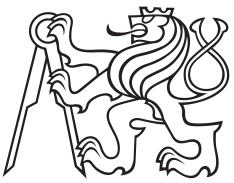
Bakalárska práca



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačov

Mobilná aplikácia pre užívateľov systému zdieľania automobilov viac užívateľmi

Filip Ravas
Študijný program: Otvorená informatika, Odbor: Softwarové systémy

Máj 2017
Vedúci práce: doc. Ing. David Šišlák Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počíta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Ravas Filip

Studijní program: Otevřená informatika
Obor: Softwarové systémy

Název tématu: Mobilní aplikace pro uživatele systému sdíleni automobilu vice uživateli

Pokyny pro vypracování:

Navrhnete a implementujte mobilní aplikaci pro uživatele systému sdíleni automobilu vice uživateli.
1) Dohodnete s ostatními studenty pracujícími na tomto systému funkcionalitu systému a zdokumentujte všechny způsoby použiti funkci pomoci UML diagramu.
2) Navrhnete intuitivní vzhled mobilní aplikace, jez bude využívaná uživateli systému pro správu rezervaci.
3) Implementujte mobilní aplikaci pro systém Android.
4) Vyzkoušejte intuitivnost aplikace na uživatelích. Tyto testy zdokumentujte a vyhodnoťte.
5) Navrhnete budoucí vylepšení mobilní aplikace.

Seznam odborné literatury:

[1] Fowler, M.: UML Distelled - Third Edition. Addison-Wesley, 2003.
[2] Meike, G. B., Mednieks, Z., Nakamura, M., Dornin, L.: Programmin Android. O?Reilly, 2011.
[3] Novotny, T.: Technicka zprava: Technicka vize CarSharingu. CVUT FS, 2015.

Vedoucí: doc. Ing. David Šišlák, Ph.D.

Platnost zadání do konce letního semestru 2017/2018

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25.1.2017

Poďakovanie / Prehlásenie

Chcel by som poďakovať doc. Ing. Davidovi Šišlákovi, Ph.D. za odbornú pomoc, cenné vedomosti a poznatky, ktoré som získal počas konzultácií.

Prehlasujem, že som predloženú prácu vypracoval samostatne. Uviedol som všetky použité informačné zdroje v súlade Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Prahe dňa 13. 6. 2017

.....

Abstrakt / Abstract

Táto bakalárska práca sa zaoberá návrhom a implementáciou mobilnej aplikácie pre operačný systém Android. Výstupom je aplikácia spĺňajúca funkcionalitu, ktorá je definovaná požiadavkami na klientskú aplikáciu študentského projektu pre zdieľanie vozidiel medzi univerzitami. Práca obsahuje popis požiadaviek, wireframes navrhnutého užívateľského rozhrania, popis technológií použitých v systéme. V niektorých prípadoch sú uvedené dôvody, ktoré vysvetľujú použitie konkrétnych technológií. Práca sa zaoberá implementáciou aplikácie, obsahuje vyhodnotenie testu užívateľského rozhrania, popisuje implementáciu častí systému nevyhnutných pre funkcionalitu a testovanie aplikácie.

Kľúčové slová: mobilná aplikácia; Android; zdieľanie automobilov;

This bachelor thesis's subject is design and implementation mobile application for operating system Android. A main goal of thesis is an application that meets functionality that is defined by requirements for client application of student's project for carsharing between universities. The thesis contains a description of requirements, wireframes designed user interface, description of technologies used in system. In some cases there is described a reason why we decided to use that particular technologies. Next there there is described whole implementation of application, evaluation of user interface tests, implementation of parts of system that was needed for functionality and testing of application.

Keywords: mobile application; Android; carsharing;

Title translation: Android application for car sharing system

Obsah /

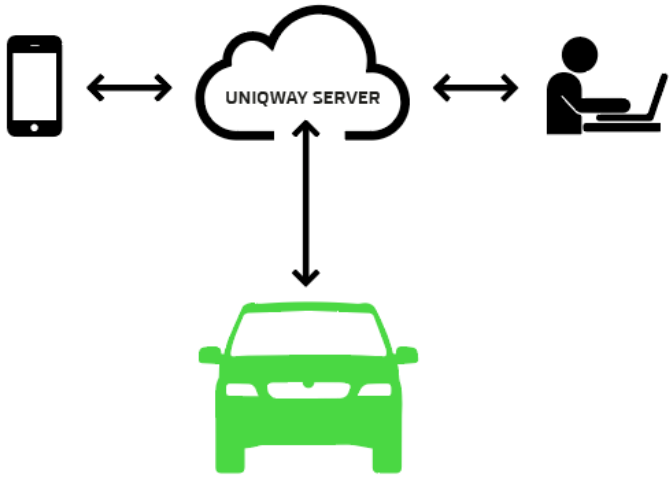
Tabuľky /

Kapitola 1
Úvod

Slovo carsharing je v súčasnosti skloňované čoraz častejšie, čo svädčí o progresivite myšlienky zdieľania prostriedkov v spoločnosti. V niektorých krajinách (napr. v Nemecku), sa už zdieľané automobily stali bežnou súčasťou každodenného života ľudí. Rastúci záujem o podobné služby sa prejavil aj v Prahe, kde už jedna spoločnosť poskytujúca carsharing vznikla. Existuje však veľká a v mnohých smeroch špecifická skupina ľudí, ktorým princípy jej fungovana nevyhovujú - študenti. Táto situácia pripravila pôdu pre vznik ich vlastného študenského projektu, na ktorom spolupracujú tri pražské univerzity: České vysoké učení technické v Praze, Česká zemědělská univerzita v Praze a Vysoká škola ekonomická v Praze. Jeho cieľom je vytvoriť službu zdieľania vozidiel, na mieru ušitú pre študentov a pracovníkov vysokých škôl. Táto cieľová skupina je veľmi špecifická a má v mnohých smeroch odlišné požiadavky ako bežný zákazník. Tím, ktorého som členom, robí všetko preto, aby im v čo najväčšej miere vyhovel. Z tohoto dôvodu prebehol výskum pomocou dotazníkov, ktorého sa zúčastnilo cez 6000 študentov.

Smartfón je dnes neoddeliteľne spätý s mladými ľudmi, ktorí sa snažia čo najviac využiť jeho potenciál. Denne ho používajú na komunikáciu, na navigáciu pri pohybe v meste, vyhľadávanie spojov verejnej dopravy, platenie účtov a veľa ďalších činností. Preto nebolo prekvapením, že viac ako 60 percent študentov v dotazníku na otázku aký spôsob by preferovali v prístupe k službe odpovedalo práve aplikáciou v smartfóne.

Za kompletné technické riešenie projektu sú zodpovední študenti ČVUT. Celý systém je logicky rozdelený na štyri časti. Server, klientskú mobilnú aplikáciu, hardverovú jednotku v aute a webovú aplikáciu pre správu vozového parku.



Obrázok 1.1. Znázornenie schémy celého systému.

Mojou úlohou v tíme a zároveň predmetom tejto bakalárskej práce je implementácia klientskej mobilnej aplikácie. Jej neodeliteľnou súčasťou je ale aj server, s ktorým komunikuje, preto sme sa aktívne podieľali aj na návrhu a implementácií jeho API pre mobilnú aplikáciu. Pomocou aplikácie je realizovaná aj interakcia s autom (odomknutie/zamknutie) a pre jej testovanie by preto bolo potrebné vozidlo s funkčným modulom pripojeným k internetu. Preto sme vytvorili script, ktorý správanie modulu simuluje. Hardware, určený na server pre projekt, nám poskytnutla Fakulta strojní. Na starosti sme mali jeho nastavenie a sprevádzkovanie nutných služieb.

Práca sa skladá z nasledujúcich hlavných častí:

- **Mobilné aplikácie:** V tejto časti sú rozobraté rôzne prístupy vývoja mobilných aplikácií s oddôvodnením výberu pre bakalársku prácu. Ďalej sa kapitola venuje popisu vybraného prístupu (napríklad štruktúri projektu, základným komponentám aplikácie)
- **Backend:** Popisuje výber technológie pre backend systému a API pre komunikáciu s mobilnou aplikáciou.
- **Funkcionalita systému:** Obsahuje požiadavky na mobilnú aplikáciu.
- **Návrh vzhľadu:** Táto kapitola obsahuje wireframes grafického návrhu aplikácie.
- **Implementácia:** Kapitola, ktorá sa venuje popisu implementácie aplikácie - zahŕňa hlavné aktivity, súbory pre automatizáciu zostavovania programu a Android manifest
- **Záver:** Zhrnutie práce. Sú v ňom zahrnuté plány do budúcnosti projektu.
- **Prílohy:** V prílohách sa nachádzajú požiadavky pre mobilnú aplikáciu ktoré je nutné implmentovať do fáze minimálneho cenného produktu.

Kapitola 2
Mobilné aplikácie

Pri rozhodovaní medzi vývojom mobilnej a webovej aplikácie, ktorá by bola optimalizovaná aj pre zariadenia s malým displejom, sme sa priklonili k mobilnej aplikácii a to hneď z niekoľkých dôvodov. Prvým z nich je potreba prístupu k perifériám, (GPS, Bluetooth), na ktorých funkcionalite je systém závislý. Ďalší dôvod vychádza z charakteristiky cieľovej skupiny: študenti často nemajú finančné prostriedky na to, aby si u mobilného operátora zaplatili veľký objem dát. Ak by klientská aplikácia bola realizovaná ako webová, bolo by nevyhnutné sťahovať pomerne veľké objemy dát (napríklad obsah nutný pre grafickú časť). Pomocou mobilnej aplikácie sme schopní prenosy obmedziť na requesty, v ktorých sa prenášajú minimalizované správy JSON. Majoritný podiel na trhu so smartfónmi (asi 70 percent) má operačný systém Android, čo dokazujú aj odpovede študentov v dotazníkovom výskume. Preto sme sa rozhodli klientskú aplikáciu vytvoriť práve pre tento systém. V budúcnosti plánujeme pokryť aj operačný systém iOS.

Existuje veľa rôznych spôsobov vývoja aplikácií pre Android, napr. použitím knižníc a frameworkov umožňujúcich vývoj v rôznych programovacích jazykoch (Python, Javascript, kombinácia HTML5, CSS a Javasriptu). Niektoré z nich majú výhodu v tom, že vytvorený kód sa dá skopmilovať aj do IOs aplikácie, v porovnaní sa natívnou však takáto aplikácia neposkytuje dobrý výkon. Ďalšou obrovskou výhodou natívneho prístupu je podpora a komunita. Vývoj Androidu zastrešuje firma Google, ktorá poskytuje veľmi kvalitnú a detailnú dokumentáciu, spolu s množstvom príkladov, tutoriálov a dokonca aj online kurzov. Dodáva tiež nástroje pre vývojárov, ako napríklad Android Studio (IDE pre Android vývoj) alebo Virtual device manger (nástroj pre emulovanie zariadení android). Vďaka spomenutým výhodám je komunita naozaj početná a iné prístupy sa s ňou môžu len ťažko porovnávať.

Pri použití natívneho prístupu sa predpokladá, že aplikácia bude mať určitú štandardnú štruktúru. Už spomínané IDE Android Studio ponúka možnosť vytvorenia základnej spustiteľnej aplikácie s touto štruktúrou.

Následujúci text až na výnimky prevzatý z [1].

2.1 Štruktúra zdrojového kódu aplikácie

Zdrojový kód Android aplikácie pozostáva z viacero logicky oddelených častí:

- **automatizácia zostavovania programu:** Automatizácia zostavovania je proces, pri ktorom sa zostaví spustiteľná aplikácia zo zdrojových kódov a resource súborov. Tento proces typicky zahŕňa napríklad: kompiláciu zdrojových súborov, balíčkovanie vzniknutých súborov, spúšťanie automatizovaných testov. Pri programovaní netriviálnej aplikácie je využívanie takéhoto nástroja takmer nevyhnutné. V prípade Android aplikácie je štandardom určeným na tento účel open source systém Gradle. Jeho veľkou výhodou pre vývojára je i pokročilý manažment závislostí projektu.
- **manifest:** Súbor AndroidManifest.xml musí mať každá aplikácia. Sú v ňom definované základné informácie o aplikácií, nevyhnutné pre operačný systém ešte pred

spustením samotného aplikačného kódu. Patria medzi ne komponenty, z ktorých aplikácie pozostáva. Sú to napríklad Service, Activity, Broadcast Receivery a Content Providery. Ďalšou veľmi dôležitou funkciou je deklarácia oprávnení aplikácie - permissions, určujúca oprávnenia, ktoré musí aplikácia mať, aby mohla pristupovať k chráneným častiam API a komunikovať s ostatnými aplikáciami. Stanovuje aj oprávnenia nutné pre iné aplikácie, aby mohli interagovať s komponentami aplikácie.

- **java:** Zložka java obsahuje zdrojové kódy, oddelené od seba Javovskými balíčkami a nachádzajú sa v nej aj JUnit testy.
- **res:** V zložke res sú súbory, ktoré nepatria medzi zdrojové, ale sú potrebné pre užívateľské rozhranie, napríklad obrázky, XML layouty, zobrazované texty atď.

2.2 Activity

Aktivita reprezentuje jednu obrazovku aplikácie, s ktorou užívateľ takmer vždy komunikuje. Je zodpovedná za vytvorenie obrazovky, do ktorej sa umiestní UI definované v resourcoch.

2.2.1 Životný cyklus Activity

Aktivity sú v systéme manažované ako zásobník. Keď je spustená nová aktivita, umiestni sa na vrch zásobníka a stane sa z nej bežiaca aktivita - predchádzajúca aktivita zostane v zásobníku pod ňou a neprejde do popredia, kým beží nová aktivita.

Aktivita má 4 základné stavy:

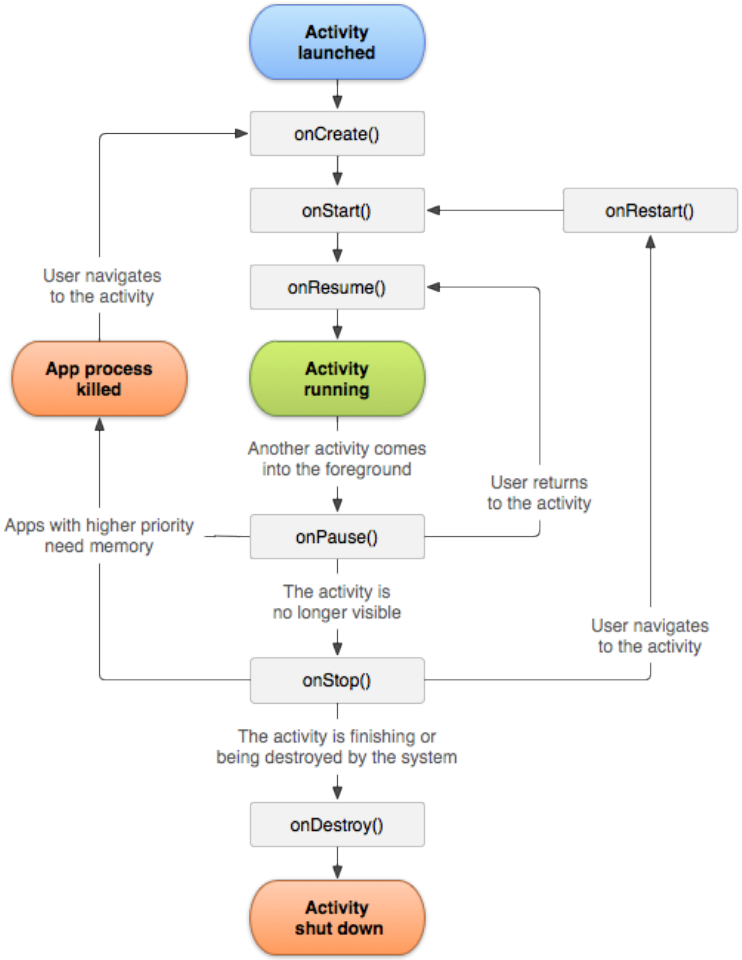
- Ak je aktivita v popredí obrazovky (na vrchu zásobníku), je aktívna alebo bežiaca.
- Ak aktivita stratila zameranie ale je stále viditeľná (je nad ňou iná aktivita, ktorá nemá plnú veľkosť, alebo je priehľadná), aktivita je pozastavená. Je stále kompletne `živá`, ale môže byť stopnutá systémom.
- Ak je aktivita kompletne prekrytá inou aktivitou, je stopnutá. Stále uchováva všetky informácie o stave a členoch, ale už nie je viditeľná pre užívateľa, jej okno je skryté a často stopnuté systémom v prípade, keď je pamäť potrebná inde.
- Ak je aktivita pozastavená alebo stopnutá, systém ju môže `pustiť` z pamäte tým, že ju požiada o ukončenie alebo `zabije` jej proces. Keď je opäť zobrazená užívateľovi, musí byť kompletne reštartovaná a obnovená do predchádzajúceho stavu.

2.2.2 Fragmenty

S verziou Androidu 3.0 (Honeycomb) môžu aplikácie využívať triedu Fragment, umožňujúcu lepšiu modularizáciu kódu a budovanie sofistikovanejších užívateľských rozhraní. Na fragment sa dá pozerať ako modulárnu časť aktivity s vlastným životným cyklom, ktorá prijíma vstupné udalosti, môže byť pridaná alebo odobratá počas behu aktivity alebo použitá vo viacerých aktivitách.

2.3 Services

Service je komponenta bez užívateľského rozhrania bežiaca na pozadí aplikácie a vykonávajúca časovo náročné operácie (napríklad http požiadavku pre získanie dát). Iná komponenta sama môže Service spustiť, aby s ňou ďalej mohla komunikovať.



Obrázok 2.1. Znázornenie životného cyklusu Activity

Service má dve hlavné funkcionality:

- Informovanie systému o niečom, čo chce vykonávať v pozadí (aj keď užívateľ priamo neinteraguje s aplikáciou).
- Vystavenie nejakej funkcionality iným aplikáciám - dovoľuje dlhodobé pripojenie k service za účelom interakcie s ňou.

2.4 Broadcast Receivery

Broadcast Receiver je komponenta, ktorá odpovedá na broadcast správy iným aplikáciam alebo systému. Je to dobre definovaný vstup do aplikácie a systém dokáže doručiť broadcasty aj do takých aplikácií, ktoré práve nebežia.

Ako príklad poslúži budík, ktorý môže byť naplánovaný, aby informoval užívateľa o nadchádzajúcej udalosti. Pretože je na notifikáciu budíkom použitý Broadcast Reveiver, nie je nutné aby aplikácia budíka bola otvorená celý čas, kým budík nezazvoní.

Veľa broadcastov je iniciovaných systémom, napríklad broadcast oznamujúci, že obrazovka bola vypnutá, batéria je takmer vybitá alebo bola spravená fotografia.

2.5 Content Providery

Content provider manažuje zdieľané sety dát aplikácie, ktoré môžu byť uložené v súborovom systéme, SQLite databáze, na webe, alebo na akomkoľvek inom perzistentnom úložisku, do ktorého má aplikácia prístup. Pomocou contet providera môžu iné aplikácie pristupovať k dátam, alebo modifikovať dáta, ku ktorým content provider dovoľuje prístup.

Napríklad systém poskytuje content provider manažujúci kontaktné informácie užívateľa. Každá aplikácia s príslušnými právami môže k pristúpiť k content providerovi ako je `ContactsContract.Data` a čítať a zapisovať informácie o konkrétnom užívateľovi.

2.6 Použité prvky

V tejto sekcii sú popísané niektoré bežne používané prvky určené pre vývoj Android aplikácií.

2.6.1 RecyclerView

Tento widget je kontajner pre zobrazovanie veľkých data setov, cez ktoré sa dá veľmi efektívne skrolovať. To je umožnené udržovaním obmedzeného počtu častí užívateľského rozhrania reprezentujúceho jednotlivé položky data setu. Trieda RecyclerView zjednodušuje zobrazovanie a prácu s veľkými datasetmi tým, že poskytuje napríklad:

- Funkcionalitu pre umiestňovanie jednotlivých prvkov.
- Štandardné animácie pre bežné operácie s prvkami ako napríklad odstraňovanie a pridávanie prvkov.

2.6.2 Adapter

Objekt Adaptér sa správa ako most medzi medzi AdapterView (v našom prípade RecyclerView) a date setom, ktoré pre View. Adapter poskytuje prístup k jednotlivým prvkom v data sete a je zodpovedný aj za vytváranie užívateľského rozhrania pre každý prvok z data setu.

2.6.3 SharedPreferences

Použitie SharedPreferences je jednoduchý spôsob uloženia základných stavových informácií a užívateľských dát, tak ako spolu so spôsobom použitia podrobnejšie popisuje [2].

2.6.4 Navigation Drawer

Podľa [3] je Navigation Drawer skrytý panel, ktorý zobrazí hlavné navigačné menu aplikácie. To pomáha užívateľovi rýchlo sa zorientovať v štruktúre aplikácie. Keď klikne na ikonu umiestnenú vpravo hore, alebo potiahne zľava doprava Navigation Drawer sa roztiahne na časť obrazovky a zobrazí zoznam položiek v menu.

Kapitola 3
Backend

Navrhnutá architektúra nášho systému je postavená na backende, ktorý komunikuje s ostatnými časťami (správcovskou aplikáciou, klientskou aplikáciou hardvérovej jednotky vo vozidle) pomocou vystaveného API. Aj keď implementácia backedu nie je predmetom našej práce, z časti sme sa vo forme implementácie API potrebného pre komunikáciu s mobilnou aplikáciou podieľali aj na ňom. Preto sa táto kapitola venuje aj odôvodneniu výberu technológie a popísaniu prístupových bodov aplikačného rozhrania.

3.1 Výber technológie

Pretože sa na vývoji backendu podieľalo niekoľko ľudí a v budúcnosti je možné, že sa ich počet ešte zväčší, museli sme sa rozhodnúť, aký prístup na jeho implementáciu použijeme. Technológií existuje veľa, počas štúdia sme pracovali napríklad s platformou Java Enterprise Edition. Kvôli zbytočnej komplexnosti celej technológie sme ju ale zamietli.

Rozhodli sme sa pre open-source framework s architektúrou MVC (tomuto návrhovému vzoru sa venuje [4]) Play, vyvíjaný firmou Lightbend. Dôvody našej voľby sú nasledovné:

- **Možnosť vývoja v jazyku Java:** Aj keď je framework napísaný v programovacom jazyku Scala, použitie Javy je plne podporované. To bolo pre nás veľmi dôležité, lebo tento programovací jazyk sme používali od začiatku štúdia.
- **Kvalitná dokumentácia:** Všetka dokumentácia je prístupná pre jazyk Scala aj Java na stránke [1]).
- **Početná komunita:** Framework má (okrem vývojárov z Lightbend) početnú komunitu developerov, ktorí sa podieľajú na jeho vývoji na Githube a existuje aj aktívna komunita ľudí ochotných odpovedať na otázky na portáli [2]).
- **Moderný stack trace a error handling:** pri chybe pri kompilácií ako aj za behu aplikácie framework v prehliadači zobrazí číslo riadku, správu chyby, cestu k súboru a dokonca aj útržok kódu. Oproti obrovským stack tracom Javy EE, alebo frameworku Spring je to výhoda, ktorá výrazne urýchli vývoj systému.
- **Predchádzajúca skúsenosť s frameworkom**

Aplikačné rozhranie definuje komunikáciu medzi backendom a ostatnými časťami systému, s ktorými server komunikuje. Popísané bude rozhranie pre komunikáciu s mobilnou aplikáciou.

Autentifikácia funguje na princípe JWT authentification. Podľa [5] je JWT kompaktný formát reprezentácie nárokov určený pre prostredie s obmedzeným priestorom ako sú HTTP hlavičky určené na autorizáciu. JWT zakóduje nároky určené na prenos vo formáte JSON. Pri použití JWT je možné, že príde k zneplatneniu užívateľovho JWT tokenu. V tomto prípade aplikácia pri vykonaní requestu vráti odpoveď return kódom 401 - neautorizovaný a pomocou intentu sa vráti na prvú obrazovku, kde bude chcieť od užívateľa prihlásenie.

[1]) `https://www.playframework.com/documentation`
[2]) `https://stackoverflow.com`

3.1.1 Prihlásenie užívateľa

V odpovedi na požiadavku sa vráti JWT token pre autentifikáciu.

- **Metóda:** POST
- **URL:** /login
- **Parametre:** uvedené v tabuľke 3.1

Názov parametra	Povinný	Dátový typ	Popis
email	áno	string	prihlasovací email
password	áno	string	prihlasovacie heslo

Tabuľka 3.1. API - Prihlásenie užívateľa (parametre)

3.1.2 Predregistrácia užívateľa

Požiadavka, ktorá slúži na predregistráciu užívateľa. Ten vyplní iba niekoľko povinných hodnôt a ľubovoľný počet nepovinných. Tieto údaje bude musieť potom pri osobnej registrácii doplniť.

- **Metóda:** POST
- **URL:** /signup
- **Parametre:** uvedené v tabuľke 3.2

Názov parametra	Povinný	Dátový typ	Popis
email	áno	reťazec	prihlasovací email
password	áno	reťazec	prihlasovacie heslo
firstName	áno	reťazec	krstné meno
lastName	áno	reťazec	priezvisko
gender	nie	celé číslo	pohlavie
idenfiticationNumber	nie	reťazec	číslo občianskeho preukazu
university	nie	celé číslo	univerzita
nationality	nie	celé číslo	národnosť
dateOfBirth	nie	reťazec	dátum narodenia DD-MM-YYYY
stateOfBirth	nie	reťazec	krajina narodenia
cityOfBirth	nie	reťazec	mesto narodenia
permanentResidence	nie	zoznam parametrov	trvalé bydlisko
permanentResidence.street	nie	reťazec	ulica
permanentResidence.doorNumber	nie	celé číslo	číslo dverí
permanentResidence.city	nie	reťazec	mesto
permanentResidence.country	nie	celé číslo	štát
contactAddress.street	nie	reťazec	ulica
contactAddress.doorNumber	nie	celé číslo	číslo dverí
contactAddress.city	nie	reťazec	mesto
contactAddress.country	nie	celé číslo	štát

Tabuľka 3.2. API - Predregistrácia užívateľa (parametre)

3.1.3 Zoznam voľných automobilov

V odpovedi na požiadavku sa vráti JSON obsahujúci zoznam voľných automobilov.

- **Metóda:** GET

- **URL:** /car
- **Očakávaná odpoveď servera:**


```
[{
  "id": 1,
  "licenceNumber": "3SY 5562",
  "carModel": {
    "name":"Škoda Fabia",
    "seats": 5,
    "name": "JmenoFabie",
    "color": "#00000",
    "engine": {
      "type":"1.2 TSI",
      "power": "81kW",
      "transmission": 6,
      "fuelType": "gasoline"},
  "tankPercentage": 50,
  "tankRangeKm": 400
},
{
  "id": 2,
  "licenceNumber": "3SY 5562",
  "carModel": {
    "name":"Škoda Fabia",
    "seats": 5,
    "name": "JmenoDruheFabie",
    "color": "#FFFFF",
    "engine": {
      "type":"1.2 TSI",
      "power": "81kW",
      "transmission": 6,
      "fuelType": "gasoline"},
  "tankPercentage": 55,
  "tankRangeKm": 400
}]
```


3.1.4 Zoznam GPS súradníc voľných automobilov s ich menami

V odpovedi na požiadavku sa vráti JSON obsahujúci zoznam GPS súradníc voľných automobilov s ich menami.

- **Metóda:** GET
- **URL:** carposition
- **Očakávaná odpoveď servera:**


```
{
  [{
    "car": {
      "id": 1,
      "name": "JmenoFabie"
      },
    "position": {
      "latitude": 21.111023
      "longitude": 53.112432
```

```
      }}
  ],[{
    "car": {
      "id": 2,
      "name": "JmenoDruheFabie"
      },
    "position": {
      "latitude": 23.871232
      "longitude": 51.53211
      }
    }]
}
```

3.1.5 Detailné informácie o vozidle

V odpovedi sa vráti JSON obsahujúci detailné informácie o vozidle. Identifikátor vozidla sa posiela v požiadavke ako URL parameter {id vozidla}.

- **Metóda:** GET
- **URL:** cardetail/{id vozidla}
- **Očakávaná odpoveď servera:**

```
{
  "id": 1,
  "licenceNumber": "3SY 5562",
  "carModel": {
    "name":"Škoda Fabia",
    "seats": 5,
    "trunk": {
      "seatsUp": 200,
      "seatsDown": 400
    }},
  "name": "JmenoFabie",
  "color": "#00000",
  "engine": {
    "type":"1.2 TSI",
    "power": "81kW",
    "transmission": 6,
    "fuelType": "gasoline",
    "consumption": {
      "urban": 7.6,
      "extraUrban": 5.6,
      "combined": 6
      }},
 "features": [{
      "navigation": true
   }],
  "tankPercentage": 50,
  "tankRangeKm": 400,
  "position": {
  "latitude": 50.102594,
  "longitude": 14.393586
  }
}
```

3.1.6 GPS súradnica konkrétneho vozidla

V odpovedi server vráti GPS súradnicu konkrétneho vozidla. Identifikátor vozidla sa posiela v požiadavke ako URL parameter {id vozidla}. Tento prístupový bod API sa používa v prípade keď aplikácia chce získať polohu voľného vozidla, alebo pri vykresľovaní trasy na mape počas jazdy.

- **Metóda:** GET
- **URL:** carposition/{id vozidla}
- **Očakávaná odpoveď servera:**

```
{
  "latitude": 23.871232
  "longitude": 51.53211
}
```

3.1.7 Vytvorenie rezervácie

V odpovedi server vráti identifikátor novo vytvorenej rezervácie. Identifikátor vozidla ktoré má byť zarezervované sa posiela v požiadavke ako URL parameter {id vozidla}.

- **Metóda:** POST
- **URL:** reservations/{id vozidla}
- **Očakávaná odpoveď servera:**

```
{
  "id": 1
}
```

3.1.8 Zrušenie rezervácie

Požiadavok slúži na zrušenie rezervácie, telo odpovedi sa očakáva prázdne. Identifikátor rezervácie ktorá má byť zrušená sa posiela v požiadavke ako URL parameter {id rezervácie}.

- **Metóda:** DELETE
- **URL:** reservations/{id rezervácie}

3.1.9 Vytvorenie inštrukcie pre auto

V odpovedi sa očakáva identifikátor vytvorenej inštrukcie pre auto.

- **Metóda:** POST
- **URL:** ./instruction/
- **Očakávaná odpoveď servera:**

```
{
  "id": 1
}
```

3.1.10 Získanie stavu inštrukcie pre auto

Požiadavok slúžiaci pre získanie stavu inštrukcie pre auto. Identifikátor inštrukcie ktorej stav sa má vrátiť v odpovedi sa posila ako URL parameter {id inštrukcie}.

- **Metóda:** GET
- **URL:** /instruction/{ id inštrukcie }
- **Očakávaná odpoveď servera:**

```
{
  "status": 1
}
```

3.1.11 Získanie aktuálnej pozície vozidla

V odpovedi sa očakáva JSON s GPS súradnicami aktuálnej polohy vozidla.

- **Metóda:** GET
- **URL:** /carinfo/{id vozidla}
- **Očakávaná odpoveď servera:**

```
{
  "latitude": 23.871232
  "longitude": 51.53211
}
```

3.1.12 Získanie aktuálnych jazdných údajov

V odpovedi sa očakávava JSON s aktuálnymi jazdnými údajmi.

- **Metóda:** GET
- **URL:** /rideinfo/
- **Očakávaná odpoveď servera:**

```
{
  "gpsSpeed": 100,
  "speed": 101,
  "rpm": 4000,
  "fuelLevel": 42,
  "odometer": 14000,
  "consumption": 7.4
}
```

Kapitola 4
Funkcionalita systému (požiadavky)

Pre každú časť systému sme v rámci projektu vytvorili samostatné požiadavky. Vypracovali sme aj požiadavky na klientskú časť aplikácie, ktoré budú implementované v našej bakalárskej práci. Požiadaviek na celý systém bude samozrejme oveľa viac a sú uvedené v prílohách práce.

4.1 Autá

- **Zobraziť autá k dispozícii** Systém užívateľovi umožní zobraziť zoznam áut, ktoré sú k dispozícii na rezerváciu. Pri jednotlivých vozidlách budú uvedené nasledujúce informácie:
  - Typ auta
  - Poznávacia značka (ŠPZ)
  - Farba karosérie
  - Motor (typ, výkon)
  - Prítomnosť navigácie v aute
  - Počet miest na sedenie vrátane vodiča
  - Informácie o prevodovke (manuálna/automatická)
  - Vzdialenosť od miesta, kde je auto zaparkované (predpoklad zapnutej GPS v smartphone)
  - Stav palivovej nádrže v percentách a v litroch
  - Predpokladaný dojazd v km
  - Typ paliva (benzín, diesel)
- **Interakcia s vybraným autom:** Systém zobrazí užívateľovi informácie o aute spolu s možnosťou navigácie k nemu pomocou externej aplikácie Google Maps. Poskytne samostatné zobrazenie na mape a rezervovanie vozidla.
- **Zobraziť detail auta:**
- **Odomknutie auta:** Systém umožní užívateľovi bez potreby použitia kľúča auto odomknúť, pričom musí byť fyzicky v jeho bezprostrednej blízkosti. Prvým odomknutím užívateľ auto prevezme a stáva sa zaň automaticky zodpovedným v zmysle všeobecných obchodných podmienok prevádzkovateľa služby. Začína tiež plynúť čas, ktorý sa používa na výpočet ceny za použitie auta. Stav auta sa zmení z “Reserved” (Rezervované) na “On the road” (Vypožičané).
- **Uzamknutie auta:** Systém umožní užívateľovi zamknúť auto, opäť bez použitia kľúča. Zamknutie nie je možné, ak nie je užívateľ fyzicky v bezprostrednej blízkosti auta.
- **Odovzdanie auta:** Systém umožní užívateľovi odovzdať auto (to znamená, ukončiť výpožičku). Stav auta sa zmení z “On the road” (Vypožičané) na “Available” (Dostupné), čím sa opäť sprístupní ostatným záujemcom. V systéme sa uloží doba, počas ktorej bolo auto vo výpožičke a počet najazdených kilometrov. Odovzdaním auta zaň užívateľ stráca zodpovednosť.

4.2 Užívatelia

- **Predregistrácia užívateľa:** Systém umožní užívateľovi predregistráciu do služby carsharingu. Pri využití tejto možnosti sa skráti čas nutný pri osobnej registrácii. V tomto prípade nebude nutné vypĺňať osobné údaje, postačí ich overenie.
- **Prihlásenie užívateľa:** Systém umožní užívateľovi vyplniť prihlasovacie údaje a prihlásiť sa, čo je nevyhnutné pre sprístupnenie ostatných funkcií.

4.3 Rezervácie

- **Vytvoriť rezerváciu:** Systém umožní užívateľovi vytvoriť rezerváciu auta. Počas jej trvania auto zmizne zo zoznamu áut voľných pre ostatných užívateľov. Užívateľ, ktorý si auto zarezervoval, získa možnosť aut odomknúť alebo zamknúť. Stav auta sa zmení z "Available" (Dostupné) na "Reserved" (Rezervované).
- **Zrušenie rezervácie:** Systém umožní užívateľovi zrušiť rezerváciu. Stav auta sa zmení z "Reserved" (Rezervované) na "Available" (Dostupné).

4.4 Jazdy

- **Zobrazenie prehľadu jázd:** Systém umožní užívateľovi zobraziť prehľad rezervácií a jázd, ktoré sa k nemu viažu. Prehľad bude obsahovať napríklad tieto informácie:
  - Dátum výpožičky DD.MM.YY
  - Uhradená čiastka za výpožičku v CZK
  - Trvanie rezervácie v HH.MM.SS
  - Požičané auto
  - Dĺžka trasy v km
  - Zobrazenie trasy na mape
- **Zistenie predpokladanej ceny jazdy:** Ak užívateľ pozná cieľ svojej jazdy, môže si po zadaní plánovanej destinácie zistiť, aká bude jej predpokladaná cena.
- **Zistenie aktuálnych informácií o jazde:** Systém umožní počas jazdy zobraziť aktuálne informácie ako sú:
  - Cena v CZK
  - Dĺžka jazdy v km
  - Spotrebované palivo v litroch
  - Rýchlosť v km/hod
  - Otáčky motora za minútu
- **Odkaz na aplikáciu Google Maps pre navigáciu:** Ak užívateľ pozná smer svojej jazdy systém mu umožní po zadaní cieľovej destinácie využiť rýchly odkaz na aplikáciu Google Maps s predvyplnenými údajmi.
- **Navigácia k autu pomocou Google Maps:** Systém umožní užívateľovi možnosť navigácie k autu pomocou odkazu na aplikáciu Google Maps s predvyplnenými údajmi.

Kapitola 5
Návrh vzhľadu

Úspech mobilnej aplikácie veľmi záleží na jej dizajne. Ak nie je užívateľsky prívetivá, nikto ju nebude používať. Užívateľské prostredie **nesmie** prekážať v interakcii so systémom a brániť tak jeho využívaniu, ani užívateľovi navodiť pocit vlastnej frustrácie alebo neschopnosti. Mobilnú aplikáciu bude používať každý študent využívajúci služby carsharingu, preto návrhu užívateľského rozhrania bolo nevyhnutné venovať veľkú pozornosť.

Pri návrhu som sa vyhýbal všetkým neštandardným užívateľského rozhrania prvkom a riadil som sa základnými zásadami Material dizajnu.

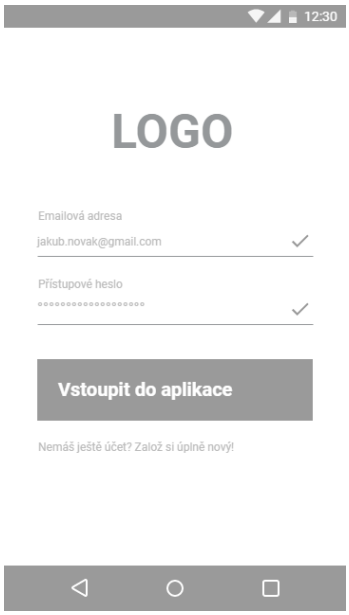
5.1 Wireframes

Wireframe je schematická reprezentácia užívateľského rozhrania, ktorá nezachádza do príliš hlbokých detailov. Nejedná sa o grafický návrh - neobsahuje obrázky, farby ani iné dizajnové špecifiká. Viac v [6].

Priložené wireframes boli vytvorené na základe poskytnutých podkladov grafikom, ktorý pracuje na projekte.

5.1.1 Prihlasovacia obrazovka

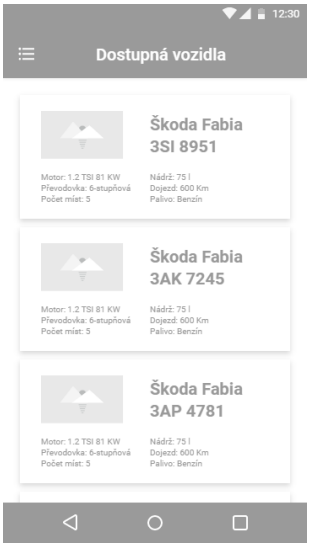
Užívateľ zadá svoj e-mail a heslo a následne stlačí tlačidlo „Vstoupit do aplikace“. Ak ešte prihlasovacie údaje nemá, klikne na text „Nemáš ještě účet? Založ si nový!“.



Obrázok 5.1. Wireframe prihlasovacej obrazovky.

5.1.2 Zoznam vozidiel k dispozícii

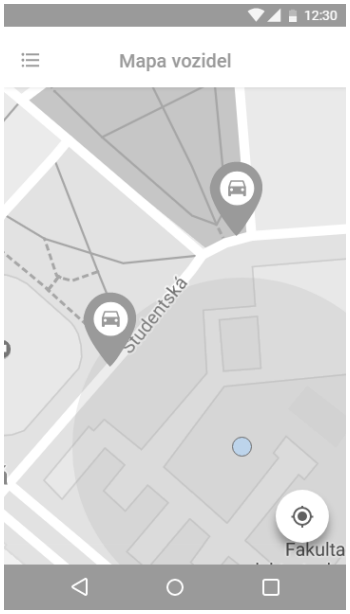
Na obrazovke je vidieť skrolovací zoznam vozidiel, ktoré sú k dispozícii na požičanie s ich základnými parametrami: model auta, ŠPZ, motor, prevodovka, počet miest na sedenie, stav palivovej nádrže, približný dojazd a typ paliva.



Obrázok 5.2. Wireframe zoznamu voľných vozidiel.

5.1.3 Mapa voľných vozidiel

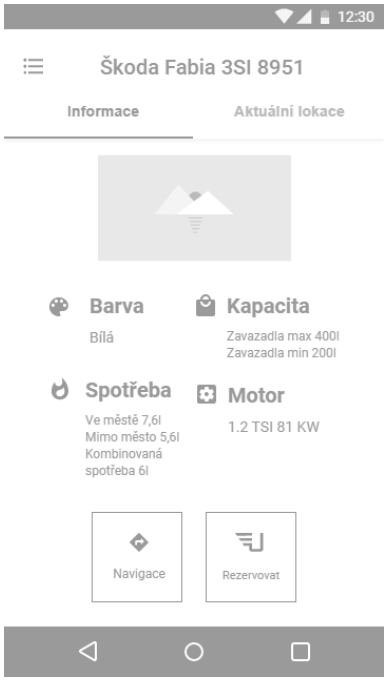
Na tejto obrazovke sa nachádza mapa so zaznačenými polohami vozidiel, ktoré sú k dispozícii na požičanie.



Obrázok 5.3. Wireframe mapy voľných vozidiel.

5.1.4 Detailné informácie o vozidle

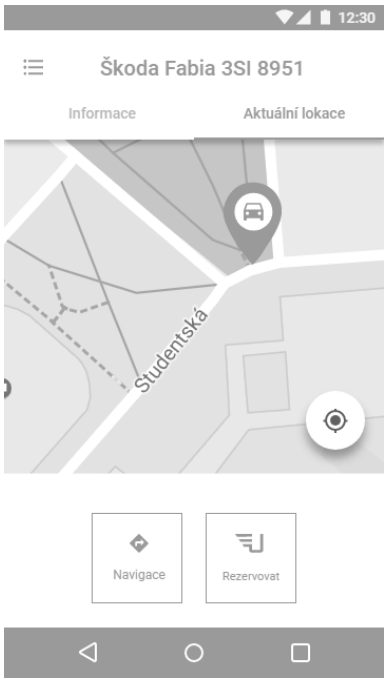
Na obrazovke sú detailné informácie o vozidle spolu s tlačidlom pre navigáciu k vozidlu a tlačidlom pre vytvorenie rezervácie.



Obrázok 5.4. Wireframe detailných informácií o vozidle.

5.1.5 Mapa vybraného vozidla

Na obrazovke je mapa s markerom vozidla, na detail ktorého sa užívateľ pozerá, spolu s tlačidlom pre navigáciu k vozidlu a tlačidlom pre vytvorenie rezervácie.



Obrázok 5.5. Wireframe mapy vybraného vozidla.

5.1.6 Aktuálna rezervácia

Na obrazovke sú informácie o rezervovanom aute spolu s tlačidlom pre navigáciu k vozidlu a tlačidlom pre zrušenie rezervácie.

Obrázok 5.6. Wireframe aktuálnej rezervácie.

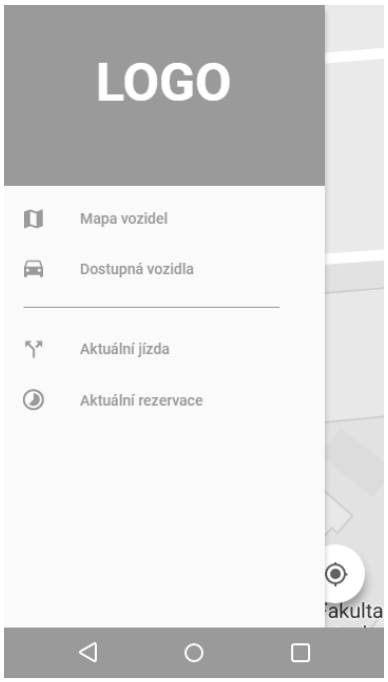
5.1.7 Potvrdenie rezervácie

Na obrazovke je dialógové okno s potvrdením rezervácie vozidla.

Obrázok 5.7. Wireframe potvrdenia rezervácie.

5.1.8 Navigation Drawer

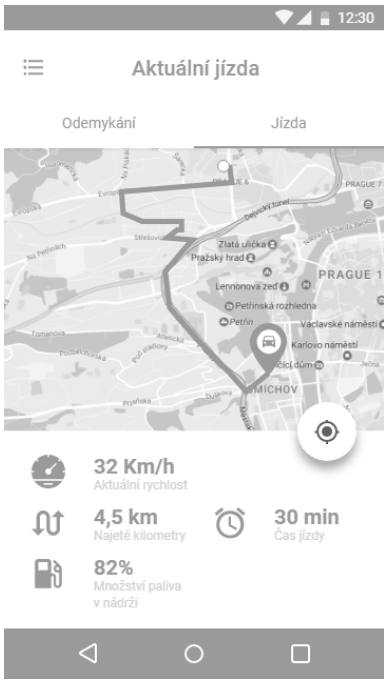
Na obrazovke sa nachádza Navigation Drawer s položkami „Mapa vozidel“, „Dostupná vozidla“, „Aktuální jízda“, „Aktuální rezervace“



Obrázok 5.8. Wireframe navigation draweru.

5.1.9 Mapa a informácie o aktuálnej jazde

Na obrazovke sa nachádza mapa so zaznačenou trasou jazdy a jazdné informácie.



Obrázok 5.9. Wireframe obsahujúci mapu a informácie o aktuálnej jazde.

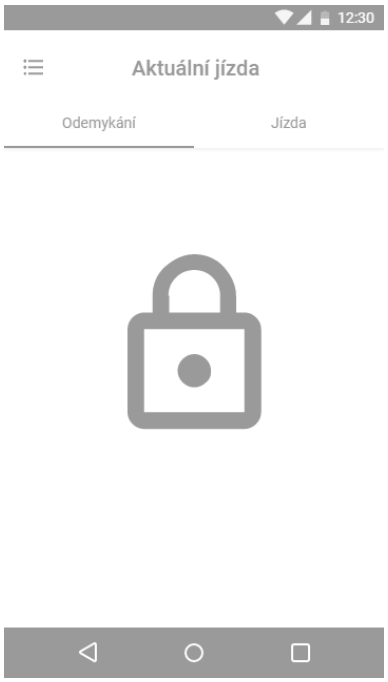
5.1.10 Odomknutie vozidla

Na obrazovke je tlačidlo slúžiace na odomknutie vozidla.

Obrázok 5.10. Wireframe pre odomknutie vozidla.

5.1.11 Zamknutie vozidla

Na obrazovke je tlačidlo slúžiace na zamknutie vozidla.



Obrázok 5.11. Wireframe pre zamknutie vozidla.

Kapitola 6
Implementácia

Táto kapitola sa venuje hlavnej časti práce - implementácii. Popisuje hlavné java triedy aplikácie ako sú aktivity, fragmenty, ale aj súbory pre automatizáciu zostavovania programu a Android manifest.

6.1 build.gradle

Tento súbor obsahuje zostavovací konfiguračný skript pre Gradle. Podstatné údaje, vyžadujúce definíciu sú:

- **compileSdkVersion:** Verzia Androidu používaného pre testovanie a debug aplikácie. V našom prípade je to API 25 - Android 7.1 (Nougat)
- **minSdkVersion:** Najstaršia verzia Android SDK na ktorej môže aplikácia bežať. Tento údaj je nutný, pretože starším verziám môže chýbať požadovaná funkcionalita. V našom prípade API 19 - Android 4.4 (KitKat), podľa aktuálneho podielu verzií systému je teda podporovaných 89.1 percent zariadení.
- **targetSdkVersion:** Verzia na ktorú bol vývoj aplikácie cielený. V našom prípade 25 API 23 - Android 6.0 (Marshmallow) ktorý tvorí aktuálne najviac, čiže 31.2 percent zariadení.
- **dependencies:** Definované závislosti. Medzi nimi sú napríklad knižnice appcompat (set podporných knižníc ktoré sa používajú na to aby sa mohli aplikácie vyvíjať na nové verzie systému a pritom boli stále komplatibilné), play-services-maps (knižnice pre prácu s Google Maps), play-services-location (knižnice pre prácu s lokáciou zariadenia).

  Zaujímavé sú ale aj knižnice tretej strany ako napríklad Retrofit 2 od firmy Square, Inc. Táto knižnica slúži ako HTTP client pre Android. API sa definuje ako Java interface. Retrofit trieda generuje implementáciu interfacu. Pomocou tejto triedy sa vykonávajú asynchrónne HTTP requesty na externý server. Asynchronita volaní je veľmi dôležitá vlastnosť, pretože sa vývojár nemusí starať o komplikovanú a zdĺhavú implementáciu abstraktnej triedy AsyncTask. Ďalej sa používa knižnica od Google - GSON ktorá sa používa pre konvertovanie Java objektov do ich JSON reprezentácie a späť.

  Ďalšou použitou knižnicou je Butterknife. Slúži pre injectovanie Views do komponent Android, čo skracuje a veľmi sprehľadňuje kód. Vývojár sa tak môže sústrediť na iné časti vývoja.

6.2 balíček service

Balíček service obsahuje interface RestApi a balíček model s POJO triedami, ktoré reprezentujú vstupné parametre HTTP volaní aj odpovede na ne.

6.2.1 Príklad definície HTTP requestu:

```
@GET("cardetail/{id}")
Call⟨CarDetail⟩ getCarDetail(@Path("id") Long id);
```

Anotácia @GET označuje HTTP metódu. Ako parameter preberá URL konkrétneho API prípadne s URL parametrom. Metóda vždy vracia generický objekt Call a prijme triedu, ktorú má HTTP request vrátiť. Táto trieda môže byť aj list tried. Metóda preberá už spomenuté URL parametre alebo telo requestu (pomocou anotácie @Body).

6.3 LoginActivity

Táto aktivita je zodpovedná za proces prihlásenia užívateľa. Ten musí najskôr do vstupných polí zadať e-mail, potom heslo ku službe. Aktivita skontroluje, či užívateľ údaje vyplnil, prípadne či zadal e-mail v požadovanom tvare. Po overení údajov aplikácia pošle http request na routu servera, kde je vystavené API pre prihlásenie. V prípade, že zadané heslo zodpovedá e-mailu, server vráti token JWT autentifikácie. Tento token si aplikácia uloží do Shared Preferences a pomocou intentu zavolá hlavú activitu - MainActivity.

Na prihlasovacej obrazovke je možnosť kliknúť na text Nemáš účet? Pridaj sa k nám!, čím sa pomocou intentu spustí UserActivity.

6.4 UserActivity

Táto aktivita má na starosti predregistráciu užívateľa a editovanie jeho údajov. Aktivita sa pozrie do SharedPreferences a skontroluje, či sa tam nachádza token z JWT autentifikácie. Ak nie, jedná sa o predregistráciu. V opačnom prípade o editáciu údajov a aktivita pošle http request na API serveru ktorý vráti údaje o prihlásenom užívateľovi. Tieto údaje sú zobrazované v užívateľskom rozhraní.

Pri kliknutí na editovateľný údaj sa vždy spustí nová aktivita s užívateľským rozhraním vytvoreným za účelom získanie údajov pre konkrétnu časť informácií (napríklad adresa trvalého bydliska). Tieto údaje sú propagované späť do UserActivity.

6.4.1 MainActivity

Hlavná aktivita aplikácie, ktorej užívateľské rozhranie pozostáva hlavne z Navigation Draweru. Tento prvok material designu obsahuje hlavné menu s nasledujúcimi prvkami:

- **Mapa:** Po kliknutí na tento prvok menu sa pomocou inštancie triedy FragmentManager (trieda ktorá slúži na prácu s fragmentami) zavolá a zobrazí AvailableCarsMapFragment
- **Dostupná vozidla:** Po kliknutí sa zobrazí AvailableCarsListFragment
- **Jízdy:** Pomocou intentu sa spustí aktivita RideActivity
- **Rezervace:** Pomocou intentu sa spustí aktivita ReservationActivity

6.4.2 AvailableCarsList Fragment

Užívateľské rozhranie tohoto fragmentu pozostáva z RecyclerView. Po vytvorení fragmentu sa zavolá metóda, ktorá vytvorí http request na server. Ten vráti JSON obsahujúci java List voľných vozidiel s ich základnými parametrami. Po úspešnej HTTP odpovedi sa k RecyclerView z užívateľského rozhrania priradí novo vytvorený adapter - AvailableCarsListAdapter, ktorého konštruktor preberá list voľných vozidiel.

6.4.3 AvailableCarsMap Fragment

Tento fragment implementuje rozhrania OnMapReadyCallback, ConnectionCallbacks, OnConnectionFailedListener, LocationListener. Sú dôležité pre funkcionalitu spojenú s Google Maps a lokáciou zariadenia. Po vykonaní nutných metód pre získanie Google Mapy sa spustí metóda pre vytvorenie HTTP requestu na API pre získanie lokácií voľných vozidiel zo serveru. Po úspešnom vykonaní requestu sa pre každé voľné vozidlo na Google mape vytvorí marker. Užívateľ môže naň kliknúť, čím sa pomocou intentu vyvolá aktivita CarDetailActivity.

6.5 CarDetailActivity

Aktivita pre zobrazenie detailných informácií o vozidle, ktoré je k dispozícií. Jej užívateľské rozhranie pozostáva z panelu slúžiaceho na prepínanie medzi fragmentami.

6.5.1 CarDetailInfoFragment

Fragment, ktorého užívateľské rozhranie sa skladá zo ScrollView obsahujúceho obrázok auta a detailných informácií o ňom, vrátane spotreby, objemu kufru a vybavenia. Pre ScrollView som sa rozhodol z dôvodu možného výskytu takého množstva informácií o aute, že ich grafická reprezentácia by mohla presahovať dĺžku obrazovky. ScrollView umožní zoskrolovať na údaje,ktoré sa nezmestili na obrazovku. Okrem spomenutých prvkov sú pod ScrollView s informáciami na obrazovke umiestnené ešte 2 tlačidlá - jedno pre navigáciu k automobilu a druhé pre jeho rezerváciu. Ak užívateľ klikne na tlačidlo pre navigáciu k vozidlu, pomocou intentu sa pre navigáciu zavolá Google maps. Po kliknutí na tlačidlo rezervácie sa zobrazí dialógové okno, ktoré upozorní užívateľa, že po potvrdení rezervácie sa mu začne podľa aktuálne platného cenníka účtovať poplatok za čas.

6.5.2 CarDetailMapFragment

Tento fragment má užívateľské rozhranie pozostávajúce z Google Mapy, na ktorej je marker s aktuálnou polohou auta, získanou HTTP requestom na server. Aj na tomto fragmente sú umiestnené tlačidlá pre rezerváciu a navigáciu, ich akcie sú rovnaké ako pri predchádzajúcej aktivite.

6.6 RideActivity

Aktivita slúžiaca na reprezentáciu aktuálne prebiehajúcej jazdy. Jej užívateľské rozhranie obsahuje panel, slúžiaci na prepínanie medzi fragmentami. Jedným z nich je fragment mapy jazdy a realtime informácií o jazde - MapInfoFragment a druhým fragment, ktorý obaľuje logiku komunikácie s autom - CarInstructionFragment.

6.6.1 MapInfoFragment

Fragment je zobrazovaný iba v prípade, že užívateľ už jazdu začal - prvýkrát otvoril auto. Jeho užívateľské rozhranie sa skladá z Google mapy, pod ktorou sú umiestnené a periodicky obnovované jazdné údaje (rýchlosť, otáčky, spotreba, atď). Časový interval pre obnovenie údajov je zatiaľ nastavený na 10 sekúnd, po otestovaní aplikácie s autom sa však môže zmeniť. Refresh prebieha HTTP requestom na API. Server vráti okrem spomínaných údajov aj aktuálne GPS súradnice auta. Podľa nich sa na Google mape zaktualizuje poloha markeru a vykresľovaná trasa ktorú užívateľ išiel.

6.6.2 CarInstruction Fragment

Tento fragment má za úlohu interakciu užívateľa s autom. Užívateľské rozhranie sa skladá z jediného tlačidla. Pri jeho inicializácií sa pošle HTTP request na server. Ten vráti informáciu, či je auto zamknuté, alebo odomknuté a podľa nej sa nastaví ikona tlačidla.

Po aktivácii tlačidla sa overí, či sa užívateľ nachádza v blízkosti auta. V hardvérovej jednotke sa nachádza Bluetooth modul, ktorý je permanentne zapnutý. Aplikácia si vyžiada, aby užívateľ zapol Bluetooth v svojom zariadení a keď to urobí, zoskenuje okolité zariadenia. Ak sa medzi nimi nachádza zariadenie s MAC adresou zodpovedajúcou MAC adrese Bluetooth zariadenia jednotky v aute, auto odošle HTTP request s inštrukciou otvorenia vozidla. Toto overenie je zatiaľ v prvej verzii - je jednoduché a nie je ťažké ho obísť. Aj keď je unikátnosť MAC adresy zaručená IEEE, dá sa nastaviť softvérovo. Útočník by si mohol sám zoskenovať zariadenia, zistiť MAC adresu Bluetooth zariadenia auta, inému zariadeniu nastaviť na danú hodnotu a tým pádom by toto overenie zlyhalo. V ďalšej verzii sa uvažuje overenie s úpravami spomenutými v závere práce. Veľkou výhodou tohoto typu overenia ale je, že auto ani užívateľ nemusia mať k dispozícii GPS a je teda plne funkčné aj v podzemných garážach.

Po poslaní HTTP requestu sa spustí progressbar a zároveň sa periodicky začnú posielať HTTP requesty pre skontrolovanie stavu inštrukcie poslanej vozidlu. Keď sa požiadavka vykoná, progressbar zmizne, requesty sa prestanú posielať a ikona tlačidla sa vymení.

6.7 ReservationActivity

Aktivita slúžiaca pre správu rezervácie. Ak užívateľ má aktuálne zarezervované auto, s ktorým ešte nezačal jazdu (neotvoril ho), zobrazia sa mu jeho základné parametre. Po kliknutí na prvok reprezentujúci auto sa pomocou intentu spustí aktivita CarDetailActivity 6.5 odkaz. K dispozícii je i tlačidlo presmerovania na aktivity aktuálnej jazdy - RideActivity.

Posledným prvkom je tlačidlo slúžiace na zrušenie rezervácie. Po jeho aktivácii kliknutím sa zobrazí potvrdzovacie dialógové okno. Po jeho potvrdení sa odošle HTTP request na server a zobrazí sa dialógový progressbar, ktorý počká na jeho odpoveď. Ak zrušenie rezervácie prebehne v poriadku, pomocou intentu sa aplikácia vráti späť na hlavnú aktivitu - MainActivity 6.4.1.

6.8 utils

V balíčku utility sa nachádza trieda CarImageUtils so statickou metódou, ktorá preberá meno modelu auta. Podla neho určí obrázok auta, ktorý má adaptér zobraziť.

6.9 AndroidManifest.xml

Bezpečnosť sa stala jednou z najdôležitejších požiadaviek užívateľov mobilných zariadení. Preto sú oprávnenia aplikácie jednou z jej kritických častí. Užívateľ je pri inštalácii aplikácie požiadaný o schválenie oprávnení a mal by si z nich zvoliť čo najmenšiu množinu, aby tak minimalizoval potenciálne riziká.

V našom prípade sú to tieto oprávnenia:

- **BLUETOOTH, BLUETOOTH_ADMIN:** Oprávnenia nutné pre prácu s Bluetooth perifériou zariadenia. Sú nevyhnutné pre overenie fyzicke vzdialenosti užívateľa od auta.
- **ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION:** Oprávnenie nutné pre lokalizáciu užívateľa na Gooogle mapách. V prípade, že toto overenie užívateľ nepovolí, na mape ktorá sa bude zobrazovať nebude vidieť svoju lokáciu.
- **INTERNET, ACCESS_NETWORK_STATE:** Tieto oprávnenia sú nutné pre všetku komunikáciu so serverom.
- **READ_GSERVICES, MAPS_RECEIVE:** Oprávnenia nutné pre zobrazovanie Google mapy.

Kapitola 7
Testovanie s užívateľom.

Bakalárska práca sa okrem návrhu a implementácie aplikácie zaoberá aj testovaním jej užívateľského rozhrania. Pre testovanie jeho intuitívnosti existuje niekoľko metód, my sme sa rozhodli použiť nižšie popísanú metódu heuristickej evaluácie. Dôvodom nášho rozhodnutia je, že jej použitie trvá iba zlomok času oproti empirickým metódam, ako napríklad usability testovanie. Teória k testovaniu je prevzatá z [7]

7.1 Testovanie heuristickou evaluáciou

Heuristické hodnotenie (heuristic evaluation) predstavil v roku 1990 expert na použiteľnosť Jakob Nielsen. Je to jedna zo základných metód testovania použiteľnosti, spočívajúca v prechádzaní a hodnotení stránok odborníkmi na použiteľnosť pomocou overených princípov použiteľnosti, tzv. heuristík. Tieto pravidlá alebo kritériá, pomáhajú odhaliť problémy, na ktoré by s vysokou pravdepodobnosťou narazil koncový užívateľ systému.

Najčastejšie používanú množinu heuristík tvorí desať všeobecných princípov:

1. **Viditeľnosť stavu systému:** Systém by mal užívateľovi vedieť, čo sa práve odohráva
2. **Väzba medzi systémom a reálnym svetom:** Komunikácia systému s užívateľom by sa mala odohrávať užívateľsky príjemným spôsobom (zrozumiteľný jazyk bez odborných termínov)
3. **Užívateľské riadenie a sloboda:** Možný únikový východ pre návrat do predchádzajúceho stavu v prípade chyby počas práce v systéme.
4. **Konzistencia a štandardy:** Užívateľ by nemal byť nútený rozmýšľať, či rôzne termíny znamenajú to isté, preto by mali byť dodržiavané všeobecné zásady.
5. **Prevencia chýb:** Vyvarovať sa chybovým hláseniam bezpečným designom, preventívne pôsobiacim proti problémom.
6. **Rozpoznávanie pred spomínaním:** Užívateľ by nemal byť nútený spomínať si na vykonávanie operácií v systéme, inštrukcie by mali byť v systéme vždy viditeľne umiestnené.
7. **Flexibilita a efektivita používania:** Umožnenie zrýchlenia práce so systémom pre pokročilého užívateľa.
8. **Estetický a minimalistický design:** Bez nepotrebných informácií.
9. **Rozoznávanie chýb a zotavenie z nich:** Chybové hlášky by mali byť uvedené v prirodzenom jazyku a mali by navrhovať riešenie.
10. **Podpora a dokumentácia:** Všetky informácie sa musia dať jednoducho vyhľadať, nápoveda by mala obsahovať postupy v krokoch.

Výstup hodnotenia predstavuje zoznam nájdených problémov vrátane odpovedajúcej heuristiky, ktorú rozhranie porušuje, aby bolo možné rozhranie upraviť. Vo väčšine prípadov sa každý problém ohodnotí číselnou hodnotou vyjadrujúcou závažnosť problému.

- **0** – nie je problém týkajúci sa použiteľnosti
- **1** – veľmi malý problém, ktorý nie je nutné opraviť pri nedostatku času

- **2** – malý problém, ktorého oprava má nízku prioritu
- **3** - veľký problém, ktorého oprava je dôležitá
- **4** - závažný problém, ktorý musí byť opravený pred vydaním systému

7.2 Cieľová skupina

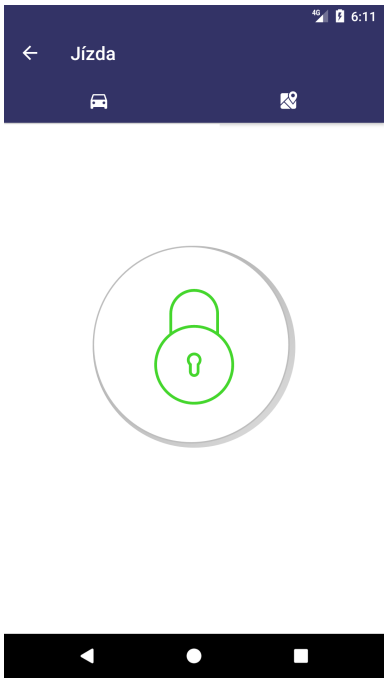
Cieľovou skupinou sú študenti vysokej školy, ktorí mobilnú aplikáciu nikdy nepoužili, ale so službou carsharingu sú oboznámení. Predpokladá sa u nich, že denne používajú niekoľko mobilných aplikácií, preto sa neočakávajú problémy s prácou s ňou.

7.3 Testovanie

V projekte pracuje tým grafikov, ktorí sa podieľajú aj na grafike mobilnej aplikácie. Pre testovanie som vybral iba také prípady užitia, ktorých grafická časť aplikácie má finálny návrh a bolo ho možné naimplementovať. Sú to následovné prípady užitia:

- Prihlásenie sa do aplikácie
- Zobrazenie zoznamu voľných automobilov
- Zobrazenie detailu vozidla
- Odomknutie automobilu

7.3.1 Nálezy



Obrázok 7.1. Odomknutie automobilu

- **Prípad užitia:** Odomknutie automobilu
- **Heuristika:** 1 - Viditeľnosť stavu systému
- **Závažnosť:** 4 - Závažný problém, ktorý musí byť opravený pred vydaním systému

- **Popis problému:** Užívateľ by ikonu zámku mohol pochopiť 2 spôsobmi. V prvom prípade by mohol usúdiť, že zámok znázorňuje stav auta - ak je zámok zamknutý je zamknuté aj vozidlo. V druhom prípade by ikonu mohol pochopiť ako stav do ktorého sa vozidlo uvedie - ak je zámok zamknutý auto sa po kliknutí naň zamkne.
- **Návrh riešenia:** Pridať tlačidlo a prísne tak rozdeliť dve rôzne tlačidlá - jedno pre zamknutie auta, druhé pre odomknutie - ako to je na reálnom kľúči od auta.

Kapitola 8
Záver

Cieľom našej práce bolo vytvoriť mobilnú aplikáciu pre systém zdieľania automobilov užívateľmi. Práca sa zaoberá jej návrhom, implementáciou a tiež testovaním užívateľského rozhrania.

V kapitole Mobilné aplikácie sme vysvetlili dôvody, ktoré viedli k rozhodnutiu vytvoriť natívnu mobilnú aplikáciu, namiesto webovej aplikácie optimalizovanej aj pre mobilné zariadenia. Zdôvodnili sme i výber operačného systému Android, pre ktorý je aplikácia určená. Popísali sme štruktúru zdrojového kódu aplikácie, vysvetlili základné komponenty, tvoriace aplikáciu a charakterizovali niektoré prvky, ktoré boli v implementácii použité.

V kapitole, ktorá sa venuje backendu celého systému, sme zdôvodnili výber frameworku pre jeho implementáciu. V najdôležitejšej časti sme popísali aplikačné rozhranie backendu pre komunikáciu s mobilnou aplikáciou.

Neoddeliteľnou súčasťou vývoja softvéru je fáza návrhu jeho funkcionality, požiadaviek, ktoré musí spĺňať. Bolo potrebné navrhnúť najzákladnejšie prípady využitia aplikácie, medzi ktoré patrí registrácia, prihlásenie do služby, zobrazenie voľných vozidiel na mape spolu s ich základnými parametrami, zobrazenie detailu vozidla, rezervácia vozidla a podobne.

Pri návrhu užívateľského rozhrania sme sa snažili, aby bolo jednoduché a užívateľsky prívetivé. Jeho prílišná zložitosť by mohla nielen skomplikovať implementáciu, ale aj odradiť užívateľov od používania aplikácie.

Jadrom našej práce bola samotná implementácia mobilnej aplikácie. Postupne sme popísali kritické časti súboru AndroidManifest.xml a build.gradle, rozobrali takmer všetky aktivity, z ktorých aplikácia pozostáva. Výnimku tvorili tie z nich, ktoré sú použité pre správu údajov spojených s užívateľom a to najmä z dôvodu ich triviálnosti. Pri vačšine aktivít sme vysvetlili aj časti (fragmenty), z ktorých sa skladajú. Popísali sme tiež dve knižnice, použité pre zjednodušenie vývoja a zvýšenie jeho produktivity. Prvá z nich sa nazýva Retrofit a zaobaľuje logiku posielania a handlovania HTTP reqestov. Druhá, Butteknife, je použitá na zjednodušenie prepojenia java tried so súbormi, ktoré definujú užívateľské rozhranie. Určitú časť práce sme venovali popisu problematiky overenia bezprostrednej fyzickej prítomnosti užívateľa pri vozidle počas interakcie s ním.

Na záver sme vysvetlili metodiku, ktorá bola zvolená pre testovanie užívateľského rozhrania aplikácie. V teste sa objavil aj dôležitý nález - funkcia tlačidla pre otváranie a zatváranie auta by mohla byť užívateľmi chápaná rozdielne. Tlačidlo s ikonou odomknutého zámku by mohol jeden užívateľ interpretovať ako príkaz na odomknutie vozidla a iný užívateľ ako stav, že vozidlo je odomknuté a príkaz na zamknutie vozidla. To by sa však dalo vyriešiť rozdelením tlačidla na dve.

8.1 Budúcnosť

Bakalárska práca je súčasťou iného projektu, preto ďalšie kroky priamo súvisia s jeho pokračovaním. Ku dňu 1. 9. 2017 je plánované spustenie testovacej prevádzky, z toho

dôvodu sa dovtedy musí dostať do fázy MVP (Minimum Viable Product). Ten je daný požiadavkami, ktoré sú v prílohe práce.

Projekt má veľký potenciál rásť a v nasledujúcich fázach budeme pracovať na pridávaní funkcionalít a ponúkanú službu postupne rozvíjať. Budú implementované nasledujúce nové časti:

- Cenové balíčky: V dôsledku veľkého množstva spôsobov využívania služby (napr. krátka jazda z internátov do školy, víkendový výlet, atď.) plánujeme ponúknuť možnosť výberu z viacerých cenových balíčkov.
- Optimalizácia jázd vozidiel: V budúcnosti bude potrebné vozidlá udržiavať na dobre prístupných a frekventovaných miestach. Užívatelia budú motivovaní k tomu, aby vozidlo, zaparkované v menej obývanej časti Prahy, odovzdali napríklad pred jednou z vysokých škôl. Motivácia bude spočívať v cenovom zvýhodnení jazdy zakončenej na požadovanom mieste.
- Spolujazda: Chceli by sme pridať možnosť využitia zdieľania jedného vozidla viacerými užívateľmi. Ten z nich, ktorý má auto zarezervované, by oznámil cieľ svojej cesty, napríklad zo Strahova na Karlovo námestie a iný užívateľ, s rovnakou cestou, by sa mohol odviesť s ním.
- Rôzne typy rezervácií: V rôznych situáciách užívateľ môže potrebovať rôzny typ rezervácie. Napríklad ak má užívateľ v úmysle ísť nakúpiť nábytok, je pre neho dôležitejší model auta (požiadavka na objem kufru, možnosť sklopenia zadných sedadiel) ako miesto, na ktoré by si prišiel vozidlo vyzdvihnúť.

Literatúra

[1] *Android dokumentácia pre vývojárov*. 2017. `https://developer.android.com/reference/classes.html`.

[2] Lauren Darcey a Shane Conder. *Sams teach yourself Android application development in 24 hours*. Pearson Education, 2011.

[3] Khalid Alharbi a Tom Yeh. *Collect, decompile, extract, stats, and diff: Mining design pattern changes in Android apps.* In: *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2015. 515–524.

[4] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Joaquim Romaguera i Ramió, Max Jacobson a Ingrid Fiksdahl-King. *A pattern language*. Gustavo Gili, 1977.

[5] Michael Jones, John Bradley a Nat Sakimura. *Json web token (jwt)*. .

[6] Cennydd Bowles a James Box. *Undercover user experience design*. Pearson Education, 2010.

[7] Bc. Petra Ivašková. *Hodnocení použitelnosti a prototypování uživatelských rozhraní webových aplikací*. Diplomová práca, Masarykova univerzita. Brno, 2013.

Príloha A
Prílohy

A.1 Rozšírenie funkcionality sytému (požiadavky)

Požiadavky ktoré neboli obsaho práce, no je nevyhnutné aby boli implementované dokedy projekt nebude vo fázy minimálneho cenného produktu.

- **Filtrácia áut podľa vybavenia** Systém umožní užívateľovi použiť filter výbavy auta. Užívateľovi sa zobrazia iba autá, ktoré spĺňajú jeho požiadavky a zároveň sú k dispozícii. Požiadavkami výbavy myslíme napríklad:
  - Prítomnosť navigácie v aute
  - Manuálna resp. automatická prevodovka
  - Počet miest na sedenie vrátane vodiča
  - Dojazd auta v km bez dotankovania
- **Zistenie informácií o aute z QR kódu:** Užívateľ môže pomocou aplikácie zoscanovať QR kód, ktorý je umiestnený na aute a následne si zobraziť dostupné informácie o aute.
- **Odoslať report o poškodení auta:** Ak užívateľ zistí pred použitím auta, že je poškodené, špinavé alebo inak znehodnotené, je potrebné oznámiť to pomocou aplikácie spolu s priloženou fotografiou. Tá bude uložená v systéme pre potreby ďalšej evidencie. Aplikáciou je užívateľ povinný oznámiť aj škodu, ktorú spôsobil počas výpožičky na aute on sám.
- **Deaktivácia účtu:** Ak užívateľ nebude chcieť služby carsharingu ďalej využívať, bude môcť svoj účet deaktivovať, čím sa jeho účet stane neaktívny.
- **Zistenie miesta kde je možné odovzdať auto:** Systém užívateľovi zobrazí lokality, v ktorých môže auto odovzdať. Informuje ho aj o možnosti odovzdania v tej lokalite, kde sa práve nachádza.
- **Zobrazenie prehľadu jázd:** Systém umožní užívateľovi zobraziť prehľad rezervácií a jázd, ktoré sa k nim viažu. Prehľad bude obsahovať napríklad tieto informácie:
  - dátum výpožičky DD.MM.YY
  - uhradená čiastka za výpožičku v CZK
  - trvanie jazdy v HH.MM.SS
  - požičané auto
  - dĺžka trasy v km
  - zobrazenie trasy na mape
- **Zistenie predpokladanej ceny jazdy:** Ak užívateľ pozná cieľ svojej jazdy, môže si po jeho zadaní zistiť, aká bude jej predpokladaná cena.

Príloha B
Zkratky a symboly

B.1 Zkratky

JSON	JavaScript Object Notation - formát pre výmenu dát
MVP	Minimum Viable Product - produkt s nejmenšou možnou funkcionalitou, aby ho použila podstatná časť cieľovej skupiny