FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Rozšíření funkcionalit Android aplikace Uniqway
Student:	Rostislav Osvald
Vedoucí:	Ing. Václav Jirovský, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je návrh a implementace rozšíření Android aplikace Uniqway pro sdílení vozidel o funkce historie jízd, tankování (včetně nahrání fotografie účtenky na backend) a péče o vozidla.
1. Ve spolupráci s týmem, který se podílí na projektu Uniqway, specifikujte požadavky a navrhněte funkcionalitu rozšíření. Návrh dokumentujte pomocí vhodných UML diagramů.
2. Návrh implementujte a napojte na existující API projektu.
3. Řešení vhodným způsobem zdokumentujte a řádně otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 28. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Rozšíření funkcionalit Android aplikace Uniqway

Rostislav Osvald

Katedra Softwarového Inženýrství
Vedoucí práce: Ing. Václav Jirovský, Ph.D.

30. července 2020

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Václavu Jirovskému, Ph.D. za cenné rady a připomínky při vedení této bakalářské práce. Také chci poděkovat své rodině za podporu v průběhu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona a to na dobu určitou do skončení trvání ochrany dle Smlouvy.

Nakládání s předloženou prací se řídí Smlouvou o spolupráci uzavřenou v návaznosti na spolupráci mezi Českým vysokým učením technickým v Praze a společností ŠKODA AUTO a.s. a Smart City Lab s.r.o. na výzkumném projektu „CarSharing pro vysokoškolské studenty“, uveřejněné v registru smluv na adrese https://smlouvy.gov.cz/smlouva/5973503.

Jsem vázán Smlouvou o zachování mlčenlivosti, že nezpřístupním třetí osobě důvěrné informace, které jsem při své práci na Projektu získal.

Tímto má předložená práce nemůže být zveřejněna po dobu platnosti závazku mlčenlivosti, tj. do 31. května 2023.

V Praze dne 30. července 2020 ……

Odkaz na tuto práci

Osvald, Rostislav. *Rozšíření funkcionalit Android aplikace Uniqway.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Cílem této práce je návrh a implementace rozšíření Android aplikace pro sdílení automobilů Uniqway o specifické funkce (Historie jízd, Tankování, Péče o vozidlo). Funkcionality byly implementovány, otestovány, zdokumentovány a začleněny do systému. Výsledek práce zlepší uživatelský zážitek uvnitř Android aplikace Uniqway a zjednoduší správu vozidel pro tým provozu. V příloze se nachází snímky obrazovek obsahující tyto funkcionality.

Klíčová slova sdílení automobilů, Android, Java, rozšíření mobilní aplikace, Uniqway

Abstract

Goal of this thesis is design and implementation of Android carsharing application extention with specific functionalities (Ride history, Refueling, Car care). Functionalities were implemented, tested, documented and integrated into the system. The result of this thesis enhance user experience of Uniqway Android Application and make vehicle managment easier for operation team. In the attachments are located screenshots containing these functionalities.

Keywords carsharing, Android, Java, mobile application extension, Uniqway

Obsah

Seznam obrázků

Seznam zdrojových kódů

Úvod

Carsharing neboli sdílení automobilů je rychle rostoucí odvětví služeb. Využívat této služby se vyplatí zejména lidem, kterým by se nevyplatil provozovat vlastní automobil z důvodu málo častého použití. Díky eliminaci potřeby uživatelů carsharingů vlastnit automobil ubude prostor, který by tato vozidla zabírala na parkovištích, což je ve velkých městech problém. Další výhodou může být ekologický dopad, jelikož carsharingy často mají k dispozici modernější vozidla, buď elektrická nebo s nižšími emisemi.

Uniqway je jedním z carsharingů, které vlastní půjčovaná vozidla, na rozdíl od carsharingů, kde si uživatelé půjčují vozidla mezi sebou. Uniqway je carsharing provozován a vyvíjen studenty ČVUT v Praze, ČZU v Praze a VŠE v Praze a je určen exkluzivně pro studenty a zaměstnance vysokých škol. Vozidla si uživatelé půjčují za použití mobilní aplikace.

Výsledek práce zlepší uživatelský zážitek uvnitř Android aplikace Uniqway a zjednoduší správu vozidel pro tým provozu. Tato aplikace byla vytvořena na základě bakalářské práce Filipa Ravase a jeho následnou prací společně se zbytkem technického týmu.[2] Součástí práce technického týmu byl obchod s odměnami, který vytvořil Petr Prouza jako součást své bakalářské práce a některé funkce obsažené v této práci budou jeho existence využívat.[3]

Kapitola 1

Cíl práce

Cílem práce je návrh a implementace rozšíření Android aplikace Uniqway pro sdílení vozidel o specifické funkce (Tankování, Historie jízd a Péče o vozidla). Pro vytvoření těchto funkcí je třeba nejdříve specifikovat požadavky a navrhnout funkcionalitu řešení ve spolupráci s týmem podílejícím se na projektu Uniqway a tento návrh následně dokumentovat pomocí vhodných UML diagramů. Následně proběhne implementace funkcí, přičemž výsledek bude komunikovat s existujícím API projektu. Nakonec je třeba všechny tyto funkce řádně otestovat a zdokumentovat.

Kapitola 2

Analýza

Kapitola obsahuje analýzu služby Uniqway a její Android aplikace, kterou má tato práce za cíl rozšířit. Dále se věnuje rešerši podobných funkcionalit v jiných aplikacích a na závěr jsou stanoveny požadavky kladené na rozšíření aplikace.

2.1 Služba Uniqway

„*Projekt Uniqway byl zahájen v roce 2015 kreativní studií zaměřenou na analýzu možnosti aplikace sdílené ekonomiky v oblasti mobility studentů s konkrétním zaměřením na studenty pražských univerzit. Projekt je realizován třemi pražskými univerzitami, Českým vysokým učením technickým v Praze, Vysokou školou ekonomickou v Praze a Českou zemědělskou univerzitou v Praze. Výzkumný projekt je po celou dobu financován společností Škoda Auto a.s. a její dceřinou společností Škoda Auto DigiLab s.r.o.* “[4]

Služba byla spuštěna do provozu 17. října 2018. Funguje tedy již déle než rok a pracuje na ní kolem čtyřiceti studentů.[5]

Uniqway patří mezi tzv. free-floating carsharingy, kde automobily vlastní provozovatel a tato vozidla jsou rozmístěna po určené lokaci. Opakem tohoto typu sdílení vozidel je peer-to-peer, kde vozidla vlastní uživatelé a půjčují si je navzájem mezi sebou.

Pro používání služby mohou uživatelé vyjma Android aplikace využít také iOS aplikaci, kterou vytvořil Michal Černý ve své bakalářské práci „iOS aplikace pro car-sharing Uniqway“[6]. Štěpán Severa ve své práci „Webová aplikace pro uživatele systému sdílení automobilů“[7] vybudoval aplikaci pro web, ale současně není dostupná uživatelům.

2.2 Současný stav aplikace

Základ aplikace Uniqway položil Filip Ravas v rámci své bakalářské práce „Mobilná aplikácia pre užívateľov systému zdieľania automobilov viac užíva-

teľmi“[2]. Tento základ byl rozšířen Filipem Ravasem a ostatními členy Android týmu za následující roky.

Mezi tyto rozšíření se řadí také výsledek bakalářské práce Petra Prouzy s názvem „Rozšíření Android aplikace pro uživatele systému sdílení automobilů o reward shop“[3]. Toto rozšíření je v upravené podobě nově zapojené do systému a zpřístupněné uživatelům.

Aplikace má z velké části dynamický obsah, který získává z API, které je vyvíjeno a provozováno Uniqway backend týmem. Často měnící se data jako je poloha vozidel se aktualizuje častěji než data, která se zřídka mění jako informace o zónách, kde mohou vozidla parkovat.

2.2.1 Současné funkce aplikace

V této podsekci jsou řešené funkce (obrazovka či množina obrazovek sloužící k jednomu účelu) a způsob jakým s nimi uživatel interaguje.

Přihlášení Umožňuje uživateli se přihlásit, aby mohl využít funkce aplikace jako je například rezervace vozidla.

Mapa vozidel Uživatel zde vidí na mapě všechna vozidla, která si může rezervovat. Po kliknutí na jedno z vozidel se zobrazí jeho detail.

Dostupná vozidla Podobně jako mapa vozidel obsahuje vozidla přístupná k rezervaci, ale jsou zobrazeny ve formě seznamu. Po kliknutí na položku se zobrazí detail zvoleného vozidla.

Detail vozidla Ukazuje uživateli základní informace o vozidlu a v případě, že je to možné, umožňuje rezervovat toto vozidlo.

Jízda Mírně upravený detail vozidla. Umožňuje vrátit vozidlo a přejít na další funkce zpřístupněné rezervováním vozidla (například zamykání a odemykání)

Informace o vozidlu Zobrazuje rozšiřující technické informace o vozidlu.

Podpora a informace Obsahuje informace o službě, návod co dělat v určitých situacích, kontaktní informace a další.

Profil Slouží jako rozcestí pro funkcionality spojené s uživatelským účtem (Změna platební karty, Obchod s odměnami, Odhlášení, ...)

Změna platební karty Obsahuje možnost změny platební karty a zobrazení současně nastavené karty.

Obchod s odměnami Uživatel zde má možnost uplatnit body, které nasbíral při užívání služeb aplikace.

Přehled bodů Zobrazuje seznam transakcí provedených s body do obchodu s odměnami.

Odemknutí/Zamknutí Umožňuje uživateli odemknout nebo zamknout vozidlo.

Při zapnutí aplikace se uživateli zobrazí Mapa vozidel. Mezi funkcionalitami lze přecházet buď pomocí postranního menu nebo pomocí tlačítek v jednotlivých obrazovkách.

Pokud uživatel není přihlášený, má přístup pouze na omezenou podmnožinu funkcí (Přihlášení, Mapa vozidel, Detail vozidla, Informace o vozidlu a Podpora a informace). Ostatní funkce jsou vázány na uživatele.

Nejčastější použití aplikace probíhá následujícím způsobem (předpokládá se provedené přihlášení):

1. výběr automobilu na mapě či v seznamu vozidel (Mapa vozidel/Dostupná vozidla)
2. rezervace vozidla (Detail vozidla)
3. odemykání a zamykání vozidla (Odemknutí/Zamknutí)
4. vrácení vozidla po provedené jízdě (Jízda)

Tato sekvence akcí byla zjištěna jako nejčastější z analytického nástroje Firebase a z konverzací s uživateli. Vzhledem k účelu aplikace jako celku není nijak překvapivá.

2.2.2 Technologie použité v aplikaci

Pro automatizaci sestavování aplikace je použit nástroj Gradle. Jak je zmíněno v [8], Gradle umožňuje použití deklarativního modelování za použití silného a expresivního domain-specific jazyka (jazyk zaměřený na konkrétní problémovou doménu), který je implementován v Groovy místo XML, který je používán u starších často používaných nástrojích Ant a Maven.

Aplikace je napsána v jazyce Java za použití Android frameworku. Je využita možnost kompatibility s Java 8, ovšem některé funkcionality této verze Javy nejsou dostupné kvůli nízké minimální Android SDK verzi.

Pro aplikaci Uniqway, stejně jako pro mnoho dalších aplikací, jsou nejdůležitější technologie tvořící uživatelské rozhraní a technologie pro síťovou komunikaci. Vyplývá to z dynamičnosti obsahu a potřebě jej zobrazit uživateli. V této podsekci jsou zmíněny pouze technologie považovány za nejdůležitější pro vývoj nových funkcionalit.

Technologie pro uživatelské rozhraní:

Android Framework - pohledy Jak je zmíněno v [9], Android framework obsahuje bohatý a rozšiřitelný systém pohledů použitelný pro budování UI aplikace. Zahrnuje tlačítka, textová pole a další.

Android Framework - správce zdrojů Zdroje jsou přídavné soubory a obsah, který se používá v kódu. Obsahují například obrázky, definice rozložení, lokalizované texty, instrukce pro animace a další. Je umožněno také vytvářet alternativní zdroje pro jiná rozlišení či jazyky. Odkazuje se na ně pomocí typu a názvu zdroje, případně pomocí jména balíčku.[10]

Spolu s pohledy jsou zdroje nejdůležitější pro vytváření uživatelského rozhraní a interakci s ním. Pro tuto práci jsou nejdůležitější zdroje typu drawable (obrázky), layout (rozložení uživatelského rozhraní), color (barvy), font (řezy písma) a string (texty).

Android Framework - správce aktivit Aktivity jsou jedny ze základních stavebních kamenů Android aplikací. Slouží jako vstupní bod pro uživatelskou interakci s aplikací a jsou také centrální k navigaci uvnitř aplikace nebo mezi aplikacemi. Správce aktivit řídí životní cyklus aplikací.[9][11]

S aktivitami jsou spjaté objekty `Intent`. Jedná se o objekt zprávy, které je možné využít pro zaslání požadavku akce od jiné komponenty aplikace. Jednou z těchto akcí je zahájení nové aktivity. Aktivita nemusí být pouze lokální, může se jednat i o spouštění externích aktivit z jiných aplikací pro získání výsledku (například fotoaparát).[12]

Butterknife Butterknife je nástroj na provázání pohledů (prvky uživatelského rozhraní) pomocí anotací. Například definice členské proměnné `@BindView(R.id.title) TextView title;` způsobí, že při pozdějším spuštění statické metody `ButterKnife.bind()` vyhledá podle identifikátoru uvedeného v anotaci BindView pohled a spojí ho s členskou proměnnou. Od toho momentu lze s těmito pohledy pracovat.[13]

Google Maps SDK Vývojová sada Google Maps umožňuje přidat do aplikace mapy zakládající se na Google mapách. Stará se o přístup na Google Maps servery a umožňuje měnit vzhled mapy a přidat do nich grafické prvky (polygony, značky, ...).[14]

Aplikace na mapách pomocí této vývojové sady zobrazuje polohy vozidel a zón, kam uživatelé mohou vozidla vracet.

Technologie pro síťovou komunikaci:

HTTPS Podle [15], je HTTP protokol použitý pro přenos dat přes web. Používá server-klient model, kde klient může být například telefon či domácí počítač a ten pošle HTTP požadavek (kupříkladu při přístupu na webovou stránku) na server, který jej zpracuje a vrátí odpověď s HTTP stavovým kódem. V případě webového serveru by se u odpovědi s úspěšným stavovým kódem měla vrátit webová stránka.

Kromě stavového kódu je další vlastností HTTP požadavku jeho metoda, která určuje účel požadavku. Pro aplikaci jsou nejdůležitější metody GET a POST. Dle specifikace, server u přijetí požadavku s metodou GET pošle současný stav cílového prostředku a pokud je metodou požadavek POST, zpracuje obsah požadavku dle typu prostředku.[16]

HTTPS je rozšíření HTTP o bezpečnostní vrstvu. Posílaná data jsou zašifrována, díky čemuž útočník, který by zachytil přenos, není schopen tyto data jednoduše přečíst.[17]

V aplikaci je HTTPS využito pro přenos dat na server a ze serveru. Bezpečnost je důležitým aspektem, jelikož se přenáší citlivá uživatelská data a data patřící Uniqway, která je možno využít pro konkurenční boj.

V [16] je zmíněno, že HTTP stavový kód je celé číslo skládající se ze 3 číslic. Podle prvního čísla tohoto kódu je určena třída kódu. Kódy začínající číslicí 2 značí úspěch požadavku. Aplikace dle této specifikace považuje všechny odpovědi nezačínající číslicí 2 za neúspěšné a zbytek za úspěšný, což určuje způsob zpracování.

JSON JavaScript Object Notation (JavaScriptová objektová notace) je textový formát na serializaci strukturovaných dat. Je odvozený z objektových literálů jazyka JavaScript. V zdrojovém kódu 2.1 lze nalézt příklad tohoto formátu. [1]

V aplikaci je JSON používaný především pro komunikaci se serverem jakožto tělo požadavků a odpovědí. O konverzi JSON textu na Java objekty a JSON textu na Java objekty se u komunikace s API stará knihovna Retrofit (popsána níže). Dalším z využití JSON formátu je ukládání komplexních datových typů pro zachování dat.

```
1  {
2      "Image": {
3          "Width": 800,
4          "Height": 600,
5          "Title": "View from 15th Floor",
6          "Thumbnail": {
7              "Url": "http://www.example.com/image/481989943",
8              "Height": 125,
9              "Width": 100
10         },
11         "Animated" : false,
12         "IDs": [116, 943, 234, 38793]
13     }
14 }
```

Zdrojový kód 2.1: Příklad objektu v JSON formátu[1]

JWT JSON Web Token je standard definující kompaktní a soběstačný způsob bezpečného přenosu informací mezi stranami v podobě JSON objektu.

Tato informace může být ověřena a je důvěryhodná díky digitálnímu popisu. Token je zakódován v Base64-URL formátu, což zjednodušuje jeho přenos v HTTP prostředcích, typicky jako hlavička „Authorization“ za použití „Bearer“ schématu. Nejběžnější použití je v autorizaci. Uživatel se přihlásí do aplikace a obdrží token. Tento token uživatel přibalí k následným požadavkům a díky němu má přístup k službám a zdrojům, které jsou s tímto tokenem přístupné.[18]

To je také způsob jaký je použitý v aplikaci. U přihlášení je obdržen token, který je od té doby do odhlášení uložený trvalým způsobem (přetrvá i po uzavření aplikace) a přikládán k požadavkům na server. Pracuje se s ním pouze jako s řetězcem, žádné dekódování nebo verifikace na straně Android aplikace neprobíhá.

Retrofit Retrofit je typově bezpečný HTTP klient pro Android a Javu. Dokáže vygenerovat implementaci rozhraní, ve kterém vývojář nadefinoval specifika (metoda, část url, typ odpovědi, ...). Také dokáže deserializovat a konvertovat tělo odpovědi na Java objekty a naopak. Lze přidat podporu pro velké množství formátů. [19]

Aplikace tuto knihovnu využívá pro komunikaci a výměnu informací se serverem. Vzhledem k povaze aplikace je tato komunikace velice důležitá, neobešla by se bez ní většina funkcionalit. Pro vytváření těl požadavků a zpracování těl odpovědí využívá konvertoru pracující s formátem JSON.

Další technologie:

Google Analytics Google Analytics je součástí platformy Firebase a pomáhá porozumět chování uživatelů, díky čemuž lze optimalizovat výkon a činit informovanější marketingová rozhodnutí. Umožňuje zaznamenávání uživatelských akcí, jako jsou například vstup na obrazovku a přihlášení. Data sbíraná pomocí tohoto nástroje lze sledovat na webovém rozhraní Firebase.[20]

Espresso Jak se píše v [21], Espresso je framework určený pro testování uživatelského rozhraní. Účelem je ulehčit psaní spolehlivých UI testů.

Obsahuje 3 základní komponenty:

ViewMatchers Umožňuje nalézt pohled v současné hierarchii pohledů.

ViewActions Umožňuje provést akce nad pohledy (např. kliknutí).

ViewAssertions Umožňuje ověřit stav pohledů (jestli je zobrazen, barva pozadí, ...)

2.3 Analýza podobných funkcionalit v jiných aplikacích

Kapitola obsahuje analýzu stejných či podobných funkcionalit (tankování, péče o vozidlo, historie jízd).

2.3.1 Tankování

Nejoblíbenější carsharingy v Praze (Car4way, Anytime, ...) nemají dedikovanou obrazovku pro tankování. Některé mají komponentu zobrazující PIN karty, se kterou uživatelé palivo platí, nebo tento PIN sdělují uživateli jinak (například v SMS zprávě).

Existují aplikace pro správu vozového parku (Drivvo, Moje garáž), které obsahují funkci Tankování, ovšem jejich účel je rozdílný. Slouží pro zaznamenání tankování pro přehled majitele (např. kvůli odhadu spotřeby či nákladů) a ne pro kontrolu uživatele. Jsou tedy pro tuto práci irelevantní.

2.3.2 Historie jízd

Byly zanalyzovány funkce pro zobrazení historie jízd v 2 aplikacích: Car4way a Anytime. Jedná se o nejpopulárnější carsharing služby v Praze.

2.3.2.1 Car4way

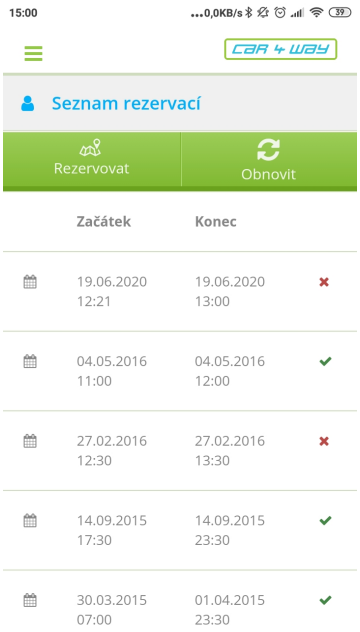
Na obrázku 2.1 je vidět současný stav historie jízd (rezervací) v aplikaci Car4way. Je zde zobrazen pouze začátek a konec rezervace a její stav (úspěšně dokončena, stornována, ...) značený malým obrázkem. Obsahuje také tlačítko pro obnovení seznamu a tlačítko pro navigaci na vytvoření nové rezervace. Po posunu na konec současně načteného seznamu se zobrazí na spodku obrazovky tlačítko pro načtení další části seznamu (pokud taková část existuje).

Po kliknutí na položku v seznamu se otevře detail rezervace, který lze vidět v obrázku 2.2. Je zde zobrazen znovu začátek a konec rezervace a přesnější popis stavu. Dále obsahuje automobil, na který byla rezervace vytvořena a další informace. Lze také přidat poznámku k rezervaci.

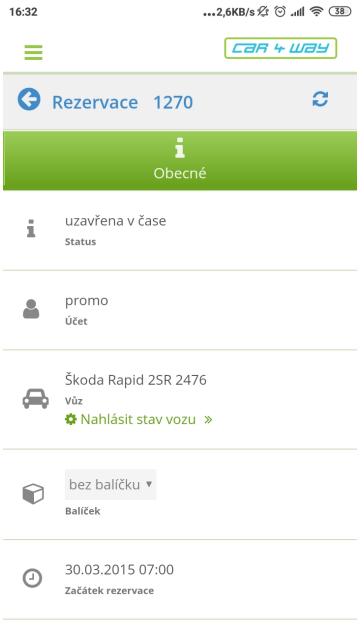
2.3.2.2 Anytime

Jak lze vypozorovat na obrázku 2.3, historie jízd v Anytime Carsharing CZ aplikaci obsahuje u jednotlivých jízd začátek a konec jízdy, poznávací značku vypůjčeného auta, cenu za jízdu a také vypočtenou dobu půjčení.

Po kliknutí na jízdu se otevře její detail (k nahlédnutí na obrázku 2.4). Tento detail obsahuje kromě počtu ujetých kilometrů model půjčeného vozidla a dobu rezervace ve větším detailu. Dále vyjma informací již obsažených v seznamu zahrnuje podrobnosti o časových událostech v jízdě a o platbě.

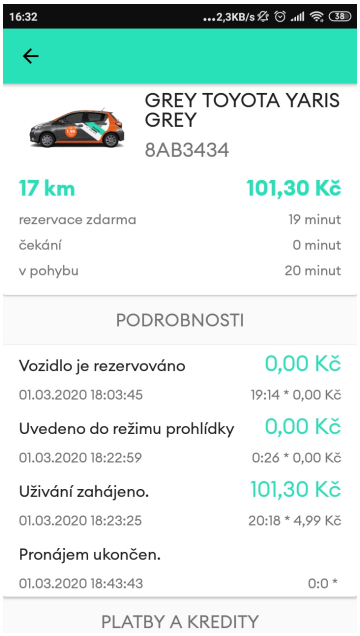


Obrázek 2.1: Historie jízd Car4way[22]



Obrázek 2.2: Detail jízdy Car4way[22]

Obrázek 2.3: Historie jízd Anytime[23]



Obrázek 2.4: Detail jízdy Anytime[23]

2.3.3 Péče o vozidlo

Nejpoužívanější carsharingy v Praze (Car4way, Anytime, ...) neřeší ve své aplikaci danou problematiku. Žádné další aplikace s podobnou funkcionalitou neexistují.

2.4 Požadavky

Tato sekce obsahuje popis požadavků specifikovaných po konzultaci s Uniqway týmem. Je rozdělená na podsekce obsahující funkční (značené F) a nefunkční (značené N) požadavky jednotlivých funkcionalit. Také jsou nejprve vypsány požadavky společné pro všechny funkcionality. Požadavky vyplývají z konzultace s týmem, grafického návrhu od týmových designerů a/nebo z procesů vytvořených ČZU týmem.

2.4.1 Společné požadavky

V této podsekci jsou uvedeny požadavky společné pro všechny funkcionality.

N1 - Lokalizace Funkcionality budou podporovat stejné jazyky jako zbytek aplikace (čeština a angličtina).

N2 - Verze Android Aplikace bude i po rozšíření podporovat operační verzi Android 5.0 a vyšší.

N3 - Jazyk Java Funkcionality budou napsány v jazyce Java z důvodu zpětné kompatibility a široké populárnosti obzvláště na akademické půdě.

2.4.2 Tankování

V této podsekci jsou uvedeny požadavky na funkcionalitu Tankování.

F1 - Zobrazení PIN a manuálu Aplikace zobrazí uživateli PIN k tankovací kartě a jak s ním nakládat.

F2 - Proces tankování Aplikace provede uživatele nahráním fotografie účtenky a vyplněním času tankování. Po dokončení zadávání zpřístupní dokončení tankovacího procesu, kdy se účtenka a čas odešlou na server.

F3 - Zmenšení fotografie před odesláním Aplikace před odesláním fotografie účtenky zmenší velikost, kterou zabírá na disku.

F4 - Zakódování fotografie Aplikace zakóduje fotografii před odesláním na řetězec znaků pomocí algoritmu Base64.

2.4.3 Historie jízd

V této podsekci jsou uvedeny požadavky na funkcionalitu Historie jízd.

F1 - Seznam jízd Aplikace zobrazí uživateli jeho uskutečněné jízdy a informace k nim (čas vzniku a zániku, cena, ...).

F2 - Stránkování Aplikace bude získávat jízdy po částech (stránkách) a bude tyto části postupně načítat.

2.4.4 Péče o vozidlo

V této podsekci jsou uvedeny požadavky na funkcionalitu Péče o vozidlo.

F1 - Proces úklidu Aplikace provede uživatele nahráním fotografií vozidla před a po úklidu. Po nahrání alespoň jedné fotografií z každé kategorie se uživateli zpřístupní dokončení procesu úklidu, kdy se fotografie odešlou na server.

F2 - Zmenšení fotografií před odesláním Aplikace před odesláním fotografií zmenší velikost, kterou zabírají na disku.

F3 - Zakódování fotografií Aplikace zakóduje fotografie před odesláním na řetězce znaků pomocí algoritmu Base64.

Kapitola 3

Návrh

Tato kapitola obsahuje návrh funkcionalit. Nejprve je řešen návrh vizuální, poté je představen návrh průběhu funkcionalit. Nakonec je popsán návrh modelů pro komunikaci se serverem.

3.1 Návrh uživatelského rozhraní

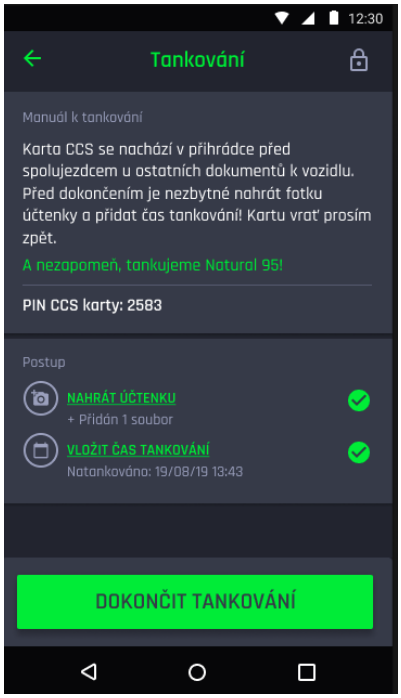
Vizuální návrhy pro aplikaci jsou tvořeny Uniqway design týmem v nástroji Figma. Zpravidla jsou konzultovány s vývojáři. Tato sekce se věnuje představením a popisu těchto návrhů, případně popisu jejich alternativy. Také je popsána interakce s uživatelským rozhraním. Návrhy jsou tvořeny způsobem vedoucím k celistvému stylu (barvy, písma, ...) v aplikaci.

3.1.1 Tankování

Vizuální návrh od design týmu je možno vidět na obrázku 3.1. Obsah se skládá ze 2 hlavních sekcí: Manuál k tankování a Postup. Manuál k tankování obsahuje popis, jak nakládat s tankovací kartou (včetně PINu) a jak dokončit Tankování.

V části obrazovky Postup se vyskytují nejprve pouze 2 tlačítka, jedno pro nahrání účtenky a druhé pro vložení času tankování. Na obrázku 3.1 je vidět stav obrazovky po provedení obou akcí, kdy se objeví tlačítko „Dokončit tankování“. Při provedení každé z akcí se uživateli také zobrazí text a ikona značící úspěch.

Obrazovka také obsahuje v horní části panel nástrojů. Obsahuje tlačítko zpět, nadpis obrazovky a tlačítko „zámek“. Tlačítko zpět ukončí Tankování a navrátí uživatele zpět na obrazovku odkud přišel. Tlačítko „zámek“ slouží pro přesměrování na odemykání či zamykání vozidla (podle stavu uzamčení). Podle stavu uzamčení je také zámek zobrazený jako otevřený (pro odemknutí) nebo uzavřený (pro uzamčení).



Obrázek 3.1: Vizuální návrh Tankování

3.1.1.1 Výběr data a času

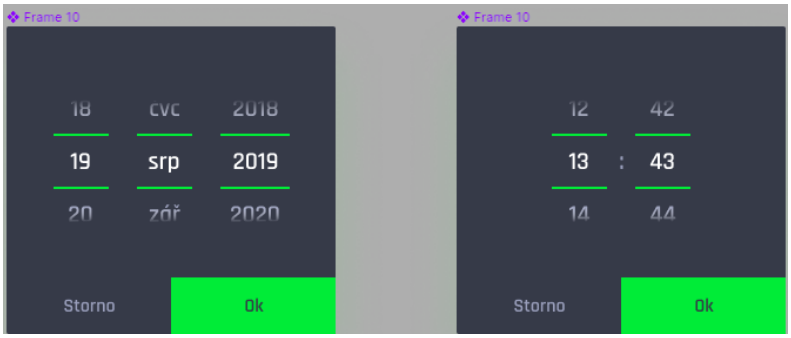
Pro výběr data a času tankování byly design týmem navrhnuty 2 dialogy (obrázek 3.2). Jeden měl sloužit pro výběr data tankování a druhý pro výběr času tankování. Po zvolení data by se otevřel dialog pro zvolení času a výsledek z obou dialogů by byl časem tankování.

Po konzultaci s týmem byl tento návrh označen za špatný a nahrazen zjednodušeným řešením. V rámci této práce byl vytvořen wireframe zjednodušeného dialogu (obrázek 3.3). Uživatel bude vybírat datum a čas ve většině případů stejný nebo následující den po tankování (v případě kdy tankuje před půlnocí). Není proto tedy potřeba mít složité vybírání data.

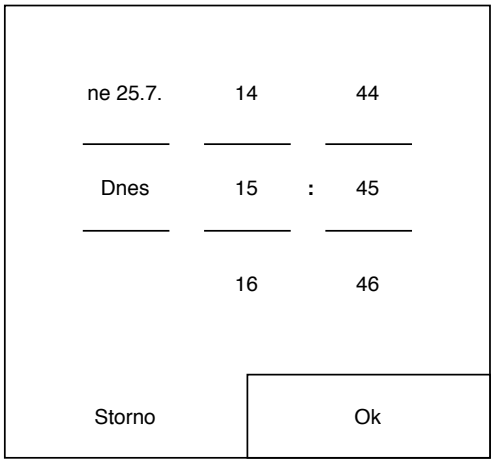
V novém návrhu se vybírá čas stejným způsobem a vedle něj je situován jednoduchý výběr dne, kdy se posouvá pouze o den a je zobrazen pouze den v týdnu, den v měsíci a měsíc. Styl tohoto dialogu bude při implementaci převzat z původního návrhu.

3.1.2 Historie jízd

Jak je vidět na obrázku 3.4, dle návrhu má být seznam rozdělen na časové úseky. Tato funkcionalita byla zrušena a bylo usneseno použití pouze jedno-



Obrázek 3.2: Vizuální návrh výběru data a času



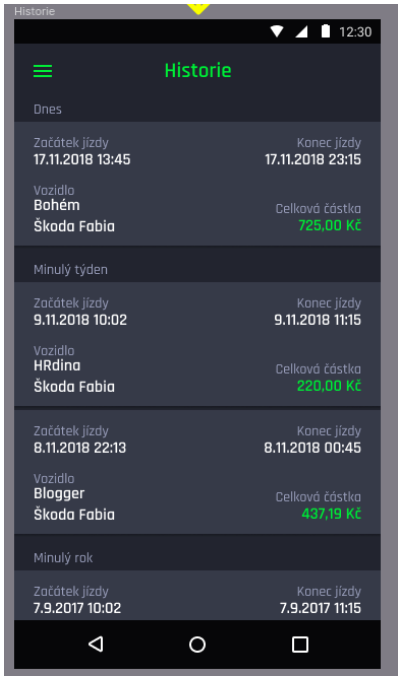
Obrázek 3.3: Wireframe zjednodušeného výběru data a času

duchého seznamu, jaký je uvnitř každé sekce pro časový úsek. Důvodem byla problematika logiky rozdělení a větší priority na jiných úkolech.

Jednotlivé položky v seznamu obsahují tato nadepsaná data o jízdě: Začátek jízdy, Konec jízdy, Vozidlo, Celková částka. Každé vozidlo má své jméno, které je zde také zobrazeno spolu s jeho modelem a značkou.

Kliknutí na položku v seznamu nemá při tomto návrhu žádný účinek. Liší se tím tedy od historií jízd z populárních carsharing aplikací analyzovaných v podsekci 2.3.2. V současné době byla tato funkcionalita odložena pro budoucí vývoj.

V návrhu je také řešen případ, kdy uživatel neabsolvoval žádnou jízdu. Jak je vidět na obrázku 3.5, uživateli se zobrazí hláška o absenci jízd a odkaz na mapu vozidel pro navedení uživatele na provedení jízdy. Na pozadí je zobrazen vodoznak loga Uniqway.



Obrázek 3.4: Vizuální návrh Historie jízd

3.1.3 Péče o vozidlo

Pro Péči o vozidlo nebyl vytvořen finální návrh kvůli nízké prioritě. Na základě několika existujících návrhů a vlastního uvážení byl vytvořen wireframe této funkcionality (obrázek 3.6). Stejně jako Tankování, obsahuje 2 sekce: Manuál (tentokrát k péči o vozidlo) a Postup. V manuálu je uživateli popsáno, jakým způsobem má odeslat fotografie pro získání bodů.

V sekci Popis jsou 2 tlačítka pro nahrání fotografií vozidla (před a po úklidu). Po nahrání fotografie se přidá její náhled pod příslušné tlačítko. Každý náhled obsahuje křížek v pravém horním rohu pro odebrání. Až uživatel nahraje alespoň 1 fotografii každého druhu (před a po), zobrazí se mu tlačítko pro dokončení. Pokud odstraní fotografie jednoho druhu, tlačítko se znovu schová.

3.2 Návrh průběhu funkcionalit

Tato sekce zahrnuje návrh průběhu funkcionalit, který byl vytvořen na základě konzultace s Uniqway týmem, grafického návrhu od týmových designerů a/nebo procesů vytvořených ČZU týmem.

Obrázek 3.5: Vizuální návrh Historie jízd pro uživatele s nulovým počtem jízd

3.2.1 Tankování

Tato podsekce se zabývá průběhem funkcionality Tankování v rámci celého systému, tedy i mimo aplikaci. Tento průběh je znázorněn na UML diagramu aktivit v obrázku 3.7. Figurují v něm 3 aktéři:

Uživatel uživatel, který provádí tankování

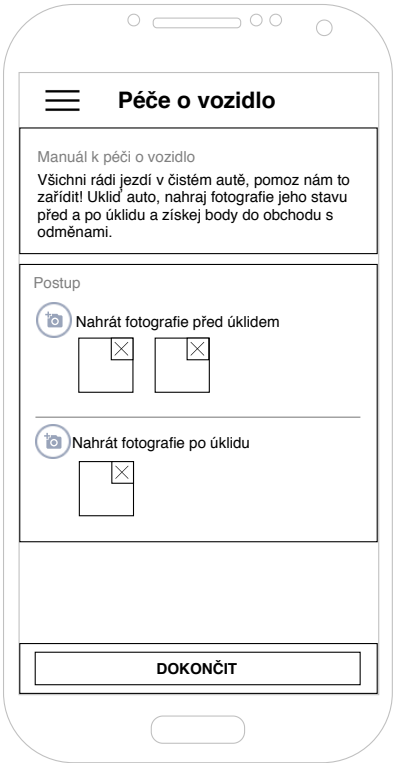
Server služba zpracovávající požadavky z aplikace

Kontrolor člověk či stroj kontrolující uskutečnění tankování

Proces začíná v momentě, kdy Uživatel otevře Tankování v aplikaci. Uživatel může kdykoliv před odesláním požadavku o dokončení tankování tento proces přerušit zavřením Tankování. Nejprve musí uživatel nahrát fotografii účtenky a čas tankování. Poté dá uživatel pokyn pro dokončení tankování, čímž předá fotografie a čas aktérovi Server.

Server provede kontrolu správnosti dat a stavu rezervace. Pokud byla neúspěšná, dá se příčina vědět Uživateli, který má možnost problém napravit a znovu provést dokončení tankování. Pokud kontrola proběhla úspěšně, předá Server fotografii a čas aktérovi Kontrolor.

Kontrolor nejprve čeká na export dat z tankovacích karet. Po obdržení těchto dat se pokusí o spárování času tankování, záznamu z tankovacích karet



Obrázek 3.6: Vizuální návrh Péče o vozidlo

a záznamu o změně v nádrži vozidla. Pokud toto spárování proběhlo úspěšně, uživateli jsou připočteny body do obchodu s odměnami. Pokud spárování selhalo, je tento případ předán k individuálnímu řešení.

3.2.2 Historie jízd

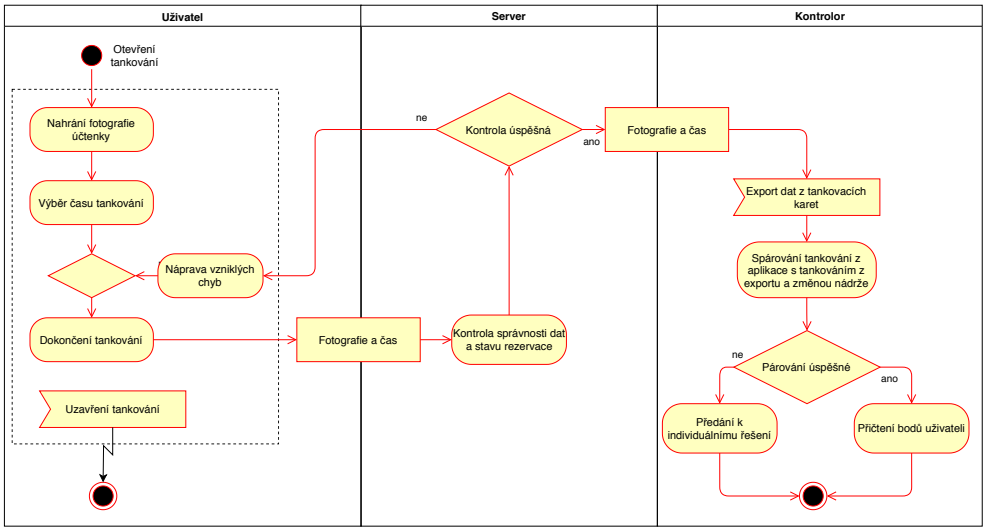
Průběh funkcionality Historie jízd je velice přímočarý. Uživatel otevře Historii jízd v aplikaci, která mu nejnovější část historie jeho jízd získá ze serveru a zobrazí. Když uživatel dosáhne konce současného seznamu jízd, získá a zobrazí se další část uskutečněných jízd (pokud existuje).

3.2.3 Péče o vozidlo

Tato podsekce se zabývá průběhem funkcionality Péče o vozidlo v rámci celého systému, tedy i mimo aplikaci. Tento průběh je znázorněn na UML diagramu aktivit v obrázku 3.8. Figurují v něm 3 aktéři:

Uživatel uživatel, který provádí mytí

Server služba zpracovávající požadavky z aplikace



Obrázek 3.7: Průběh procesu Tankování

Kontrolor člověk či stroj kontrolující uskutečnění mytí

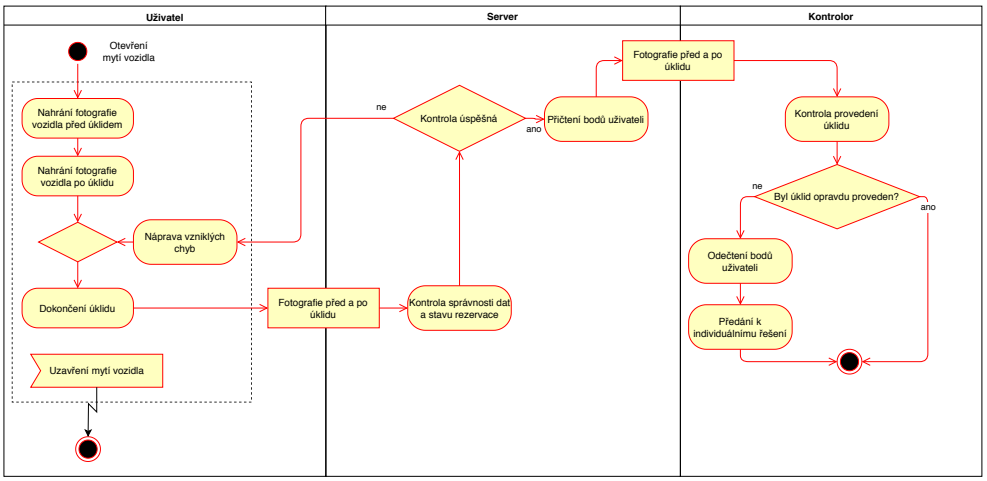
Proces začíná v momentě, kdy Uživatel otevře Péči o vozidlo v aplikaci. Uživatel může kdykoliv před odesláním požadavku o dokončení úklidu tento proces přerušit zavřením Péče o vozidlo. Nejprve musí Uživatel nahrát fotografie vozidla před a po úklidu. Poté dá pokyn pro dokončení mytí. Fotografie se tímto předají aktérovi Server.

Server provede kontrolu správnosti dat a stavu rezervace. Pokud byla neúspěšná, dá se tato skutečnost vědět Uživateli, který může tuto chybu napravit a znovu vykonat dokončení mytí. Pokud byla kontrola úspěšná, Server přičte uživateli se body do obchodu s odměnami a předá fotografie aktérovi Kontrolor.

Kontrolor provede kontrolu, zda úklid opravdu proběhl. Pokud ne, jsou uživateli odečteny body a je tento případ předán k individuálnímu řešení. V opačném případě uživateli body zůstanou.

3.3 Návrh modelových tříd pro komunikaci se serverem

Pro komunikaci se serverem je třeba nadefinovat těla požadavků a odpovědí, pro což slouží modelové třídy. Byly vytvořeny po konzultaci s Uniqway backend týmem.



Obrázek 3.8: Průběh procesu Péče o vozidlo

3.3.0.1 Tankování

Tato podsekce popisuje modelové třídy potřebné pro volání endpointu pro dokončení Tankování. Parametry těla požadavku:

fuelCard identifikační číslo tankovací karty, která se nachází v autě, do kterého se tankovalo

billPhoto fotografie účtenky zakódovaná v Base64

billTimestamp čas tankování zadaný uživatelem jakožto řetězec (rok-měsíc-den hodina:minuta:sekunda)

Parametry těla odpovědi:

id identifikační číslo vytvořeného záznamu tankování

createdAt čas vytvoření záznamu

fuelCard objekt s informacemi o použité tankovací kartě

originalTankVolume původní objem nádrže

finalTankVolume objem nádrže po natankování

3.3.0.2 Historie jízd

Tato podsekce popisuje modelové třídy potřebné pro volání endpointu pro získání stránky seznamu jízd uživatele. Tělo požadavku nebude obsahovat žádné parametry. Je třeba odeslat 2 informace: číslo strany a počet záznamů na

stranu. Dle zvyklostí je vhodnější tyto informace předat v URL požadavku. Parametry těla odpovědi:

data seznam položek v seznamu

Parametry položky v seznamu:

id identifikační číslo záznamu

car objekt obsahující indormace o autě, ve kterém byla jízda provedena (jméno, značka, ...)

price cena, kterou uživatel zaplatil za jízdu

createdAt datum a čas vytvoření rezervace

finishedAt datum a čas ukončení rezervace

meta objekt, obsahující metadata k stránkování:

hasNextPage logická hodnota značící zda existuje následná stránka

totalPageCount celkový počet stran

totalCount celkový počet jízd uživatele

3.3.0.3 Péče o vozidlo

Tato podsekce popisuje modelové třídy potřebné pro volání endpointu pro dokončení úklidu. Tělo odpovědi nebude obsahovat žádné parametry. Jediná potřebná informace z odpovědi je její úspěšnost.

Parametry těla požadavku:

photosBefore seznam fotografií před úklidem v Base64 formátu

photosAfter seznam fotografií po úklidu v Base64 formátu

Kapitola 4

Implementace

Do aplikace byly implementovány všechny požadované funkcionality. Tato kapitola se zabývá způsobem této implementace, především použité technologie. Také je popsán způsob testování, správy verzí, nasazení a dokumentace.

4.1 Použité technologie

V této sekci jsou popsány technologie použité pro implementaci a jejich použití.

4.1.1 Návrh rozložení pohledů

Pro návrh rozložení pohledů jsou v Android aplikacích použity tzv. layout zdroje (podsekce 2.2.2 Android Framework - správce zdrojů). Jedná se o XML soubory ve kterých je specifikována hierarchie jednotlivých prvků uživatelského rozhraní. Tato hierarchie se skládá z pohledů (potomků třídy View). Jsou skupovány do skupin (např. LinearLayout, ConstraintLayout), přičemž tyto skupiny jsou také považovány za pohledy. Tyto skupiny slouží především pro správné zarovnání jejích potomků, také jsou použity pro nastavení pozadí apod. Lze také vytvářet vlastní pohledové třídy.

Příklad tohoto návrhu lze vidět v zdrojovém kódu 4.1. Jsou zde 2 textová pole (TextView) a jsou seskupeny do stejné skupiny (LinearLayout). Textová pole zde mají nastavený text, řez písma, velikost a barvu textu. Také spolu s jejich nadřazenou skupinou definují jejich pozici (padding, margin, ...). K povšimnutí je zde rovněž nastavení textu zobrazeného v textovém poli. Je možné zde mít přímo vypsaný text nebo (jako v tomto případě) lze využít referenci na textové (string) zdroje. Výhodou využití referencí je lokalizace (popsána v podsekci 4.1.3).

Android Studio (vývojové prostředí) umožňuje vykreslovat tyto pohledy po každé změně, což značně zjednodušuje práci pro vývojáře. Také lze tyto návrhy

tvořit a upravovat pomocí grafického rozhraní, které generuje a upravuje XML kód.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="@color/primary_dark"
    android:paddingLeft="19dp"
    android:paddingRight="19dp"
    android:paddingTop="15dp"
    android:paddingBottom="15dp"
    android:layout_marginBottom="4dp"
    android:elevation="2dp"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="@font/rajdhani_medium"
        android:text="@string/car_care_manual_header"
        android:textColor="@color/text_gray"
        android:textSize="14sp"
        android:layout_marginBottom="10dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/rajdhani_medium"
        android:text="@string/car_care_manual_body"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:layout_marginBottom="10dp"/>
</LinearLayout>
```

Zdrojový kód 4.1: Příklad návrhu rozložení pohledů

4.1.2 Provázání pohledů

Pro provázání pohledů byl u funkcionalit Tankování a Historie jízd použita technologie Butterknife (popsán v podsekci 2.2.2). Nevýhodou této technologie je fakt, že se chyba provázání projeví až po spuštění dané obrazovky. Také je potřeba pro každý prvek uživatelského rozhraní, se kterým chceme pracovat, vytvořit členskou proměnnou.

Tyto problémy řeší technologie ViewBinding. V momentě, kdy je v projektu zapnutý ViewBinding, vytvoří se pro každý soubor s návrhem vazební třída. Tato třída (přesněji její implementace generovaná při kompilaci) obsahuje všechny pohledy v návrhu s definovaným identifikátorem jako veřejně přístupné členské proměnné. Také obsahuje statickou metodu `inflate()`, která vytvoří instanci této třídy s navázanými pohledy. Využití této metody je vidět v zdrojovém kódu 4.2.

```
1 ActivityCarCareBinding binding = ActivityCarCareBinding.inflate(
      getLayoutInflater());
```

Zdrojový kód 4.2: Vytvoření binding objektu pro péči o vozidlo

K povšimnutí je název vygenerované třídy. Jména pohledů a souborů s návrhem jsou dle zvyklostí pojmenovávány ve stylu „Snake case“, kde jsou slova psány malými písmeny oddělovány podtržítkem (např. activity_car_care). ViewBinding tyto názvy konvertuje do stylu „Camel case“, kde slova nejsou oddělena, ale každé první písmeno následného slova začíná velkým písmenem a ostatní písmena jsou malá s možnou vyjímkou prvního písmena výrazu (např. ActivityCarCare, totalPriceTitle). Názvy vygenerovaných vazebních tříd začínají dle zvyklostí velkým písmenem a názvy proměnných v těchto třídách začínají malým písmenem.

Technologie ViewBinding byla využita pouze pro Péči o vozidlo. V době implementace Tankování a Historie ji nebylo možné použít, jelikož nebyla vydána. Pro Tankování a Historii jízd byla použita technologie Butterknife.

4.1.3 Lokalizace

Lokalizace v implementovaných funkcionalitách je řešena pomocí Android zdrojů. Texty jsou vypsány pro každý jazyk v souboru strings.xml v příslušné složce (values pro výchozí hodnoty, values-cs pro hodnoty v českém jazyce). Při referenci na texty v těchto souborech (např. v layout souboru jak je ukázáno v podsekci 4.1.1) se automaticky v aplikaci vybere správný jazyk podle nastaveného systémového jazyka na telefonu uživatele.

Pro jednodušší správu textů v aplikacích (Android, iOS, web) jsou použity Google Sheets a plugin Spreadsheet Localizer od Ackee, který tyto texty automaticky importuje do správných souborů.

4.1.4 Výsledky aktivit

Aktivity (popsány v podsekci 2.2.2 Android Framework - správce aktivit) lze spustit s očekáváním výsledku. Tato funkcionalita je využívána pro funkcionality Tankování a Péče o vozidla (pro každou z těchto funkcionalit byla vytvořena aktivita). Zdrojový kód 4.3 obsahuje statickou metodu pomocné třídy ActivityStarter. Účel této třídy je oddělení logiky pro spouštění jednotlivých aktivit mimo jiné pro zamezení duplicity.

Jak je vidět v ukázkovém kódu, spuštění aktivity s očekávaným výsledkem se provádí zavoláním metody `startActivityForResult()` na současnou aktivitu. Do této metody se předá objekt `Intent` obsahující informaci o aktivitě, která se má otevřít, a celé číslo identifikující požadavek pro pozdější zpracování.

```
1 public static void startRefuelingActivity(Activity actualActivity) {
2     Intent intent = new Intent(actualActivity, RefuelingActivity.
          class);
3     actualActivity.startActivityForResult(intent, RequestCode.
          REFUELING);
4 }
```

Zdrojový kód 4.3: Start aktivity s očekávaným výsledkem

Zpracování probíhá ve aktivitě, ze které se nová aktivita otevírala (actualActivity v zdrojovém kódu 4.3) či v současně zobrazeném fragmentu (aktivita v sobě může obsahovat jeden či více fragmentů). Činí se tak v metodě `onActivityResult()`. Tuto metodu zpracovávající výsledek aktivit pro Tankování a Péči o vozidla lze nalézt v zdrojovém kódu 4.4. Větví se dle identifikačního čísla požadavku specifikovaný při spouštění aktivity a kontroluje se zde, zda byl požadavek úspěšný dle předaných parametrů. Pokud Tankování nebo Péče o vozidla (jejich aktivity), byly ukončeny s úspěšným výsledkem, zobrazí se uživateli zpráva o úspěchu pomocí pomocné třídy `SnackbarMaker`.

```
1 protected void onActivityResult(int requestCode, int resultCode,
      Intent data) {
2     super.onActivityResult(requestCode, resultCode, data);
3     if (requestCode == ActivityStarter.RequestCode.REFUELING &&
          resultCode == RESULT_OK) {
4         SnackbarMaker.refuelingSuccessTSnackbar(getWindow().
              getDecorView().getRootView());
5     } else if (requestCode == ActivityStarter.RequestCode.CAR_CARE
          && resultCode == RESULT_OK) {
6         SnackbarMaker.cleaningFinishSuccessTSnackbar(getWindow().
              getDecorView().getRootView());
7     }
8 }
```

Zdrojový kód 4.4: Metoda onActivityResult v třídě MainActivity

Výsledek aktivity je nastavován v metodách dané aktivity. Pro nastavení je použita metoda `setResult()`, do které se předá celočíselná konstanta značící výsledek (např. konstanta RESULT_OK použita v zdrojovém kódu 4.4), případně přídavná data. U Tankování a Péče o vozidlo je při vytvoření aktivity nastaven touto metodou výchozí stav výsledku na RESULT_CANCELED (zrušená operace), při úspěchu dokončení je nastaven na RESULT_OK (úspěch) a aktivita je ukončena.

4.1.5 Získání obrázků

Ve funkcionalitách Tankování a Péče o vozidlo je třeba získat fotografie (účtenky, vozidla, ...). Lze tento problém řešit externě či interně. Interní způsob by zahrnoval implementaci fotoaparátu a/nebo výběru obrázků ze souborů uvnitř

aplikace. U externího způsobu jsou používány cizí aplikace typu fotoaparát a galerie. Vzhledem k složitosti implementace interního řešení a předpokládané malé četnosti používání této funkcionality (oproti jiným funkcionalitám v aplikaci) byl zvolen přístup externí.

Externí přístup funguje na principu aktivit (v tomto případě externích) a jejich výsledků, obdobně jsou použity pro interní aktivity, jak bylo ukázáno v podsekci 4.1.4. V zdrojovém kódu 4.5 lze vidět spouštění aktivity pro získání obrázku z galerie. Typ aktivity je předán do objektu `Intent` pomocí definované konstanty pro tento typ akce. Také lze pomocí objektu `Intent` definovat například soubor kam uložit data, která jsou výsledkem aktivity. Jeden ze způsobů je vidět v zmíněném ukázkovém kódu.

```
1 public static void startImagePickExternalActivity(Activity
      actualActivity, int requestCode) {
2     Intent intent = new Intent(Intent.ACTION_PICK, android.provider.
          MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
3     actualActivity.startActivityForResult(intent, requestCode);
4 }
```

Zdrojový kód 4.5: Metoda pro spouštění aktivity pro získání obrázku z galerie

Stejně jako u interních aktivit, výsledek externích aktivit je zpracován v metodě `onActivityResult()`. Jak je vidět v zdrojovém kódu 4.4, do metody je předán objekt typu `Intent`, který může obsahovat polohu souboru obrázku z čeho lze tento obrázek získat. V zdrojovém kódu 4.6 je příklad zpracování tohoto obrázku a vytvoření `Bitmap` objektu (proměnná data je objekt typu `Intent`) předaný do `onActivityResult()`).

```
1 final Uri imageUri = data.getData();
2 final InputStream imageStream = getContentResolver().openInputStream
      (imageUri);
3 final Bitmap selectedImage = BitmapFactory.decodeStream(imageStream)
      ;
```

Zdrojový kód 4.6: Zpracování obrázku z galerie vybraný uživatelem

4.1.6 Seznamy a stránkování

Tato podsekce se zabývá implementací seznamů a stránkování v funkcionalitě Historie jízd. Také se zabývá řešením problematiky nulového počtu položek v seznamu.

4.1.6.1 Seznamy

Pro zobrazování seznamu položek je použit pohled zvaný RecyclerView (v tomto případě jeho členy týmu rozšířená verze RecyclerViewWithEmptyState). Pro použití RecyclerView je třeba vytvořit adaptér (potomek

třídy RecyclerView.Adapter, u historie jízd RideHistoryItemAdapter). Ten se spolu s třídou, která je potomkem třídy RecyclerView.ViewHolder (RideHistoryItemAdapter.HistoryItemViewHolder), stará o vytváření pohledů pro každou položku v seznamu a nastavování hodnot do vytvořených pohledů.

Konstruktor a metody implementované v třídě RideHistoryItemAdapter lze vidět v zdrojovém kódu 4.7. Konstruktor přijímá seznam položek v seznamu (jejich datovou reprezentaci) a předává jej do konstruktoru nadtřídy ListItemAdapter, která byla vytvořena pro zjednodušení práce s těmito seznamy. Metoda `onCreateViewHolder()` řeší vytvoření pohledu a objekt podtřídy RecyclerView.ViewHolder pro položku v seznamu (v tomto momentě ne pro specifickou položku). Metoda onBindViewHolder řeší propojení vytvořeného objektu s konkrétním objektem, respektive s příslušnými daty (více níže).

```
1  public RideHistoryItemAdapter(List<ReservationHistoryItem> items) {
2      super(items);
3  }
4
5  @NonNull
6  @Override
7  public HistoryItemViewHolder onCreateViewHolder(@NonNull ViewGroup
       parent, int viewType) {
8      View view = LayoutInflater.from(parent.getContext())
9              .inflate(R.layout.ride_history_item, parent, false);
10     return new HistoryItemViewHolder(view);
11 }
12
13 @Override
14 public void onBindViewHolder(@NonNull HistoryItemViewHolder holder,
       int position) {
15     Log.d(TAG, "#" + position);
16     holder.bind(position);
17 }
```

Zdrojový kód 4.7: Metody a konstruktor třídy RideHistoryItemAdapter

V zdrojovém kódu 4.8 lze vidět zmiňovanou třídu HistoryItemViewHolder, která je podtřídou RecyclerView.ViewHolder. Drží si všechny potřebné pohledy pro zobrazení potřebných dat a po zavolání metody `bind()` z metody `onBindViewHolder` třídy RideHistoryItemAdapter (v kódu výše) také tato data zobrazí. V ukázce je vynechána většina pohledů a nastavování jejich hodnot pro přehlednost. Tato data jsou získána ze seznamu, který je definován v nadtřídě ListItemAdapter.

```
class HistoryItemViewHolder extends RecyclerView.ViewHolder {

    private View itemView;

    private TextView carName;
    ...

    public HistoryItemViewHolder(View itemView) {
        super(itemView);
        this.itemView = itemView;
        carName = itemView.findViewById(R.id.
            ride_history_item_car_name);
        ...
    }

    void bind(int position) {
        ReservationHistoryItem item = items.get(position);
        carName.setText(item.getCar().getName());
        ...
    }
}
```

Zdrojový kód 4.8: Třída HistoryItemViewHolder (zjednodušená)

4.1.6.2 Problematika nulového počtu položek

V návrhu uživatelského rozhraní byl řešen případ, kdy uživatel neabsolvoval žádnou jízdu (obrázek 3.5). Tato problematika je řešena ve výše zmíněné pohledové třídě RecyclerViewWithEmptyState. Kód 4.9 je proveden při každé změně v seznamu. Pokud je seznam prázdný, schová sama sebe a zobrazí uživateli pohled značící prázdný seznam, jak byl navržen. Jeho vzhled a další specifika jsou nastaveny v zdrojích a pohledových třídách.

```
private void initEmptyView() {
    if (mEmptyView != null) {
        mEmptyView.setVisibility(
                getAdapter() == null || getAdapter().getItemCount()
                    == 0 ? VISIBLE : GONE);
        RecyclerViewWithEmptyState.this.setVisibility(
                getAdapter() == null || getAdapter().getItemCount()
                    == 0 ? GONE : VISIBLE);
    }
}
```

Zdrojový kód 4.9: Metoda initEmptyView v RecyclerViewWithEmptyState

4.1.6.3 Stránkování

Načtení další strany v Historii jízd je řešeno pomocí vytvořené instance třídy RecyclerView.OnScrollListener. V zdrojovém kódu 4.10 je možno nahlédnout

na logiku, která toto načítání obstarává. Provede se při každé změně stavu posunu v seznamu. Pokud nelze provést posun směrem dolů, seznam obsahuje další stranu (zjištěno z metadat posílaných u seznamu, jak byly definované v) a nová strana se současně nenačítá, provede se požadavek o následující stranu seznamu. Zde použitá třídní proměnná loadingPage je instancí třídy AtomicBoolean a zajišťuje pomocí atomických operací řešení pro problémy se souběžným přístupem vláken ke zdrojům.

```
1 if(!recyclerView.canScrollVertically(1) && hasNextPage) {
2     if(!loadingPage.compareAndSet(false, true)) {
3         return;
4     }
5     getHistoryPage(++currentPage);
6 }
```

Zdrojový kód 4.10: Načítání další stránky seznamu

Je nutno podotknout, že tento způsob stránkování v současnosti není optimální. Lepší řešení problému poskytuje knihovna Paging, která je součástí souboru knihoven Android Jetpack. Tento soubor knihoven je přístupný pouze pro projekty, které migrovaly z tzv. Support knihoven na knihovny Androidx, což bylo provedeno až po implementaci Historie jízd.

4.1.7 Komunikace s API

V podsekci 2.2.2 byly představeny technologie pro síťovou komunikaci současně použité v aplikaci: HTTPS, JSON, JWT a Retrofit. Použití HTTPS a JSON je řešeno na pozadí především pomocí knihovny Retrofit. Použití JWT bude popsáno spolu s použitím Retrofit.

Pro vytvoření metody, která vytváří volání API endpointu, je třeba napsat metody, které specifikují typ, na který se má tělo odpovědi namapovat, cestu k danému endpointu, parametry volání apod., a které jsou umístěné v rozhraní, ze kterého se později generuje implementace provádějící volání API endpointů.

Takto definované metody pro tuto práci lze nalézt v zdrojovém kódu 4.11. Jsou umístěné v rozhraní RestApi. Jak je v útržku zdrojového kódu vidět, každá metoda je nadepsána anotací definující HTTP metodu (popsány v podsekci 2.2.2) a statickou část cesty. Každá z metod vrací typ Call z Retrofit knihovny (u tohoto objektu lze například spustit volání synchronně či asynchronně) se specifikovaným generickým typem (např. RideHistoryPage u historie jízd) značící typ úspěšné odpovědi. Všechny metody také mají jako parametr řetězec s JWT tokenem. Předáním do tohoto parametru se přiloží k požadavku jako součást hlavičky, identifikovaný pomocí řetězce „Authorization“.

Historie jízd má v statické cestě definovaný tzv. query parametr (za otazníkem v URL), který zajišťuje správné řazení, tedy od nejnovějšího záznamu.

Kromě JWT tokenu jsou předávány další query parametry, tentokrát dynamické. Značí číslo požadované stránky a počet záznamů ve stránce. V těle úspěšné odpovědi je definován model dle návrhu (podsekce 4.1.6.3). HTTP metoda tohoto volání je nastavena na GET.

Metody pro Tankování a Péči o vozidlo jsou definovány podobně. Mají nastavenou HTTP metodu POST a těla odpovědi a požadavku dle návrhu (podsekce 3.3.0.1 a 3.3.0.3).

```
1  public interface RestApi {
2      ...
3
4      //Historie jizd
5      @GET("client/reservations?sorts=id:desc")
6      Call<RideHistoryPage> getReservations(@Header(Headers.
           AUTHORIZATION) String JWToken, @Query("page") Integer page,
           @Query("pageLimit") Integer pageLimit);
7
8      ...
9
10     //Tankovani
11     @POST("client/refuels")
12     Call<StoreRefuelResponse> createRefuel(@Header(Headers.
           AUTHORIZATION) String JWToken, @Body Refuel refuel);
13
14     //Pece o~vozidlo
15     @POST("client/car-care")
16     Call<ResponseBody> finishCleaning(@Header(Headers.AUTHORIZATION)
            String JWToken, @Body Cleaning cleaning);
17
18     ...
19 }
```

Zdrojový kód 4.11: Definice volání endpointů

Pro vygenerování rozhraní RestApi pomocí Retrofit je třeba vykonat kód 4.12 (tento kód byl zjednodušen). Nejdříve se vytvoří stavební objekt třídy Retrofit. Do něj se poté předají potřebné informace: základní URL, upravený http klient (jeho implementace není důležitá pro tuto práci), konvertor (pro konverzi JSON formátu). Pokud by nebyly specifikovány, byly by použity výchozí stavy. Poté je zavoláním metody build vytvořen objekt třídy Retrofit. Pomocí něj a jeho metody create, do které se předá třída rozhraní, pro kterou se vytvoří implementace.

```
1 Retrofit retrofit = new Retrofit.Builder().baseUrl(url)
2             .client(httpClientBuilder.build())
3             .addConverterFactory(GsonConverterFactory.create(gson))
4             .build();
5 RestApi restApi = retrofit.create(RestApi.class);
```

Zdrojový kód 4.12: Vygenerování implementace RestApi

Je nutno podotknout, že v současně době zavolání cesty pro Péči o vozidlo vyprodukuje vždy chybu 404 (nenalezeno), jelikož není na serveru implementována. Pro vývoj byla odpověď s touto chybou zpracována jako úspěšná, což bude před vydáním uživatelům smazáno. Pro testování byla využita imitace serveru jak je popsáno v podsekci 4.2.1.1.

4.2 Testování

Testování je velice důležitá část tvorby softwaru. Umožňuje kontrolu správné funkčnosti jednotlivých komponent a také jejich zkombinování. Existuje mnoho typů testování. V této práci je řešeno automatické UI testování, tedy testování chování uživatelského rozhraní a UX testování (testování použitelnosti). Unit (jednotkové) testování není zahrnuto, kvůli vysoké závislosti tříd a metod na prvky uživatelského rozhraní a na Android framework, což vysoce ztěžuje vytváření těchto testů.

4.2.1 UI testování

Pro UI testování byl zvolen framework Espresso. Je nejvíce podporovaný a splňuje všechny naše požadavky. Pro každou funkcionalitu byly napsány testy, které provedou akce podle scénáře a zkontrolují, zda tyto akce měly správný efekt (především vizuální). Tyto scénáře zahrnují pozitivní i negativní situace. Tyto testy mohou plní více účelů. Jedním z nich je kontrola správnosti implementace při vývoji. Obzvláště jsou užitečné v případě nefunkčnosti serveru či absence potřebných API cest. Další uplatnění najdou jako regresní testy, které ověřují, zda provedené úpravy neměly negativní vliv na fungující funkcionality.

4.2.1.1 Požadavky na server u testů

Při vytváření testů uživatelského rozhraní v této aplikaci může být komunikace se serverem problém. Nelze zaručit obsah, úspěšnost či další aspekty komunikace. Také není vhodné například vytvářet rezervace na reálných automobilech kvůli testování. Možností by bylo použití testovacího serveru s falešnými auty, který v systému Uniqway existuje, ovšem problematická je změna stavu rezervace, odemykání vozidla apod.

Tento problém je řešen technologií MockWebServer. Lze do něj pomocí instance třídy Dispatcher nastavit odpovědi pro volání API z aplikace. Volání jsou rychlá a odpovědi mají definovaný návratový kód a obsah, díky čemuž je možno testovat, jak reaguje aplikace na úspěch a neúspěch, či jak reaguje na konkrétní přijatá data.

4.2.2 UX testování

Testování použitelnosti bylo pro všechny funkcionality provedeno distribucí aplikace napříč zaměstnanci Uniqway a následného sběru zpětné vazby. Historie jízd je již více než rok přístupné všem uživatelům a Tankování je současně v dlouhodobém beta testování z interních důvodů (je tedy přístupné podmnožině uživatelů). Během doby po zveřejnění těchto funkcionalit proběhly různými způsoby dialogy o zmíněných funkcionalitách.

4.3 Správa verzí a nasazení

Pro správu verzí je využit distribuovaný systém správy verzí Git. Umožňuje spolupráci v týmu nad stejnou množinou zdrojových kódů (repozitář). Jako online úložiště je v projektu využíván a služba GitLab.

Android aplikace jsou nejběžněji distribuovány mezi uživatele pomocí služby Google Play, která je také použita pro aplikaci Uniqway. Lze takto vydat verzi aplikace mezi všechny uživatele, mezi beta testery, alfa testery apod.

Při vytváření vydání se vygeneruje z kódu aplikační balík podepsaný certifikátem kvůli bezpečnosti. Tento balík je poté nahrán do konzole Google Play a spolu s názvem a popisem tvoří nové vydání aplikace. Google Play při vydání využije aplikační balík pro instalaci aplikace jednotlivým uživatelům.

Interní testovací verze (většinou verze komunikující s testovacím serverem) jsou v současné době distribuovány pomocí podepsaných apk souborů distribuovaných pomocí komunikačních nástrojů. Je připravena funkcionalita pro odeslání těchto apk na projektový interní informační web pomocí GitLab služeb pro nepřetržité dodávání, ovšem v současnosti není tento web pro funkcionalitu připraven.

4.4 Dokumentace

Pro vytvořené funkcionality byla vytvořena vývojářská dokumentace ve formátu Markdown. Je obsažena v repozitáři obsahují kód aplikace. Také byla vytvořena uživatelská příručka pro funkcionality Tankování a Péče o vozidlo (lze nalézt v příloze B)

Závěr

Cílem práce bylo navrhnout a implementovat rozšíření Android aplikace Uniqway pro sdílení vozidel o funkce Tankování, Historie jízd, Péče o vozidlo.

Aplikace nyní umožňuje uživateli zobrazit historii jeho jízd, provede ho procesem tankování a péče o vozidlo (mytí, čištění...). Historie jízd je již přes rok v aplikaci přístupná všem uživatelům. Tankování je v současné době umístěno v dlouhodobém beta testování, může ji využít tedy jen podmnožina uživatelů. Péči o vozidlo chybí serverová část, není tedy možné tuto funkcionalitu vydat uživatelům. Ostatní funkcionality byly napojeny na existující API projektu.

Za tankování a péči o vozidla bude v budoucnu uživatel získávat body do obchodu s odměnami. V současně době není implementováno přidání bodů na serverové části.

Pro tyto funkce byly napsány automatické UI testy, které ověřují správné fungování v pozitivních i negativních situacích. Pro vytvoření těchto testů byl použit framework Espresso. Byly také manuálně otestovány členy týmu Uniqway.

Pro dokumentaci byla vytvořena uživatelská příručka provádějící uživatele funkcionalitami Tankování a Péče o vozidlo. Také byla rozšířena existující vývojářská dokumentace pro lepší orientaci v projektu.

Možná rozšíření

Tankování v současné době nepodporuje vozidla s elektrickým pohonem. Vzhledem k faktu, že v nedávné době přibyl takový automobil do Uniqway flotily, by bylo vhodné zobrazit návod na provedení nabíjení a změnit proces této funkcionality pro tato elektrická vozidla.

Možným rozšířením historie jízd je přidání detailu jízdy. Mohl by obsahovat podrobnější informace o jízdě, možnost hodnocení jízdy apod.

Do budoucna by bylo vhodné přidat funkce pro kontrolu vozidel uživateli a zaměstnanci, kteří se o tato vozidla starají. Tato funkcionalita by jim umožnila provádět přímo v aplikaci kontroly, které nyní vykonávají přes Google formulář, a dala by jim přehled o již provedených kontrolách a nahlášených závadách. Tyto závady by mohli nahlašovat také uživatelé, což by zrychlilo jejich nápravu.

Bibliografie

1. BRAY T. Ed., Textuality.
   The JavaScript Object Notation (JSON) Data Interchange Format.
   RFC Editor, 2017. ISSN 2070-1721.
   Dostupné také z: `https://www.rfc-editor.org/info/rfc8259`. RFC.
   RFC Editor.
2. RAVAS, Filip.
   Mobilni aplikace pro uzivatele systemu sdileni automobilu vice uzivateli.
   2017. B.S. thesis.
   České vysoké učení technické v Praze. Vypočetní a informační centrum.
3. PROUZA, Petr. *Rozšíření Android aplikace pro uživatele systému sdílení automobil o reward shop*. 2019. B.S. thesis.
   České vysoké učení technické v Praze. Vypočetní a informační centrum.
4. JIROVSKỲ, V et al. Ověřená technologie: Carsharing Uniqway. 2018.
5. *UNIVERZITNÍ CARSHARING UNIQWAY OSLAVIL PRVNÍ ROK PROVOZU* [online]. 2019 [cit. 2019-12-17].
   Dostupné z: `https://aktualne.cvut.cz/stalo-se/20191024-univerzitni-carsharing-uniqway-oslavil-prvni-rok-provozu`.
6. ČERNỲ, Michal. *iOS aplikace pro car-sharing Uniqway*. 2019.
   B.S. thesis.
   České vysoké učení technické v Praze. Vypočetní a informační centrum.
7. SEVERA, Štěpán.
   Webová aplikace pro uživatele systému sdílení automobil. 2019.
   B.S. thesis.
   České vysoké učení technické v Praze. Vypočetní a informační centrum.
8. MUSCHKO, Benjamin. *Gradle in action*. Manning, 2014.
9. *Platform Architecture* [online].
   https://developer.android.com/guide/platform, 2020 [cit. 2020-07-22].

10. *App resources overview* [online]. https://developer.android.com/guide/topics/resources/providing-resources, 2020 [cit. 2020-07-22].

11. *Activities* [online]. https://developer.android.com/guide/components/activities, 2019 [cit. 2020-07-22].

12. *Intents and Intent Filters* [online]. https://developer.android.com/guide/components/intents-filters, 2019 [cit. 2020-07-22].

13. WHARTON, Jake. *Retrofit: A type-safe HTTP client for Android and Java* [online]. https://jakewharton.github.io/butterknife [cit. 2020-07-20].

14. *Maps SDK for Android: Overview* [online]. https://developers.google.com/maps/documentation/android-sdk/overview, 2020 [cit. 2020-07-20].

15. CHRISTENSSON, P. *HTTP Definition* [online]. https://techterms.com/definition/http: Sharpened Productions, 2015 [cit. 2020-07-20]. Dostupné z: `https://techterms.com`.

16. FIELDING R. Ed. a Reschke J., Ed. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* RFC Editor, 2014. ISSN 2070-1721. Dostupné také z: `https://www.rfc-editor.org/info/rfc7231`. RFC. RFC Editor.

17. CHRISTENSSON, P. *HTTPS Definition* [online]. https://techterms.com/definition/https: Sharpened Productions, 2008 [cit. 2020-07-20]. Dostupné z: `https://techterms.com`.

18. AUTH0.COM. *JSON Web Tokens Introduction.* Dostupné také z: `https://jwt.io/introduction/`.

19. SQUARE INC. *Retrofit: A type-safe HTTP client for Android and Java* [online]. https://square.github.io/retrofit [cit. 2020-07-20].

20. *Google Analytics* [online]. https://firebase.google.com/docs/analytics, 2020 [cit. 2020-07-20].

21. VOGEL, Lars. *Android user interface testing with Espresso - Tutorial* [online]. https://www.vogella.com/tutorials/AndroidTestingEspresso/article.html, 2016 [cit. 2020-07-22].

22. CAR4WAY A.S. *CAR4WAY*. 2019. Verze 0.9.62.
Dostupné také z: `https://play.google.com/store/apps/details?id=com.phonegap.car4way`.

23. ANYTIME CARSHARING CZECH REPUBLIC.
Anytime Carsharing CZ. 2020. Verze 22.337.
Dostupné také z: `https://play.google.com/store/apps/details?id=cz.anytime.mobile.android`.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CCS Czechoslovak Card Services

ČVUT České Vysoké Učení Technické

ČZU Česká Zemědělská Univerzita

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

JSON JavaScript Object Notation

JWT JSON Web Token

PIN Personal Identification Number

SDK Software Development Kit

SMS Short Message Service

UI User Interface

UML Unified Modeling Language

URL Uniform Resource Locator

UX User Experience

VŠE Vysoká Škola Ekonomická

XML Extensible markup language

Příloha B

Uživatelská příručka

Tato příloha obsahuje uživatelskou příručku pro funkcionality Tankování a Péče o vozidlo včetně akcí mimo aplikaci.

B.1 Tankování

Pro natankování Uniqway vozidla proveďte následující kroky:

1. Otevřete si Tankování v aplikaci Uniqway.
2. Natankujte do vozidla palivo zobrazené v aplikaci.
3. Zaplaťte za palivo pomocí CCS karty, kterou naleznete v přihrádce před spolujezdcem. PIN k této kartě naleznete v aplikaci.
4. Pomocí tlačítka v aplikaci nahrajte fotografii účtenku, kterou jste obdržel/a.
5. Pomocí tlačítka v aplikaci vyplňte čas tankování uvedený na účtence.
6. Dokončete kliknutím na tlačítko ve spodní části obrazovky.

B.2 Péče o vozidlo

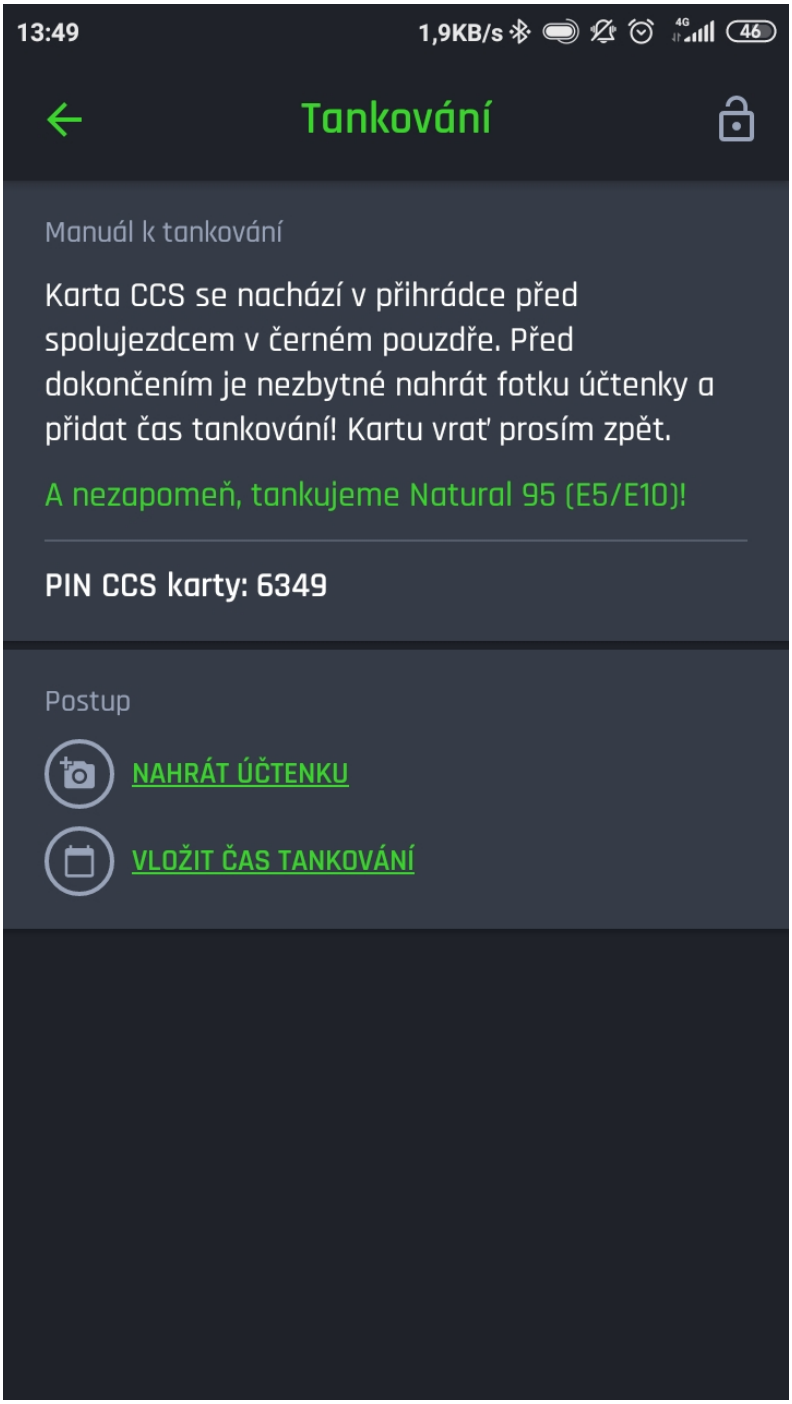
Pro získání bodů za úklid Uniqway vozidla proveďte následující kroky:

1. Otevřete si Péči o vozidla v aplikaci Uniqway.
2. Pomocí tlačítka v aplikaci nahrajte libovolný počet fotografií částí vozidla, která máte v plánu umýt.
3. Umyjte tyto části vozidla.

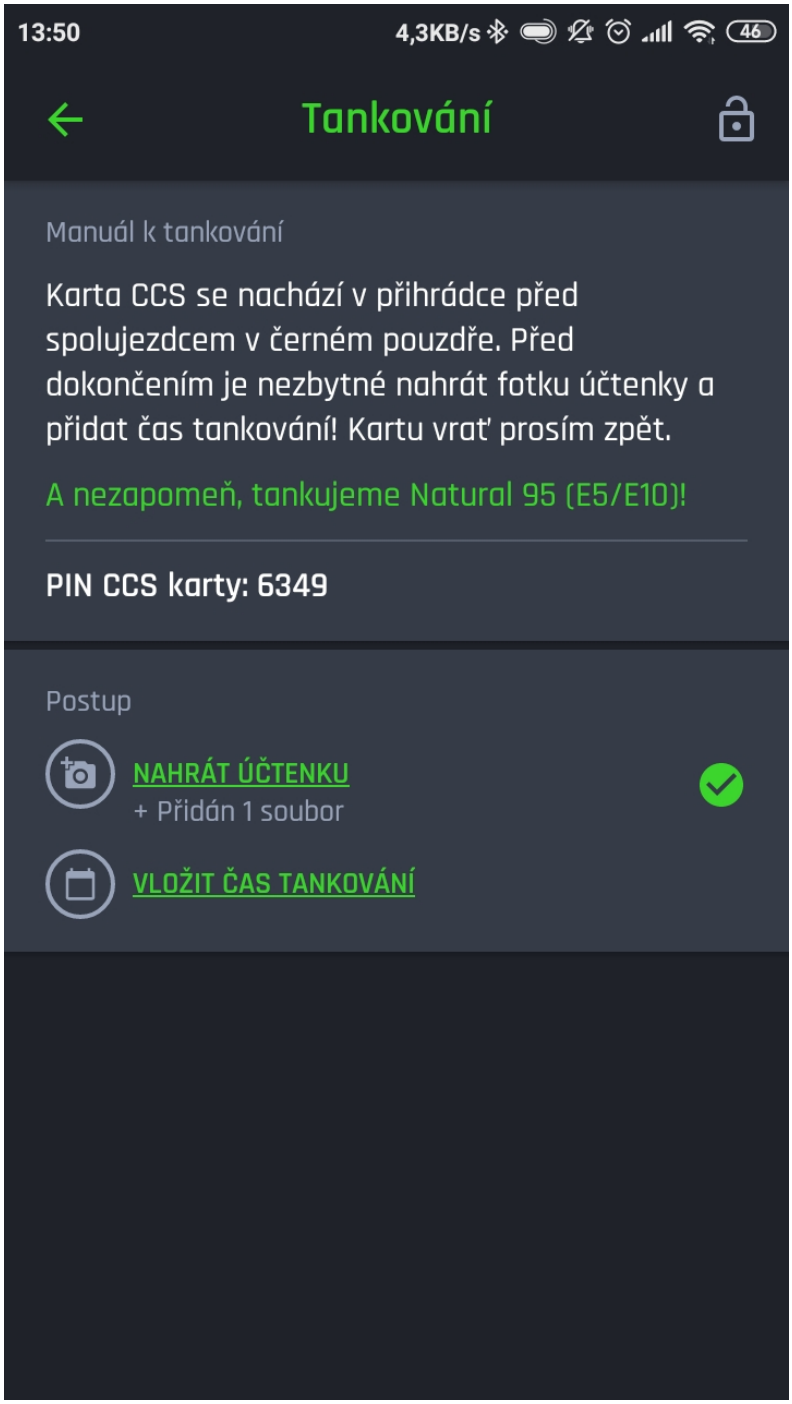
4. Pomocí tlačítka v aplikaci nahrajte libovolný počet fotografií Vámi umytých částí vozidla.
5. Dokončete kliknutím na tlačítko ve spodní části obrazovky.

Příloha C

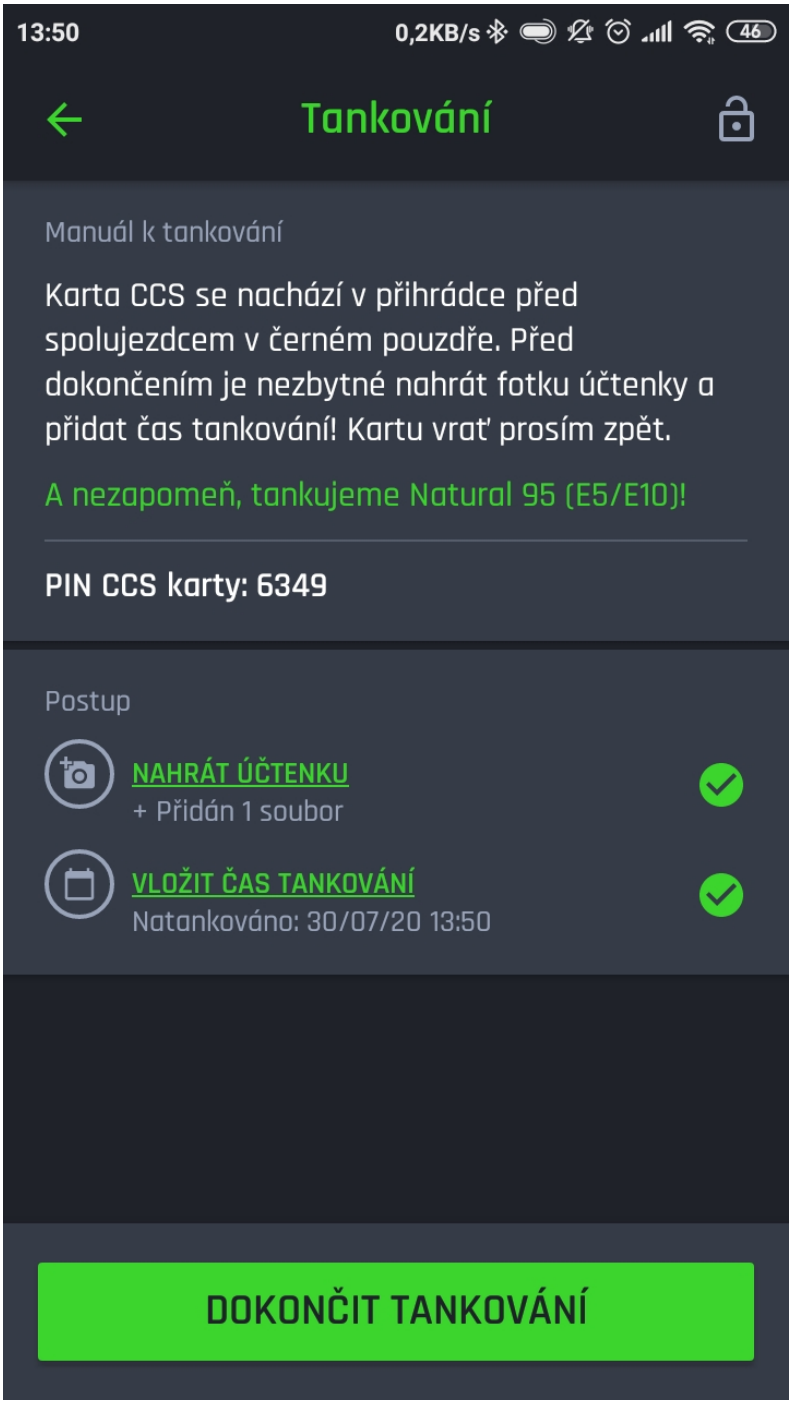
Snímky obrazovek implementovaných funkcionalit



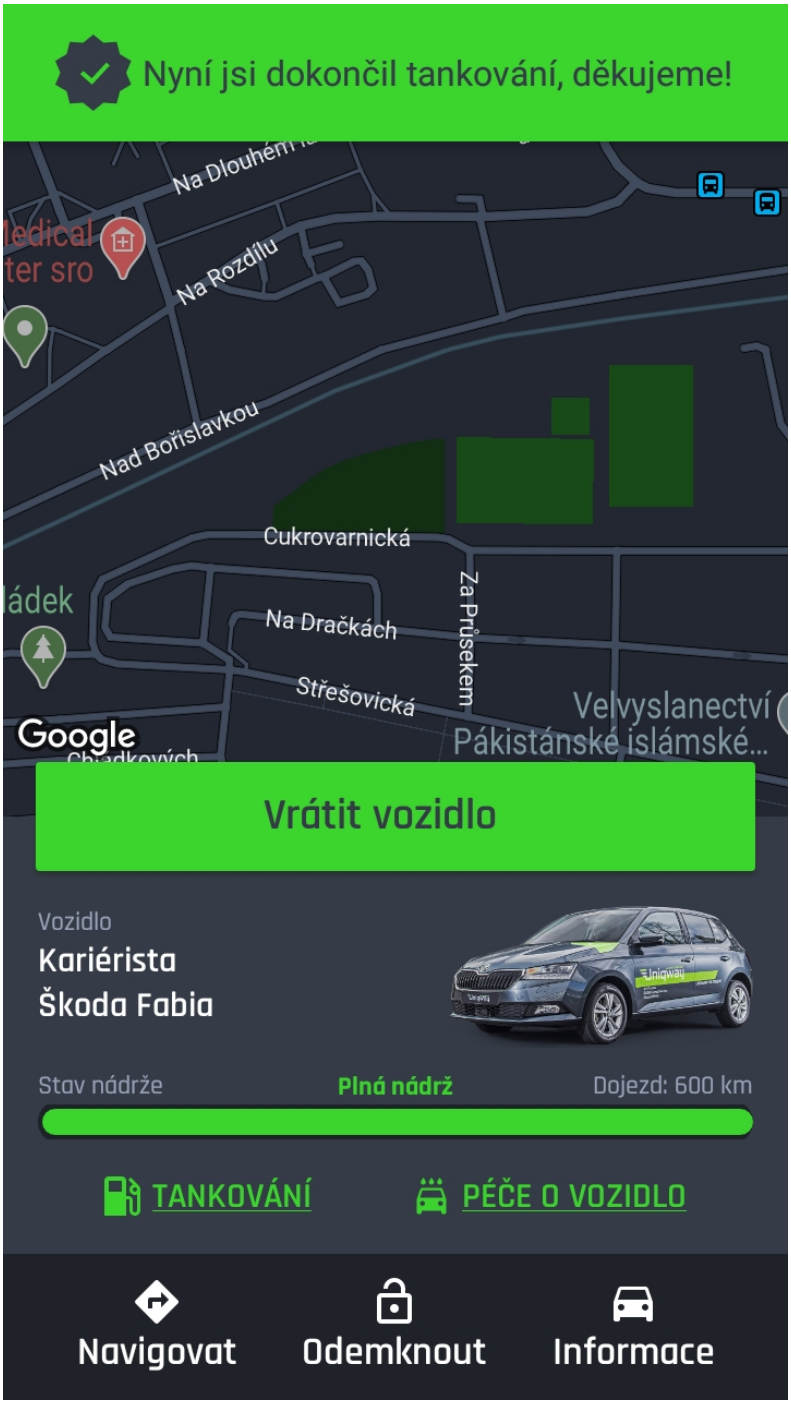
Obrázek C.1: Tankování po otevření



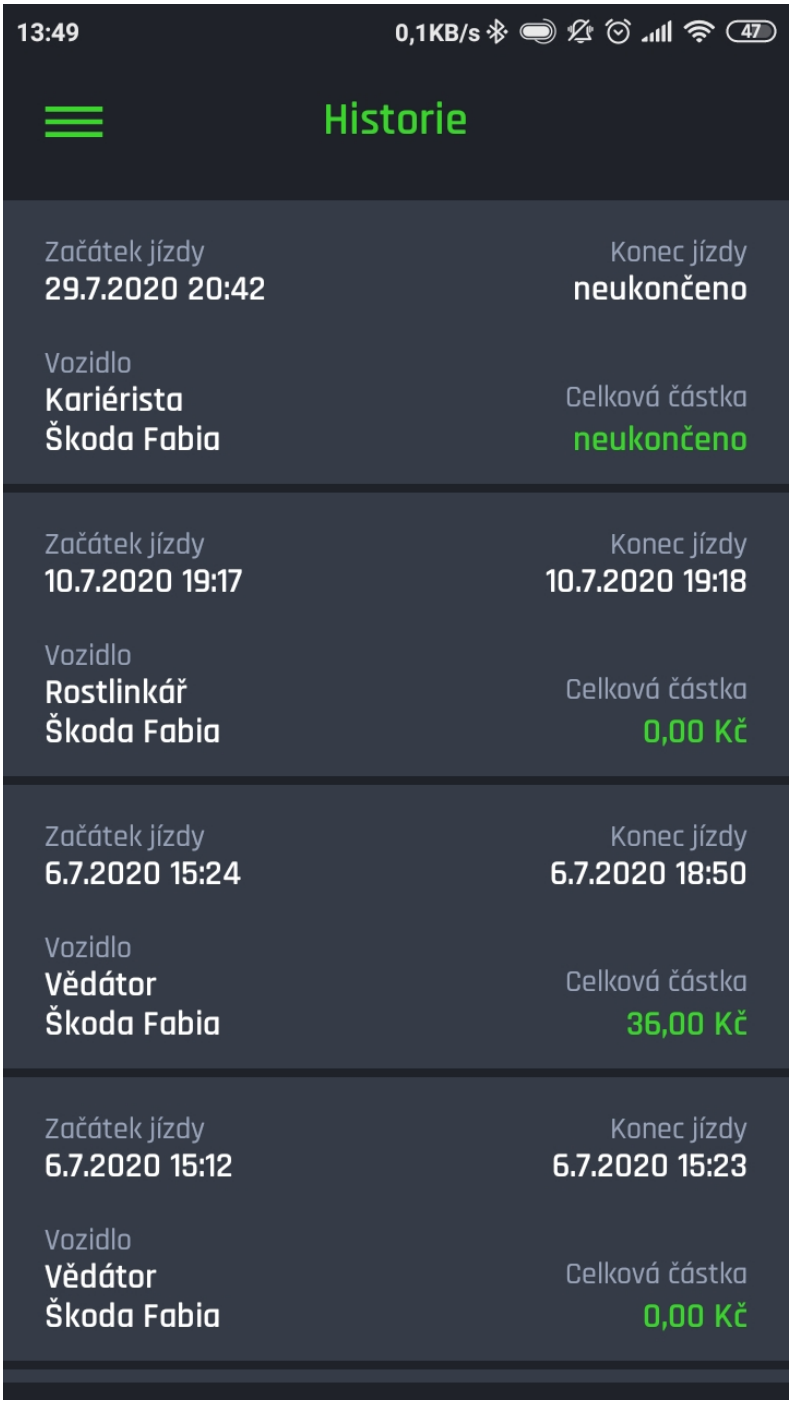
Obrázek C.2: Tankování s nahranou fotografií účtenky



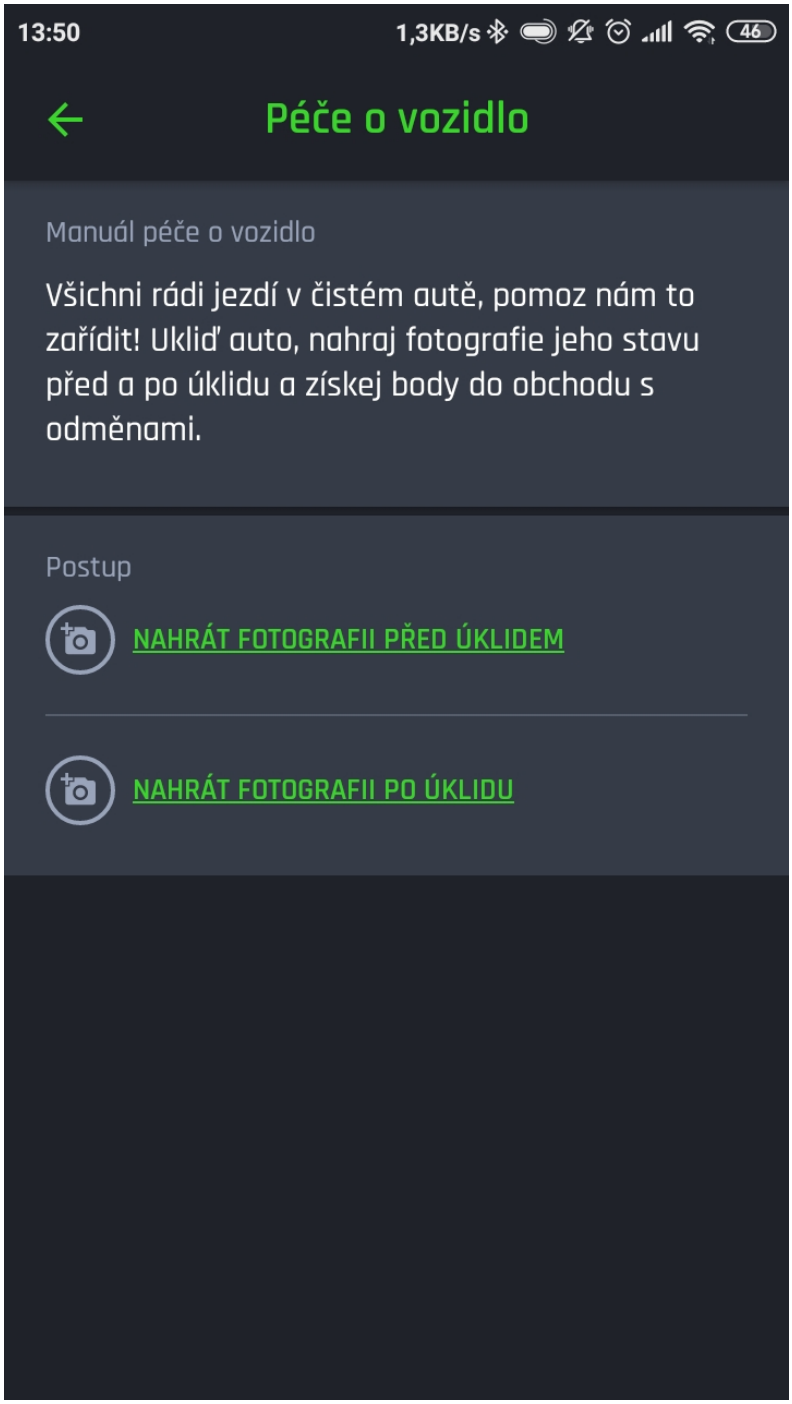
Obrázek C.3: Tankování s nahranou fotografií účtenky a vyplněným časem



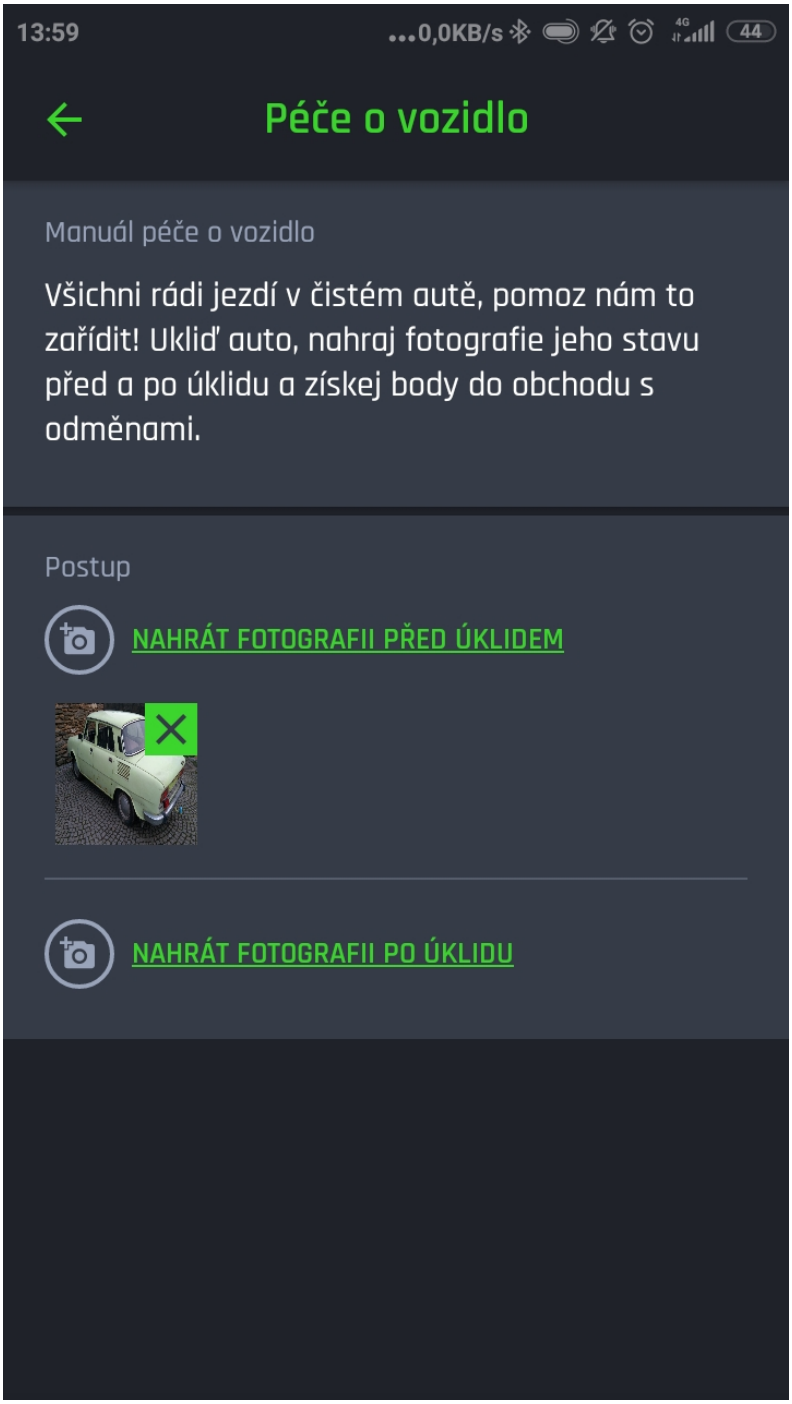
Obrázek C.4: Tankování - zpráva o úspěchu



Obrázek C.5: Historie jízd

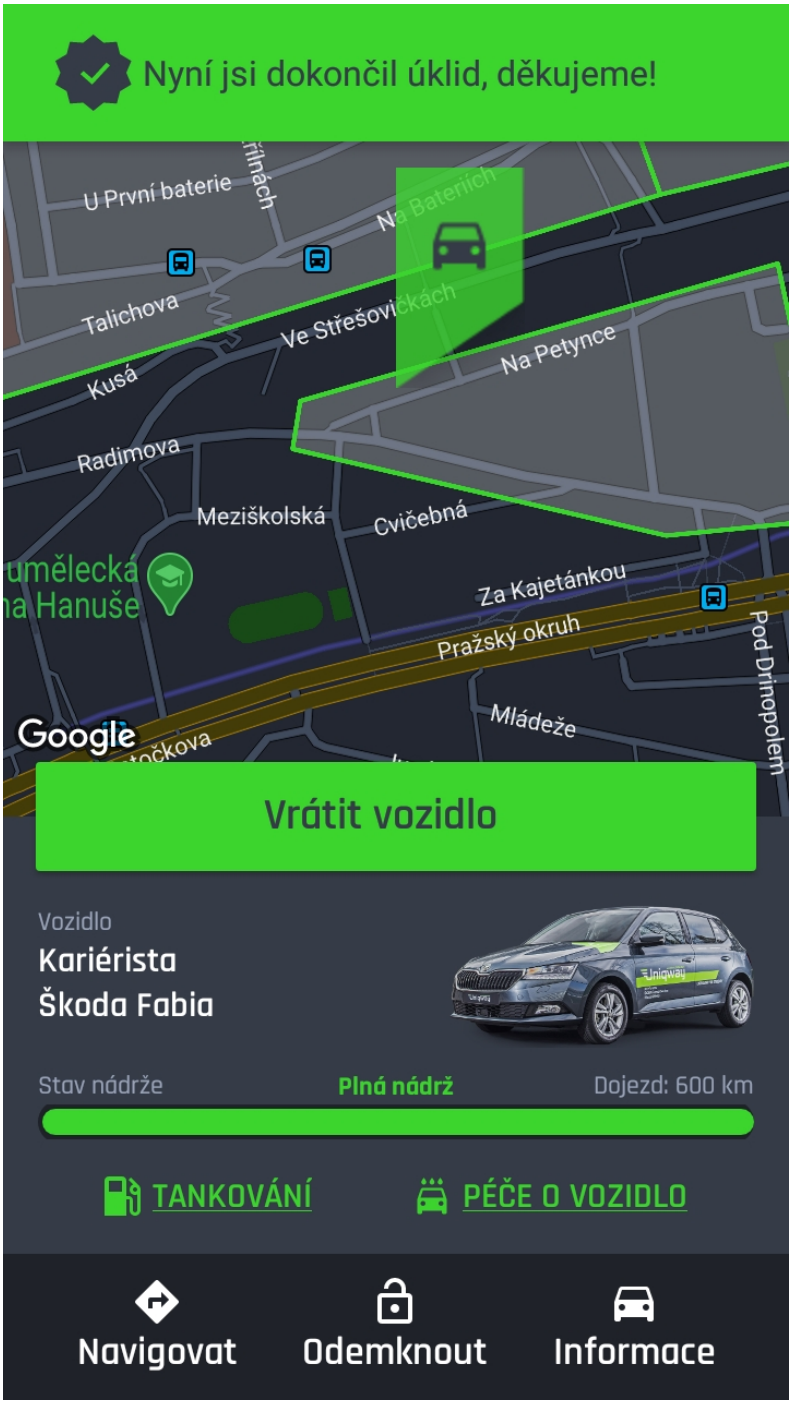


Obrázek C.6: Péče o vozidlo po otevření



Obrázek C.7: Péče o vozidlo s nahranou fotografií před úklidem

Obrázek C.8: Péče o vozidlo s nahranou fotografií před úklidem a fotografiemi po úklidu



Obrázek C.9: Péče o vozidlo - zpráva o úspěchu

Příloha D

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `apk` ..... adresář s instalovatelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF