena s otvorenou klávesnicou.

2.4.5 Zmeny na základe kognitívneho priechodu

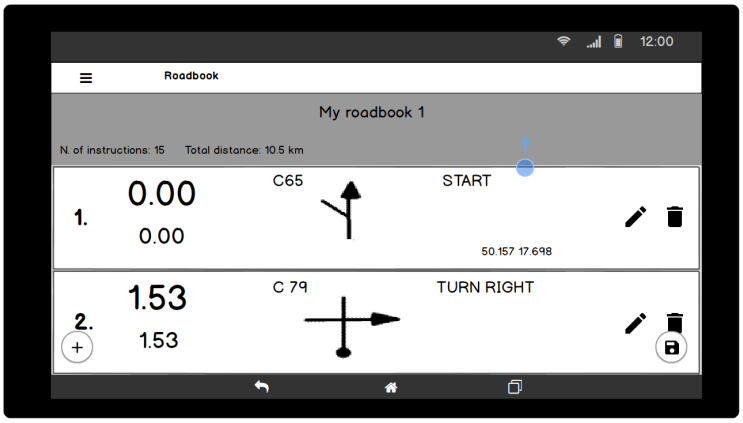
Zodpovedaním otázok kognitívneho priechodu bolo nájdených niekoľko nedostatkov v návrhu užívateľského prostredia, preto v ňom boli vykonané zmeny a rovnako boli upravené wireframy príslušných obrazoviek.

2.4.5.1 Vytváranie trasy

Prvou zmenou pri vytváraní trasy je automatické zobrazenie klávesnice pri vstupe do procesu vytvárania, keďže prvým krokom je zadanie mena. Nákres obrazovky po zmene je zobrazený na obrázku 2.11, pôvodný na obrázku 2.3.

Druhou zmenou je pridanie správ pre užívateľa o úspešnom uložení inštrukcie a roadbook, tie sa zobrazujú v spodnej časti obrazovky. Wireframy zobrazujúce tieto správy je možné vidieť v prílohe diplomovej práce.

Tretia zmena sa týka spôsobu zobrazenia informácií o práve vytváranej trase. Tieto dáta budú zobrazené vždy, pokiaľ užívateľ neposunie zoznam gestom posunutia smerom hore, čím roluje na nižšie položky zoznamu, vtedy sa dáta plynule so zoznamom zasunú do vrchnej lišty. Tým pádom užívateľ vie, že tam dané údaje sú a rolovaním zoznamu do opačnej strany ich opätovne zobrazí. Tlačidlo "i" bolo teda z obrazovky odstránené a zároveň tlačidlá "plus" a "uložiť" boli presunuté do ľavého dolného rohu, respektíve do pravého, čo je zaužívanejšie umiestnenie akčných tlačidiel. Pôvodné wireframy sú zobrazené na obrázku 2.4 a nákres po zmene na obrázku 2.12.



Obr. 2.12: Zoznam inštrukcií pri vytváraní trasy po zmene.

2.4.5.2 Zoznam roadbook

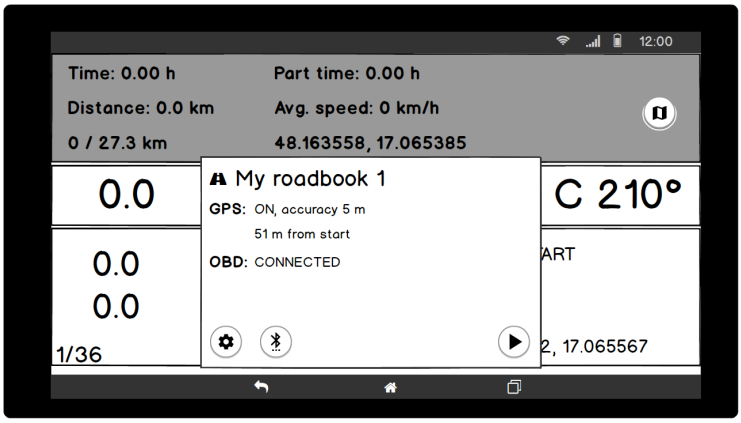
Tu nastala jediná drobná zmena, a síce do rozbaľovacieho menu ponúkajúceho možnosti vymazania, kopírovania, zdieľanie trasy a zobrazenia jej štatistík, bola pridaná akcia "Detail", ktorá zobrazí podrobnosti a zoznam inštrukcií jednej trasy. Toto zobrazenie je prístupné aj po kliknutí na položku zoznamu trás, akcia bola do bočného menu pridaná pre menej skúsených užívateľov operačného systému android.

2.4.5.3 Navigácia

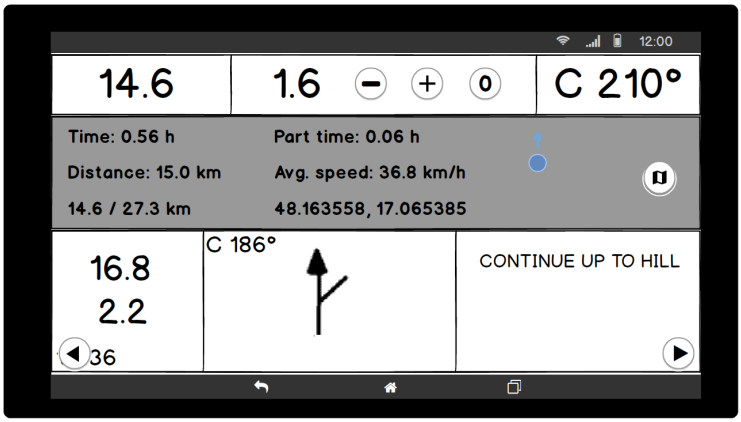
V návrhu navigácie bolo pomocou kognitívneho priechodu nájdených niekoľko nedostatkov, ktoré vyžadovali pridanie novej obrazovky na začiatku navigácie a niekoľko menších zmien v návrhu užívateľského prostredia. Prvou zmenou je teda pridanie úvodnej obrazovky, ktorú možno vidieť na obrázku 2.13. Tá na začiatku navigácie užívateľovi zobrazuje informácie o GPS a o priebehu pripájania sa na zariadenie ELM327, ak je definované v nastaveniach aplikácie, ďalej ponúka akcie na otvorenie nastavení systému a bluetooth zariadení, no a samozrejme možnosť spustenia navigácie.

Druhá zmena sa týka rozbaľovacích dodatočných informácií o priebehu navigácie. Podobne ako pri vytváraní aj v navigácii sa budú tieto údaje zobrazovať vždy, až kým ich užívateľ rolovaním zoznamu neskryje. Ďalšou zmenou je premiestnenie týchto údajov pod hlavný panel zobrazujúci prejdené vzdialenosti tak, aby tieto dôležité informácie mali svoje pevné miesto na obrazovke. Spomenuté zmeny je možné vidieť na obrázku 2.14, pôvodný wireframe zobrazuje obrázok 2.9.

Treťou a poslednou zmenou je pridanie tlačidla na opustenie mapy, dôvodom je, že na niektorých zariadeniach nie sú hardwerové tlačidlá a tie softwerové nie sú vždy viditeľné (zobrazujú sa gestom), takto má užívateľ tlačidlo

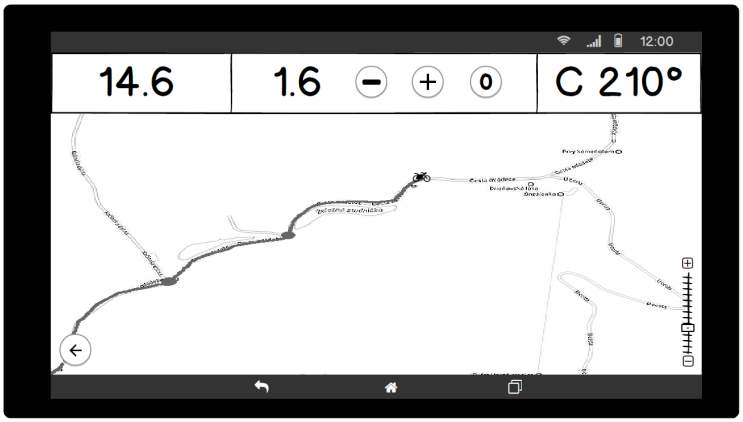


Obr. 2.13: Obrazovka pred spustením navigácie.



Obr. 2.14: Zmenené rozloženie prvkov obrazovky navigácie.

na opustenie mapy vždy prístupné. Zmena je ukázaná na obrázku 2.15.



Obr. 2.15: Mapa v navigácii – pridané tlačidlo “späť”.

Kapitola 3

Návrh

3.1 Android

Pred samotným návrhom aplikácie určenej pre operačný systém Android sa treba oboznámiť s možnosťami, ktoré sú pre vývoj dostupné. Android aplikácie je možné všeobecne rozdeliť na tri druhy a to natívne, hybridné a webové aplikácie. Prvý druh sú aplikácie vyvíjané použitím Android Framework, implementačným jazykom je Java, alebo nový oficiálne podporovaný jazyk Kotlin. Tieto aplikácie sa vyznačujú lepšou spoluprácou s hardwerom zariadenia a často sú užívateľsky najprívetivejšie, keďže dokážu poskytnúť viac a lepšie funkcie ako ostatné, na druhej strane sú zložitejšie a časovo náročnejšie na vývoj a údržbu. Hybridné aplikácie sú vyvíjané použitím niektorého z multiplatformných frameworkov, kedy je celá aplikácia, alebo aspoň jej veľká časť, implementovaná použitím jedného programovacieho jazyka a frameworku, ktorý následne dokáže vygenerovať natívne aplikácie pre viacero mobilných, či webových platforiem. Samozrejme takýto vývoj je obmedzený na schopnosti použitého frameworku, na druhej strane pre jednoduchšie aplikácie je tento druh výhodný a šetrí potrebný čas a peniaze na vývoj. Posledným druhom sú webové aplikácie, pod ktorými je možné rozumieť akýkoľvek web, ktorý sa prispôsobí danému zariadeniu a na jeho spustenie stačí webový prehliadač. Do tejto kategórie spadajú aj aplikácie či už natívne, alebo hybridné, ktoré však okrem základných ovládacích prvkov všetok obsah zobrazujú v takzvaných WebView. [6]

Na vývoj aplikácie Roadbook bol vybraný prvý spôsob – natívna aplikácia, najmä z dôvodu, že ide o náročnú aplikáciu, čo sa týka spolupráce s hardwerom zariadenia, napríklad práce s bluetooth či GPS, ktoré sú funkcie systému veľmi potrebné. Aplikácia bude teda vyvíjaná pomocou Android SDK, Android Framework, Google APIs a knižníc tretích strán. Hlavným programovacím jazykom bude Java.

3.1.1 Komponenty android aplikácie

Android framework ponúka 5 základných stavebných prvkov:

- **Activity** – aktivita, zjednodušene je možné povedať, že ide o hlavný prvok každej aplikácie, ktorý zabezpečuje zobrazenie UI komponentov na obrazovku a zachytáva používateľove interakcie s nimi. Každá aktivita má svoj životný cyklus (lifecycle), na ktorý je potrebné myslieť pri implementácii, nakoľko je práca v nesprávnom stave častým zdrojom pádov aplikácií. Každá aktivita musí byť definovaná v Manifeste.

- **Fragment** – je pevne zviazaný s aktivitou a rovnako tak slúži na zobrazenie UI komponentov. Jedna aktivita môže obsahovať viacero fragmentov, či už súčasne viditeľných alebo nie, rovnako tak fragment môže obsahovať ďalšie fragmenty. Výhodou jeho použitia je oddelenie častí aplikácie a možnosť použiť rovnaký fragment vo viacerých aktivitách.

- **Service** – komponent aplikácie, ktorý beží na pozadí a priamo nemení, nezasahuje do UI komponentov. Najčastejšie sa používa na náročnejšie výpočty alebo sťahovanie dát. Výhodou je možnosť, že service ostane bežať aj keď je aplikácia – aktuálna aktivita na pozadí.

- **Broadcast receiver** – už názov napovedá, že tento komponent slúži na prijímanie správ. Tieto správy môžu byť vysielané systémom android, napríklad správy o prichádzajúcom hovore, zmene stavu pripojenia na internet, upozornenie na začatie nabíjania alebo slabú batériu. Rovnako je možné zachytávať správy vysielané inými, alebo aj vlastnou aplikáciou.

- **Content provider** – komponent, ktorý poskytuje prístup k dátam iných aplikácií a samotného systému, ako napríklad kontakty, kalendár, ale aj samotné súbory v pamäti zariadenia.

3.2 Architektúra

Výber správnej architektúry je dôležitý pre každý väčší projekt a obzvlášť pre systémy s grafickým – užívateľským rozhraním. Pri mobilných aplikáciách, určených pre operačný systém Andorid, o to viac, že výber nie je nijak obmedzený, keďže Android framework a SDK neurčujú žiadne obmedzenia na architektúru aplikácie. Preto je aj častým problémom a chybou pri vývoji, najmä u začiatočníkov, implementovať všetky funkcie, pravidlá, prácu s UI vrámci aktivít prípadne fragmentov. To si uvedomuje aj samotný Google a na github pod názvom Andorid Architecture Blueprints [7] ponúka hneď niekoľko ukážkových kódov jednoduchej TODO aplikácie, implementovanej s rôznymi vzormi a použitými technológiami. Nájdeme tu MVP (Model-View-Presenter) a MVP s použitím rôznych technológií ako napríklad Dagger a RxJava. Ďalším

vzorom je MVVM (Model-View-ViewModel) s použitím DataBinding alebo implementácia s novými architektonickými komponentami predstavenými na Google I/O 2017. Pripravované sú aj ukážky MVP a MVVM v jazyku Kotlin. Ku každej ukážke – architektúre je podrobný popis jej komponentov a pravidiel ako ich správne použiť a tiež zaujímavé porovnania s ostatnými vzormi, napríklad v počte riadkov Java kódu, alebo v XML.

Po naštudovaní týchto ukážok a prečítaní ďalších článkov [8] a odporúčaní [9] bola vybraná architektúra MVVM a pri jej implementácii budú použité architektonické komponenty z android knižnice predstavenej pod názvom Android Architecture Components. Medzi komponenty patrí trieda ViewModel a LiveData, ktoré slúžia na komunikáciu medzi UI a ViewModel.

3.2.1 MVVM

Architektonický vzor MVVM pozostáva z troch hlavných komponentov a to Model, View a ViewModel, kde každý má svoju úlohu a pravidlá. Tento vzor sa zameriava na oddelenie GUI a business logiky, napomáha modularite, lepšiemu deleniu kódu a tým aj jeho jednoduchšiemu testovaniu. [10]

- **Model** – podobne ako pri iných vzoroch ako MVC a MVP, aj tu Model predstavuje dáta, stav – status a business pravidlá, čiže logiku. Model je nezávislý na užívateľskom rozhraní, teda View.
- **ViewModel** – slúži ako mediátor – lepidlo medzi modelom a view, pripravuje a poskytuje potrebné dáta pre view, prijíma od view užívateľské vstupy, pritom však neobsahuje žiadnu referenciu na view. Preto je potrebné, aby ViewModel poskytoval sadu dát – udalostí, na ktoré view reaguje a upravuje podľa nich užívateľské rozhranie. Na tento účel slúži už spomínaná trieda LiveData.
- **View** – spravuje UI komponenty, ich zobrazenie a vyplnenie dát získaných od ViewModel. V tomto prípade sú komponentami View aktivity, fragmenty a ďalšie triedy slúžiace na zobrazenie dát.

3.2.2 DataBinding

Alebo inak Data Binding Library, je knižnica od Google slúžiaca na písanie deklaratívnych XML layoutov, čím znižuje množstvo kódu potrebného na spojenie dát a UI komponentov. Do XML, ktoré slúži na definíciu UI prvkov, sa pridá špeciálny element určujúci napríklad POJO (plain-old java object) triedu a jej názov. Následne je pri ostatných prvkoch možné vlastnosti (dáta, výsledky metód) tohto objektu využiť, napríklad do textového pola vložiť text, ktorý jedna z metód danej triedy vracia. Ďalej je možné vložiť triedu, ktorá bude reagovať na používateľove vstupy – kliknutia, čím sa dá v kóde aktivít a fragmentov vyhnúť zdĺhavému definovaniu reakcií na vstupy. [11]

Vrámci tejto aplikácie bude knižnica na DataBinding použitá iba v niektorých špeciálnych XML layoutoch a to najmä tých, ktoré budú zobrazované v zoznamoch, napríklad zoznam inštrukcií trasy, alebo zoznam absolvovaných jázd. Dôvodom prečo práve v týchto prípadoch bude databinding implementovaný je, že tieto dáta sú počas celého procesu zobrazenia nemenné.

3.3 Návrh ukladania dát – databáza

Operačný systém android ponúka podporu pre ukladanie dát do SQLite databázy nachádzajúcej sa priamo v zariadení. V Android SDK tak nájdeme API slúžiace na interakciu s databázou. Pre náročnosť, najmä časovú, priameho použitia tohto API bolo vytvorených viacero knižníc na uľahčenie práce s databázou. Medzi takéto riešenia patria aj knižnice implementujúce princíp ORM (Object-relational mapping). Jedná sa o techniku mapovania – konverzie dát medzi relačnou databázou a objektovo orientovaným programovacím jazykom. Výhodou takéhoto riešenia je, že vývojárom aplikácií umožňuje ukladanie a získavanie dát z SQLite databázy bez písania SQL dotazov, tí tak pracujú s objektami – entitami a knižnica zabezpečuje ich perzistenciu. [12]

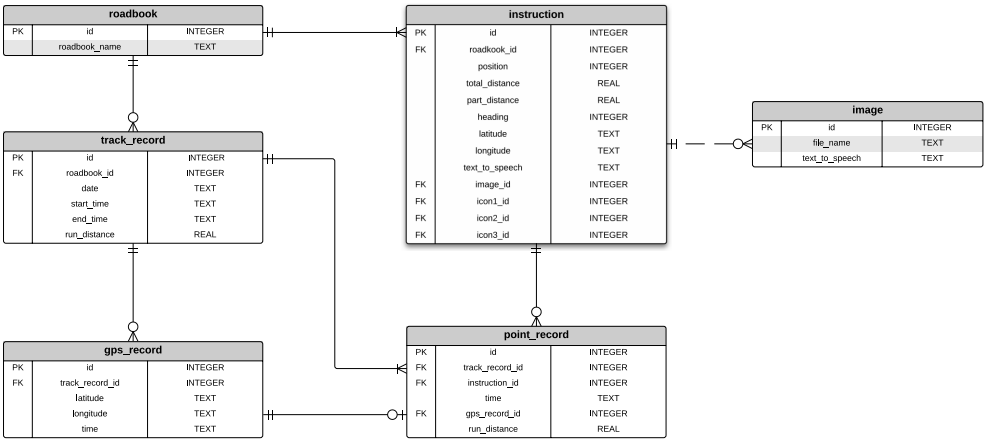
Medzi takéto knižnice patrí napríklad Room Persistence Library od spoločnosti Google, ktorá bola predstavená len nedávno na Google I/O 2017. Ďalšou je knižnica greenDao od spoločnosti Greenrobot, ide o open-source projekt. Práve táto knižnica bude použitá pri navrhovanej aplikácii, nakoľko autor práce túto knižnicu už používal a má s jej využitím pri implementácii skúsenosti.

Mapovanie entít – java objektov funguje na princípe anotácií, ktoré ponúka daná knižnica. Tak je možné definovať jednotlivé atribúty entít a tiež ich vzťahy. [13]

Pri návrhu dát – entít, ktoré aplikácia potrebuje uchovávať v zariadení bol vytvorený ER diagram, ktorý je zobrazený na obrázku 3.1. Podľa neho sa pri implementácii vytvoria entity – POJO objekty s anotáciami pre ORM knižnicu, ktorá už zabezpečí vytvorenie tabuliek v SQLite databáze a následnú prácu s dátami.

3.4 Triedy diagram entít

Z entít uvedených v ER diagrame a analytických tried popísaných v predošlej kapitole bude vytvorený diagram tried spolu s atribútmi, ich typom a vzťahmi. Cieľom pre vytvorenie tohoto modelu je získanie presne špecifikovaných tried a ich atribútov na takom stupni, aby ich bolo možné implementovať. Tieto triedy budú pri implementácii rozšírené o anotácie potrebné pre ORM knižnicu – greenDao tak, aby bola zabezpečená ich perzistencia. Model tried je uvedený na obrázku 3.2.



Obr. 3.1: ER diagram.

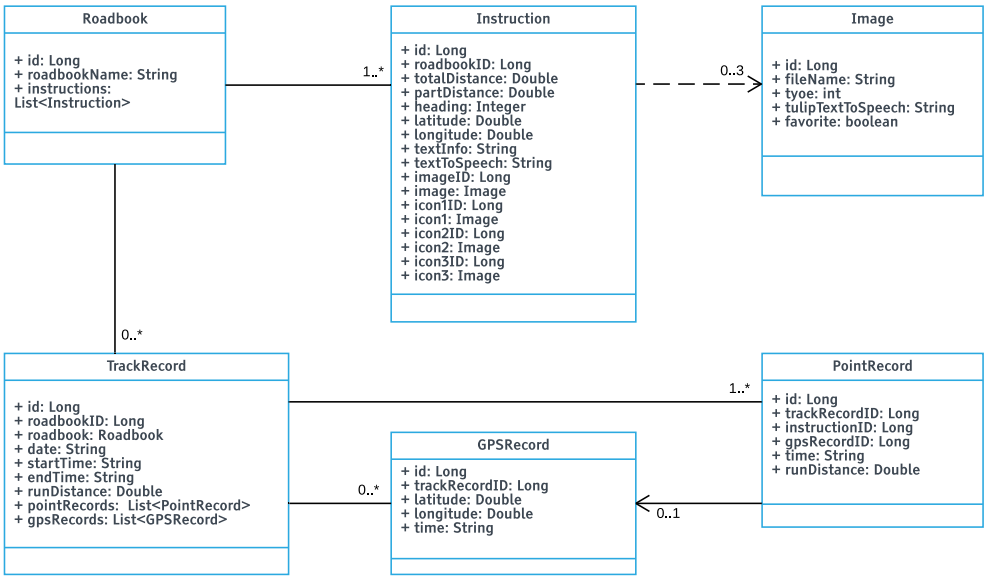
Navrhované entity je možné rozdeliť na dva typy, a síce triedy obsahujúce dáta o roadbook (Roadbook, Instruction, Tulip) a triedy s dátami zaznamenanými počas navigácie (TrackRecord, PointRecord, GPSRecord).

Entity:

- Trieda **Roadbook** obsahuje názov trasy a zoznam jej inštrukcií.
- **Instruction** je entita obsahujúca všetky dáta k jednej zmene na trase.
- **Tulip** predstavuje entitu, ktoré má informáciu o jednom obrázku – nákrese situácie.
- Trieda **TrackRecord** obsahuje údaje o jednej absolvovanej trase, kedy navigácia začala a skončila, koľko kilometrov jazdec prešiel. Ďalej pozostáva z listov záznamov dát jednotlivých inštrukcií a GPS polôh.
- **PointRecord** je entita predstavujúca záznam absolvovania jednej inštrukcie, jej čas, prejdenú vzdialenosť a zaznamenanú polohu.
- Trieda **GPSRecord** obsahuje jeden záznam polohy, získanej počas navigácie a čas jej zaznamenania, môže sa vzťahovať k PointRecordu.

3.5 Balíky aplikácie

Cieľom návrhu balíkov vyvíjanej aplikácie je premyslenie a vytvorenie tried s ohľadom na zvolenú MVVM architektúru tak, aby ešte pred samotnou im-



Obr. 3.2: Model tried.

plementáciou bolo jasne definované rozloženie tried do príslušných vrstiev architektúry. Navrhnuté budú najdôležitejšie a zároveň aj najzložitejšie časti systému, ktorými sú vytváranie trasy, navigácia a štatistiky jázd. Každý proces v systéme bude prebiehať vrámci jednej android aktivity, ktoré patrí do view vrstvy. Aby sa o všetku prácu nestarala iba táto jedna aktivita, je potrebné pridať ďalšie fragmenty, ktoré sa spravidla starajú o jednu ucelenú časť, napríklad zobrazenie zoznamu inštrukcií pri vytváraní roadbook. Tým sa dosiahne väčšej modularity systému a kód aplikácie bude lepšie udržiavateľný. Pre každý balík bude navrhnutá jedna trieda predstavujúca vrstvu model, dôvodom prečo sa priamo nepoužijú jednotlivé už navrhnuté entity je skutočnosť, že procesy potrebujú pracovať s viac ako jednou entitou naraz.

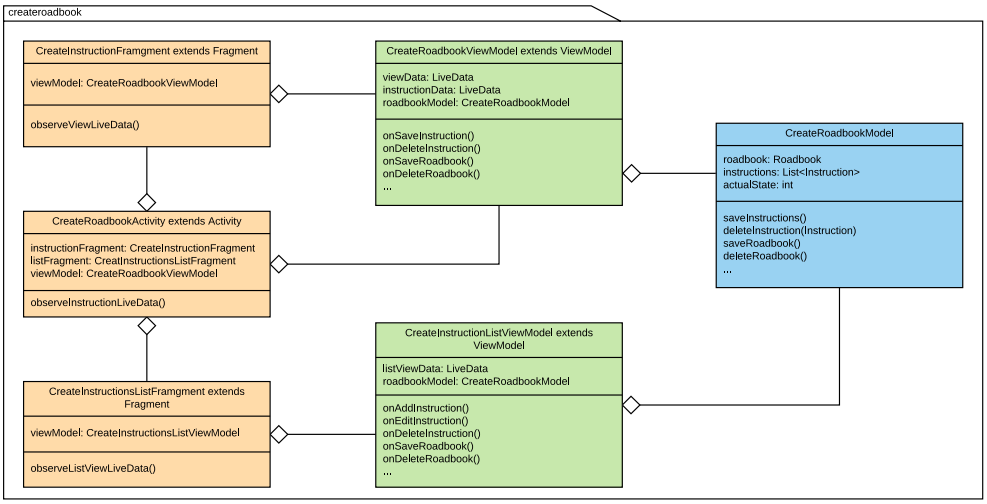
3.5.1 Balík Vytvárenia roadbook

Class diagram znázorňujúci triedy podieľajúce sa na procese vytvorenia itineráru sú zobrazené na obrázku 3.3.

Medzi triedy patriace do view komponentu v zvolenej architektúre MVVM patrí aktivita CreateRoadbookActivity a fragmenty CreateInstructionFragment a CreateInstructionsListFragment.

Vrstvu ViewModel reprezentujú rovno dve triedy, CreateRoadbookViewModel a pre spomínaný list fragment je to CreateRoadbookInstructionsListViewModel.

Tretiu zložku architektúry, model, predstavuje trieda CreateRoadbookModel, ktorá pracuje s entitami Roadbook a Instruction. Obsahuje logiku a



Obr. 3.3: Model balíka Vytváranie roadbook.

metódy na pridávanie, ukladanie, úpravu inštrukcií a tiež celej trasy.

3.5.2 Balík Navigácia

Základom každého procesu je android aktivita, v prípade navigácie je to NavigationActivity, ktorá s fragmentom NavigationInstructionsListFragment zobrazujúcim zoznam inštrukcií, reprezentuje View vrstvu.

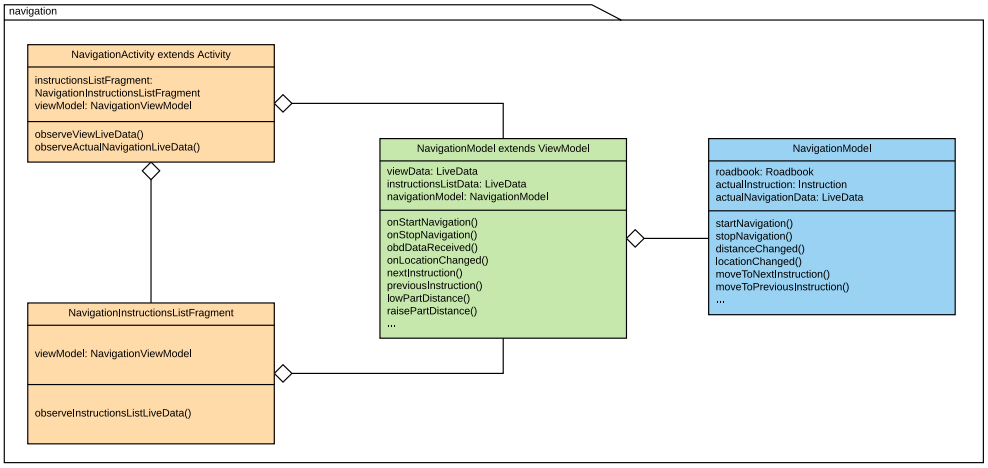
ViewModel zodpovedný za poskytovanie dát a notifikáciu na ich zmeny pre aktivitu a fragment je NavigationViewModel.

Zaznamenávanie prejdených vzdialeností, sledovanie aktuálnej polohy a všetka ďalšia logika aplikácie, potrebná na správne navigovanie užívateľa, je v triede NavigationModel. Tá pracuje v podstate so všetkými entitami systému, ako s triedami reprezentujúcimi trasu, tak aj s entitami slúžiacimi na zaznamenávanie dát počas navigácie. Model balíka navigácie je zobrazený na obrázku 3.4.

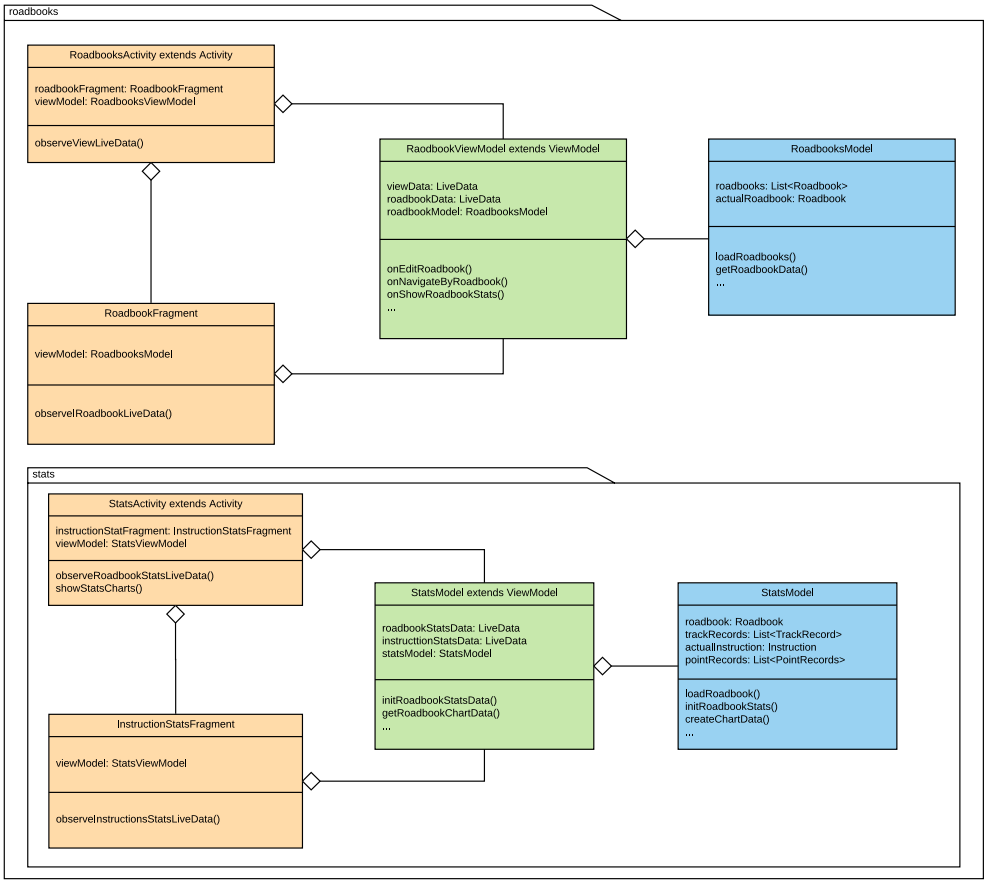
3.5.3 Balík Roadbooks

Táto časť aplikácie slúži na zobrazenie zoznamu trás uložených v zariadení a tiež na vykreslenie detailu jednej trasy a jej inštrukcií. Za vykreslenie zodpovedá aktivita RoadbooksActivity, detail trasy zobrazuje fragment RoadbookFragment. Obom týmto triedam patriacim do vrstvy View odovzdáva dáta trieda RoadbooksViewModel. O načítanie potrebných entít sa stará RoadbooksModel.

Model balíka Roadbook aj s balíkom Štatistiky, ktorý je jeho súčasťou, zobrazuje obrázok 3.5.



Obr. 3.4: Model balíka Navigácia.



Obr. 3.5: Model balíka Roadbooks.

3.5.3.1 Balík Štatistiky

Tento balík je súčasťou balíka Roadbook, ktorý zobrazuje obrázok 3.5 a ako názov napovedá ide o časť systému zodpovedajúcu za výpočet a zobrazenie štatistík k absolvovaným navigáciam na užívateľových trasách. View vrstvu reprezentujú aktivita StatsActivity a fragment InstructionStatsFragment. Spojivo medzi View a Model predstavuje trieda StatsViewModel. Načítanie a spracovanie dát o jazdách zabezpečuje StatsModel.

3.6 Návrh procesu navigácie

Pri navigácií pomocou roadbook je dôležité správne meranie prejdenej vzdialenosti. Systém bude poskytovať dva spôsoby merania.

Prvým spôsobom je získavanie prejdenej vzdialenosti z dát, ktoré poskytuje riadiaca jednotka vozidla, na ktorú je možné pripojiť zariadenie ELM327, s ktorým bude aplikácia komunikovať cez bluetooth rozhranie. OBDII protokol poskytuje dopyt na prejdenú vzdialenosť od posledného vymazania chybových hlásení. Táto hodnota je však udávaná s presnosťou na kilometre, čo pre účely aplikácie nie je dostačujúce, preto sa vzdialenosť bude vypočítavať z času a aktuálnej rýchlosti vozidla, ktorá sa dá získať ďalším dopytom. Dôležitými parametrami pri výpočte bude čas medzi jednotlivými dopytmi na riadiacu jednotku, čím častejšie sa ich podarí odoslať a spracovať, tým presnejší bude výpočet vzdialenosti.

Druhým spôsobom je využitie sledovania aktuálnej polohy a výpočet vzdialenosti medzi dvoma zaznamenanými polohami, na ktorý bude použitá funkcia distanceBetween poskytovaná Android framework v balíku Location. Pri tomto spôsobe bude dôležité získavať dáta čo najčastejšie, no okrem toho bude záležať aj na presnosti získaných lokácií, preto bude aplikáciou požadované zapnutie najpresnejšieho spôsobu lokácie podporovaného zariadením.

Kapitola 4

Implementácia

Táto kapitola pojednáva o priebehu implementácie aplikácie navrhnutej v predošlých častiach. Budú ukázané použité knižnice, technológie a práca s nimi.

Na implementáciu bol vybraný Android framework s použitím programovacieho jazyka Java. Na uľahčenie práce bolo použité vývojárske prostredie Android Studio. Dôvodom spomenutých technológií je viacročná skúsenosť autora práce s vývojom aplikácií pomocou ich použitia.

4.1 Knižnice

Pri implementácii bolo použitých viacero knižníc určených na riešenie rôznych problémov súvisiacich s potrebami aplikácie.

4.1.1 Support Library

Android Support Library už nie je len jedna knižnica, ale vyvinula sa do súboru knižníc. Tie slúžia na spätnú kompatibilitu, čo znamená, že vývojárom ponúkajú komponenty predstavené v novších verziách androidu a tí tak nemusia zvyšovať minimálnu verziu operačného systému na ktorom bude aplikácia bežať. [6]

4.1.2 Google Play Services

Google play services je balík služieb a APIs od Google pre Android, ktoré sú nainštalované na každom zariadení s balíkom google služieb. Týmto spôsobom Google môže svoje nové API dostať do starších zariadení bez nutnosti aktualizácie operačného systému. Medzi služby patria napríklad Google mapy, drive, účty atď. [14]

4.1.3 GreenDao

Knižnica GreenDao [13] a jej účel už boli spomenuté pri návrhu ukladania dát vo vyvíjanej aplikácii.

Použitie knižnice spočíva v anotovaní entity, ktorú chceme do databázy ukladať. Na ukážke kódu 4.1 je možné vidieť entitu **Roadbook**. Hlavnou anotáciou je *@Entity*, ktorou knižnici určíme, že má túto entitu ukladať do databázy. Ďalej *@Id(autoincrement = true)* označuje primárny kľúč danej entity, knižnica momentálne podporuje iba typy *long* a *Long* ako primárne kľúče. Položkou *autoincrement* je určené, že sa má id automaticky zvyšovať pri pridaní entity do databázy. Anotáciu *@NotNull* sa určí obmedzenie na daný parameter, teda pri ukladaní entity musí byť daný atribút inicializovaný.

Ďalšou použitou anotáciu je *@ToMany(referencedJoinProperty = "roadbookID")*, ktorá slúži na modelovanie relácie jedna k mnoho medzi entitami. Tento konkrétny príklad hovorí, že entita **Roadbook** obsahuje zoznam entít **PersistentInstruction**, kedy každá jedna má parameter *roadbookID*, podľa ktorého je relácia riešená. Pre reláciu jedna k jednej slúži anotácia *@ToOne*.

Knižnica po builde aplikácie vygeneruje do entít metódy set, get a konštruktory. Tiež pre každú entitu vygeneruje triedu s postfixom *Dao*, ktorá slúži na ukladanie, získavanie, mazanie a úpravu danej entity v databáze. K týmto triedam je možné pristupovať cez inštanciu triedy **DaoSession**, ktorú je vhodné inicializovať pri štarte aplikácie. Na jej získanie slúži trieda **DaoMaster**, ktorá vytvorí databázu, v prípade prvého štartu aplikácie po nainštalovaní, alebo sa pripojí na už existujúcu. [15]

```
@Entity
public class Roadbook {

    @Id(autoincrement = true)
    private Long id;

    @NotNull
    private String roadbookName;

    private boolean workingSave;

    @ToMany(referencedJoinProperty = "roadbookID")
    private List<PersistentInstruction> persistentInstructions;
}
```

Zdrojový kód 4.1: Ukážka anotácií entity Roadbook pomocou knižnice greenDao.

4.1.4 Glide

Knižnica Glide [16] uľahčuje načítanie obrázkov z internetu alebo pamäte zariadenia, na čo je aj v aplikácií použitá. Knižnica ponúka automatické prispôsobenie veľkosti načítavaného obrázku k veľkosti *ImageView*, do ktorého bude obrázok vykreslený, ďalej tiež ponúka rôzne cache stratégie. Jednoduchosť načítania obrázku s použitím Glide knižnice je možné vidieť na ukážke kódu 4.2.

4.1.5 Material dialogs

Knižnica Material dialogs [17] slúži na jednoduché a rýchle zobrazenie dialógových okien s pripraveným material design štýlom. V ukážke kódu 4.3 jo zobrazené ako jednoducho sa dialóg okno vytvára a ako je možné definovať akcie reagujúce na pozitívnu a negatívnu odpoveď.

4.1.6 HelloCharts

HelloCharts knižnica [18] ponúka sadu rôznych prispôsobiteľných grafov. Jej použitie nie je najjednoduchšie kvôli slabej dokumentácii a tak je potrebné správny postup implementácie hľadať v ukážkovom kóde.

V aplikácii sa zobrazujú čiarové grafy časov absolvovaných jázd a inštrukcií. V ukážke kódu 4.5 je zobrazené definovanie grafu v XML layout a v ukážke 4.5 odovzdanie dát tomuto elementu. Knižnica ponúka široké možnosti prispôsobenia grafu, ako napríklad zmena farby, štýl bodov, možnosť priblíženia, posunutia grafu, nastavenie šírky čiary a ďalšie.

4.2 Layout

Android framework ponúka niekoľko layoutov – UI elementov, ktoré rozširujú triedu **ViewGroup**, tieto triedy môžu obsahovať ďalšie (vnorené) layouty, preto slúžia na umiestňovanie konkrétnych UI komponentov, ako napríklad textové polia, či tlačidlá. Tieto komponenty rozširujú triedu **View**. [6]

Základné layouty:

```
public void loadImage(Context context, String imageFilePath,
    ImageView imageView) {
   Glide.with(context)
        .load(imageFilePath)
        .into(imageView);
}
```

Zdrojový kód 4.2: Ukážka kódu metódy slúžiacej na zobrazenia obrázku z pamäte zariadenia pomocou knižnice Glide.

- **LinearLayout** – radí vnorené prvky do horizontálneho alebo vertikálneho radu.
- **RelativeLayout** – umožňuje umiestňovať prvky relatívne medzi sebou, alebo rodičom, napríklad tlačidlo bude pod textovým polom, ktoré je na vrchu rodičovského layoutu.
- **FrameLayout** – je špeciálny layout, ktorý sa používa na zobrazenie jedného priameho potomka, pretože umiestňovanie viacerých je komplikované a niekedy až nemožné, tak, aby sa viaceré prvky neprekrývali. V aplikácii sa používa ako kontajner – miesto pre vkladanie rôznych fragmentov v runtime.
- **ConstraintLayout** – je pomerne novým layoutom v android frameworku, podobný s RelativeLayout, kde všetky prvky sú umiestnené podľa vzťahov medzi nimi a rodičovským elementom. Na rozdiel od svojho predchodcu je ConstraintLayout flexibilnejší a umožňuje vytvárať komplexnejšie layouty bez potreby použitia vnorenia ďalších **ViewGroup** elementov. Taktiež je jednoduchší na použitie, nakoľko Android Studio poskytuje Layout Editor, ktorý umožňuje vytvoriť celý layout pomocou drag-and-drop prostredia, bez potreby úpravy XML kódu. Práve tento layout je použitý na vytvorenie väčšiny UI v aplikácii. [19]
- **CoordinationLayout** – je vylepšením FrameLayout-u, používa sa ako top level layout aplikácie, alebo ako kontajner umožňujúci jednoduché

```
MaterialDialog.Builder builder = new MaterialDialog
   .Builder(activity)
   .title(dialogTitleText)
   .content(dialogContentText)
   .positiveText(dialogPositiveButtonText)
   .onPositive(new MaterialDialog.SingleButtonCallback() {
     @Override
         public void onClick(MaterialDialog dialog, DialogAction
             which) {
                 onDialogPositiveAction();
         }
   })
   .onNegative(new MaterialDialog.SingleButtonCallback() {
     @Override
     public void onClick(MaterialDialog dialog, DialogAction
         which) {
       onDialogNegativeAction();
     }
   });
MaterialDialog dialogFragment = builder.show();
```

Zdrojový kód 4.3: Ukážka zobrazenia dialógu a reakcie na jeho akcie použitím knižnice MaterialDialog.

definovanie interakcií medzi potomkami. Konkrétnym príkladom môže byť **Toolbar** (panel nástrojov, umiestnený vo vrchnej časti obrazovky aplikácie), ktorý sa zmenšuje a zväčšuje počas skrolovania zoznamu prvkov pod ním. Keď je panel zväčšený, môže zobrazovať dodatočné informácie, alebo poskytovať viaceré akcie, ktoré sú inak pri skrolovaní zoznamu skryté a tak uvoľňujú miesto na obrazovke. [20]

Existuje veľa ďalších layouts so špecifickejším zameraním použitia, medzi také patrí napríklad **DrawerLayout** spolu s **NavigationView**, ktoré slúžia na zobrazenie bočného menu aplikácie. Ďalej **CardView**, ktoré je špeciálnym FrameLayout-om so zaoblenými rohmi a tieňovaním okolo okrajov.

4.3 Architecture Components

Súbor android knižníc pod názvom Architecture Components bol spomenutý už v návrhu aplikácie. V tejto časti budú priblížené jednotlivé komponenty a ich využitie pri implementácii aplikácie.

4.3.1 Lifecycle-Aware components

Tieto komponenty vykonávajú akcie podľa zmien životného cyklu v inom komponente, napríklad aktivite alebo fragmente. Pomáhajú vytvárať lepšie organizovaný, kratší a ľahšie udržateľný kód. Medzi základné komponenty patrí trieda **Lifecycle**, ktorá drží stav životného cyklu komponentu a umožňuje iným objektom tento stav pozorovať (observe). Ďalšími sú rozhrania **LifecycleOwner** a **LifecycleObserver**. Triedy implementujúce prvé rozhranie poskytujú stav životného cyklu triedam implementujúcim druhé spomínané rozhranie, ktoré tento cyklus sledujú a podľa neho vykonávajú potrebné úkony. Medzi aktivity implementujúce **LifecycleOwner** patrí **LifecycleActivity** z balíka *android.arch.lifecycle* a **AppCompactActivity** zo Support Library v7:26+, rovnako tak fragmenty **LifecycleFragment** z ba-

```
<lecho.lib.hellocharts.view.LineChartView
   android:id="@+id/stats_graph_tracks_time"
   android:layout_width="0dp"
   android:layout_height="200dp"
   android:layout_marginEnd="8dp"
   android:layout_marginStart="8dp"
   android:layout_marginTop="16dp"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toBottomOf="@id/stats_roadbook_name"
      />
```

Zdrojový kód 4.4: Ukážka definovania elementu grafu z knižnice HelloCharts.

líka *android.arch.lifecycle*, respektíve **Fragment** zo Support Library v4:26+. [21]

V aplikácii boli použité aktivity dediace z **AppCompactActivity** a fragmenty z triedy **Fragment**. Dôvodom pre tento výber je ich podpora starších verzií operačného systému android.

4.3.2 ViewModel

Trieda **ViewModel** z už spomínaného balíka *android.arch.lifecycle* je navrhnutá na uchovávanie a spravovanie dát súvisiacich s užívateľským prostredím. Dáta uchovávané v tejto triede prežívajú konfiguračné zmeny aktivít, medzi ktoré patrí napríklad zmena orientácie zariadenia, na rozdiel od tých uchovávaných v aktivitách a fragmentoch. Ďalšou výhodou komponentu je skutočnosť, že je v ňom možné spravovať asynchrónne volania, napríklad do databázy, ktoré sú rovnako ako dáta oddelené a teda uchránené od konfiguračných zmien. [22] [23]

V aplikácii sú triedy odvodené od **ViewModel** použité na uchovávanie tried predstavujúcich vrstvu model vo zvolenej architektúre MVVM. Tieto triedy poskytujú aktivitám, fragmentom a ďalším triedam z view vrstvy dáta, ktoré potrebujú zobraziť a vystavujú metódy reagujúce na užívateľské vstupy.

4.3.3 LiveData

Trieda **LiveData** slúži ako pozorovateľný kontajner dát, čo znamená, že uchováva inú triedu a upozorňuje iné prihlásené triedy – pozorovateľov (observer)

```
LineChartView chart = findViewById(R.id.stats_graph_tracks_time);
chart.setContainerScrollEnabled(true,
    ContainerScrollType.HORIZONTAL);

ChartData chartData = viewModel.getPointsValuesForTrackRecords();
Line line = new Line(chartData.points);
line.setHasLines(true);
line.setHasLabels(true);
List<Line> lines = new ArrayList<>(1);
lines.add(line);

LineChartData lineChartData = new LineChartData(lines);
Axis axisX = new Axis();
axisX.setValues(chartData.axisValues);
Axis axisY = new Axis();
axisY.setHasLines(true);

chart.setLineChartData(lineChartData);
```

Zdrojový kód 4.5: Ukážka použitia knižnice HelloCharts, zobrazenie čiarového grafu.

na zmeny v týchto dátach. Hlavnou výhodou použitia **LiveData** je skutočnosť, že ide o lifecycle-aware komponent, čo znamená, že táto trieda rešpektuje životný cyklus svojich pozorovateľov, ktorí musia implementovať **LifecycleOwner** interface. Tým je zabezpečené, že zmeny dát nebudú upravovať UI komponenty, ktoré napríklad ešte nie sú inicializované, alebo je aktivita už zastavená, napríklad na pozadí. [22]

Pri implementácii bola použitá trieda **MutableLiveData**, ktorá rozširuje **LiveData** a ponúka metódy na opätovné nastavenie triedy, ktorú uchováva. Medzi triedy pozorujúce tieto dáta patria už spomínané aktivity a fragmenty rozširujúce **LifecycleOwner**.

Medzi dáta uchovávané v tejto triede patria údaje o viditeľnosti rôznych UI komponentov vrámci jedného procesu, príkladom môže byť údaj o viditeľnosti zoznamu inštrukcií alebo mapového podkladu počas navigácie. Ďalším príkladom sú aktuálne dáta získané z riadiacej jednotky vozidla, alebo zo zachytávania aktuálnej polohy zariadenia. Triedu **MutableLiveData** v aplikácii uchovávajú vrstvy ViewModel, prípadne Model.

4.4 Použitie DataBinding v RecycleViewAdapter

Ako už bolo popísané v návrhu architektúry, technológia databinding slúži na prepojenie popisu grafického rozhrania pomocou XML a tried – java objektov, uchovávajúcich dáta, ktoré sa majú zobraziť.

Táto knižnica je v aplikácii použitá pri implementácii UI zoznamov – listov dát, napríklad zoznam inštrukcií pri navigácii, či vytváraní trasy, alebo zoznam absolvovaných jázd a ich časov. Pre definovanie kontajneru obsahujúceho list prvkov sa v aplikácii používa **RecyclerView** zo Support Library v7. Tento element je následne v aktivite, alebo fragmente načítaný a je mu potrebné nastaviť adaptér pre dáta rozširujúci triedu **RecyclerView.Adapter<RecyclerView.ViewHolder>**, celé nastavenie **RecyclerView** je zobrazené na ukážke kódu 4.6. Táto trieda slúži na inicializáciu a ďalšiu úpravu zoznamu prvkov. Pre každý jeden prvok je vytvorená jedna inštancia triedy rozširujúca triedu **RecyclerView.ViewHolder**, ktorá vykonáva samotný binding. [24]

Existujú tri hlavné metódy, ktoré adaptér rozširujúci triedu **RecyclerView.Adapter** musí definovať:

- *onCreateViewHolder* – vytvára **RecyclerView.ViewHolder**, ktorému odovzdáva view vytvorené z XML súboru. Toto XML obsahuje elementy popisujúce UI komponenty predstavujúce jeden prvok zoznamu. Časť tohto XML, kde je použitý databinding, je zobrazená v ukážke kódu 4.7. Hlavným elementom layoutu s databinding musí byť *<layout>* a pre definovanie tried s dátami slúži element *<data>* a v ňom vnorený element *variable* popisujúci jednu konkrétnu premennú zavedenú

do XML. Konkrétne sa definuje typ a názov, ktorý bude ďalej použitý. Využitie zavedenej java triedy môže následne vyzerať následovne *android:text="@instruction.getRoundTotalDistance()*, kde výsledok metódy **getRoundTotalDistance** vracajúcej *String* je nastavený do textového pola.

- *onBindViewHolder* – v tejto metóde sa nakoniec odohráva samotné nastavenie – vloženie inštancií do layoutu, ktoré je stručné a jednoduché. DataBinding knižnica pre každý layout obsahujúci element *<data>* vytvorí triedu s názvom layoutu a postfixom **Binding**, ktorá ponúka *set* metódy pre triedy definované v elemente *variable*. Získanie a použitie tejto triedy je zobrazené na ukážke kódu 4.8.

- *getItemCount* – vracia počet prvkov zoznamu.

4.5 OBDII a komunikácia so zariadením ELM327

Získavanie dát zo zariadenia ELM327, ktoré sa pripája do diagnostickej zásuvky vo vozidle, prebieha formou dopyt – odpoveď. Vybrané bolo zariadenie s bluetooth komunikáciou, ktorá bola aj implementovaná. Dôvodom zvolenia tohto typu bola predošlá skúsenosť s daným zariadením a implemntáciou pripojenia a samotnej komunikácie s ním.

Zariadenie ELM327 rozpoznáva dva typy dopytov, jednými sú príkazy pre samotné zariadenie, ktoré slúžia na jeho nastavenie, napríklad určenie formátovania odpovedí. Tieto dopyty začínajú s predponou AT. Druhým typom sú príkazy určené pre diagnostickú jednotku vozidla, ktorými sa získavajú dáta. Tieto dopyty sa skladajú z dvoch hexadecimálnych čísiel, pričom prvé určuje mód – skupinu príkazov, ktorých je v štandarde definovaných 9, druhé samotný príkaz. Pri návrhu spôsobu výpočtu prejdenej vzdialenosti bolo zistené, že je potrebné získavať aktuálnu rýchlosť vozidla, a pre korekcie vo výpočte aj prejdenú vzdialenosť, ktorá ako už bolo uvedené je uvádzaná v kilometroch.

```
RecyclerView instructionsRecycleView =
    findViewById(R.id.nav_instructions_recyclerview);
instructionsRecycleView.setHasFixedSize(true);
instructionsRecycleView.setLayoutManager(new
    LinearLayoutManager(getActivity()));
NavigationInstructionsRecycleViewAdapter instructionsListAdapter
    = new NavigationInstructionsRecycleViewAdapter(viewModel);
instructionsRecycleView.setAdapter(instructionsListAdapter);
```

Zdrojový kód 4.6: Ukážka nastavenia RecyclerView zobrazujúceho zoznam UI prvkov.

Dopyt pre aktuálnu rýchlosť je **010D** a pre prejdenú vzdialenosť **0131**, za každý dopyt je ešte potrebné pripojiť ukončovací znak `\r`. [25]

Komunikácia cez bluetooth je v aplikácii riešená pomocou android komponentu **Service**, ktorý sa stará o nadviazanie spojenia, pravidelné odoslanie dopytu a prijatie odpovede. Android Framework ponúka radu tried pre prácu s bluetooth zariadeniami pod názvom **Android Bluetooth APIs**. Na nadviazanie spojenia je potrebné poznať konkrétne zariadenie, na ktoré sa chceme pripojiť, to je reprezentované triedou **BluetoothDevice** a užívateľ ho určuje v nastaveniach aplikácie. Táto trieda ponúka metódu *createRfcommSocketToServiceRecord*, ktorá vracia inštanciu triedy **BluetoothSocket**, pomocou jej metódy *connect* sa vytvorí spojenie. Trieda ďalej poskytuje metódy *getInputStream* a *getOutputStream*, slúžiace na získanie výstupného, respektíve vstupného kanála, pomocou ktorých budú odosielané dopyty a prijímané odpovede. [26]

Odoslanie a spracovanie odpovede zo zariadenia je zobrazené na ukážke kódu 4.9. Vstupný stream sa číta pokiaľ nie je nájdený ukončovací znak odpovede, ktorým je >. Následne sa v odpovedi nájde postupnosť označujúca začiatok odpovede, tá je reprezentovaný číslom 41, po ktorom nasleduje medzera a číslo samotného dopytu. Po nich je potrebné načítať správny počet znakov, pretože odpovede sú rôzne dlhé a obsahujú viacero hodnôt. Ku každému dopytu je uvedený vzorec na výpočet požadovanej hodnoty za použitia získaných hodnôt.

4.6 Získavanie polohy zariadenia

Pre získavanie polohy zariadenia bola použitá Google Play services APIs, konkrétne triedy z balíka *com.google.android.gms.location*. Postup implementácie je celý výborne popísaný vo vývojárskej príručke androidu [27]. Pre získavanie polohy je možné nastaviť jej presnosť a čas jej obnovenia. V aplikácii bola použitá **PRIORITY_HIGH_ACCURACY**, ktorá vracia najpresnejšiu možnú polohu. Spustenie, odchytenie chýb a nastavenie požadovanej presnosti sú riešené v triede **LocationProvider**, ukážka jej inicializácie je zobrazená v ukážke kódu 4.10, ktorá tiež zobrazuje implemntáciu **LocationCallback**. Jej metóda *onLocationResult(LocationResult locationResult)* v nastavených intervaloch vracia novú polohu zariadenia, tá je ďalej odovzdaná triede implementujúcej **LocationProviderInterface**. [28]

4.7 TTS – Text-to-Speech

Android Framework ponúka triedu **TextToSpeech** na interakciu s TTS engine nastaveným v zariadení, ktorý slúži na syntézu textu do hovoreného slova, ktorý je možné prehrať. Implementácia pomocou spomínanej triedy je nezávislá od konkrétneho TTS engine používaného daným zariadením, na

ktorom aplikácia beží. Jej použitie je veľmi jednoduché, celý proces spočíva v inicializácii triedy. Konštruktor vyžaduje inštanciu implementujúcu rozhranie **TextToSpeech.OnInitListener**, ktoré obsahuje jedinú metódu *onInit(int status)*. Jej parameter *status* informuje o úspešnom, respektíve neúspešnom, inicializovaní. Po úspešnom inite je možné triedu začať používať. Trieda poskytuje niekoľko metód súvisiacich s nastavením enginu a samotným spustením čítania textu. V aplikácii je použitá jediná metóda a to *speak*, ktorej sa odovzdáva samotný text na prečítanie, režim pridania do fronty a prípadne ďalšie parametre, ktoré však aplikácia nepotrebuje. Režimy pridania sú dva a síce **TextToSpeech.QUEUE_ADD** a **TextToSpeech.QUEUE_FLUSH**. Prvý zmienený spôsobí pridanie textu na koniec fronty spravovanej TTS engine, a druhý túto frontu vyprázdni a spustí čítanie textu okamžite. Pri implementácii aplikácie je použitý druhý režim, nakoľko čítanie povelov nie je také časté, a tiež nový pokyn je väčšinou dôležitejší ako dokončenie predošlého povelu.

```
<layout
    xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto">

   <data>
     <variable
       name="instruction"
       type="sk.roadbook.entities.Instruction" />
   </data>

   <android.support.constraint.ConstraintLayout
     android:layout_width="match_parent"
     android:layout_height="match_parent">

   <TextView
     android:id="@+id/nav_inst_total_distance"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@{instruction.getRoundTotalDistance()}"
         ...  />

   <TextView
     android:id="@+id/nav_inst_part_distance"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@{instruction.getRoundPartDistance()}"
     ...  />

    ...

    </android.support.constraint.ConstraintLayout>
</layout>
```

Zdrojový kód 4.7: Ukážka použitia databindingu v XML súbore popisujúcom view.

```
@Override
public void onBindViewHolder(InstructionViewHolder holder, int
    position) {
   Instruction instruction =
       viewModel.getInstructionInList(position);
   NavigationInstructionListItemBinding binding =
       DataBindingUtil.bind(holder.itemView);  //itemView je
       instancia triedy View, inicializovana z~XML suboru
       v~metode onCreateViewHolder
   binding.setInstruction(instruction);
   binding.executePendingBindings();
}
```

Zdrojový kód 4.8: Ukážka metódy onBindViewHolder s nastavením premenných pre layout.

```
socketOutputStream.write(("4131\r").getBytes());
char c = '.';
String response = "";
while (c != '>' && socketInputStream.available() > 0) {
   int rInt = socketInputStream.read();
   if (rInt != 0)
     response += (char) rInt;
     c = (char) rInt;
   }
int index = response.indexOf("41 31");
String hexaString = response.substring(index + 6, index + 11);
String[] value = hexaString.split(" ");
double a = Integer.parseInt(value[0], 16);
double b = Integer.parseInt(value[1], 16);
double distance = (a * 256) + b;
```

Zdrojový kód 4.9: Ukážka odoslania dopytu, prijatia a spracovania odpovede zo zariadenia ELM327.

```
public void init(Activity activity, LocationProviderInterface
    navViewModel) {
   this.viewModel = navViewModel;
   mFusedLocationClient =
       LocationServices.getFusedLocationProviderClient(activity);
   mSettingsClient =
       LocationServices.getSettingsClient(activity);

   mLocationCallback = new LocationCallback() {
     @Override
     public void onLocationResult(LocationResult locationResult)
         {
       super.onLocationResult(locationResult);
       viewModel.onLocationChanged(mCurrentLocation);
   }};
   mLocationRequest = new LocationRequest();
   mLocationRequest
    .setInterval(UPDATE_INTERVAL_IN_MILLISECONDS);
   mLocationRequest
    .setFastestInterval(FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS);
   mLocationRequest
    .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
   LocationSettingsRequest.Builder builder = new
       LocationSettingsRequest.Builder();
   builder.addLocationRequest(mLocationRequest);
   mLocationSettingsRequest = builder.build();
}
```

Zdrojový kód 4.10: Ukážka inicializácie triedy zabezpečujúcej získavanie aktuálnej polohy zariadenia.

Kapitola 5

Testovanie

Testovanie je dôležitou súčasťou vývoja akéhokoľvek softweru, ktorou sa overuje jeho správnosť, funkčnosť, bezpečnosť, či výkon. Spôsobov a typov testovania je veľké množstvo, každý je vhodný pre iný typ systémov, alebo sa zameriava na iný jeho aspekt. Vo všeobecnosti je možné testovanie rozdeliť na dva druhy a síce automatické testovanie a testovanie vykonávané testerom, či samotným užívateľom.

Základ automatických testov vyvíjanej android aplikácie predstavujú takzvané lokálne testy, alebo inak unit testy, ktoré dopĺňajú inštrumentačné testy.

Ďalším spôsobom testovania je užívateľské testovanie zameraná na celkovú funkčnosť a užívateľské prostredie. To bolo tiež otestované už po jeho návrhu pomocou kognitívneho priechodu. Užívateľské testovanie prebiehalo v dvoch fázach, prvé bolo testovanie beta verzie s neúplnou funkčnosťou. Po úpravách, ktoré vzišli z testovania a doplnení funkčnosti prebehlo užívateľské testovanie alfa verzie aplikácie, ktoré bola aj nasadená na Google Play.

5.1 Lokálne (unit) testy

Pri vývoji natívnej android aplikácie sa pod pojmom lokálne testy rozumejú automatické testy spúšťané na JVM (Java Virtual Mechine), čiže bez potreby reálneho zariadenia, alebo emulátora s operačným systémom android. Preto je nimi možné testovať iba obmedzenú časť systému, čo je ale výhodou, a núti to developerov deliť kód tak aby ho bolo možné testovať.

5.2 Inštrumentačné testy

Inštrumentačné testy sú špeciálnym druhom unit testov pre operačný systém android. Tieto testy už nie sú spúšťané lokálne na JVM, ale musia byť spustené na reálnom zariadení, alebo na emulátore. Je tak možné testovať celú aplikáciu a jej UI komponenty.

Tento typ testov bol využitý na testovanie tried predstavujúcich ViewModel vrstvu aplikácie a predovšetkým nimi poskytovaných LiveData tried, ktoré View vrstva pozoruje (observe) a podľa nich mení zobrazené informácie, či celé UI komponenty.

5.3 Beta testovanie

Prvé testovanie aplikácie ešte s neúplnou funkcionalitou bolo vykonané iba s jedným testerom a to samotným autorom nápadu jej vytvorenia. Toto testovanie bolo uskutočnené najmä pre konzultáciu smerovania jej vývoja a overenie zatiaľ implementovanej funkcionality.

Tester mal za úlohu vytvoriť reálny roadbook, podľa ktorého sa bude následne navigovať. Pri vytváraní bolo odhalených niekoľko nedostatkov, ale väčšina z nich, až na jeden, sa netýkala užívateľského rozhrania, ale nesprávnych reakcií systému – chýb. Jedinou požiadavkou bola zmena spôsobu pridávania tulip tak, že kliknutie na konkrétny tulip spôsobí jeho pridanie ako hlavný nákres inštrukcie, a dlhý klik zobrazenie ostatných možností pre daný nákres. Táto požiadavka bola do finálnej aplikácie zapracovaná.

Inštrukcie a ich vzdialenosti tester pridával pomocou Google máp, čo sa pri navigácii prejavilo. Prejdené vzdialenosti boli merané pomocou GPS, čiže s určitou nepresnosťou sa počítalo, ale v kombinácii s nepresnými vzdialenosťami medzi bodmi získanými zo spomínaných máp, bola navigácia náročná, a bez neustálej úpravy prejdenej vzdialenosti a manuálnom prepínaní inštrukcií nemožná. Preto sa potvrdila potreba pridania funkcie vytvárania roadbook online, čiže zaznamenávanie polohy a prejdených vzdialeností pri prechode trasou a vytváranie jej jednotlivých inštrukcií. Táto funkcia bolo do aplikácie pridaná a otestovaná pri ďalšom testovaní.

Tester mal ešte jednu výhradu k spôsobu zobrazenia inštrukcií, požadované bolo viditeľnejšie odlíšenie aktuálnej inštrukcie voči po nej nasledujúcich, aj toto vylepšenie bolo do výslednej aplikácie implementované.

5.4 Užívateľské testovanie

Užívateľské testovanie je dôležitou zložkou overenia kvality a použiteľnosti akéhokoľvek softweru s užívateľským rozhraním. Testovanie uskutočňujú reálne osoby, tie sú volené podľa typu samotného systému, ktorý bude testovaný. Pre vyvíjanú aplikáciu je vhodné zvoliť testerov, ktorý majú skúsenosti s používaním roadbook.

Užívateľské testovanie uskutočňovali štyria testeri. Rovnako tak boli použité aj štyri rôzne zariadenia, dva telefóny (One Plus One, One Plus 5T) a dva tablety (Asus Memo Pad 7, Asus ZenPad 8).

5.4.1 Scenáre

5.4.1.1 Vytváranie roadbook offline

Zadanie pre testera:

1. Vytvorte roadbook v offline móde.
2. Pridajte k trase inštrukciu.
3. Vyplňte vzdialenosti, nákres situácie, polohu bodu, dodatočnú textovú informáciu a ikonu.
4. Uložte inštrukciu a pridajte niekoľko ďalších.
5. Pri jednej z inštrukcii vytvorte vlastný nákres.
6. K jednej z inštrukcii pridajte hlasový povel.
7. Uložte roadbook a pozrite si jeho detail.

Vyhodnotenie testu

Počas testovania bolo odhalených niekoľko nedostatkov. Testeri sa zhodli na tom, že by pri vzdialenostiach mali byť uvedené jednotky. Jeden z testerov navrhoval možnosť voľby jednotiek medzi metrami a kilometrami. Iný z testerov mal problém s pridaním dodatočnej ikony, respektíve nevedel akým spôsobom je možné ikonu pridať. Pokúsil sa pridať ju kliknutím na riadok pod textovým popisom, čiže na miesto, kde sa ikona neskôr zobrazí. Ďalší problém nastal pri vytváraní vlastného nákresu. Pri pokuse o jeho vytvorenie hľadal jeden z testerov túto možnosť na nesprávnom mieste – v záložke Vlastné hľadal možnosť pridania. Testeri ďalej odhalili niekoľko drobných nedostatkov, zväčša preklepov v popisoch.

5.4.1.2 Vytváranie roadbook online

V tomto teste bol roadbook vytváraný raz pomocou merania vzdialenosti s GPS a druhý krát cez OBDII. Scenáre sa líšia iba v prvom bode – nastavenie merania vzdialenosti.

Zadanie pre testera:

1. Nastavte meranie vzdialenosti pomocou GPS/OBDII.
2. Vytvorte roadbook v online móde.
3. Pridajte k trase inštrukciu.
4. Postupujte po trase do bodu inštrukcie.

5. Pridajte nákres situácie, prípadne aj dodatočnú textovú informáciu a ikonu.
6. Uložte inštrukciu a postupujte ďalej po trase k ďalšiemu bodu.
7. Rovnakým spôsobom pridávajte ďalšie inštrukcie, až do cieľa trasy.
8. V cieli uložte roadbook a pozrite si jeho detail.

Vyhodnotenie testu

Pri pridávaní inštrukcií má užívateľ dve možnosti, buď klikne na ikonu diskety alebo na ikonu ďalšieho kroku. Pri zvolení diskety aplikácia vráti užívateľa na obrazovku so zoznamom inštrukcií a kým nie je zvolené pridanie ďalšej inštrukcie, nemeria vzdialenosť. Testeri túto možnosť vyhodnotili ako nevhodnú, pretože pri vytváraní za jazdy preferujú druhú možnosť. Ďalej bolo odhalené, že pri pripojení zariadenia OBDII, sa nezobrazuje správne jeho názov.

5.4.1.3 Navigácia

V tomto teste bola spustená navigácia raz pomocou GPS a raz pomocou merania vzdialenosti z dát získaných cez protokol OBDII.

Zadanie pre testera:

1. Vyberte roadbook pre navigovanie.
2. Nastavte meranie vzdialenosti:
   - pomocou GPS.
   - pomocou OBDII a pridajte spárované zariadenie.
3. Počkajte na informácie o zapnutom GPS, respektíve úspešnom pripojení sa na OBDII zariadenie.
4. Spustite navigáciu.
5. Po dosiahnutí polohy inštrukcie sa posuňte na ďalšiu inštrukciu.
6. Postupujte po inštrukciách trasy až do cieľa.

Vyhodnotenie testu

Testeri pri navigácii nemali problém s používaním aplikácie. Jeden z testerov navrhoval, že by bolo vhodné pridať možnosť úpravy celkovej prejdenej vzdialenosti. Pri testovaní bolo odhalené, že meranie vzdialenosti pomocou OBDII nedosahuje takej presnosti ako meranie pomocou GPS. Táto nepresnosť je zapríčinená spôsobom merania vzdialenosti cez OBDII, ktoré, ako bolo

zistené už pri návrhu aplikácie, ponúka prejdenú vzdialenosť iba s presnosťou na kilometre a preto bol implementovaný výpočet pomocou aktuálnej rýchlosti a času s korekciou po kilometroch, no tá funguje iba pri dlhších úsekoch. Pri menších vzdialenostiach jednotlivých inštrukcií sa nepresnosť znásobuje a pri testovaní v cieli dosahovala niekoľko desiatok metrov voči meraniu vzdialenosti pomocou GPS, ktoré dosahovalo uspokojivú presnosť.

Testeri sa ďalej zhodli, že by aplikácia mohla ponúkať možnosť externého ovládania, ktoré by bolo realizované pomocou ovládača ponúkajúceho aspoň dve tlačidlá na prechod medzi inštrukciami.

5.4.1.4 Štatistiky absolvovanej trasy

Zadanie pre testera:

1. Zvoľte možnosť štatistiky jázd.
2. Zobrazte si štatistiky jedného roadbook.
3. Zobrazte si štatistiky jednej konkrétnej inštrukcie.

Vyhodnotenie testu

Jeden z testerov navrhoval, že by bolo vhodné pridať možnosť zobrazenia mapy trasy. Iné problémy pri testovaní nenastali.

5.5 Záver testovania

Testovanie, najme beta a užívateľské, odhalilo niekoľko nedostatkov a požiadaviek zo strany užívateľov. Všetky zistené chyby boli odstránené. Najväčšou zmenou v aplikácii bolo pridanie online vytvárania roadbook po beta testovaní, to vyžadovalo niekoľko väčších zmien v procese vytvárania. Aj po užívateľskom testovaní boli požiadavky užívateľov implementované do výslednej aplikácie, ako napríklad pridanie uvádzania jednotiek pri vzdialenostiach, zobrazovanie prejdenej trasy na mape v sekcii štatistík jázd, pridanie akcie plus do záložky Vlastné pre vytvorenie vlastného nákresu situácie, úprava celkovej prejdenej vzdialenosti pri navigácii. Všetky zmeny boli následne riadne otestované a výsledná aplikácia zverejnená na Google Play.

Záver

V tejto diplomovej práci bola navrhnutá a potom aj úspešne implementovaná a otestovaná aplikácia Roadbook pre OS Android. Aplikácia slúži na vytváranie a navigáciu podľa roadbook a je určená pre všetkých nadšencov a hobby jazdcov navigačných závodov, no nie len pre nich. Poskytuje tak inú možnosť ako drahé profesionálne riešenia alebo vlastnoručne vytvorené zariadenia.

V jednotlivých kapitolách práce je možné sledovať postup pri vývoji aplikácie. Najskôr analýzu existujúcich riešení, určenie požiadaviek, či definíciu prípadov použitia. Následne bolo navrhnuté užívateľské rozhranie a jeho otestovanie pomocou kognitívneho priechodu. Ďalším krokom bol návrh architektúry, ukladania dát aplikácie a jednotlivých jej častí. Po návrhu nasledovala implementácia a prvý test s užívateľom. Po dokončení implementácie bolo vykonané záverečné užívateľské testovanie, v ktorom bolo vznesených niekoľko požiadaviek a nájdených zopár nedostatkov. Tie boli ešte pred nahraním aplikácie na Google Play odstránené a doplnené.

Aplikácie je momentálne zverejnená v takzvanej alfa testovacej verzii na Google Play, kde je verejne prístupná skupine užívateľov – alfa testerov (https://play.google.com/store/apps/details?id=roadbook.com.roadbook). Po zozbieraní užívateľských recenzií, prípadných chybových hlásení a ich odstránení bude zverejnená produkčná verzia aplikácie.

Zadanie a stanovené ciele práce boli splnené a navyše aplikácia obsahuje niekoľko dodatočných funkcií, ktoré prinieslo najmä testovanie a úzka spolupráca s autorom nápadu, príkladom môže byť online vytváranie roadbook, či zobrazovanie prejdenej trasy v štatistikách jázd.

Budúcnosť a rozšírenia aplikácie

Ďalším krokom postupu bude zozbieranie recenzií a chybových hlásení z alfa testovacej verzie zverejnenej na Google Play. Po odstránení nedostatkov a prípadných menších rozšírení by som chcel aplikáciu vydať do produkčnej verzie.

Čo sa týka možných rozšírení aplikácie do budúcna, okrem vylepšení súčasných funkcií, by bolo vhodné pripraviť väčšie množstvo tulip a dodatočných ikon používajúcich sa pri rally roadbook. Ďalšími rozšíreniami by mohlo byť pridanie externého ovládača, ktoré bolo navrhnuté jedným testerom pri užívateľskom testovaní. Na trhu je niekoľko bluetooth ovládačov, ktoré by mohli byť vhodné na prepínanie inštrukcií a aj úpravu prejdenej vzdialenosti. Nakoľko sa meranie vzdialenosti pomocou OBDII ukázalo ako nepresné možným rozšírením by bolo získavať prejdenú vzdialenosť z iného externého zariadenia, napríklad odometra.

Literatúra

[1] Modelování uživatelů (personas). [cit. 2017-12-11]. Dostupné z: https://wikisofia.cz/wiki/Modelov%C3%A1n%C3%AD_u%C5%BEivatel%C5%AF_(personas)

[2] Personas. [cit. 2017-12-11]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/personas.html

[3] Bradáč, B. J.: *Diplomová práce - Mojeskolka.info - tvorba uzavřené komunitní sítě pro školky*. České vysoké učení technické v Praze, 2015. Dostupné z: https://dspace.cvut.cz/handle/10467/62999

[4] Návrh užívateľského prostredia. [cit. 2017-12-14]. Dostupné z: https://wikisofia.cz/wiki/N%C3%A1vrh_u%C5%BEivatelsk%C3%A9ho_rozhran%C3%AD

[5] Cognitive Walkthrough. [cit. 2017-12-12]. Dostupné z: http://www.usabilitybok.org/cognitive-walkthrough

[6] Krabač, B. T.: *Diplomová práce - Historie ČVUT (Android mobilní aplikace)*. České vysoké učení technické v Praze, 2016. Dostupné z: https://dspace.cvut.cz/handle/10467/63014

[7] Google: *Android Architecture Blueprints*. [cit. 2017-10-7]. Dostupné z: https://github.com/googlesamples/android-architecture

[8] The MVC, MVP, and MVVM Smackdown. [cit. 2017-11-27]. Dostupné z: https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/

[9] *Guide to App Architecture*. [cit. 2017-11-27]. Dostupné z: https://developer.android.com/topic/libraries/architecture/guide.html

[10] Android Architecture Components MVVM. [cit. 2017-11-27]. Dostupné z: https://proandroiddev.com/android-architecture-components-mvvm-part-1-1bd138959535

[11] *Data Binding Library*. [cit. 2017-11-28]. Dostupné z: https://developer.android.com/topic/libraries/data-binding/index.html

[12] Object-relational mapping. [cit. 2017-11-27]. Dostupné z: https://en.wikipedia.org/wiki/Object-relational_mapping

[13] *greenDAO: Android ORM for your SQLite database*. [cit. 2017-11-25]. Dostupné z: http://greenrobot.org/greendao/

[14] Overview of Google Play Services. [cit. 2017-11-27]. Dostupné z: https://developers.google.com/android/guides/overview

[15] *greenDAO Documentation*. [cit. 2017-11-25]. Dostupné z: http://greenrobot.org/greendao/documentation/

[16] About Glide. [cit. 2017-11-29]. Dostupné z: https://bumptech.github.io/glide/

[17] *Afollestad - Material Dialogs*. [cit. 2017-11-26]. Dostupné z: https://github.com/afollestad/material-dialogs

[18] *HelloCharts for Android*. [cit. 2017-11-26]. Dostupné z: https://github.com/lecho/hellocharts-android

[19] Build a Responsive UI with ConstraintLayout. [cit. 2017-11-27]. Dostupné z: https://developer.android.com/training/constraint-layout/index.html

[20] Mastering the Coordinator Layout. [cit. 2017-11-27]. Dostupné z: http://saulmm.github.io/mastering-coordinator

[21] Handling Lifecycles with Lifecycle-Aware Components. [cit. 2017-11-15]. Dostupné z: https://developer.android.com/topic/libraries/architecture/lifecycle.html

[22] Shvarts, J.: Android Architecture Components by Example. [cit. 2017-11-15]. Dostupné z: https://proandroiddev.com/architecture-components-modelview-livedata-33d20bdcc4e9

[23] ViewModel. [cit. 2017-11-15]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel.html

[24] Mount, G.: Android Data Binding: RecyclerView. [cit. 2017-11-17]. Dostupné z: https://medium.com/google-developers/android-data-binding-recyclerview-db7c40d9f0e4

[25] *OBD-II PIDs*. [cit. 2017-12-11]. Dostupné z: https://en.wikipedia.org/wiki/OBD-II_PIDs

[26] Chlup, J.: *Bakalárska práca - Asistent riadenia motorového vozidla*. Univerzita Komenského v Bratislave, 2015. Dostupné z: http://alis.uniba.sk/storage/dzb/dostupne/FM/2015/2015-FM-63733

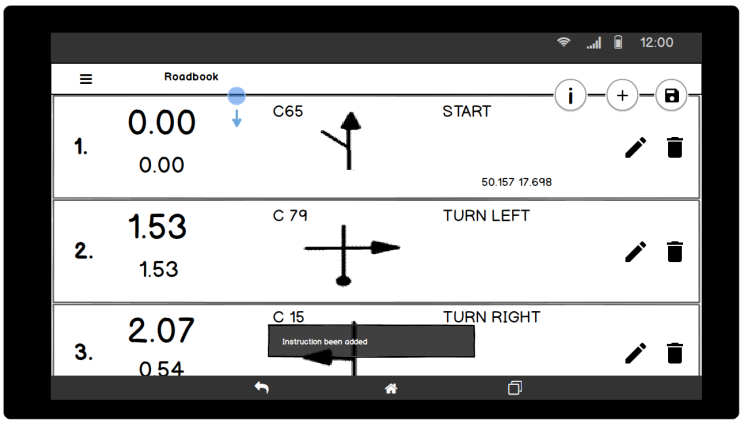
[27] *Location Strategies*. [cit. 2017-12-11]. Dostupné z: https://developer.android.com/guide/topics/location/strategies.html

[28] *Getting the Last Known Location*. [cit. 2017-12-11]. Dostupné z: https://developer.android.com/training/location/retrieve-current.html

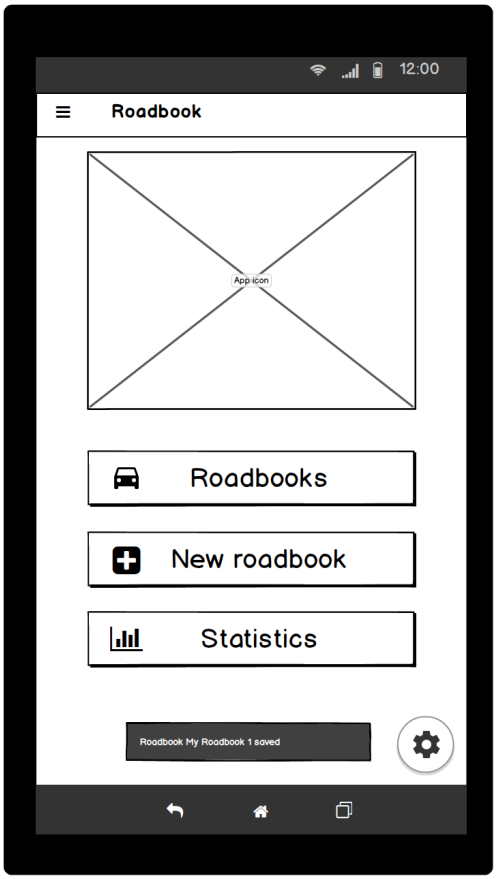
Dodatok A

Wireframy

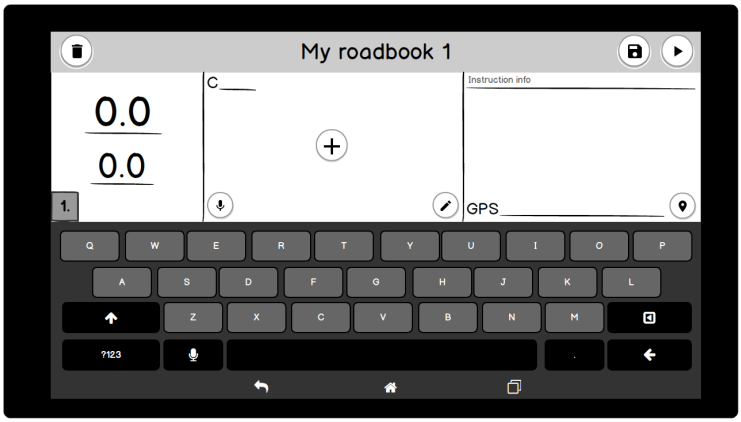
Všetky vytvorené wireframy (mock-up systému) vo formáte PDF sú na priloženom CD.



Obr. A.1: Wireframe správy o uložení inštrukcie.



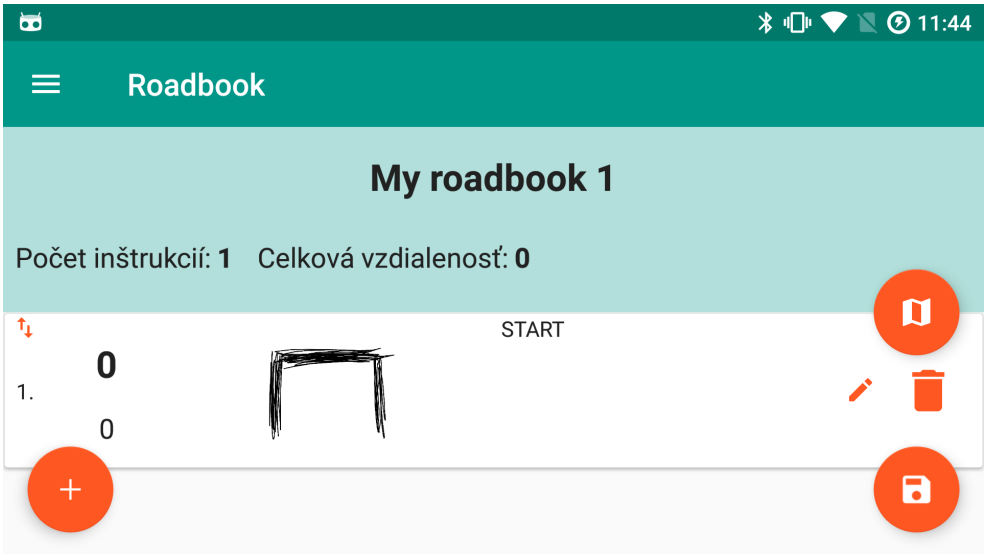
Obr. A.2: Wireframe správy o uložení roadbook.



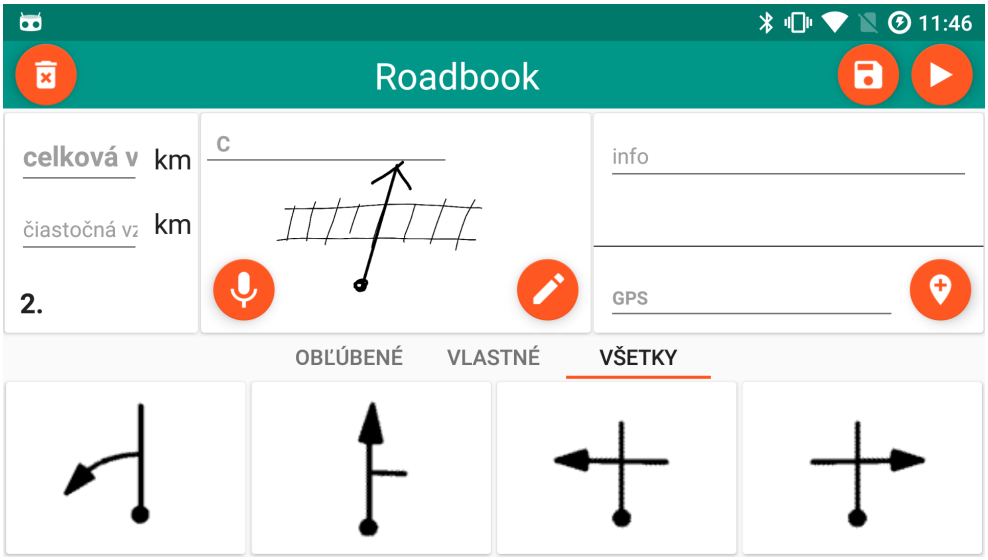
Obr. A.3: Vytváranie inštrukcie so zobrazenou klávesnicou.

Dodatok B

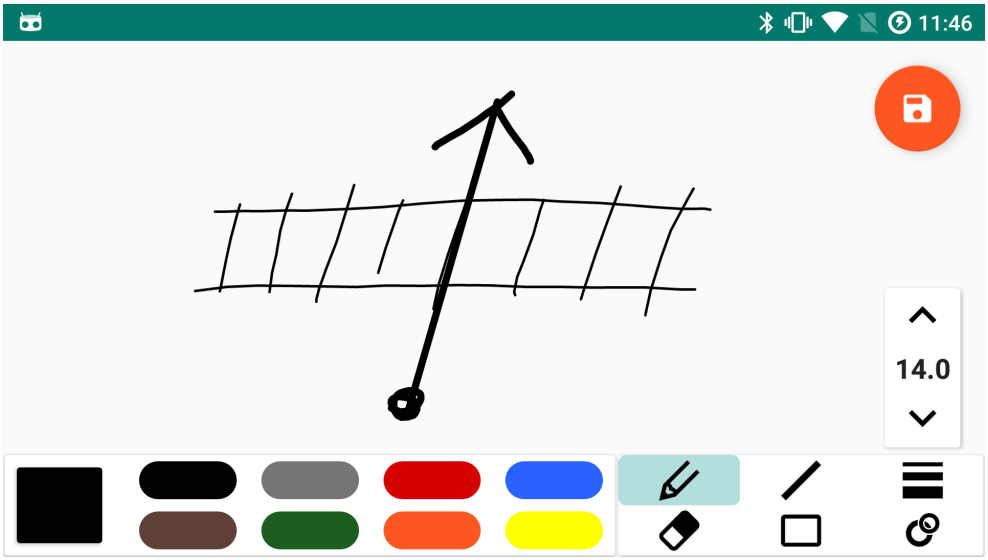
Ukážky výslednej aplikácie



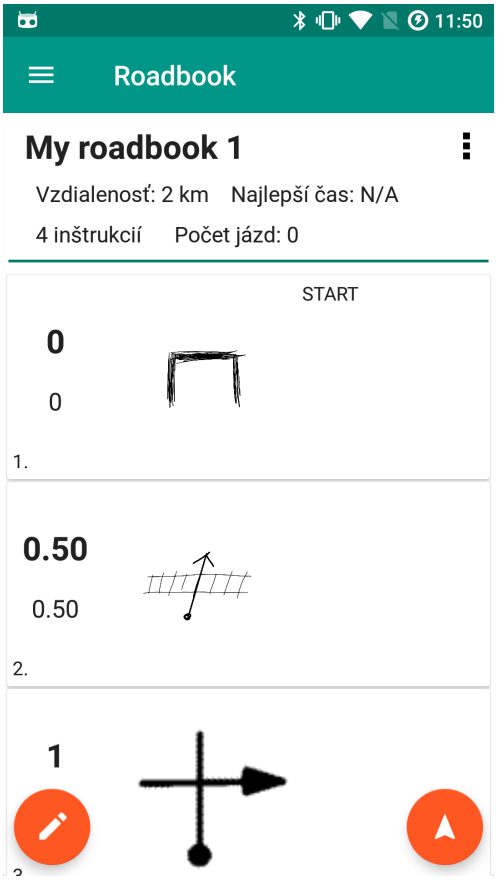
Obr. B.1: Ukážka aplikácie – vytváranie roadbook.



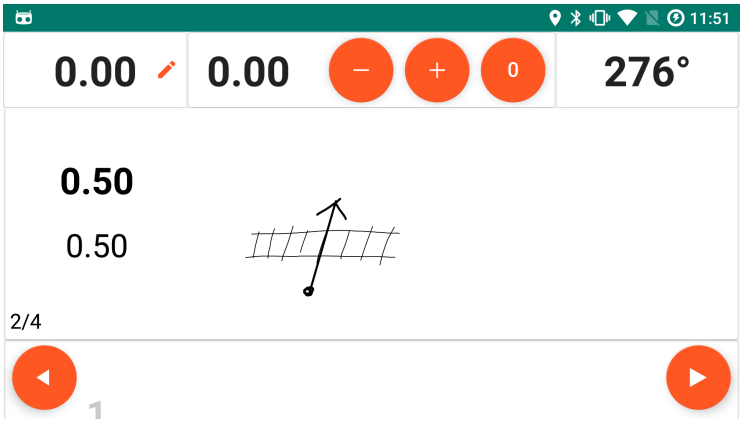
Obr. B.2: Ukážka aplikácie – vytváranie inštrukcie.



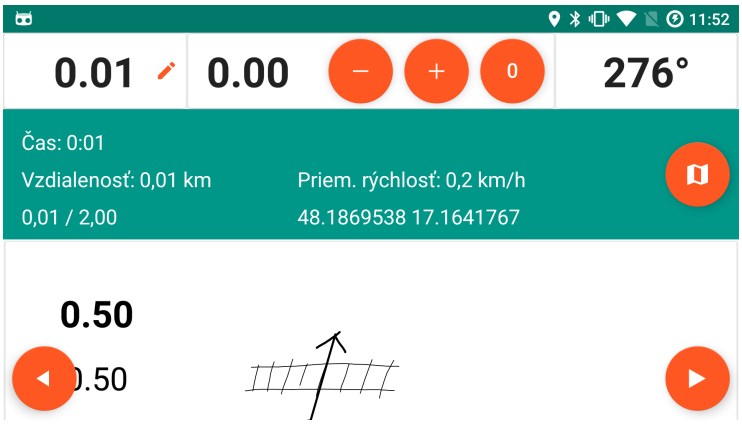
Obr. B.3: Ukážka aplikácie – kreslenie vlastného nákresu situácie.



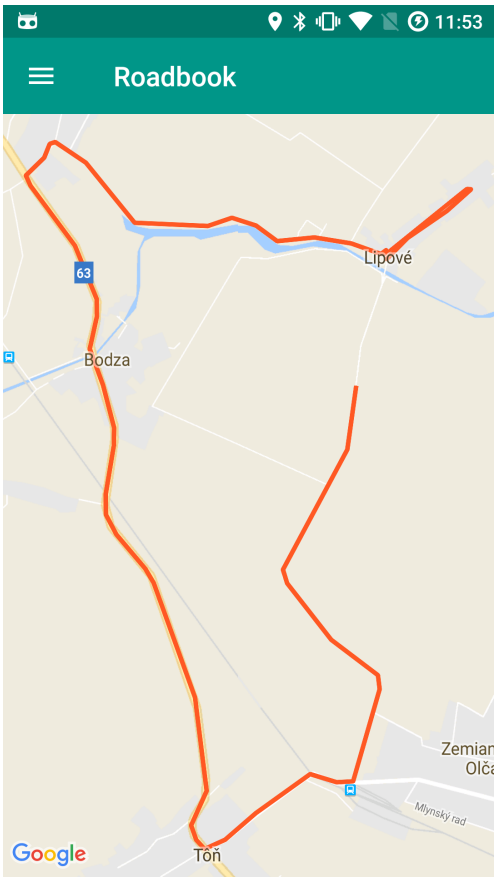
Obr. B.4: Ukážka aplikácie – detail roadbook.



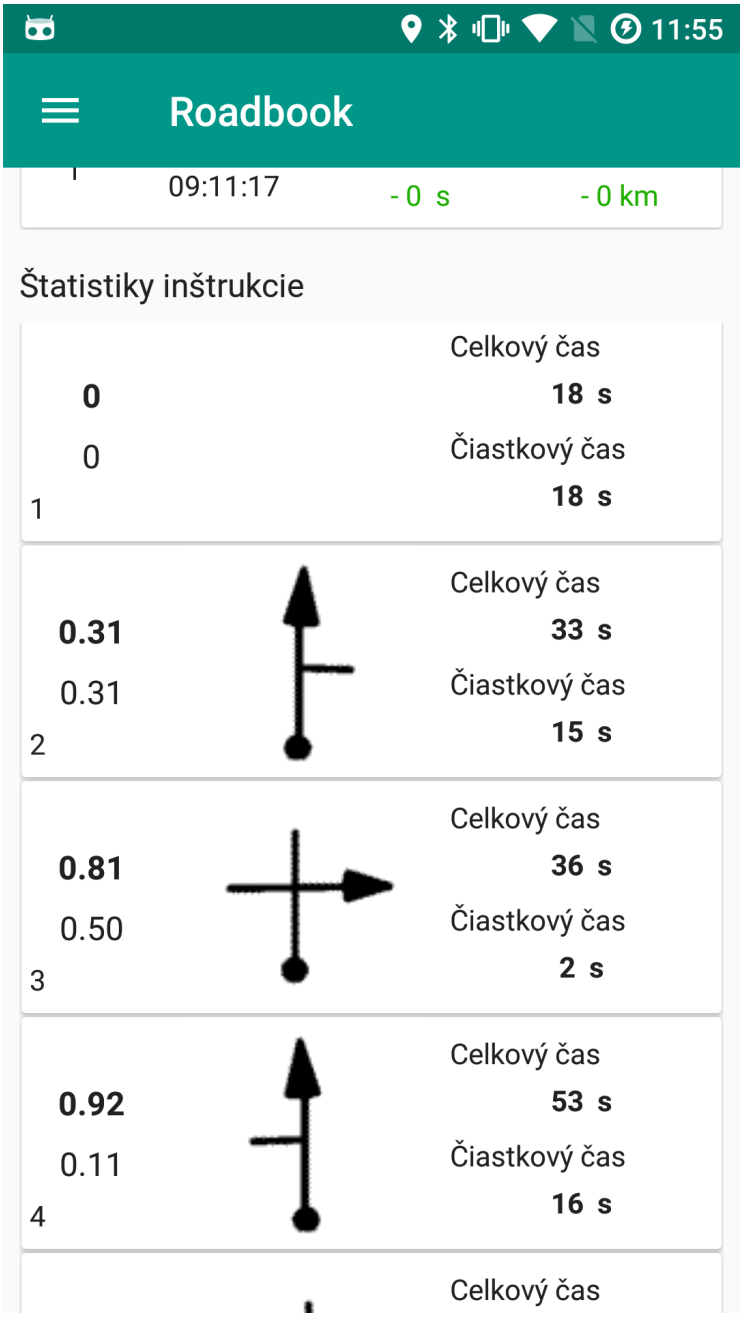
Obr. B.5: Ukážka aplikácie – navigácia.



Obr. B.6: Ukážka aplikácie – navigácia so zobrazenými dodatočnými informáciami.



Obr. B.7: Ukážka aplikácie – mapa absolvovanej jazdy.



Obr. B.8: Ukážka aplikácie – štatistiky absolvovaných inštrukcií.

Dodatok C

Zoznam použitých skratiek

GPS Global positioning system

UI User interface

XML Extensible markup language

GPX inak GPX Exchange Format, je XML schéma navrhnutá pre často používané dáta spojené s GPS. Používa sa napríklad na zapísanie trasy a jej jednotlivých bodov.

Dodatok D

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... súbor obsahujúci aplikáciu pre OS Android
  - `app-release.apk` ..... spustiteľný súbor na OS Android
- `roadbook.pdf` ..... mock-up aplikácie
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF