F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Mobilní aplikace pro bezkontaktní tankování aut

Kurbanov Ruben
Softwarové inženýrství a technologie

Květen 2023
Vedoucí práce: doc. Ing. Ivan Jelínek, CSc.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kurbanov** Jméno: **Ruben** Osobní číslo: **495630**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mobilní aplikace pro bezkontaktní tankování aut

Název bakalářské práce anglicky:

Mobile application for contactless refueling of cars

Pokyny pro vypracování:

Navrhněte multiplatformní mobilní aplikaci, která uživatelům umožní natankovat do auta pomocí QR kódu, aniž by vozidlo opustili. Bude především určena lidem se zdravotním postižením. Aplikace umožní uživatelům kontrolovat platbu za tankování vozu Aplikace bude zároveň průvodcem, který poskytne všechny potřebné informace o dostupných čerpacích stanicích a navrhne cestu k nejbližší stanici nebo podle zadaných preferencí.
Proveďte rešerši stávajících řešení, rozbor funkčnosti těchto aplikací.
Identifikujte slabé a silné stránky těchto řešení a závěry rešerše vyhodnoťte.
Implementujte jak serverovou tak klientskou stranu.
Na straně serveru bude řešeno API pro realizaci komunikace mezi čerpacími stanicemi a zákazníky.
Aplikace musí mít pohodlný, strukturovaný, jasný a přátelský design navržený na webu Figma a implementována jako skutečné mobilní aplikace.
Aplikaci otestujte na operačním systému Android ze strany klienta podle konkrétního scénáře s cílem identifikovat nedostatky, nejasnosti a chyby. Výsledky testů vyhodnoťte.
Otestujte i stranu serveru, správné fungování logiky a implementaci přístupu a ukládání aplikačních dat. Zvolte vhodný systém průkazných testů.
Zhodnoťte užitečnost aplikace pro zdravotně postižené

Seznam doporučené literatury:

http://octo-technology.cz
https://krasnodarmedia.su/news/960119/?from=78
http://octo-technology.cz/jak-probiha-platba-na-bezobsluzne-cerpaci-stanici/
https://www.globusbonus.cz/
https://www.globusbonus.cz/clanky/1639-tankujte-ve-kteroukoliv-hodinu

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Ivan Jelínek, CSc. kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

doc. Ing. Ivan Jelínek, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

Poděkování / Prohlášení

Nejprve bych chtěl vyjádřit poděkování vedoucímu mé bakalářské práci panu doc.Ing. Ivanu Jelínkovi, CSc, který mi pomáhal ve všech fázích mého projektu. Také chci poděkovat své rodině a přátelům, kteří mě všemožně podporovali.

Prohlašuji, že jsem práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Motodickým pokynem a etické přípravě vysokoškolských závěrečných prací. Nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským.

V Praze dne 23. 05. 2023

.....

Abstrakt / Abstract

Navrhněte multiplatformní mobilní aplikaci, která uživatelům umožní natankovat do auta pomocí QR kódu, aniž by vozidlo opustili. Bude především určena lidem se zdravotním postižením. Aplikace umožní uživatelům kontrolovat platbu za tankování vozu Aplikace bude zároveň průvodcem, který poskytne všechny potřebné informace o dostupných čerpacích stanicích a navrhne cestu k nejbližší stanici nebo podle zadaných preferencí.

Klíčová slova: Flutter, Dart, Spring Boot, PostgreSQL, Mobile application, Google Maps

Design a cross-platform mobile application that allows users to refuel their car using a QR code without leaving the vehicle. It will primarily be intended for people with disabilities. The application will allow users to control the payment for refueling the car. The application will also be a guide that will provide all the necessary information about available gas stations and suggest a route to the nearest station or according to the preferences entered.

Keywords: Flutter, Dart, Spring Boot, PostgreSQL, Mobile application, Google Maps

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Cílem projektu je navrhnout a vytvořit uživatelsky přívětivou aplikaci, která svým uživatelům umožní tankovat auta, aniž by ji opustili. Tato aplikace poskytne uživatelům snadný a rychlý způsob, jak naplnit svůj vůz pomocí QR kódu, všechny potřebné informace o čerpacích stanicích, konkrétně nejbližší ziskové čerpací stanice a poskytne aktuální informace o cenách. Podnětem k výběru tématu práce byly životní zkušenosti, které se mi jevily jako dobrá příležitost se pokusit vytvořit něco užitečného, co by zlepšilo, zrychlilo, zjednodušilo proces doplňování paliva do auta, a aby byl bezpečnější a pohodlnější.
Stává se, že po příjezdu na čerpací stanici se nejprve zařadíte do fronty složené z aut a poté z lidí u pokladny. V takových případech může dojít k několika důsledkům:
1) Klient naléhavě potřebuje natankovat, protože jeho šipka paliva ukazuje na nulu a on tráví čas ve frontě na tankování, takže si v následujících časech bude pamatovat, že na této čerpací stanici strávil spoustu času a s největší pravděpodobností nepřijede znovu;
2) Klient se jednoduše otočí a pojede na další nejbližší čerpací stanici; První ani druhý případ nevyhovuje jak tankujícímu zákazníkovi, tak společnosti zásobující PHM. Tato aplikace tedy výrazně zkrátí dobu tankování, protože nemusíte trávit čas cestou k pokladně a zpět, čekáním ve frontě, dokud se nějaký hladový zákazník před vámi nerozhodne o výběru náplně do svého burgeru. Další výhodou tohoto způsobu tankování - bezpečnost. Často mohou být našimi pasažéry v autě děti nebo zvířata, která opravdu nechceme a není příliš bezpečné je nechávat samotné v uzavřeném autě. Do auta potřebují natankovat i osoby se zdravotním postižením, např. můj děda, který kvůli nemoci přišel o jednu nohu (během psaní této práce již přišel o druhou nohu, ale stále řídí auto), a aby se dostal nejprve na čerpací stanici, pak k pokladně a zpět, je třeba hodně úsilí. Proto, aby mohl natankovat své auto, musíme cestovat do čerpací stanici společně. Existuje však řešení (implementované v této práci), které pomůže handicapovaným využít čerpací stanici samostatně – možnost přivolat obsluhu u výdejního stojanu vybraného v aplikaci pro další pomoc.
Každý je odhodlán osvobodit lidstvo od banální, rutinní práce, jako je stání u pokladny, která poškozuje nejen zdraví, ale i emocionální složku. Mnoho z toho, co se den za dnem, čas od času opakuje, se lidé snaží automatizovat. Zlepšení a optimalizace těchto procesů je možná pouze automatizací, zbaví člověka této potřeby, a tím mu poskytne více času pro sebe, rodinu a rozvoj. Nesmíme zapomínat na takové zdroje, jako je papír a elektřina. Při každém tankování a platbě paliva se na pokladně vytiskne doklad o zaplacení, který se svou velikostí může zdát jako nepodstatná součást ekologických škod, ale když se tak děje denně a ve velkém množství, škody se mnohonásobně zvyšují. Nejčastěji tyto doklady vyhodíme, ale když jsou potřeba, je úplně jedno, v jakém formátu jsou, v papírové nebo elektronické podobě. Elektronická verze je rozhodně pohodlnější než ta papírová, která se může ztratit, ušpinit nebo zaneřádit přihrádku na rukavice nebo cormanové džíny. Existují také čerpací stanice, například od Globus Bonus, kde je využití lidských zdrojů nahrazeno terminály, které jsou umístěny u každé čerpací

stanice . Například, máme 4 čerpací stanice a každá z nich je oboustranná a každá strana potřebuje nainstalovat platební zařízení. To je nejen velká spotřeba energie, ale také drahé na vybavení.

Kapitola 2
Existující řešení

K dnešnímu dni již existují některé aplikace, které zákazníkům umožňují natankovat auto, aniž by ho opustili. Nejčastěji se jedná o aplikace konkrétních společností zabývajících se prodejem pohonných hmot. Uživatelé je mohou používat na svých mobilních zařízeních, stejně jako zobrazit potřebné informace na oficiálních stránkách, konkrétně - návod k použití, popis funkčnosti, zobrazení na mapě všech dostupných čerpacích stanic a mnoho odpovědí na otázky, které mohou uživatelé mít.

Některé aplikace jsou zaměřeny spíše na informační obsah, který pomůže najít nejbližší čerpací stanice nebo zorientuje ve věrnostním systému, který umožňuje zobrazit nasbírané body, zboží a služby, které konkrétní čerpací stanice poskytuje.

Výběr společností, které poskytují služby tankování aut, je dán tím, že sám jsem řidič a opakovaně jsem musel na čerpací stanice v ČR. Pro úplnost informací bylo v prohlížeči Google vyhledány funkčně podobné čerpací stanice s dotazy: „bezkontaktní čerpací stanice v ČR“, „tankování bez opuštění auta“ a „tankování auta pomocí QR kódu“.

V rámci této analýzy bude největší pozornost věnována aplikacím určeným zejména pro realizaci tankování vozu bezkontaktním nebo obdobným způsobem. Podobné mobilní aplikace budou analyzovány podle kritérií popsaných v následujících bodech:

- Přehlednost aplikace
- Způsoby platby
- Vzhled aplikace
- Obsah
- Péče o uživatele
- Práce s mapou

2.1 Globus Bonus

Globus Bonus je organizace, která poskytuje věrnostní program. Tato společnost má širokou škálu služeb/produktů. Při využívání této služby zákazníci získávají peněžní bonusy, za které mohou nakupovat zboží/služby této značky.

Web stránka:[1] Webová stránka je spíše informačního charakteru, konkrétněji - seznam produktů ve slevě s podrobným popisem, různé akce.

Mobilní aplikace: Má velmi velký výběr různých funkcí - od Scan and go (umožňuje skenovat produkty na prodejnách a následně nemusíte čekat na pokladně, dokud není zkontrolováno veškeré zboží) až po bezkontaktní tankování do auta Mobilní aplikace je k dispozici ke stažení pro iOS a Android. Abyste mohli začít používat aplikaci, musíte

[1] `https://www.globusbonus.cz`

provést následující kroky: Stáhnout - Registrovat - Vybrat si město v ČR - HOTOVO. V následujících dílech bude aplikace hodnocena pouze pro část - tankování aut.

Funkcionalita:

- **Aplikace umožňuje tankovat do auta pomocí QR kódu** - Pro natankování je potřeba naskenovat QR kód, který je umístěn na sloupu u čerpací stanice, vybrat částku, za kterou chcete natankovat, potvrdit platbu kartou (před nastartováním je vhodné kartu zaregistrovat v přihlášce předem).
- **Prohlédnout pokyny k doplňování paliva** - Před zahájením tankování si uživatel může přečíst stručný návod.
- **Zobrazit na mapě aktivních čerpacích stanic** - Je možné otevřít mapu a prohlédnout si všechny čerpací stanice této značky, které poskytují možnost bezkontaktního tankování.
- **Informace o ceně benzínu** - K dispozici je seznam druhů pohonných hmot, které jsou na čerpací stanici k dispozici spolu s cenami.

Zpráva o aplikaci: Aplikace je na první pohled velmi podobná webu. Dostatečně dlouhé načítání a pomalé odezvy z aplikace. Design působí velmi jednoduše a klidně, ale v době registrace se vyskytly nedostatky z hlediska kolize jednoho prvku s druhým. Vzhledem k tomu, že činnost této společnosti není soustředěna pouze na prodej benzínu, abych našel sekci, ve které bych mohl používat aplikace k účelu, pro který jsem si ji stáhl (konkrétně tankování auta), musel jsem utratit dostatek času na to, abych měl chuť jej zavřít a smazat.
Byly také identifikovány Plusy a Mínusy této aplikace:

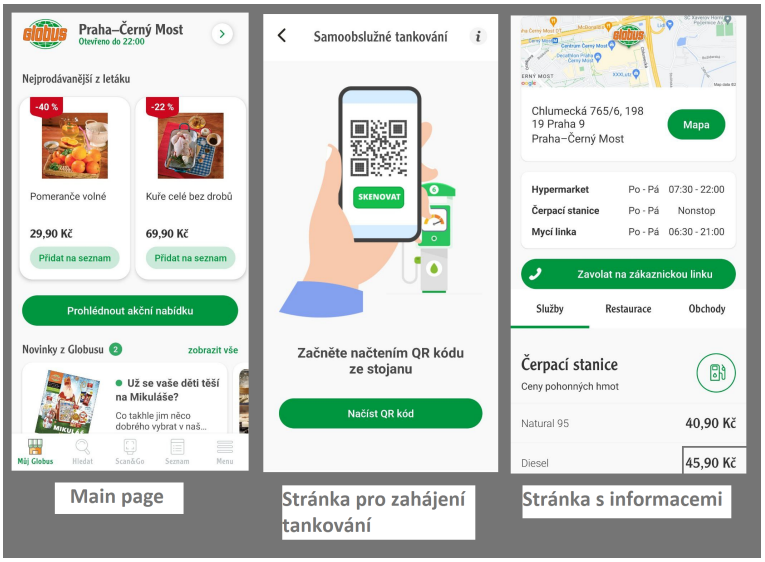
- [+] Při přihlašování se používá autorizace pomoci Google
- [+] Nenápadný design
- [+] Obsah aplikace nemá přebytek
- [-] Při výběru města je potřeba vybrat ze seznamu. Výhodnější by bylo, kdyby se město nabízelo automaticky geolokací.
- [-] Při registraci - vyplnění pole s datem narození se tečky automaticky nenahrazují, musel jsem je vložit sam. Byl jsem opravdu zmatený, když došlo na vyplňování formuláře adresy. PSČ, město, adresa. Pokud vše zadáte samostatně, pak formulář neurčuje zadanou adresu, ale pokud vyplníte pouze pole s ulicí, aplikace automaticky navrhne možnosti.

Hodnocení aplikace na Google Play: 4,5[2] (Poslední kontrola 03.11.2022)

Platformy: IOS[3], Android[4]

Vzhled aplikace:

[2] https://play.google.com/store/apps/details?id=com.pragonauts.globus
[3] https://apps.apple.com/cz/app/m%C5%AFj-globus/id1425012509?l=cs
[4] https://play.google.com/store/apps/details?id=com.pragonauts.globus



Obrázek 2.1. Ukázka vzhledu aplikace Globus Bonus.

2.2 ORLEN Benzina

Patří do skupiny ORLEN Unipetrol, která je součástí nadnárodní skupiny ORLEN, a provozuje největší síť čerpacích stanic v České republice. Značka byla založena již v roce 1958 a dnes nabízí řadu produktů a služeb souvisejících s pohonnými hmotami a dalšími. Firma poskytuje uživatelům možnost tankovat do svých vozů pomocí aplikace. Všechny jejich stanice, které podporují bezkontaktní platby na čerpacích stanicích, jsou otevřené 24 hodin denně. Kromě pohonných hmot mohou uživatelé nakupovat značkové produkty také pomocí věrnostního programu a značkové karty TANKART.

Web stránka:[5] Na jejich stránkách se uživatelé mohou seznámit s aktuálními akcemi, věrnostním programem, zjistit informace o společnosti, produktech, ale i seznámit se s bezkontaktním tankovacím systémem, návody a jeho funkčností.

Mobilní aplikace: Všechny funkce začínají v aplikaci. Po registraci / autorizaci se uživatel může seznámit s aktuálními akcemi, produkty. Bonusový systém umožňuje uživatelům sbírat body „koníky“ a platit s nimi v aplikaci. Vzhledem k tomu, že aplikace pracuje přímo s platební kartou uživatele, vyžaduje aplikace přístupový kód nebo biometrii otisků prstů, což zajišťuje bezpečnost používání aplikace třetími stranami. Abyste mohli zahájit proceduru tankování vozu, musíte nejprve naskenovat QR kód, vybrat si, jakým způsobem chcete za tankování auta zaplatit, poté vybrat typ paliva, částku, za kterou je třeba vůz natankovat, a potvrdit způsob platby.

Funkcionalita:

- **Mobilní platby za tankování** - Nabízejí rychlý způsob platby za pohonné hmoty prostřednictvím mobilní aplikace, aniž byste museli chodit k pokladně.
- **TANKARTA Easy and Business** - Karta, která uživatelům umožňuje dobít ji, aby zaplatili za tankování.
- **Odměny za věrnost** - Věrnostní systém, který uživatelům umožňuje mít spořicí účet, který se bude doplňovat při každém tankování vozu, poté mohou platit za značkové zboží.
- **Navigace ke stanicím** - Uživatelé si mohou v aplikaci zobrazit všechny stanice společnosti pomocí mapy a aplikace automaticky určí nejbližší čerpací stanici k vám.

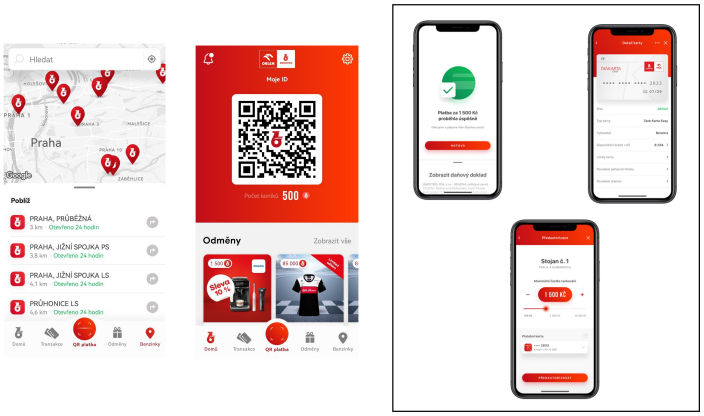
[5] `https://www.orlenbenzina.cz`

Zpráva o aplikaci: Tato aplikace překvapila svým vzhledem. Aplikace navíc umožňuje změnit jazyk. Vyhovovala mi také karta TANKART, která umožňuje mít společnou kartu pro firmu nebo rodinu. Stačí ji jen doplnit a na přání s ní může platit několik lidí najednou. Pokud jde o návod, tato aplikace téměř žádný nemá. Tato aplikace nemá nic nadbytečného a vše lze bez problémů najít, ale nezkušení uživatelé se v ní mohou zmást.

Hodnocení aplikace na Google Play: 3,5[6] (Poslední kontrola 03.11.2022)

Platformy: IOS[7], Android[8]

Vzhled aplikace:



Obrázek 2.2. Ukázka vzhledu aplikace ORLEN Benzina.

2.3 OMV MyStation

OMV je předním ropným a plynárenským koncernem ve střední Evropě. Kromě pohonných hmot nabízí společnost širokou škálu služeb pro mytí aut, občerstvení a další. Služby této značky jsou z velké části zapojeny do věrnostního systému. Jelikož se společnost zaměřuje na mnoho aspektů, všechny jsou zahrnuty do věrnostního systému. Za nákup zákazníci dostávají slevy, bonusy a mohou jimi platit.

Web stránka:[9] Jejich stránka slouží pouze pro informační účely. Na něm se můžete seznámit se seznamem funkčnosti mobilní aplikace, údaji o firmě atd.

Mobilní aplikace: V mobilní aplikaci se uživatelé mohou seznámit s aktuálními slevami/nabídkami/kupony, zobrazit si všechny stanice této značky na mapě. Aplikace má také charakter elektronické peněženky, která uživateli umožňuje využívat služby společnosti pomocí věrnostního systému pomocí elektronické karty (QR kódu). Bonusový systém může také využívat více lidí (například rodina nebo firma). Filtr pomůže uživateli zobrazit na mapě ty čerpací stanice, které mají myčky aut nebo třeba kavárny.

Funkcionalita:

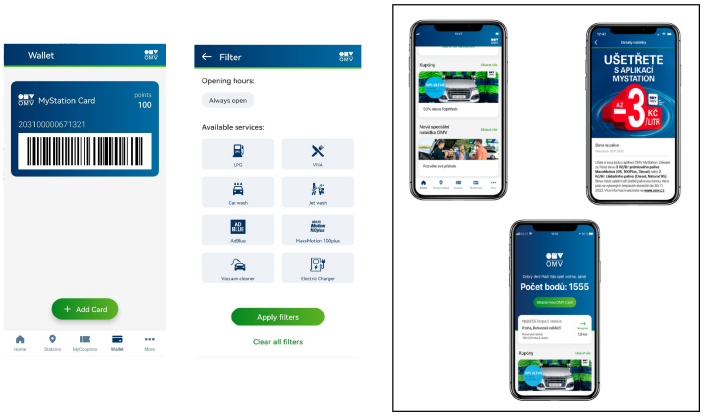
- **Kuponový a bodový systém** - V aplikaci je již při registraci vytvořena karta, která je kumulativní. Systém umožňuje shromažďovat body a utrácet je za produkty společnosti.

[6] https://play.google.com/store/apps/details?id=cz.benzina.aplikace
[7] https://apps.apple.com/cz/app/benzina-orlen-aplikace/id1337659525
[8] https://play.google.com/store/apps/details?id=cz.benzina.aplikace
[9] https://www.omv.cz/cs-cz/omv-mystation-v-cesku

- **Online peněženka** - Pomocí elektronické peněženky můžete uložit všechny věrnostní karty do systému a použít je místo plastových karet předložením čárového kódu u pokladny.
- **Hledání** - Aplikace nabízí vyhledávání na mapě čerpacích stanic pomocí filtru. Vybrat si můžete čerpací stanici například s kavárnou nebo takovou, která podporuje nabíjení elektromobilů.

Zpráva o aplikaci: Aplikace dle mého názoru působí velmi nudně a smutně. Monochromatický design a dostatek informací, které se někdy nacházejí na těch místech aplikace, kde by neměly být nebo jsou tam nadbytečné. Funkčnost, která je v aplikaci přítomná, je velmi vzácná a lze ji nahradit mnoha jinými aplikacemi, kromě věrnostního systému. Návod jako takový neexistuje.

Hodnocení aplikace na Google Play: 2,7[10] (Poslední kontrola 03.11.2022)
Platformy: IOS[11], Android[12]
Vzhled aplikace:



Obrázek 2.3. Ukázka vzhledu aplikace OMV MyStation.

2.4 Shell

Jde o globální skupinu energetických a petrochemických společností, která vyrábí plyn a ropu, provádí údržbu letadel a vyrábí také benzín, motorové oleje a maziva. Na území Československa byla společnost Shell založena v roce 1991, později se po rozdělení státu stala v roce 1993 samostatnou společností na území České republiky. Společnost má velké množství čerpacích stanic, které uživatelům umožňují platit za tankování ze svého telefonu. Společnost také zohlednila věrnostní systém, pomocí kterého mohou uživatelé získávat slevy, sbírat body a platit za produkty stejnými body.

Web stránka:[13] Na stránce můžete najít různé informace o společnosti, produktech, aplikaci atd. Vzhledem k tomu, že činnost společnosti je zaměřena spíše na těžbu a výrobu paliv, informace o tom, co jejich aplikace dělá, a doplňující informace o čerpacích stanicích se hledaly poměrně obtížně.

Mobilní aplikace: Registrace v aplikaci poskytuje především možnost vytvořit si věrnostní kartu, která bude sbírat body za platby za zboží a pohonné hmoty. Mobilní

[10] https://play.google.com/store/apps/details?id=com.comarch.clm.mobileapp.omvcz
[11] https://apps.apple.com/cz/app/omv-mystation-v-%C4%8Desku/id150236175
[12] https://play.google.com/store/apps/details?id=cz.benzina.aplikace
[13] https://www.shell.cz

aplikace také implementuje vyhledávání nejbližších čerpacích stanic a platbu za tankování prostřednictvím mobilní aplikace. Způsob tankování se však od těch, které byli popsány výše, liší tím, že místo skenování QR kódu musíte ručně zadat číslo čerpací stanice.

Funkcionalita:

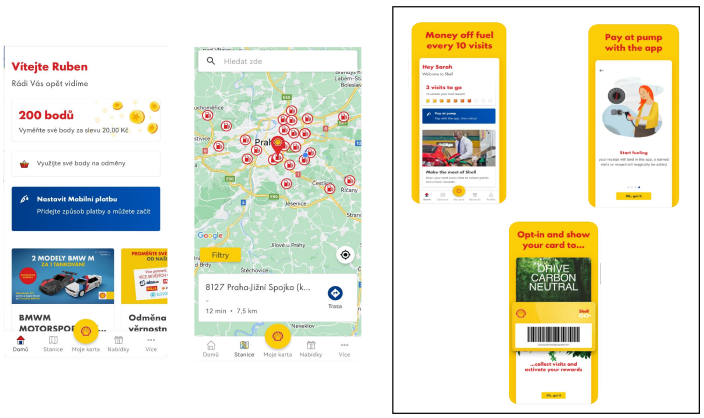
- **Digitální karta** - Virtuální karta, která umožní hromadit bonusy a utrácet je na čerpacích stanicích Shell.
- **Shell SmartPay - pohodlná platba z auta** - Virtuální karta, která umožní hromadit bonusy a utrácet je na čerpacích stanicích Shell.
- **Vyhledávač stanic** - Navrhne nejbližší čerpací stanici na základě geolokace a také pomůže najít ty čerpací stanice, které vyhovují potřebám.

Zpráva o aplikaci: Bohužel v době psaní této zprávy a den poté přestala aplikace fungovat kvůli technickým pracím nebo se něco pokazilo. Ale v každém případě některé funkce stále fungovaly. Vzhled návrhu mi taky připadá nudný a místy až zastaralý. Neexistují také žádné podrobné pokyny k používání aplikace, stejně jako pohodlný způsob, jak získat zpětnou vazbu od služby podpory.

Hodnocení aplikace na Google Play: 4,3[14] (Poslední kontrola 04.11.2022)

Platformy: IOS[15], Android[16]

Vzhled aplikace:



Obrázek 2.4. Ukázka vzhledu aplikace Shell.

2.5 Závěrečné srovnání

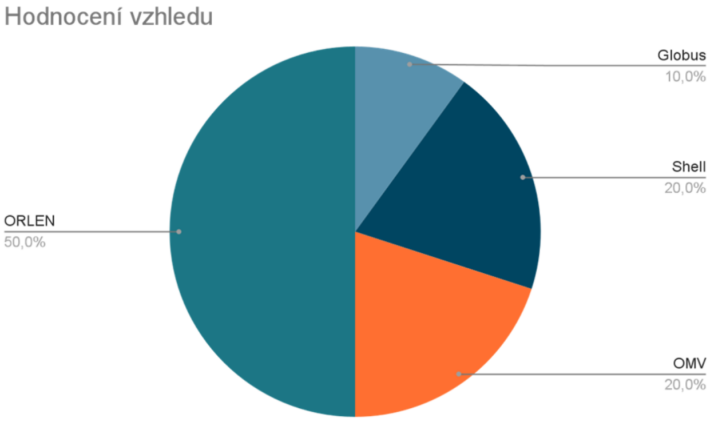
Aplikace bude hodnocena na stupnici od 1 do 4, kde 1 - výborně, 2 - velmi dobře, 3 - dostatečný, 4 - nedostatečný. Skóre bude založeno na množství funkcí, které konkrétní aplikace podporuje.

Funkčnost

[14] https://play.google.com/store/apps/details?id=com.shell.sitibv.retail
[15] https://apps.apple.com/cz/app/shell/id1464649113
[16] https://play.google.com/store/apps/details?id=com.shell.sitibv.retail&hl=cs

Funkce	Globus	ORLEN	OMV	Shell
Práce s mapou	Ano	Ano	Ano	Ano
Hledání nejbližší čerpací stanice	Ne	Ano	Ano	Ano
Platba za tankování kartou přes aplikace	Ano	Ano	Ne	Ano
Tankování pomocí QR kódu	Ano	Ano	Ne	Ne
Firemní využití aplikace	Ne	Ano	Ano	Ne
Podpora ukládání více vozidel	Ne	Ne	Ne	Ne
Technická podpora (formou chatu/ telefonu)	Ano	Ano	Ne	Ano
Instruktáž aplikace	Ano	Ne	Ne	Ano
Informace o aktuálních cenách na čerpací stanici	Ano	Ne	Ne	Ne
Podpora lidí se zdravotním postižením	Ne	Ne	Ne	Ne
Funguje bez přerušení	Ne	Ano	Ano	Ne
HODNOCENI	2	1	4	3

Tabulka 2.1. Závěrečné srovnání existujících řešení.



Obrázek 2.5. Závěrečné srovnání existujících řešení. Vzhled.

Vzhled Celkem hlasovalo 10 respondentů včetně mě. A soudě podle grafu níže je vidět, že většině se nejvíce líbila aplikace ORLEN Benzina. Výsledky jsou uvedeny na obrázku 2.5.

2.6 Závěr

Hlavním účelem analýzy stávajících řešení bylo zjistit silné a slabé stránky aplikací pro bezkontaktní tankování aut. Analyzovány byly 4 aplikace, které jsou lídry co do počtu stažení. Dá se dokonce říci, že tyto aplikace jsou navzájem téměř totožné, pouze různé společnosti. Zákazníci čerpacích stanic se s největší pravděpodobností neřídí značkou a docházejí na čerpací stanici podle potřeby na tu, která je nejblíže. Pro každou čerpací stanici není používání různých aplikací také příliš pohodlný postup, a proto jsem chtěl navrhnout systém, který by propojil všechny čerpací stanice a poskytnul celé to spektrum funkčnosti všem a pomocí jedné aplikace.
Dle výsledků hodnocení je nejúspěšnější aplikace - ORLEN Benzina. Tato aplikace má opravdu velmi příjemný vzhled, téměř veškerou potřebnou funkcionalitu pro každodenní a pohodlné tankování do auta.

Kapitola 3
Analýza funkčnosti aplikací

3.1 Analýza potenciálních uživatelů

Aplikace je určena především pro řidiče automobilů. Funkcionalita je určena pro různé typy řidičů, pro ty, kteří pravidelně cestují za prací, studiem, pro turisty, kteří přijedou na víkend a krátkodobě si vezmou carsharing, ale i pro lidi s handicapem.. Funkce doplnění informací o voze nebo platebních metod zjednoduší kontrolu tankování vozů, pokud jde o použití správného paliva a kontrolu nákladů na prostředky přímo na tankování. Funkce vyhledání nejbližší čerpací stanice může být pro řidiče velmi praktická. Uživatelé mohou také využívat služby Google Maps, které jsou součástí aplikace. Funkce přivolání personálu k čerpací stanici je vhodná pro osoby se zdravotním postižením. Každá čerpací stanice v aplikaci má podrobné informace o dostupných službách, otevírací době, aktuálních cenách pro ty, kteří se špatně orientují ve městě.

3.2 Funkční požadavky

V této kapitole budou vygenerovány funkční požadavky, které určují, jaká funkcionalita bude uživatelům k dispozici, čímž vznikne ucelený seznam funkcí. Při vývoji softwaru je důležité nejprve přesně definovat, co může uživatel se softwarem dělat.

3.2.1 Autorizace a registrace

- FP [1]: Aplikace umožňuje registrace uživatelům. Pro registrace je potřeba do systému zadat: Jméno (povinné), Příjmení (povinné), Email (povinné), Tel. číslo (volitelné), Pohlavi (povinné), Datum narozeniny (povinné).
- FP [2]: Aplikace umožňuje autorizace uživatelům. Pro manualni autorizace je potřeba do systému zadat E-mail + Heslo.
- FP [3]: Uživatel si může obnovit zapomenuté heslo.

3.2.2 Práce s mapou

- FP [4]: Uživatel si zobrazí mapu se všemi čerpacími stanicemi, které podporují systém bezkontaktních čerpacích stanic.
- FP [5]: Systém uživateli nabízí nejbližší čerpací stanice.
- FP [6]: Uživatel si může vytvořit trasu k nejbližší čerpací stanici.

3.2.3 Informace

- FP [7]: Aplikace umožní uživatelům prohlédnout seznam všech nabídek konkrétních čerpacích stanic.
- FP [8]: Uživatel si může přečíst návod k použití aplikace.
- FP [9]: Systém podporuje více jazyků (angličtinu, češtinu, ruštinu).

3.2.4 Tankování

- FP [10]: Uživatel může naskenovat QR kód pomocí kamery na stojanu čerpací stanice a identifikovat konkrétní čerpací stanici a stojan v systému.
- FP [11]: Uživatel může ručně zadat kód výdejního stojanu čerpací stanice za účelem identifikace konkrétní čerpací stanice a výdejního stojanu v systému.
- FP [12]: Uživatel si může vybrat druh paliva.
- FP [13]: Uživatel si může zvolit množství paliva k tankování v litrech/kroonech.
- FP [14]: Uživatel si může vybrat vůz, který naplní, pokud je v systému vytvořeno více než 1 vůz. Je nutné, aby nedošlo k záměně typu paliva, který je vhodný pro tento vůz, jinak systém nedovolí natankovat do vozu nesprávné palivo.
- FP [15]: Uživatel může zaplatit za tankování výběrem bankovní karty ze seznamu registrovaného v systému.
- FP [16]: Uživatel může přivolat obsluhu čerpací stanice, aby mu pomohla s tankováním (pro handicapované).
- FP [17]: Uživatel si může přečíst návod k používání bezkontaktního tankování.
- FP [18]: Uživatel může ohodnotit náplň (udělit hodnocení od 1 do 5 hvězdiček, kde 1 je velmi špatná, 5 je vynikající), stejně jako zanechat komentář nebo tuto položku přeskočit.

3.2.5 Podpora

- FP [19]: Uživatel bude moci kontaktovat podporu pomocí komentáře.
- FP [20]: Uživatel může kontaktovat službu podpory pro konkrétní případ tankování (vybrat z historie tankování) pomocí komentáře.

3.2.6 Transakce

- FP [21]: Uživatel může filtrovat seznam transakcí podle následujících parametrů: datum, částka (od - Kč; do - Kč), natankovaný vůz, použitá platební metoda (karta).
- FP [22]: Uživatel si může prohlédnout seznam všech tankování, která provedl. Každá transakce obsahuje následující informace: datum a čas, číslo a značka vozu (pokud je v systému nastaveno), způsob platby (bankovní karta), počet litrů, částka.
- FP [23]: Uživatel může kontaktovat službu podpory pro konkrétní případ tankování (vybrat z historie transakcí) pomocí komentáře (FP[20]).
- FP [24]: Uživatel si bude moci stáhnout/sdílet potvrzení o tankování.

3.2.7 Platba

- FP [25]: Uživatel může zaregistrovat více než jednu platební metodu (platební kartu). Pro přidání nové karty je potřeba zadat: číslo karty, datum expirace, cvv kód.
- FP [26]: Uživatel si může prohlédnout seznam všech registrovaných platebních karet.
- FP [27]: Uživatel může odebrat platební kartu ze seznamu.

3.2.8 Auto

- FP [28]: Uživatel může zaregistrovat více než jeden vůz. Pro přidání nového vozu je potřeba zadat: číslo vozu, značka vozu, typ paliva.
- FP [29]: Uživatel si může prohlédnout seznam všech registrovaných aut
- FP [30]: Uživatel může odebrat vůz ze seznamu.

3.3 Nefunkční požadavky

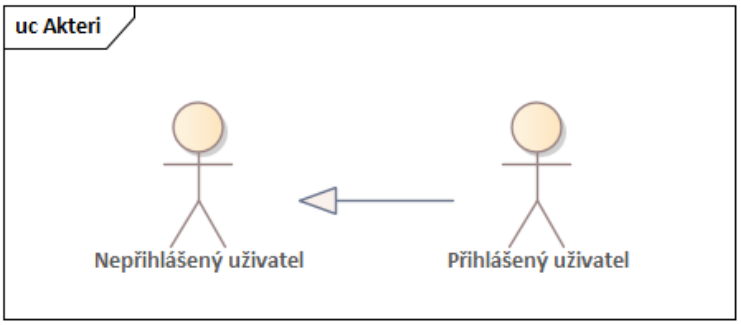
- NFP [1]: Ukládání dat do databáze. Po registraci/autorizaci budou všechna data uživatele uložena do databáze, pokud uživatel zadal data nesprávně, pak se data neuloží.
- NFP [2]: Systém nerozhoduje o platebním systému. Databáze neukládá daty karet, ale jen poslední 4 čísla karty. Kvůli tomu, že systém nevyhovuje standardům PCI DSS.
- NFP [3]: Geolokace. Aby aplikace správně fungovala pro FP[5] - FP[6], bude systém vyžadovat přístup ke geolokaci uživatele, ale základní funkce aplikace budou dostupné i bez uvedeného povolení.
- NFP [4]: Kamera. Aby aplikace správně fungovala pro FP [10], bude systém vyžadovat přístup ke kameře, ale základní funkce aplikace budou dostupné i bez uvedeného povolení.
- NFP [5]: Format komunikace mezi klientem-serverem. Data jsou odesílána ve formátu JSON.
- NFP [6]: Architektura serveru. Komunikace je realizována pomocí architektury REST - HTTP request.

3.4 Případy užití (Use Case)

Diagramy případů užití definují, kdo bude systém používat a jaká budou mít práva, vycházejí z definice funkčních požadavků, které popsány v kapitole 3.2. Diagramy zahrnují: subjekty, které jsou zapojeny do systému; akce, které popisují chování systému. Kompletní schéma případu použití je uvedeno v příloze B.

3.4.1 Aktéři

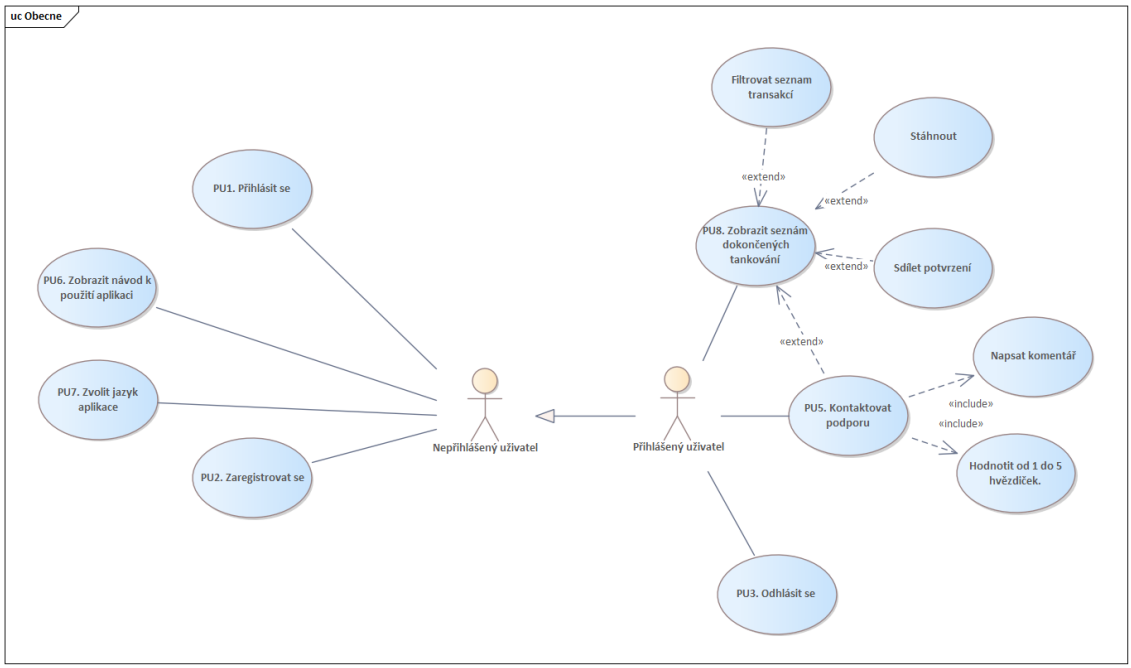
Nepřihlášený uživatel - nebude omezen v používání aplikace, ale nebudou mu bez oprávnění dostupné funkce jako: tankování bezkontaktním způsobem; tvorba platebních metod, aut a manipulace s nimi; hodnocení, komentář k čerpacím stanicím.
Autorizovaný uživatel - má všechna privilegia, která má neoprávněný uživatel, nicméně se mu otevírá celá škála možností, které aplikace nabízí: tankování bezkontaktním způsobem; tvorba platebních metod, aut a manipulace s nimi; hodnocení, komentář k čerpacím stanicím.



Obrázek 3.1. Diagram aktérů (Use Case)

3.4.2 Obecné

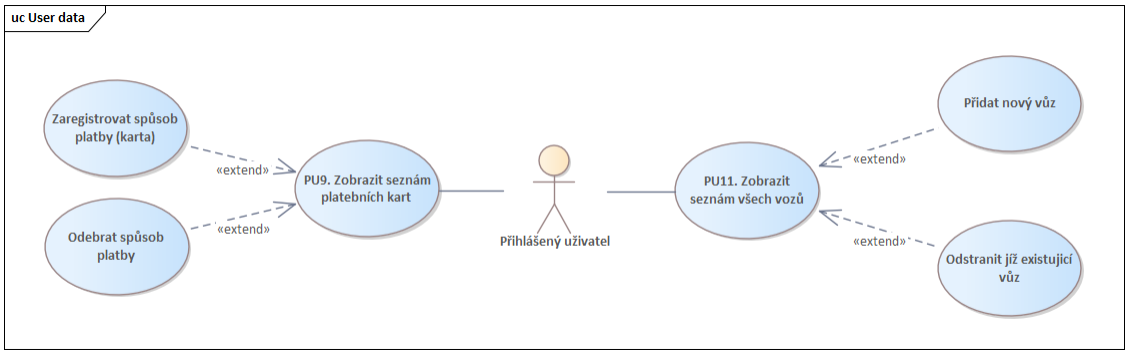
Aplikace umožňuje neoprávněnému uživateli přihlásit se (PU1) pomocí e-mailu a hesla, přečíst si pokyny (PU6), změnit jazyk aplikace (PU7), zaregistrovat si nový účet (PU2) v případě, že uživatel zadal email, který ještě není registrován v systému a všechny potřebné údaje při registraci byly vyplněny bez chyb. Aplikace umožňuje oprávněnému uživateli požádat o pomoc při podpoře (PU5) napsáním komentáře (recenze) ke konkrétnímu případu tankování nebo zanechat recenzi na konkrétní tankování; zobrazit seznam všech transakcí (PU8), kontaktovat službu podpory pro konkrétní transakci, stáhnout potvrzení , filtrovat seznam transakcí podle parametrů (datum, částka (od - Kč; do - Kč), natankovaný vůz, použitá platební metoda (karta)) nebo je sdílet v messengeru/zprávách.



Obrázek 3.2. Diagram případu použití. Oběcné.

3.4.3 Uživatelská data

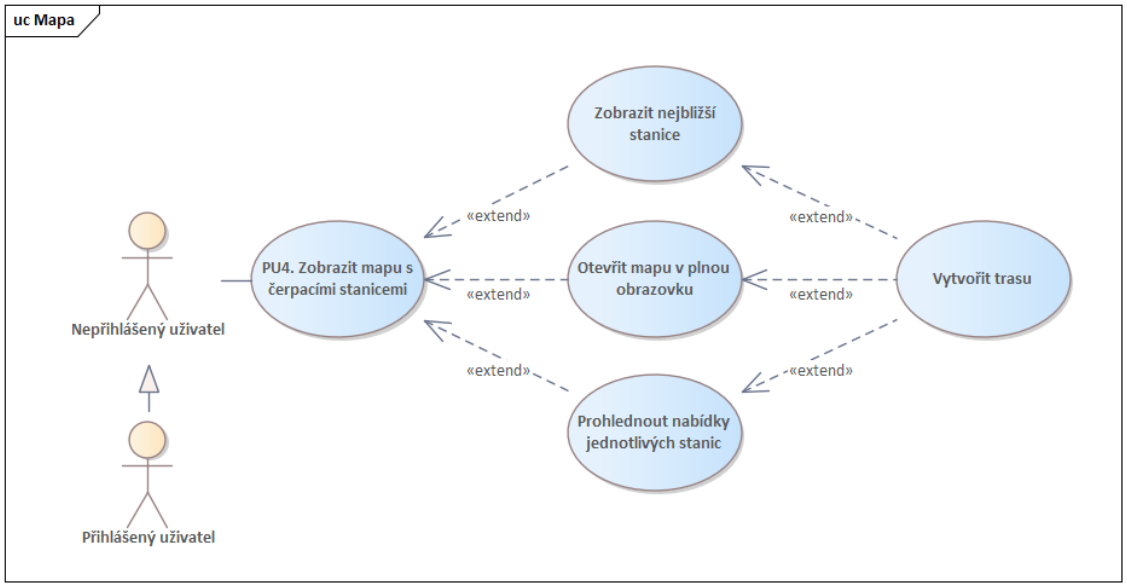
Aplikace umožňuje oprávněnému uživateli prohlížet seznam registrovaných platebních metod (PU9), vytvářet nové a odstraňovat stávající ze seznamu; umožňuje oprávněnému uživateli prohlížet seznam registrovaných vozů (PU11) i vytvářet nové a mazat stávající ze seznamu.



Obrázek 3.3. Diagram případu použití. Uživatelská data.

3.4.4 Mapa

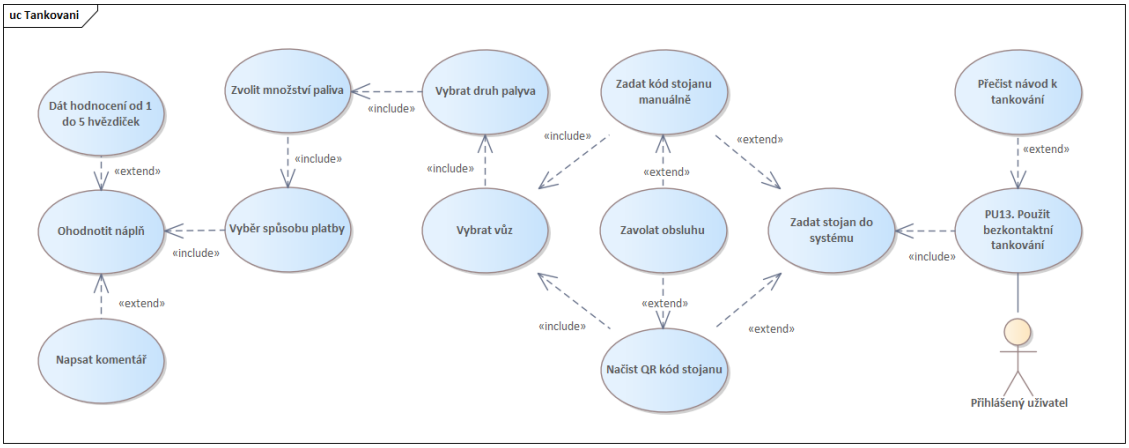
Aplikace umožňuje oprávněným i neoprávněným uživatelům pomocí mapy zobrazit umístění čerpacích stanic (PU4), které využívají systém bezkontaktního tankování. Uživateli se také zobrazí nejbližší čerpací stanice, ke které můžete získat trasu v Google maps; zobrazit podrobnosti o konkrétní čerpací stanici nebo otevřít mapu na celé obrazovce, abyste získali lepší přehled o místě.



Obrázek 3.4. Diagram případu použití. Mapa.

3.4.5 Tankování

Aplikace umožňuje oprávněnému uživateli natankovat auto pomocí bezkontaktního tankovacího systému (PU13), ale před zahájením procesu si uživatel může přečíst pokyny pro tankování, pro zahájení procesu tankování musí uživatel identifikovat čerpací stanici v systému naskenováním QR kódu nebo nastavením kódu ručně, k tankování pro osoby se zdravotním postižením může uživatel v systému označit, že potřebuje pomoc personálu. Abyste mohli zahájit tankování, musíte vybrat auto, které bude tankováno, poté vybrat typ benzínu, jeho množství, způsob platby a na konci procesu tankování může uživatel ohodnotit tankování zanecháním komentáře a / nebo hodnocení od 1 do 5 hvězdiček, kde 1 - velmi špatné, 5 - vynikající.



Obrázek 3.5. Diagram případu použití. Tankování.

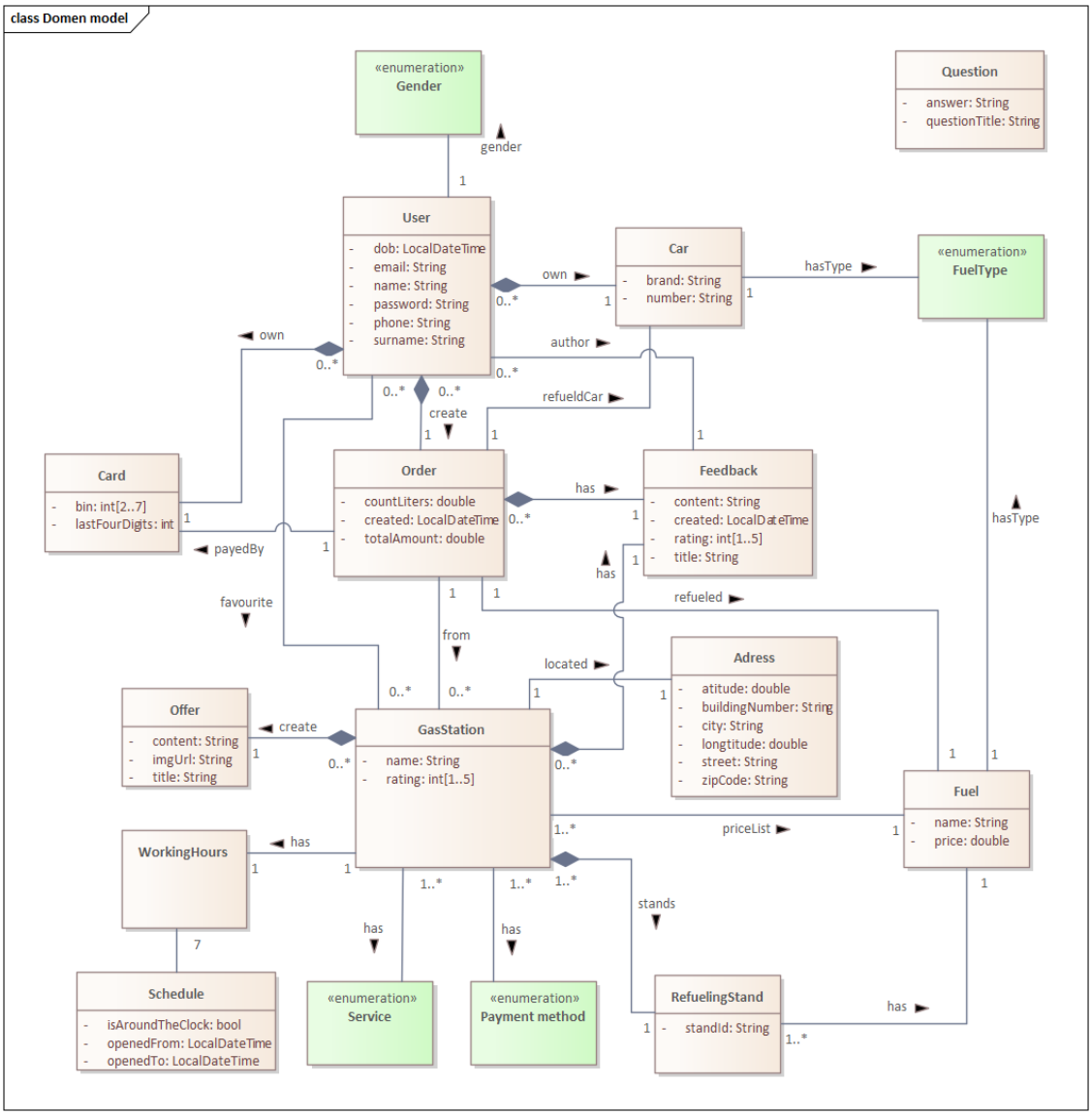
Kapitola 4
Návrh

V této části bude popsány navrh architektury aplikace, stejně jako jednotlivé části aplikace - klient a server a definice entit aplikace. Architektura aplikace je typu klient-server, na kterém se podílejí dva faktory KLIENT a SERVER. Server je mozkem této aplikace, závisí na něm business logika, sprava zdroje, starost o bezpečnost. Pracuje s požadavky HTTP. Klient zase vytváří požadavky na server a také od něj přijímá odpovědi a poté s těmito daty pracuje. Klientská strana poskytuje uživatelské rozhraní, které jim umožňuje zadávat tyto požadavky na server. Klient často také provádí práci s daty (výpočty, třídění atd.) za účelem snížení zátěže serveru.
Komponenty aplikace klient-server:

- **Pracovní stanice.** Pracovní stanice se také nazývají uživatelské zařízení, jako je telefon, tablet a počítač. Používají se k odesílání požadavků na server.
- **Servery.** Obvykle se jedná o vysokorychlostní hardware, který je centrem pro ukládání dat, programů a zásad. Musí být schopni zpracovat velké množství požadavků, které přicházejí z různých pracovních stanic. Servery také fungují jako poštovní servery, databáze, souborové servery atd.
- **Síťová zařízení.** Síťová zařízení jsou nástroje, které pomáhají propojit dvě klíčové komponenty v architektuře klienta a serveru, jednou z nich jsou rozbočovače, směrovače, opakovače a mosty.

4.1 Návrh doménového modelu

Hlavním subjektem je Gas Station, který obsahuje klíčové údaje, se kterými uživatel pracuje: Address, Offers, které čerpací stanice inzeruje, enumeration Service a Payment Method, díky kterým může uživatel filtrovat požadavky a najít vhodnou čerpací stanici. Každá Gas Station dále obsahuje ceny za ceny pohonných hmot (Fuel) a Refueling Stands, které obsahují i řadu subjektů Fuel, pro tankování u určitých výdejních stojanů. Neméně důležitou roli hraje entita Order, která shromažďuje veškeré informace o uskutečněných transakcích při platbě za tankování prostřednictvím aplikace. Subjekt User může mít v systému mnoho registrovaných entit Card a Car pro správné tankování a placení. Car a Fuel se vztahují k určitému Fuel Type, aby systém kontroloval správnost tankování a neumožňoval tankování automobilů špatným benzínem.



Obrázek 4.1. Domenový model.

4.2 Návrh použitých technologií

4.2.1 Frontend

K vývoji vnějšího pláště aplikace byl použit velmi mladý framework od Googlu – Flutter[1]. Jedná se o jedno z multiplatformních aplikačních řešení. Díky tomuto frameworku je možné snadno spustit produkt MVP na různých platformách – ať už jde o telefon, webovou stránku. Kromě Flutter existuje také React Native[2], který poskytuje podobné funkce.

Vybral jsem si Flatter, protože se velmi snadno udržuje a nevyžaduje duplikaci práce při implementaci jedné aplikace na více platformách.

Místo XML layout souborů jsou použity widgety, které lze vytvořit z existujících a nakonfigurovat je přímo v kódu.

[1] https://flutter.dev/
[2] https://reactnative.dev/

K zápisu kódu se používá jazyk Dart[3], který se příliš neliší od JavaScriptu. Proces kompilace - Flutter je kompilován do nativního kódu pro každou platformu. Za ním používá Skia[4] jako svůj grafický engine.

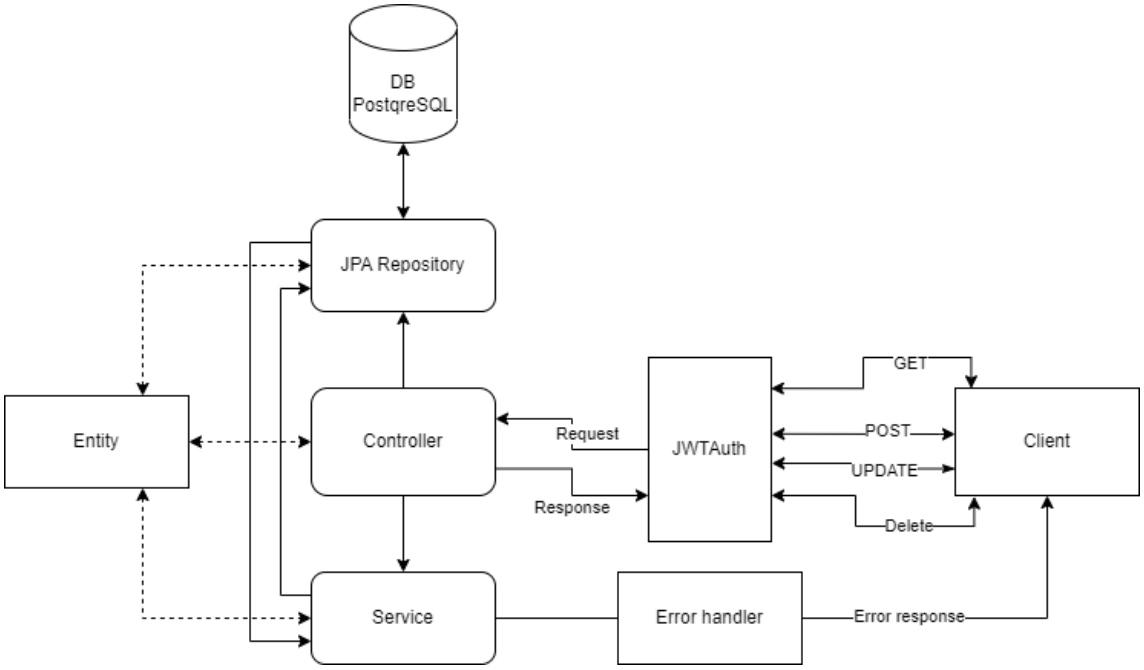
4.2.2 Backend

REST API na Kotlinu a Spring Boot[5].
Jedná se o velmi pohodlný programovací jazyk, který se projevuje nejen v syntaxi, ale i v některých momentech, což v Javě chybělo, například podpora přiřazování NULL proměnných do proměnných, což je častý problém v Javě, kdy program vyhodí chybu NULLPointerException. Jednou ze specifických výhod je snadná interakce s Javou. Taky dobrá podpora od Intellij Idea.
Budu používat Kotlin s frameworkem Spring Boot. Spring je jedním z nejpopulárnějších frameworků pro vývoj enterprise aplikací. Jednou z vlastností tohoto frameworku je použití vzoru DI, který pomáhá zjednodušit implementaci potřebné funkcionality pro aplikaci a také umožňuje vytvářet volně spojené třídy, což je činí univerzálnějšími.
Spring Boot usnadňuje používání frameworku Spring a zároveň využívá framework Spring jako jeho základ, příkladem je funkce automatické konfigurace, která nám ušetří kód a čas.
Použití této technologie má několik výhod: méně kódu, vytváření samostatných aplikací, snadné nastavení a správa.

4.3 Návrh technologií serveru

V serverové části aplikace jsem použil framework Spring Boot, ve kterém je komunikace mezi klientem a serverem realizována pomocí rozhraní REST API, které slouží k bezpečné výměně informací na internetu pomocí protokolu HTTP. Přenos dat pomocí protokolů HTTP se obvykle provádí prostřednictvím připojení TCP / IP a data jsou ve formátu XML nebo JSON. Moje aplikace pracuje s formátem JSON. Pro zajištění bezpečnosti přenášených dat je využíván zabezpečený protokol HTTPS s certifikátem SSL. Framework Spring Boot má vícevrstvou architekturu, který umožňuje vyvijet tu každou vrstvu zvlášť od ostatních, což značně zjednodušuje práci a testování aplikace. Na obrazku 4.2 znazornena výsledná schéma architektury backendu aplikace.

[3] https://dart.dev/
[4] https://levelup.gitconnected.com/flutters-skia-engine-takes-cross-platform-app-development-to-a-new-level-85cc5f92ca9b
[5] https://spring.io/projects/spring-boot



Obrázek 4.2. Výsledná schéma architektury backendu aplikace.

4.4 Návrh technologií klientu

Aplikace je realizována pomocí frameworku Flutter v jazyce Dar, který je určen k vytváření aplikací pro různé platformy. S ním můžete implementovat aplikace pro Android, IOS, Windows, macOS, Linux a webové aplikace. Hlavní část Flutteru byla implementována pomocí jazyka C++, přizpůsobena grafické knihovně Google Skia a také spolupracuje s SDK pro každou z platforem, jako je Android nebo iOS. Podle statistik[6] se Flutter řadí na šesté místo v žebříčku nejlepších frameworků. Kód se téměř výhradně skládá z widgetů, které jsou rozšířeny o animace a rozpoznávání gest.

Zvláštnosti:

- Tato platforma nepoužívá Java Script. Místo toho se Dart používá jako programovací jazyk, který lze z hlediska rychlosti kompilace srovnat s jazyky jako Objective-C, Swift, Java.
- Nepoužívá nativní komponenty. Celé rozhraní vykresluje sám.
- Konstrukce uživatelského rozhraní je realizována pomocí deklarativního přístupu. Aby se zvýšila rychlost rozhraní, widgety se kreslí podle potřeby, jinými slovy, až když se v nich něco změní.

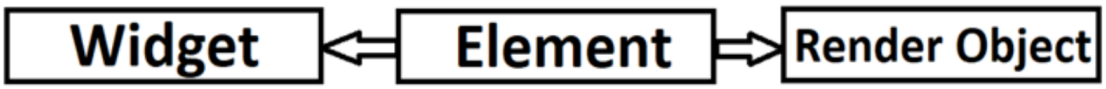
Jak tato platforma funguje.

Oficiální web uvádí, že „Všechno ve Flutteru jsou widgety“[7]. Samotné widgety jsou však popisy nějakého Elementu. Widget je ústřední třídou ve Flutteru a zároveň kolem něj existuje mnoho dalších komponent. Jeden widget může být zahrnut do stromu dalších widgetů.

Pro změnu konfigurace aplikace se používá entita State, která popisuje aktuální stav widgetů.

[6] https://www.statista.com/statistics/793840/worldwide-developer-survey-most-used-frameworks/
[7] https://flutter.dev/learn

Render Object je zodpovědný za implementaci základních kreslicích a polohovacích protokolů.
Grafické znázornění principu fungování platformy je na obrázku 4.3.



Obrázek 4.3. Grafické znázornění principu fungování platformy Flutter.

Widget (Konfigurace)

- Prohlášení
- Skladování majetku

Element (Řízení)

- Vytvoření hierarchie
- Řízení vztahů

Render Object (Obraz)

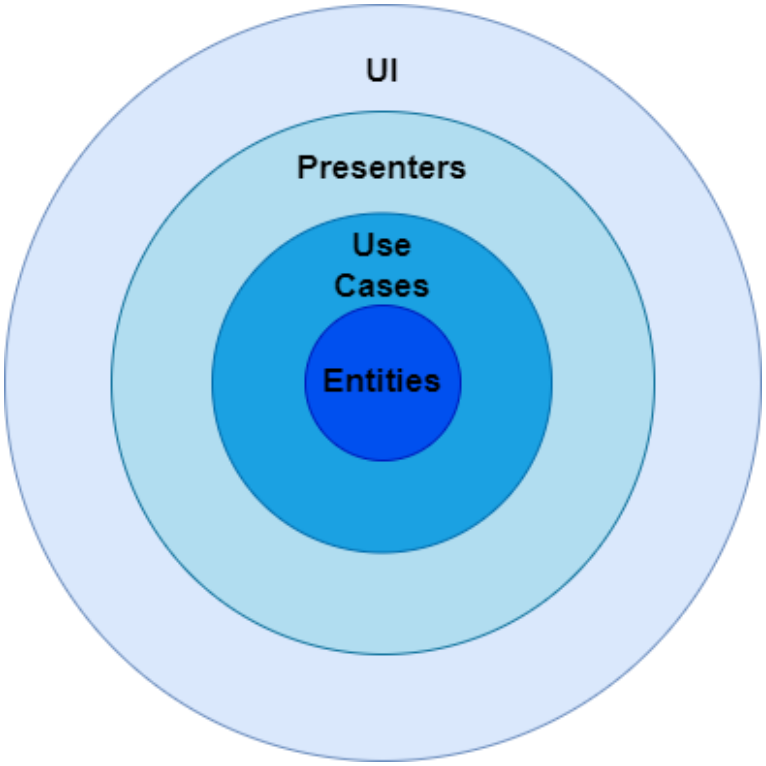
- Vykreslování
- Účtování rozměrů a omezení

4.5 Architektura frontendu aplikace

Technologie Flutter používá čistou architekturu, která znamená:

- **Nezávislost na frameworku.** Architektura je nezávislá na konkrétní knihovně, což umožňuje, aby byl framework použit jako doplněk k systému, aniž by to podléhalo omezením frameworku.
- **Testování.** Business logiku lze testovat bez externích komponent (databáze, uživatelské rozhraní atd.).
- **Nezávislost na UI.** Uživatelské rozhraní lze snadno změnit beze změny zbytku systému.
- **Databázově nezávislá.** Business logika je nezávislá na typu databáze.
- **Nezávislost na externích službách.**

Na obrazku 4.4 znazornena výsledná schéma architektury frontendu aplikace.



Obrázek 4.4. Výsledná schéma architektury frontendu aplikace.

Aplikace se skládá ze čtyř vrstev:

- **data** – datová vrstva. Tato vrstva popisuje, jak pracovat s externími rozhraními API.
- **domain** - vrstva obchodní logiky aplikace.
- **internal** - do této vrstvy se vkládají závislosti.
- **presentation** - prezentační vrstva, která popisuje UI aplikace.

Kapitola 5
Prototyp (Design)

Tato část práce obsahuje tři fáze vývoje aplikace:

- LO-FI prototyp (Příloha)
- HI-FI prototyp implementovaný v programu Figma[1]
- konečný pohled na aplikaci na smartphonu (Příloha)

Celá aplikace je rozdělena na části (stránky, sekce), podle kterých byl proveden návrh a následný vývoj aplikace. Každá část má také stručný popis toho, za co je zodpovědná a jaké má vlastnosti.
Byl zvolen minimalistický design aplikace, byly dodrženy velikosti a barvy textů, tlačítek, bloků ve stejném stylu. Všechny rozměry ve zdrojové aplikaci se automaticky vypočítají pomocí velikosti zařízení, přes které se aplikace používá pro responzivní rozhraní.

5.1 Stránka autorizace a registrace

Nepřihlášený uživatel má na výběr ze 3 možností:

- **Autorizace.** Uživatel si nastaví své údaje (e-mail, heslo)
- **Registrace.** Uživatel vyplní registrační formulář. Registrace bude dokončena zadáním ověřovacího kódu zaslaného na e-mail uživatele.
- **Přihlášení bez oprávnění.**

Obrázek 5.1. Sekce autorizace a registrace (HI-FI prototype).

[1] https://www.figma.com/file/AGdpSe5Uq5SVhZxvUYWU1V/Ruben?node-id=17%3A642&t=gLyBF-dUnqmLUjm9b-1

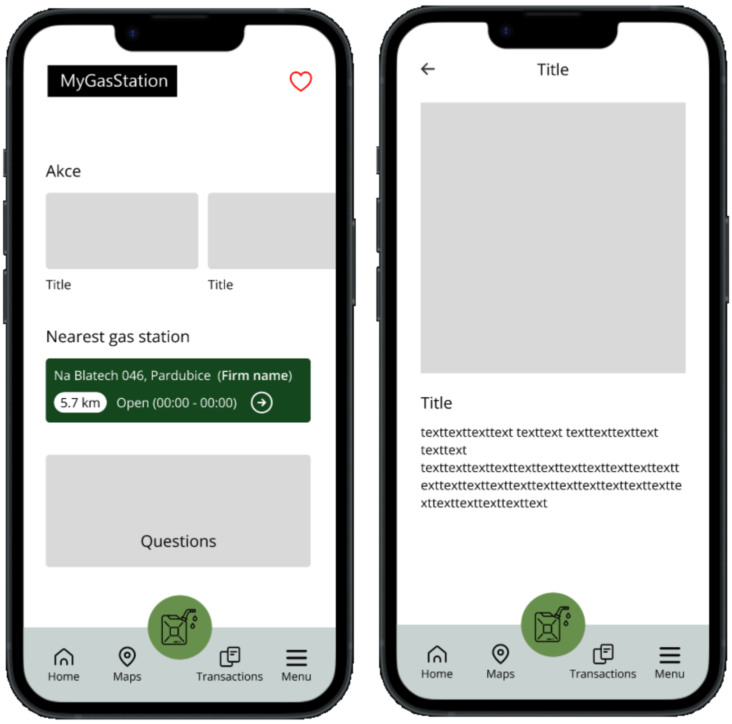
5.2 Hlavní stránka (Domovská stránka)

Tato stránka obsahuje několik sekcí:

- **Akce.** Malý seznam akčních nabídek různých čerpacích stanic. Je možné zobrazit podrobné informace o akci
- **Nejbližší čerpací stanice.** Pokud uživatel povolil přístup ke své poloze, pak tato sekce uživateli nabídne nejbližší čerpací stanici
- **Podpora.** Uživatel se může seznámit s již vyřešenými dotazy/potíže

Spodní panel je rozdělen do několika hlavních částí aplikace:

- **Domovská stránka.**
- **Mapa.** Plná práce s mapami, pomoc s hledáním vhodné čerpací stanice
- **Bezkontaktní tankování.**
- **Transakce.** Historie všech operací
- **Menu.** Nastavení atd

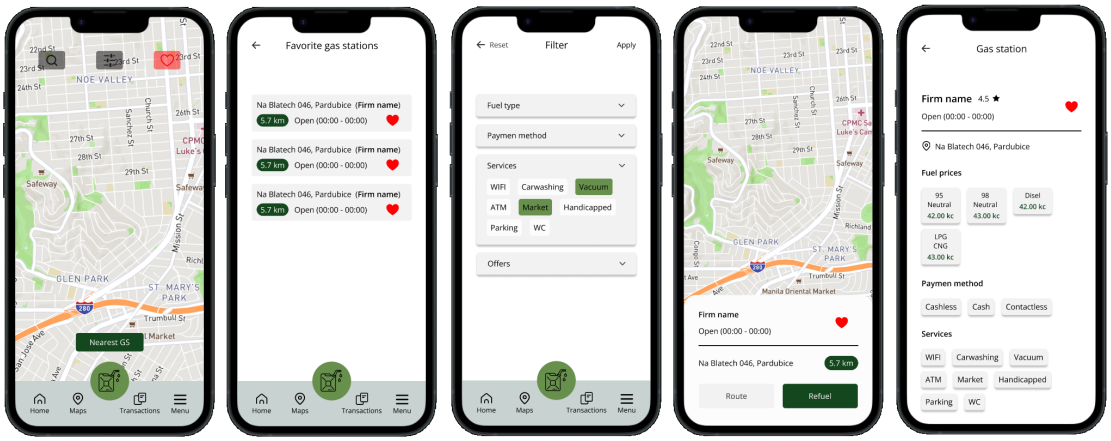


Obrázek 5.2. Sekce home a akce (HI-FI prototype).

5.3 Sekce Maps

Tato část vyzve uživatele, aby našel čerpací stanici podle svých preferencí:

- **Vyhledávání podle jména (vyhledávač).**
- **Filtr.** Uživatel bude moci filtrovat vyhledávání podle různých kategorií
- **Seznam oblíbených čerpacích stanic.**
- **Zobrazení podrobných informací o čerpací stanici.**



Obrázek 5.3. Sekce maps (HI-FI prototype).

5.4 Sekce tankování

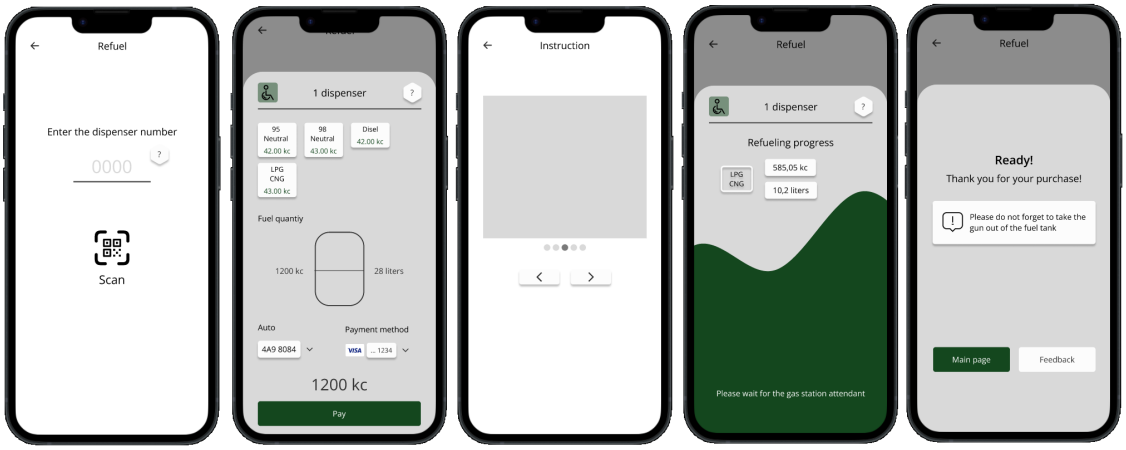
Po přechodu na stránku tankování bude mít uživatel tři možnosti pro události:

- Zobrazit pokyny
- Zadat kód z čerpací stanice ručně
- Naskenovat QR kód umístěný na čerpací stanici

Poté se otevře výsuvné menu s nastavením tankování, ve kterém budete muset vybrat:

- Benzín
- Množství paliva
- Tankovací vozidlo
- Způsob platby

Následovat bude proces tankování, který bude zakončen děkovnou zprávou a možností zhodnocení tankování.

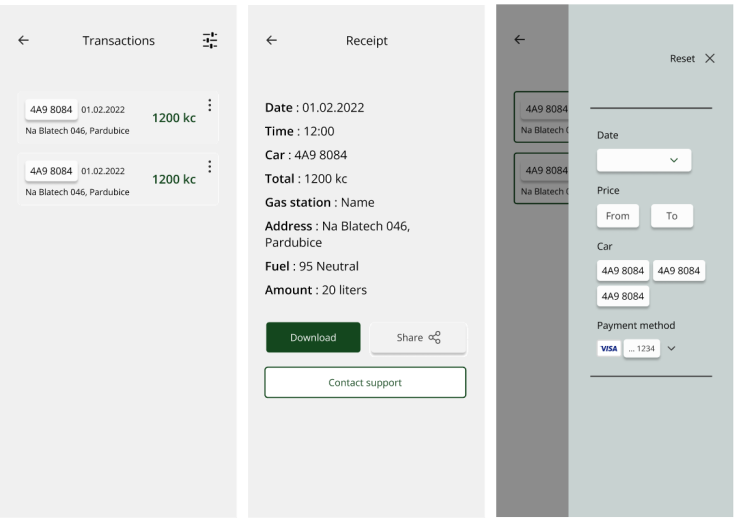


Obrázek 5.4. Sekce tankování(HI-FI prototype).

5.5 Sekce provedených transakcí

Tato sekce umožňuje uživateli zobrazit historii transakcí. První stránka zobrazuje seznam transakcí, které již byly dokončeny. Pro podrobné informace je potřeba kliknout

na konkrétní operaci, po které se uživateli zobrazí stránka s podrobnými informacemi. Uživatel má také přístup k funkcím odeslání a uložení dokumentu ve formátu PDF.



Obrázek 5.5. Sekce transakce (HI-FI prototype).

5.6 Sekce Menu

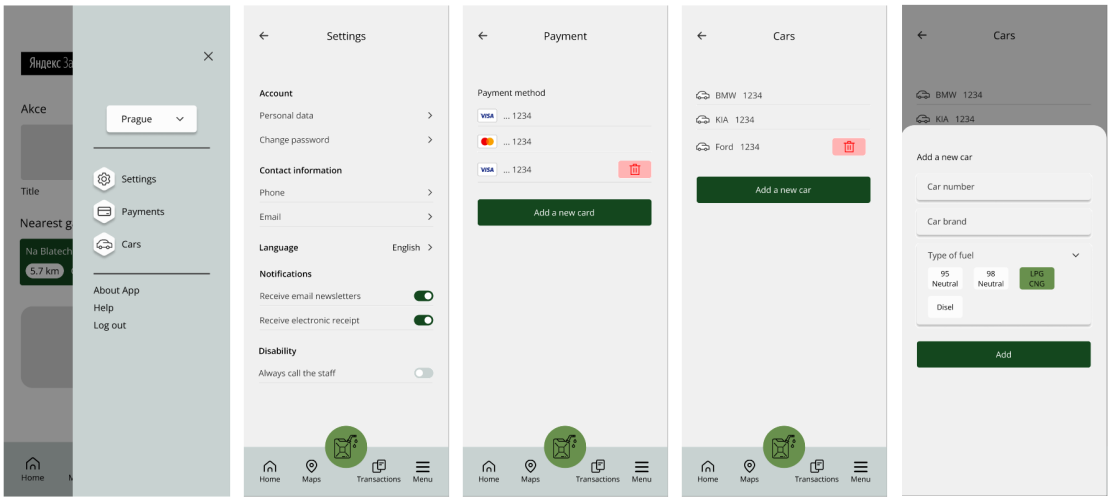
V části Menu jsou uživateli k dispozici následující funkce:

- Změna města
- Nastavení
- Sekce platebních metod
- Automobilová sekce
- Zobrazit informace o aplikaci
- Pomoc
- Odhlásit se

V sekci nastavení může uživatel:

- Zobrazit podrobnosti profilu
- Změnit heslo
- Změnit telefonní číslo
- Změnit e-mail
- Změnit jazyk aplikace
- Spravovat oznámení
- Nastavit režim pro zdravotně postižené na výchozí

V sekci platebních metod je možné vytvořit nebo smazat bankovní karty k platbě za tankování.
V sekci auta je možné vytvářet nebo mazat informace o autech.



Obrázek 5.6. Sekce menu (HI-FI prototype).

5.7 Sekce Dotazů

V této části se uživatelé mohou seznámit s popisem problémů, které mohou při používání aplikace nastat.

Obrázek 5.7. Sekce dotazů (HI-FI prototype).

Kapitola 6
Implementace

Tato část popisuje postupy pro implementaci klientské a serverové části aplikace a také příklady kódu, které byly použity v procesu implementace.

6.1 Použitá vývojová prostředí a software

- K vývoji serverové části byla použita IntelliJ IDEA[1].
- K vývoji klientské části byla použita Visual Studio Code[2].
- Docker[3] pro spouštění aplikací v izolovaných kontejnerech na jediném serveru.
- Systém pro správu verzí aplikací GIT[4].

6.2 Implementace serverové části aplikace

Serverová část byla implementována pomocí frameworku Spring Boot. Pro implementaci testování aplikací byl použit Docker, který umožňuje spouštět aplikaci v různých kontejnerech na stejném serveru. Po úspěšném testování byla aplikace hostována na Heroku.com[5] (cloudové platformě PaaS[6]), která podporuje řadu programovacích jazyků (včetně Kotlin).
Databáze byla vytvořena pomocí specifikace Java Persistence API (JPA) a zároveň implementuje koncept ORM[7].

6.2.1 Datová vrstva (Data Access Layer)

DAO se používá pro interakci aplikace s uživatelskými/systémovými daty. Aplikace Spring Boot používá rozhraní JPA. Pro implementaci specifikace JPA bylo použito ORM konkrétně Hibernate.
Výhoda Hibernate spočívá v poskytování funkcí, jako je ukládání do mezipaměti, transakce, dotazy, které jsou nezávislé na databázi. Knihovna Hibernate kromě propojení tříd Java s databázovými tabulkami poskytuje možnost automatického generování a aktualizace tabulek, výrazně zkracuje dobu vývoje tím, že nás jako programátory šetří ručním psaním SQL a JDBC kódu - automatizuje generování SQL-dotazů.

[1] https://www.jetbrains.com/idea/
[2] https://code.visualstudio.com/
[3] https://www.docker.com/
[4] https://about.gitlab.com/
[5] https://www.heroku.com/
[6] https://en.wikipedia.org/wiki/Platform_as_a_service
[7] https://en.wikipedia.org/wiki/Object–relational_mapping

6.2.2 Aplikační vrstva (Business Layer)

Veškerá business logika, která je přítomna v aplikaci, je obsažena v Business Layer. Tato vrstva kombinuje datovou vrstvu a prezentační vrstvu a je zodpovědná za správný přenos dat mezi těmito dvěma vrstvami.
Skládá se z objektů service, které implementují všechny datové operace (ověření, třídění, počítání, vyhledávání atd.), které byly poskytnuty jako vstup nebo z uložených dat.
Po obdržení požadavku business vrstva zpracuje požadavek, počínaje kontrolou správnosti požadavku (správnosti poskytnutých vstupních dat), poté shromáždí paket odpovědi s daty očekávanými na výstupu zpracování tohoto požadavku a odešle to do prezentační vrstvy, která zase odešle odpověď klientovi.

6.2.3 Prezentační vrstva (Presentation Layer)

Tato vrstva je počátečním a koncovým bodem životního cyklu jednoho požadavku klienta na server. Poté, co klient odeslal svůj požadavek na server, je to právě tato vrstva, která jej přijme a distribuuje na konkrétní metodu pomocí adresy URL. Všechny požadavky přicházejí ve formátu JSON.
Metody, které obdrží požadavek, zkontrolují soulad s právy klienta (pokud nebyla dodržena role uživatele/klienta, odešle jako odpověď chybu). Po úspěšném přijetí požadavku jsou data ve formátu JSON přenesena do objektů entity a odeslána ke zpracování do business vrstvy a vrácena v objektu/objektech entity specifických pro požadavek, načež jsou vrácena zpět klientovi.

```
@PreAuthorize("hasRole('USER')")
@GetMapping("/")
fun getAllCards(fromUser: HttpServletRequest):
          ResponseEntity List CardAssembler.CardDto {
   val user = userRepository.findByEmail(fromUser.userPrincipal.name)
   .orElse(null)
   // implementace controlleru ...
}
```

Výše uvedený příklad ukazuje příklad implementace ochrany přístupu k získání seznamu všech uložených uživatelských karet. Tato kontrola má dvě fáze:

- fromUser: HttpServletRequest - předán kontroleru jako parametr. Určuje, kdo odešle požadavek pomocí tokenu, který je předán v hlavičce požadavku (Authorization: Bearer token).
- @PreAuthorize(`hasRole('USER')`) - určuje roli osoby, která požadavek odeslal.

6.2.4 Seznam API pro komunikaci se serverem

Aplikace obsahuje implementaci API jak pro běžné uživatele, kteří budou produkt používat k zamýšlenému účelu, tak pro ty, kteří budou spravovat obecné informace aplikace (dotazy), vytvářet a upravovat informace o čerpacích stanicích. Každý požadavek bude rozdělen do sekcí, které odkazují na manipulace s určitými entitami a obsahují roli uživatele(Role - [‘USER’, ‘ADMIN’, ‘MANAGER’]), který bude mít k tomuto požadavku přístup.

UserController

- **[-] POST user/reg** - Přijímá požadavek na registraci nového uživatele. Vyžaduje uživatelská data (jméno, příjmení, email, telefonní číslo, datum narození, heslo, pohlaví).

Pokud uživatel nastavil všechny parametry správně, načte data nového uživatele do databáze a odešle data uživatele jako odpověď, v opačném případě odešle chybu (kód chyby a popis).

- **[-] POST user/login** - Přijímá žádost o autorizaci uživatele. Vyžaduje uživatelská data (e-mail a heslo). Pokud uživatel nastavil všechny parametry správně, odešle jako odpověď uživatelská data a token pro následné požadavky pro oprávněné uživatele, v opačném případě odešle chybu (kód chyby a popis).
- **[-] POST user/verify_account** - Přijímá požadavek na potvrzení emailu uživatelem pomocí kódu, který mu byl zaslán e-mailem. Pokud se uživatel zaregistroval, ale nepotvrdil svůj email, položka autorizace odešle chybu. Vyžaduje uživatelská data (e-mail, heslo a potvrzovací kód). Pokud uživatel nastavil všechny parametry správně, odešle jako odpověď uživatelská data a token pro následné požadavky pro oprávněné uživatele, v opačném případě odešle chybu (kód chyby a popis).
- **[-] POST user/email** - Přijímá žádost o zaslání ověřovacího kódu e-mailem pro resetování hesla. Vyžaduje uživatelská data (email). Pokud uživatel nastavil všechny parametry správně, odešle na email uživatele kód pro obnovení hesla, v opačném případě odešle chybu (kód chyby a popis).
- **[-] POST user/reset_pas** - Přijímá požadavek na obnovení hesla. Vyžaduje uživatelská data (e-mail, heslo, opětovné zadání hesla, kód pro obnovení hesla). Pokud uživatel nastavil všechny parametry správně, aktualizuje nové heslo v databázi, jinak odešle chybu (kód chyby a popis).
- **[USER] POST user/reset_pas_auth** - Obdrží požadavek na aktualizaci hesla. Vyžaduje uživatelská data (staré heslo, nové heslo, opětovné zadání nového hesla). Pokud uživatel nastavil správně všechny parametry a byl autorizován, aktualizuje nové heslo v databázi, jinak odešle chybu (kód chyby a popis).
- **[USER] POST user/update_phone** - Obdrží požadavek na aktualizaci telefonního čísla. Vyžaduje uživatelská data (nové telefonní číslo). Pokud uživatel správně nastavil všechny parametry a byl autorizován, aktualizuje nové telefonní číslo v databázi, jinak odešle chybu (kód chyby a popis).
- **[USER] POST user/update_email_req** - Přijímá požadavek na zaslání nového potvrzovacího kódu e-mailem.Vyžaduje uživatelská data (nová e-mailová adresa). Pokud uživatel nastavil správně všechny parametry a byl autorizován, odešle uživateli na nový email potvrzovací kód, v opačném případě odešle chybu (kód chyby a popis).
- **[USER] POST user/update_email** - Přijímá požadavek na aktualizaci e-mailu. Vyžaduje uživatelská data (nový e-mail, potvrzovací kód). Pokud uživatel nastavil správně všechny parametry a byl autorizován, aktualizuje nový email v databázi, jinak odešle chybu (kód chyby a popis).
- **[ADMIN] POST user/manager** - Přijímá požadavek na registraci nového manažera. Vyžaduje nové údaje správce (jméno, příjmení, email, telefonní číslo, datum narození, heslo, pohlaví). Pokud administrátor nastavil správně všechny parametry a byl autorizován, uloží data nového manažera do databáze, jinak odešle chybu (kód chyby a popis).

QuestionController

- **[-] GET question/** - Odešle seznam všech otázek entity Otázka (název otázky, odpověď).
- **[ADMIN] POST question/add** - Přijímá požadavek na vytvoření nové otázky. Jsou vyžadována data nové otázky (název, odpověď). Pokud administrátor správně nastavil všechny parametry a byl autorizován, uloží nový dotaz do databáze, jinak odešle chybu (kód chyby a popis).

OrderController

- **[USER] GET order/** - Odešle seznam všech transakcí entity Order, oprávněného uživatele.
- **[USER] GET order/filter** - Odešle filtrovaný seznam transakcí subjektu Order, oprávněného uživatele podle parametrů (datum, cena, natankované auto, bankovní karta).

CarController

- **[USER] GET car/** - Odešle seznam všech vozů entity Car (značka, číslo, typ benzínu) oprávněnému uživateli.
- **[USER] POST car/** - Přijímá požadavek na vytvoření nového vozidla. Jsou vyžadována údaje o vozidle (značka, číslo, typ benzínu). Pokud uživatel správně nastavil všechny parametry a prošel autorizací, uloží nový vůz do databáze, v opačném případě odešle chybu (kód chyby a popis).
- **[USER] DELETE car/car-id** - Přijímá žádost o odstranění vozidla. K odstranění potřebuje ID vozidla. Pokud uživatel správně nastavil všechny parametry a prošel autorizací, je vozidlo v databázi označeno jako smazane, jinak odešle chybu (kód chyby a popis).

CardController

- **[USER] GET card/** - Odešle seznam všech bankovních karet entity Card (poslední 4 čísla, bin) oprávněnému uživateli.
- **[USER] POST card/** - Přijímá požadavek na vytvoření nové karty. Jsou vyžadována údaje o kartě (číslo, datum expirace, cvv kód). Pokud uživatel správně nastavil všechny parametry a prošel autorizací, uloží novou kartu do databáze (jen poslední 4 čísla), v opačném případě odešle chybu (kód chyby a popis).
- **[USER] DELETE card/card-id** - Přijímá žádost o odstranění karty. K odstranění potřebuje ID karty. Pokud uživatel správně nastavil všechny parametry a prošel autorizací, je kartya v databázi označena jako smazana, jinak odešle chybu (kód chyby a popis).

FeedbackController

- **[USER] GET feedback/gs/gs_id** - Odesílá data zpětné vazby zanechané oprávněným uživatelem konkrétní čerpací stanici podle ID.
- **[USER] GET feedback/order/order_id** - Odesílá data zpětné vazby zanechaná oprávněným uživatelem pro konkrétní objednávku podle ID.
- **[USER] POST feedback/order/order_id** - Přijímá požadavek na vytvoření nové recenze. Požaduje data zpětné vazby (komentář, hodnocení). Pokud uživatel správně nastavil všechny parametry a prošel autorizací, je nová recenze uložena do databáze a přiřazena ke konkrétní objednávce, v opačném případě odešle chybu (kód chyby a popis).
- **[USER] POST feedback/gs/gs_id** - Přijímá požadavek na vytvoření nové recenze. Požaduje data zpětné vazby (komentář, hodnocení). Pokud uživatel správně nastavil všechny parametry a prošel autorizací, je nová recenze uložena do databáze a přiřazena ke konkrétní čerpací stanici, v opačném případě odešle chybu (kód chyby a popis).

GasStationController

- **[-] GET gas_station/city** - Odesílá zjednodušená data čerpací stanice pro konkrétní město.
- **[-] GET gas_station/info/gs_id** - Zasílá kompletní data čerpací stanice podle ID.

- **[USER] GET gas_station/favorite** - Zasílá zjednodušená data všech oblíbených čerpacích stanic oprávněnému uživateli.
- **[USER] POST gas_station/favorite/gs_id** - Přijímá požadavek na změnu vztahu čerpací stanice. Pokud je uživatel autorizován a má správně nastavené ID čerpací stanice, odebere ji ze seznamu nebo přidá do seznamu oblíbených čerpacích stanic, v opačném případě odešle chybu (kód chyby a popis).
- **[-] GET gas_station/filter** - Odesílá filtrovaný seznam čerpacích stanic podle kritérií (poskytované služby, platební metody, druhy benzínu poskytované čerpací stanicí).
- **[-] GET gas_station/search/req** - Odešle seznam čerpacích stanic na vyhledávací dotaz.
- **[-] GET gas_station/offers** - Odešle seznam 5 akčních nabídek náhodných čerpacích stanic.
- **[-] GET gas_station/nearest** - Odešle kompletní informace na nejbližší čerpací stanici podle souřadnic uživatele.

ManagerController

- **[MANAGER] POST manager/gs** - Přijme požadavek na vytvoření nové čerpací stanice v systému. Pokud je manažer oprávněn a správně zadal všechny potřebné údaje, vytvoří se v databázi nový záznam čerpací stanice, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/offer/gs_id** - Přijímá požadavek na vytvoření nové akční nabídky od čerpací stanice. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, vytvoří se záznam o nové akční nabídce v databázi, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] DELETE manager/gs/offer/gs_id** - Přijímá žádost o odstranění akční nabídky z čerpací stanice. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, bude záznam akční nabídky v databázi označen jako smazaný, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/schedule-add/gs_id** - Přijme požadavek na vytvoření rozvrhu pracovní doby pro čerpací stanici. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, bude harmonogram prací přidán na konkrétní čerpací stanici v databázi, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/schedule/gs_id** - Přijme požadavek na změnu rozvrhu pracovní doby čerpací stanice na konkrétní den. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje - pracovní plán pro určitou čerpací stanici na určitý den se v databázi změní, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/services/gs_id** - Přijme požadavek na změnu seznamu služeb pro čerpací stanici. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, seznam služeb poskytovaných konkrétní čerpací stanicí se v databázi změní, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/rs/gs_id** - Přijímá požadavek na vytvoření nového výdejního stojanu pro čerpací stanici. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, bude k určité čerpací stanici v databázi přiřazen výdejní stojan s číslem, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/fuel/gs_id** - Přijme požadavek na vytvoření záznamu informací o palivu pro čerpací stanici. Pokud je manažer oprávněn, je

vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, informace o PHM poskytnuté s cenou budou přiřazeny do databáze pro určitou čerpací stanici, v opačném případě odešle chyba (kód chyby a popis).

- **[MANAGER] POST manager/gs/fuel-rs/gs_id** - Přijímá požadavek na spojení výdejního stojanu a paliva poskytované čerpací stanicí. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, bude palivo přiřazeno k určitému výdejnímu stojanu v databázi, jinak odešle chybu (kód chyby a popis).
- **[MANAGER] POST manager/gs/fuel-update/gs_id** - Přijímá požadavek na aktualizaci ceny paliva poskytovaný čerpací stanicí. Pokud je manažer oprávněn, je vlastníkem čerpací stanice v systému a správně zadal všechny potřebné údaje, cena PHM pro určitou čerpací stanici se v databázi změní, jinak odešle chybu (kód chyby a popis).

6.2.5 Zajímavé služby a knihovny

Zpracování chyb (Controller Error Handler)

Ošetření chyb v jakémkoli softwaru je jednou z nejdůležitějších fází vývoje, bez které se neobejde ani jeden kvalitní produkt související s IT. Při jakémkoli požadavku na REST API může dojít k chybě, ať už z vývojářských chyb nebo z nesprávných dat, která byla odeslána ke zpracování. Bez ošetření chyb bude velmi obtížné určit příčinu problému a také nepochopit uživatele důvodu pro jejich opravu. Pro pohodlí byla vytvořena služba (ControllerErrorHandler), která implementuje zpracování a zveřejňování chyb na jednom místě. Níže je ukázka implementace error handleru (tato metoda byla implementována v jedné z mých semestrálních prací pro předmět B6B36EAR, jehož kód je umístěn na GitLabu[8]):

```
@ExceptionHandler(ControllerException::class)
fun conversationIdInvalidException(controllerException:
    ControllerException): ResponseEntity (ErrorResponse) {
   val res = ErrorResponse(controllerException.code,
    controllerException.message ?: "Empty message")
   val response = ResponseEntity.status(controllerException.status)
    .body(res)
   LOG.error("Code: ${controllerException.code} message:
   ${controllerException.message} status: ${controllerException.status}
   ${response.body.toString()}")
   return response
}
```

Příklad inicializované třídy se specifickou chybou (kód chyby a popis)

```
    // package cz.cvut.fel.mygs.exceptios.AppExceptions.kt
    ...
    class UserAlreadyRegisteredException(
    message: String = "User with this email is already registered!"):
    ControllerException(message,
    ResponseConstants.EMAIL_ALREADY_EXIST.value)
    ...
    // package cz.cvut.fel.mygs.variables.ResponseConstants.kt
```

[8] https://gitlab.fel.cvut.cz/kurbarub/ear_semestralka - uzavřený přístup

```
...
EMAIL_ALREADY_EXIST("USR003"),
...
```

Mail service

Aplikace implementuje kontakt s uživatelem prostřednictvím e-mailové služby knihovny org.springframework.boot:spring-boot-starter-mail. Pro implementaci odesílání zpráv byl použit účet Google. Níže je uveden příklad implementace zasílání zpráv prostřednictvím e-mailu:

```
@Service
class MailService(val emailSender: JavaMailSender) {
   fun send(to: String, subject: String, text: String) {
       emailSender.send(
           SimpleMailMessage().apply {
               setFrom("email@email.cz")
               setTo(to)
               setSubject(subject)
               setText(text)
           }
       )
   }
}
```

Bankovní service

Vzhledem k tomu, že aplikace pracuje s platebními metodami uživatelů (bankovní karty), bylo implementováno „workaround“, který uměle simuluje bankovní službu. Vzhledem k tomu, že aplikace nepodporuje standard PCI DSS, nejsou údaje o kartě uživatelů ukládány do databáze, kromě posledních 4 číslic karty pro orientaci uživatele. Níže je uveden příklad kódu bankovní služby:

```
@Service
class BankService(val cardData: CardData) {
   fun checkData(cardNumber: String,
        cardExpirationDate: LocalDateTime, cardCvv: Int):Boolean {
       return cardData.check(cardNumber, cardExpirationDate, cardCvv)
   }
   fun save(cardNumber: String,
        cardExpirationDate: LocalDateTime, cardCvv: Int){
       cardData.save(cardNumber,cardExpirationDate, cardCvv)
   }
   fun payForRefueling(amount: Double): Boolean{
       return cardData.pay(amount)
   }
}
```

Autorizační service

K implementaci autorizace a následného ověření oprávněných uživatelů byla k odesílání požadavků na server použita knihovna org.springframework.boot:spring-boot-starter-security. Tato knihovna je zodpovědná za získání přístupu ke službě, která pracuje s osobními údaji uživatelů. Každý požadavek, který vyžaduje autorizaci, je zkontrolován a lze jej zohlednit pouze v případě, že se uživatel úspěšně autorizuje, jinak

služba odešle chybu. JwtAuthTokenFilter se velmi snadno používá a řeší mnoho úkolů, jako je udělení tokenu oprávněnému uživateli, kontrola oprávněného uživatele, kontrola rolí, nastavení časového limitu tokenu a tak dal. Níže je uveden příklad kódu JwtProvider, který generuje token a kontroluje jeho pravost (tato metoda byla implementována v jedné z mých semestrálních prací pro předmět B6B36EAR, jehož kód je umístěn na GitLabu[9]):

```
fun generateJwtToken(username: String): String {
   return Jwts.builder()
       .setSubject(username)
       .setIssuedAt(Date())
       .setExpiration(Date((Date()).getTime() + jwtExpiration!! * 1000))
       .signWith(SignatureAlgorithm.HS512, jwtSecret)
       .compact()
}

fun validateJwtToken(authToken: String): Boolean {
   try {
       Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken)
       return true
   } catch (e: SignatureException) {
       logger.error("Invalid JWT signature -> Message: {} ", e)
   } catch (e: MalformedJwtException) {
       logger.error("Invalid JWT token -> Message: {}", e)
   } catch (e: ExpiredJwtException) {
       logger.error("Expired JWT token -> Message: {}", e)
   } catch (e: UnsupportedJwtException) {
       logger.error("Unsupported JWT token -> Message: {}", e)
   } catch (e: IllegalArgumentException) {
       logger.error("JWT claims string is empty -> Message: {}", e)
   }

   return false
}
```

Bing API service

Využití této služby je nutné pro zjištění vzdálenosti uživatele aplikace k čerpací stanici. Vzdálenost mezi dvěma body se následně zobrazí uživateli v aplikaci a také pomůže najít nejbližší. Volba padla na tento způsob určení vzdálenosti, protože je zdarma a dostatečně rychlý k použití.

Web Socket pro tankovácí process

Proces doplňování paliva je realizován pomocí Web Socketu, což je obousměrný komunikační protokol, který umožňuje výměnu dat v reálném čase mezi klientem a serverem. Vzhledem k tomu, že proces tankování auta není okamžitou akcí a zabírá čas, bylo nutné implementovat simulaci procesu tankování, která je pod dohledem serveru a zasílá uživateli aktuální informace o stavu proces tankování. Proces probíhá následujícím způsobem:

Mezi klientem a serverem je vytvořeno spojení „Handshake“, po kterém uživatel odesílá na server všechny potřebné informace pro doplnění paliva; poté, co server úspěšně

[9] https://gitlab.fel.cvut.cz/kurbarub/ear_semestralka - uzavřený přístup

inicializuje všechna data, je uživatel upozorněn, že platba proběhla a proces doplňování paliva bude brzy zahájen; pokud uživatel potřebuje pomoc obsluhy, bude upozorněn, že obsluha již odešla a tankovací proces začne poté, co obsluha vloží tankovací pistoli do nádrže vozidla, v opačném případě server upozorní uživatele, že tankování bylo úspěšně zaplaceno a je nutné vložit pistoli do nádrže, po 3 sekundách bude uživatel upozorněn na zahájení procesu tankování a začnou se měnit ukazatele ceny a počtu litrů; na konci procesu tankování je vytvořena entita objednávky, která bude umístěna u uživatele v sekci `transakce`, uživatel bude upozorněn na ukončení procesu a spojení mezi klientem a serverem bude ukončeno.

6.3 Implementace klientské části aplikace

V této části bude popsán proces tvorby klientské části aplikace, konkrétně způsob navigace na některých hlavních stránkách aplikace, služby, které byly použity k implementaci konkrétní funkcionality, implementace datového úložiště, s pomocí které jsou rozpoznávány stavy, jako je jazyk aplikace, město a autorizační data, a také způsob komunikace mezi klientem a serverem. Pro implementaci klienta byla použita platforma Flutter v jazyce Dart. Nastavení jazyků pro různé platformy, při vytváření projektu bylo zvoleno následující:

- Android language: Kotlin
- IOS language: Swift

6.3.1 Navigace

Aplikace má 5 hlavních sekcí, z nichž každá plní svou vlastní roli, a to: Domovská stránka (standardně se otevře jako první), Mapy, Stránka Tankování, Transakce, Menu. Všechny tyto prvky jsou umístěny na spodním prvku aplikace, tzv. BottomNavigationBar. Protože Flutter podporuje stavové metody, každá z těchto stránek se otevře, když se změní stav aktuální sekce. Níže je uveden příklad kódu, který inicializuje stránky a spravuje jejich stavy:

```
late int _selectedIndex;

static const List⟨Widget⟩ _pages = [
  HomePage(),
  MapsPage(),
  RefuelingPage(),
  TransactionsPage(),
  MenuDrawer(),
];

@override
void initState() {
  _selectedIndex = 0;
  super.initState();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
```

```
      key: _scaffoldKey,
      body: SafeArea(child: _pages[_selectedIndex]),
      bottomNavigationBar: _bottomNavBar(),
      endDrawer: const MenuDrawer(),
      floatingActionButton: _fab(),
      floatingActionButtonLocation:
        FloatingActionButtonLocation.centerDocked,
      resizeToAvoidBottomInset: false,
    );
  }

  void \_onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }
```

Existují však stránky, které nesouvisejí s hlavními sekcemi aplikace a na které se uživatel může pohybovat pomocí speciální třídy Navigátor, která má několik pomocných metod pro přesun na jiné stránky. Níže je uveden příklad kódu pro přesun na jinou stránku pomocí metody pushReplacement():

```
Navigator.of(context).pushReplacement(
          MaterialPageRoute(builder: (context) => const HomePage()));
```

Tento příklad ukazuje způsob, jak přejít na stránku HomePage() bez možnosti návratu na předchozí stránku, jinými slovy, tato metoda neuchovává v paměti historii stránek, které byly před přesunem na novou.

Níže je uveden příklad kódu pro přesun na jinou stránku pomocí metody push():

```
Navigator.of(context)
        .push(MaterialPageRoute(builder: (context) => const HomePage()));
```

Tento příklad ilustruje způsob přechodu na stránku HomePage() s možností návratu na předchozí stránku, jinými slovy, tato metoda ukládá do paměti historii stránek, které byly před přesunem na novou.

Tyto dvě metody jsou k implementaci navigace potřeba různými způsoby. Protože v případě autorizace uživatele bude odeslán do pracovní oblasti aplikace a není třeba mu dávat příležitost vrátit se, aby mu byla znovu dána příležitost k autorizaci.

6.3.2 Implementace datového úložiště

Vzhledem k tomu, že aplikace pracuje přímo s uživatelskými daty a každé spuštění závisí na relevanci těchto dat, konkrétně tokenu pro autorizaci uživatele, jeho umístění, jazyku aplikace, bylo k tomu nutné implementovat způsob ukládání těchto dat pomocí SharedPreferences API. SharedPreferences se snadno používá a ukládá malá data ve formátu klíč–hodnota, ke kterému lze rychle a snadno přistupovat. Pro vytváření a správu dat aplikace byl vytvořen samostatný soubor DataCollector pomocí návrhového vzoru Singleton, který poskytuje jedinou instanci této třídy v celé aplikaci, což umožňuje inicializovat nová nebo již vytvořená data, která byla uložena v paměti telefonu. Prostřednictvím této třídy jsou také požadována data ze serveru.

6.3.3 Komunikace mezi klientem a serverem

Aby aplikace fungovala správně, je nutné poskytnout dobrý způsob zpracování síťových požadavků, protože mohou obsahovat neočekávané výsledky, kterým bude obtížné porozumět. Tato část popisuje, jak aplikace zpracovává požadavky REST API pomocí balíčku Dio.

Dio je HTTP klientem pro jazyk Dart, který podporuje mnoho funkcí, jako je interceptor, globální konfigurace, nahrávání souborů a další. Jeho výhodou je snadná obsluha, která výrazně zjednodušuje a zrychluje proces implementace komunikace mezi klientem a serverem. Níže je uveden příklad inicializace Dio:

```
buildDio() {
  DateTime dateTime = DateTime.now();
  BaseOptions options = BaseOptions(
    baseUrl: apiUrl,
    connectTimeout: 15000,
    receiveTimeout: 15000,
    responseType: ResponseType.json,
  );
  dio = Dio(options);
  options.connectTimeout = 15 * 60 * 1000;
  // set up authorization
  dio.interceptors.add(
      InterceptorsWrapper(onRequest:
          (RequestOptions options, handler) async {
    options.headers['Accept-Language'] = language;
    if (isLoggedIn()) {
      options.headers['Authorization'] = 'Bearer \$token';
    }
    return handler.next(options);
  }));
  dio.interceptors.add(LogInterceptor(responseBody: true));
}
```

Pokud jde o proces komunikace mezi klientem a serverem, níže je uveden příklad registrace nového uživatele:

```
Future⟨Registration⟩ createUser(UserClass user) async {
  Map⟨String, dynamic⟩ postData = user.toRegistrationPostData();

  Response response;

  // perform action & receive response
  try {
    response = await dio.post('user/reg', data: postData);
  } on DioError catch (error) {
    print(error.response?.data['errorMessage']);

    String errCode = error.response?.data['errorCode'];
    String errMessage = error.response?.data['errorMessage'];

    print("ERROR FROM THE SERVER: $errCode, $errMessage");
```

```
      if(errCode == "USR003") return Registration.alreadyregistered;
      if(errCode == "USR002") return Registration.bademail;
      if(errCode == "USR004") return Registration.badphone;
      if(errCode == "USR005") return Registration.baduserdata;
      if(errCode == "USR006") return Registration.shortpass;

      return Registration.unknown;
    } catch (e) {
      print('request failed: \$e');
      return Registration.networkError;
    }

    if (response.statusCode == 200) {
      buildDio();
      print(response.data);
      return Registration.success;
    }

    return Registration.unknown;
  }
```

Z příkladu vidíme proces vytvoření těla požadavku, který bude adresován na adresu „user/reg“, ihned po odeslání požadavku metoda zachytí odpověď ze serveru, která je následně zpracována na chyby/data v případě úspěchu.

6.3.4 Vytváření prvků uživatelského rozhraní

Každý prvek uživatelského rozhraní se skládá z velkého počtu widgetů, které jsou prvky knihovny Flutter a mají stromovou strukturu. Na každý widget lze použít více nastavení, od barvy po chování při kliknutí. Před zahájením hlavního vývoje aplikace byly implementovány všechny často používané prvky (tlačítka, vyskakovací seznamy, pole pro zadávání textu), styly textů, velikosti, barvy, proměnné. Kromě jednoduchých, konstantních prvků bylo často nutné zobrazovat dynamické prvky na základě seznamů dat, například seznam „oblíbených čerpacích stanic“. K implementaci tohoto problému byl použit StreamBuilder, který umožňoval práci s datovými toky přicházejícími ze serveru. Umožňuje sledovat obsazenost potoka, reagovat na jeho plnění, načítání. Níže je uveden příklad vytvoření prvku seznamu čerpacích stanic:

```
body: Padding(
         padding: EdgeInsets.only(top: font22),
         child: StreamBuilder(List(GasStation))(
             stream: dc.obsGS,
             builder: (context, snapshot) {
               List(GasStation) gs = snapshot.data;
               if (gs == null || gs.isEmpty) {
                 return Container();
               } else {
                 return ListView.builder(
                     itemCount: gs.length,
                     itemBuilder: (BuildContext context, int index) {
```

```
                    return _gsTile(gs[index]);
                  });
              }
          }))
```

K implementaci vyskakovacích oken byly použity dva typy widgetů:

- Dialog – Widget pro modální okna, která jsou umístěna kdekoli na obrazovce, nad skutečným obsahem. Sloužily pro systémový dialog s uživatelem, například pro potvrzení smazání prvků. Níže je příklad z kódu, který vytváří prvek modálního okna pro potvrzení odebrání prvku ze seznamu oblíbených čerpacích stanic a jeho vizuálu.

```
Dialog(
  alignment: Alignment.bottomCenter,
  shape:RoundedRectangleBorder(borderRadius:
    BorderRadius.circular(font25)),
  child: Container(
          //content
         )
)
```

- showModalBottomSheet - Dynamický ovládací prvek obsahu. Umožňuje umístit velké množství prvků do sebe, chová se jako samostatná stránka, je umístěn ve spodní části obrazovky.

```
WidgetsBinding.instance.addPostFrameCallback((_) async {
    await showModalBottomSheet(
        enableDrag: true,
        isDismissible: true,
        barrierColor: Colors.transparent,
        isScrollControlled: true,
        backgroundColor: Colors.transparent,
        context: context,
        builder: (builder) {
          return _sheet();
        });
 });
```

6.3.5 Použité knihovny a služby

Google Maps. Pro zobrazení map v aplikaci byla použita `google_maps_flutter` knihovna, která je dobře integrována pod Flutter. Aby bylo možné mapu zobrazit na obrazovce, nabízí tato knihovna jednoduchý způsob, jak nakreslit google mapu pomocí widgetu - GoogleMap(), ve kterém je možné nastavit výchozí bod kreslení pomocí initialCameraPosition a umístění značek na mapu pomoci parametru marks.
QR Scanner. Protože hlavní myšlenkou aplikace je bezkontaktní tankování pomocí QR kódu, byla použita knihovna `flutter_barcode_scanner`, která umožňuje číst data z QR kódu pomocí kamery. Kromě toho knihovna nabízí funkce dekódování čárových kódů a generování QR a čárových kódů.
PDF dokument. Každá transakce provedená v aplikaci je uložena v databázi a lze ji stáhnout z aplikace jako soubor PDF, pro pohodlí uživatele nahrazují papírové účtenky. Pro tuto funkcionalitu byly použity knihovny pdf a `flutter_full_pdf_viewer`, které

umožňují generovat a prohlížet PDF dokument, resp.
Share . Pro nahrání PDF souboru potvrzení platby se používá `esys_flutter_share` knihovna, která umožňuje posílat dokumenty mezi kontakty.

Kapitola 7
Testování

Neexistují žádné jasné algoritmy akcí pro testování softwaru, vše závisí na typu softwaru a jeho účelu, ale je to nedílná součást vývoje. Testování softwaru je kontrola souladu s očekávanými a skutečnými výsledky programu. Účelem testování je kontrola shody softwaru s uvedenými požadavky, zajištění důvěry kvalitního produktu, aby během provozu nebyly zjištěny neočekávané chyby.

Základní principy testování:

- **Testování ukazuje přítomnost závad.** Testování snižuje pravděpodobnost chyb během provozu, ale nezaručuje jejich nepřítomnost.
- **Vyčerpávající testování není možné.** Vyčerpávající testování všech modulů je fyzicky nemožné, kromě triviálních případů.
- **Včasné testování.** Testování je vítáno v počáteční fázi vývoje, aby bylo možné je identifikovat a předcházet jim v rané fázi.
- **Pesticidní paradox.** Pokud opakujete stejné testy mnohokrát, pak v určité chvíli přestanou odhalovat nové chyby.
- **Testování závisí na kontextu.** Způsob testování závisí na typu softwaru a jeho účelu, například aplikace, ve kterých je důležitá bezpečnost, se testují jinak než aplikace poskytující služby rozvozu jídla.
- **Absence-of-fallacy.** I když chyby nebyly zjištěny, zaručuje to jejich absenci, jinými slovy, ne vždy po testování je software připraven k vydání. Systém musí splňovat očekávání a potřeby uživatele.

7.1 Unit testy

Unit testy umožňují otestovat výkon jednotlivých částí aplikace. Tento způsob testování nových funkcí umožňuje refaktorovat nový kód bez obav, že stávající funkce přestanou fungovat. Pomocí Unit Tests je možné snadno a rychle najít nové chyby v jednotlivých částech aplikace a opravit je v rané fázi.
Testování bude realizováno pomocí jednotlivých HTTP požadavků odeslaných již běžící aplikaci. Vše se děje na lokálním serveru, specifikuje se port a URL. Poté je nutné nastavit údaje (pokud jsou pro konkrétní požadavek nutné). Protože vyvolání takových testů vede k následné aktualizaci dat v databázi (záznam, editace, mazání), bylo ve fázi testování aplikace nastaveno nastavení v konfiguraci hibernate - spring.jpa.hibernate.ddl-auto= create-drop, který po každém zastavení aplikace obnoví databázi do původního stavu.

Po odeslání požadavku na server probíhá část testování, ohraničená těmito uvozovkami % //code % . Zavolá se funkce, která test implementuje.
1. řádek: Je uveden typ požadavku, v našem případě je to POST + URL
2. řádek: Určuje formát dat, která budou odeslána na server, v našem případě JSON
3. řádek: Určuje datový formát, který bude vrácen serverem, v našem případě je to

JSON
4. řádek: Tento požadavek může provést pouze oprávněný uživatel předáním uživatelského tokenu požadavku. To je také nezbytné pro určení, pro který účet bude nový záznam vytvořen.
Níže je uveden příklad použití testu jednotky ve formátu požadavku HTTP:

```
### ADD NEW CAR
POST localhost:8091/car/
Content-Type: application/json
Accept: application/json
Authorization: Bearer {{token}}

{
  "brand": "BMV X5",
  "number": "AD456ND",
  "fuelType": 1
}
```

7.2 Systémové testy

Tato část testování se příliš neliší ve způsobu testování s unit testy, nicméně nepřebírá jedinou část funkcionality, ale kombinuje mnoho částí, čímž simuluje standardní scénář použití aplikace. Při těchto testech je aplikace testována jako funkční celek. Tyto testy se používají v pozdějších fázích vývoje. Testování aplikace z pohledu klienta. Podle připravených scénářů jsou modelovány různé kroky, které mohou v praxi nastat. V našem případě byly koncové body backendu testovány lokálně během vývoje. Tyto testy byly provedeny ručně pomocí požadavků HTTP.

7.3 Kontrolní seznam testování uživatelského rozhraní.

7.3.1 Funkční testování

V této části je důležité se ujistit, že aplikace splňuje uvedené požadavky a její hlavní část funguje správně.
1. Spuštění aplikace (první spuštění, následné spuštění).
2. Výkon hlavní funkčnosti aplikace.
2.1. Autorizace (e-mailem a uživatelským heslem).
2.2. Registrace (dle zadaných údajů uživatele).
2.3. Ověření povinných polí pro zadávání údajů.
2.4. Navigace.
2.5. Editace uživatelských dat v sekci `Menu`.
2.6. Mapová práce.
2.7. Proces zobrazení procesu tankování.
2.8. Generování a předávání PDF dokumentů potvrzujících platbu.
2.9. Ověření skenováním QR kódu.
2.10. Spolehlivost při výpočtu nejbližších čerpacích stanic.
3. Správné zobrazení chyb.
4. Testování nestabilního internetu.

5. Testování modálních oken.
6. Testování rolovatelných prvků.
7. Složení/rozbalení aplikace.
8. Způsoby připojení k síti (mobilní síť, WIFI).
9. Orientace obrazovky (na výšku, na šířku).

Všechny položky byly plně testovány. Všechna pole, která umožňují zadání uživatele, byla testována s nesprávnými údaji. V místech špatného připojení (podchody, metro) vznikla nestabilita internetu. Níže je uvedena tabulka hodnot rychlosti internetu na určitých místech, doba odezvy serveru a indikátor automatického připojení k internetu. Rychlost internetu byla měřena pomocí služby speedtest.net.

Místo	Způsob	Download v Mbps	Upload v Mbps	Indikátor	Čas
Dům (ubytovna)	Wi-Fi	22.40	17.10	Wi-Fi	1s
Venku, před domem	Vodafone	16.22	13.11	LTE	1s
Uvnitř budovy	Vodafone	15.7	14.34	LTE	2s
Podzemní přechod	Vodafone	4.35	2.6	E	6s
Režim Letadlo	-	0.0	0.0	-	-

Tabulka 7.1. Výsledky testu s různým internetovým signálem.

7.3.2 Bezpečnostní testování

Tato kontrola je zaměřena na testování aplikace z hlediska její bezpečnosti.

1. Testování povolení přístupu ke kameře, umístění.
2. Tajná uživatelská data (heslo) nejsou přenášena v čisté podobě.
3. Pole pro zadání hesla/cvv kódu jsou chráněna hvězdičkami

7.3.3 Testování lokalizace a globalizace

Testování této části aplikace zahrnuje testování různých umístění, oblastí, formátů data, čísel a měn.

1. Všechny statické prvky aplikace jsou přeloženy do konkrétního jazyka nastaveného uživatelem.
2. Kontejnery obsahující variabilní text v závislosti na nainstalovaném jazyce nepřesahují a nepřekrývají se.
3. Správné zobrazení dat a časů

7.3.4 Testování použitelnosti

Tento druh testování vám umožní ujistit se, že použití aplikace je snadné a efektivní pro dosažení vašich cílů. Dá se říci, že kontrolujeme uživatelskou přívětivost produktu.

1. Zobrazení prvků aplikace na zařízeních s různým rozlišením.
2. Fonty odpovídají původní podobě prototypu.
3. Chybové zprávy se zobrazují správně bez pravopisných a gramatických chyb.
4. Správné nadpisy a statické texty.

5. Položky, které jsou neaktivní nebo pouze pro čtení, jsou zašedlé
6. Opravte přechody mezi stránkami a zpět (pokud jsou k dispozici).

Níže je uveden seznam zařízení, na kterých byl software testován:

Model	RAM	Velikost	Rozlišení px/px
Huawei P30 Lite	4.0 Gb	6.15”	2312x1080
Huawei Nova 3	8.0 Gb	6.59”	2340x1080
Emulátor (no-name)	4.0 Gb	7.59”	2208/1768
Emulátor. Pixel 4 XL	4.0 Gb	6.3”	3040x1440

Tabulka 7.2. Seznam zařízení pro testování.

7.3.5 Uživatelské testování

Uživatelské testování nebo testování použitelnosti je technika používaná k vyhodnocení toho, jak snadné je používat mobilní aplikaci. Testy se provádějí se skutečnými uživateli, aby se zjistilo, jak aplikace funguje, aby se posoudilo, jak intuitivně uživatelé dosahují svých cílů.
Do testování uživatelského rozhraní se zapojili lidé, především ti, kteří mají řidičský průkaz a zajímali se o tento způsob tankování do auta, a také lidé se zdravotním postižením.
Před testováním byl každý tester seznámen s hlavními funkcemi, cíli a malým návodem aplikace. Každý tester se řídil specifickým scénářem 7.2.1 (tento scénář zahrnuje úkoly pro každodenní použití i očekávané vzácné funkce). Testeři museli každý svůj čin hlasově komentovat.
Níže jsou uvedeny jednotlivé úkoly, které musí tester postupně provést, a také jeho očekávané chování.
Každý tester se spoléhá pouze na konkrétní úkol, nikoli na popis nebo očekávání uživatele, aby neomezoval svobodu jednání a získal co nejpřirozenější výsledek testu.

Úkol 1. Zaregistrujte se
Uživatel vyplní registrační formulář, poté email potvrdí pomocí kódu, který mu přišel uvedený email a zaregistruje se.
Vstupní data pro registrace:

- Jméno testera
- Příjmení testera
- Testovací e-mail připravený předem pro obdržení potvrzovacího kódu (rubenkur@mail.ru)
- Telefonní číslo testera
- Datum narození a pohlaví testujícího
- Kód, který potvrzuje e-mail uživatele, byl vygenerován a oznámen testerovi.
- Heslo

Úkol 2. Přihlásit se
Uživatel se přihlásí do systému pomocí uvedených údajů

Úkol 3. Přejděte do sekce Mapy
Uživatel přejde do sekce Mapy.

Úkol 4. Hledejte stanice s názvem OMV
Uživatel přejde do vyhledávací sekce a do pole zadá název OMV, načež se mu zobrazí seznam všech stanic s tímto názvem.

Úkol 5. Najděte všechny čerpací stanice, které mají bankomaty
Uživatel přejde do sekce filtr, kde ve službách označí ATM, poté použije filtr a zobrazí se mu seznam všech stanic, které mají bankomaty.

Úkol 6. Přidejte libovolnou stanici do seznamu oblíbených stanic
Uživatel klikne na ikonu srdce.

Úkol 7. Postavte dráhu k nejbližší čerpací stanici
Uživatel klikne na tlačítko `Nearest GS` na mapě a v novém okně, které se otevře, a klikne na tlačítko `Route`. Poté se mu otevře aplikace Google Maps, která mu sestaví trasu.

Úkol 8. Přidejte platební metodu (nová karta)
Uživatel přejde do sekce menu, poté do sekce Platba a přidá novou kartu. Pro usnadnění bude testerovi poskytnut následující seznam s údaji pro vytvoření nové platební metody:

- Číslo karty: 2003 4567 2123 1119
- Datum vypršení platnosti karty: 06.2024
- CVV kód: 123

Úkol 9. Přejděte do sekce tankování a zkuste natankovat za 1000 Kč
Uživatel přejde do sekce tankování, naskenuje před sebou QR kód, který mu bude poskytnuty předem, vybere auto jako "nevybrané", vybere typ benzínu, vybere požadované množství paliva, aby dosáhl 1000 korun, vybere způsob platby a zaplatí nákup.

Úkol 10. Vyberte jakoukoli účtenku ze seznamu transakcí a pošlete jej libovolné osobě v libovolném messengeru
Uživatel přejde do sekce Transakce, vybere libovolnou transakci a po kliknutí na tlačítko `Share` vybere libovolný messenger a odešle jej libovolné osobě.

Úkol 11. Změňte přihlašovací heslo
Uživatel přejde do sekce Menu, poté do Nastavení, poté na stránku pro změnu hesla a změní heslo.

Úkol 12. Změňte jazyk aplikace na český
Uživatel, aniž by opustil sekci Mneu a sekce Nastavení, změní jazyk aplikace na český.

Úkol 13. Přidejte auto
Uživatel přejde do sekce Menu, poté přejde do sekce Auta, kde vytvoří nové auto.
Pro usnadnění bude testerovi poskytnut následující seznam s údaji pro vytvoření nového vozidla:

- Číslo vozidla: 1A5 345

- Značka: Škoda Octavia
- Typ benzínu: Disel

Úkol 14. Napište recenzi na kteroukoli čerpací stanici ve vašem seznamu oblíbených
Uživatel přejde na domovskou stránku, ve které klikne na srdce v pravém horním rohu, načež se mu zobrazí seznam oblíbených čerpacích stanic. Výběrem libovolné stanice ze seznamu se otevře stránka s podrobnými informacemi a na úplném konci se kliknutím na tlačítko `Feedback` přenese na stránku `zpětná vazba`.
Pro úsporu času budou uživateli poskytnuty následující údaje pro zpětnou vazbu:

- Komentář: Velmi příjemný personál, děkujeme!
- Hodnocení: 5 hvězdiček

Úkol 15. Odhlaste se
Uživatel přejde do sekce Menu a odhlásí se.

7.3.6 Výsledek a analýza testování uživatelského rozhraní

Testování aplikace se zúčastnilo 10 lidí, z nichž většina zkušenějšími uživateli softwaru. Účastníky byli také lidé, kteří mají méně zkušeností s používáním různých druhů softwaru. Nejvíc se aplikace líbila testerovi, který má navíc více než 40letou řidičskou praxi, navíc člověk se zdravotním postižením. Každý z účastníků úspěšně prošel všemi fázemi testování scénáře a zanechal zpětnou vazbu na svůj postoj k aplikaci. Pro seřazený seznam výsledků budou použity následující parametry (Question):

- Číslo testeru (T1-T10)
- Q1: Věk
- Q2: Řidičské zkušenosti (roků)
- Q3: Konečné hodnocení vzhledu aplikace*
- Q4: Konečné hodnocení funkčnosti aplikace*
- Q5: Konečné hodnocení užitečnosti aplikace*
- Q6: Čas strávený testováním (minuty)
- Q7: Komentář (Stručná zpětná vazba o tom, co mohlo být změněno nebo co nefungovalo)
- Q8: Operační systém osobního telefonu (Android/IOS)

*Skóre od 1 do 5, kde 5 je nejvyšší skóre (výborně), 1 je nejnižší skóre (velmi špatné)

$\bar{x}$ - Označení aritmetického průměru

Z výsledků testu lze vyvodit několik závěrů, testování byli převážně lidé středního věku, kteří mají poměrně velké řidičské zkušenosti a čím jsou starší, tím více se

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
T1	23	2	5	4	3	9	Účastník moc nechápal, proč je potřebná funkčnost vytváření aut. Navíc poznamenal, že obsah je na stránkách velmi rozprostřený.	IOS
T2	31	6	4	5	5	11	Účastník byl v rozpacích z prázdného, ne světlého designu. Doporučuje se přidat pozadí a sloučit stránky, jako jsou Jazykové změny, s hlavní stránkou Nastavení pomocí vyskakovacího seznamu.	Android
T3	41	20	5	5	5	17	Účastník byl naprosto spokojen s designem, funkčností a užitečností aplikace.	IOS
T4	44	19	5	5	4	13	Účastníka zmátla funkce přidání auta, zdánlivě nadbytečná.	Android
T5	18	1	4	4	3	6	Účastník špatně pochopil systém tvorby auta, který později při tankování kontroluje kompatibilitu paliva a také se mu zdál nadbytečný. Kromě toho také poznamenal, že design se mu zdál nudný, ale velmi úhledný. Navrhl také skrýt tlačítka pro mazání bankovních karet a aut.	Android
T6	27	2	5	5	5	18	Účastník byl s aplikací ve všech ohledech naprosto spokojen, nicméně měl problém najít stránku s nastavením jazyka pro bod 12, kvůli tomu strávil více času než ostatní testeři.	IOS
T7	23	5	5	5	5	10	Účastník byl s aplikací ve všech ohledech naprosto spokojen a rychle a úspěšně prošel celým seznamem scénáře.	Android
T8	68	45	5	5	5	19	Tento účastník byl hlavním testerem, který má poměrně zralý věk, dlouholetou řidičskou praxi a je také zástupcem řidičů se zdravotním postižením. Jediná poznámka byla, že si obsluha sama mohla splést druh paliva, při nepozorném seznámení se s objednávkou.	Android
T9	22	4	5	5	5	9	Účastník byl s aplikací ve všech ohledech naprosto spokojen a rychle a úspěšně prošel celým seznamem scénáře.	Android

Tabulka 7.3. Výsledek a analýza testování uživatelského rozhran 1.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
T10	32	12	5	5	5	11	Účastník byl s aplikací ve všech ohledech naprosto spokojen a rychle a úspěšně prošel celým seznamem scénáře.	Android
$\tilde{x}$	32	11	4.8	4.8	4.5	12	-	-

Tabulka 7.4. Výsledek a analýza testování uživatelského rozhran 2.

prodlužuje doba testování, ale průměrný výsledek testu je 12 minut, což není vůbec špatné, protože žádný z testerů dříve neměl zkušenosti s prací tohoto typu aplikací. Co se týče výsledného hodnocení ve třech kategoriích, výsledky mě velmi potěšily, kritéria pro vzhled aplikace i její funkčnost jsou na stejné úrovni s průměrným skóre 4,8 bodů. Většina negativních recenzí a jediná kategorie, která několikrát získala skóre 3, je hodnocení přínosu aplikace. Rozebereme-li připomínky testerů, pak je nejvíce zmátla funkčnost přidání vozu, pro správné tankování vozu.
Níže jsou uvedeny nejčastější problémy/otázky (Issues) a jejich možná řešení. Každé položce bude přiděleno skóre závažnosti v rozmezí od 1 do 5, kde 5 je velmi důležité.

Issue 1. Funkce přidání auta.
Popis: Z 10 účastníků testu tři považovali funkci přidání vozu ke správnému tankování vozu za nadbytečnou. Účastník T5 až do samého konce testování nechápal, proč je tato funkcionalita potřeba, teprve po ukončení testování mu byla tato funkcionalita co nejpodrobněji popsána.
Závažnost: 4
Řešení: Tato funkcionalita byla původně určena pro všeobecné použití, předpokládala využití jednoho vozu mnoha uživateli, pomocí kterých by bylo možné kontrolovat kompatibilitu tankovaného paliva těmi uživateli, kteří tento vůz nevlastní, např. společnost, která má několik vozů a mohou být používány různými pracovníky. Tato myšlenka mě napadla při probírání mé práce s kolegy, kteří se mnou sdíleli problém tankování auta, do kterého bylo z nedbalosti natankováno špatné palivo a následně vyřazeno z provozu.

Issue 2. Nudný design.
Popis: Návrh se některým účastníkům zdál nedokončený a prázdný. Chyběly barvy, některé prvky se nacházely na prázdných stránkách a chtěly více obsahu, některé prvky působily velmi rušivě, jako například tlačítko pro odstranění položek ze seznamu bankovních karet a aut.
Závažnost: 1
Řešení: Znovu zvažte barevnou politiku aplikace, vyplňte prázdné místo obsahem a opravte jednotlivé momenty, jako je odstranění tlačítek pro smazání.

Kapitola 8
Závěr

Cílem práce bylo implementovat mobilní aplikaci, která umožní tankovat bezkontaktně, aniž byste opustili svůj vůz. Tato metoda je velmi užitečná pro ty, kteří mají zdravotní omezení, protože umožňuje řidiči natankovat, aniž by opustil vůz. Kromě technologie bezkontaktního tankování jsou aplikace skvělým vodičem v oblasti tankování. Uživatelé mají možnost orientovat se v oblasti tankování, protože různé společnosti nabízejí různé služby, od tankování do auta až po bankomaty umístěné na čerpacích stanicích. Celý proces implementace byl realizován díky předkompilované analýze, která popisuje technologie, model domény, uživatelské případy užití a také detailní a kompletní prototyp, který měl několik fází tvorby, od papíru (LO-FI) až po výrobu (HI-FI) implementované pomocí softwaru Figma. Aplikace má několik částí: klientskou aplikaci, která byla implementována v jazyce Dart pomocí Flutter, a serverovou aplikaci REST API, implementovanou pomocí Spring Boot v jazyce Kotlin s využitím databáze PostgreSQL. Každá část vývoje byla na tu dobu svým způsobem příjemná a těžká, ale z procesu tvorby tak rozsáhlé aplikace, kterou jsem implementoval kompletně sám, mi zůstal jen příjemný dojem. Většina z toho, co bylo koncipováno (analýza existujících řešení s definicí silných a slabých stránek a konečným závěrem analýzy; analýza a následná implementace serverové a klientské části aplikace; na straně serveru byl navržen způsob správy čerpacích stanic prostřednictvím rolí (Administrátor, Manažer) a implementace komunikace mezi klientem a serverem během procesu tankování; serverová část aplikace byla testována pomocí předem napsaných skriptů, jejichž prvky jsou HTTP požadavky; klientská část aplikace, která byla testována podle konkrétního scénáře, za účasti 10 testerů), byla implementována přesně podle plánu a požadavků, které jsem popsal, nicméně existuje určité množství funkcí, které jsem z nedostatku času nemohl implementovat: autorizace prostřednictvím služby Google Authorization, funkce sdílení bankovních karet a automobilů, pro možnosti využití této aplikace ve firmě nebo velké rodině. Do budoucna je plánováno rozšíření aplikace s přihlédnutím k mnou neimplementovaným funkcím a také vytvoření plnohodnotné služby, která by manažerům umožňovala pohodlné vytváření a editaci dat čerpacích stanic.

Rád bych také shrnul výsledky, kterých bylo dosaženo pro užitečnost aplikace pro handicapované. Podle mé osobní zkušenosti byl navržen způsob obsluhy handicapovaných osob na čerpacích stanicích, kdy nemusí vůbec vystupovat z auta. Klient k tomu potřebuje dojet co nejblíže k čerpací stanici, naskenovat QR kód, který bude údajně umístěn na samotném stojanu, nebo zadat kód výdejního stojanu ručně, poté musí uživatel v systému označit všechny potřebné údaje pro tankování (množství paliva, druh paliva, způsob platby) a hlavní částí je poznamenat si ve vyskakovacím okně, že je nutná pomoc obsluhy (klikněte na ikonu s obrázkem handicapované). Poté systém oznámí serverové části aplikace, že je nutná pomoc personálu, poté bude muset klient počkat, až bude pistole vložena do plnící nádrže a začne proces plnění. V rámci testování klientské části aplikace byl jako tester přizván zástupce této skupiny osob se zdravotním postižením – můj děda. Kvůli nemoci přišel o dvě nohy a jeho auto

bylo kompletně předělané na ruční ovládání. V jeho případě by nebyl schopen vyjet a naplnit auto sám. Tato myšlenka se mu zdála velmi výhodná, protože v současné době, aby mohl natankovat auto, se na to musí zeptat příbuzných, ale s takovou příležitostí se bude cítit jako úplnější člen společnosti, kterou dnes , je dost nepohodlné.

Literatura

[1] Jak probíhá platba na bezobslužné čerpací stanici. In: Octo-technology.cz [online]. 149 00 Praha 4 - Chodov, 2014 [cit. 2023-05-18]. Dostupné z: http://octo-technology.cz/jak-probiha-platba-na-bezobsluzne-cerpaci-stanici/

[2] Věrnostní program Globus Bonus. In: Globusbonus.cz [online]. Chlumecká 765/6, 198 19 Praha 9, 2023 [cit. 2023-05-18]. Dostupné z: https://www.globusbonus.cz

[3] Tankujte ve kteroukoliv hodinu. In: Globusbonus.cz [online]. Chlumecká 765/6, 198 19 Praha 9, 2023 [cit. 2023-05-18]. Dostupné z: https://www.globusbonus.cz/clanky/1639-tankujte-ve-kteroukoliv-hodinu

[4] OMV MyStation. In: Omv.cz [online]. Česká republika, 2023 [cit. 2023-05-18]. Dostupné z: https://www.omv.cz/cs-cz/omv-mystation-v-cesku

[5] Hodnocení aplikace App Store. In: Apple.com [online]. United states, 2023 [cit. 2023-05-18]. Dostupné z: https://www.apple.com/app-store/

[6] Hodnocení aplikace Play Store. In: Play.google.com [online]. United states, 2023 [cit. 2023-05-18]. Dostupné z: https://play.google.com/store/games

[7] Flutter - nový pohled na vývoj napříč platformami / Sudo Null IT News. In: Developers.google.com [online]. United states, 2023 [cit. 2023-05-18]. Dostupné z: https://habr.com/ru/companies/google/articles/426701/

[8] Flutter architectural overview | Flutter. In: Docs.flutter.dev [online]. United states, 2018 [cit. 2023-05-18]. Dostupné z: https://docs.flutter.dev/resources/architectural-overview

[9] Spring Framework Documentation :: Spring Framework. In: Docs.spring.io [online]. United states, 2023 [cit. 2023-05-18]. Dostupné z: https://docs.spring.io/spring-framework/reference/

[10] Architecture client-server. In: Intellipaat.com [online]. United states, 2023 [cit. 2023-05-18]. Dostupné z: https://intellipaat.com/blog/what-is-client-server-architecture/

[11] O HTTP / Sudo Null IT News. In: Habr.com [online]. RU, 2006–2023 [cit. 2023-05-18]. Dostupné z: https://habr.com/ru/articles/215117/

[12] Spring Data JPA. In: Devcolibri.com [online]. RU, 2022 [cit. 2023-05-18]. Dostupné z: https://devcolibri.com/spring-data-jpa-dao-services-2/

[13] Flutter under the hood. In: Temofeev.ru [online]. RU, 2023 [cit. 2023-05-18]. Dostupné z: https://temofeev.ru/info/articles/flutter-pod-kapotom/

[14] What is Docker and how is it used - Tproger. In: Tproger.ru [online]. RU, 2022 [cit. 2023-05-18]. Dostupné z: https://tproger.ru/articles/chto-takoje-docker/

[15] Running Spring Boot with PostgreSQL in Docker Compose | Baeldung. In: Baeldung.com [online]. 2022 [cit. 2023-05-18]. Dostupné z: https://www.baeldung.com/spring-boot-postgresql-docker

[16] Spring Boot Data Access Layer Best Practices | by Ahad Azarian | Towards Dev. In: Towardsdev.com [online]. 2022 [cit. 2023-05-18]. Dostupné z: https://towardsdev.com/spring-boot-data-access-layer-best-practices-c544d400de7b

[17] Spring Boot Data Access Layer Best Practices | by Ahad Azarian | Towards Dev. In: En.wikipedia.org [online]. Wikipedia, 2023 [cit. 2023-05-18]. Dostupné z: https://en.wikipedia.org/wiki/Hibernate_(framework)

[18] Flutter + clean architecture: an example / Sudo Null IT News. In: Habr.com [online]. RU, 2006–2023 [cit. 2023-05-18]. Dostupné z: https://habr.com/ru/articles/522640/

[19] WebSocket Token-Based Authentication - Nuvalence. In: Nuvalence.io [online]. 2020 [cit. 2023-05-18]. Dostupné z: https://nuvalence.io/insights/websocket-token-based-authentication/

[20] Kotlin, SpringBoot and WebSockets | by Wayne Ellis. In: Codemwnci.medium.com [online]. 2018 [cit. 2023-05-18]. Dostupné z: https://codemwnci.medium.com/kotlin-springboot-and-websockets-276029b22482

[21] PDF creation with Flutter | by Maneesha Erandi. In: Maneesha-erandi.medium.com [online]. 2020 [cit. 2023-05-18]. Dostupné z: https://maneesha-erandi.medium.com/pdf-creation-with-flutter-bac7e2753b89

[22] How to Download and Display a PDF Document in Flutter with PSPDFKit. In: Pspdfkit.com [online]. 2022 [cit. 2023-05-18]. Dostupné z: https://pspdfkit.com/blog/2022/download-and-display-pdf-in-flutter-with-pspdfkit/

[23] Creating PDF files with Multiple Images in Flutter - DEV Community. In: Dev.to [online]. 2023 [cit. 2023-05-18]. Dostupné z: https://dev.to/husbycodereis/creating-pdf-files-with-multiple-images-in-flutter-3eco

[24] Flutter - Simple PDF Generating App - GeeksforGeeks. In: Geeksforgeeks.org [online]. Noida, Uttar Pradesh - 201305, 2023 [cit. 2023-05-18]. Dostupné z: https://www.geeksforgeeks.org/flutter-simple-pdf-generating-app/

[25] How to Build a Flutter PDF Viewer | PSPDFKit. In: Pspdfkit.com [online]. 2023 [cit. 2023-05-18]. Dostupné z: https://pspdfkit.com/blog/2022/how-to-

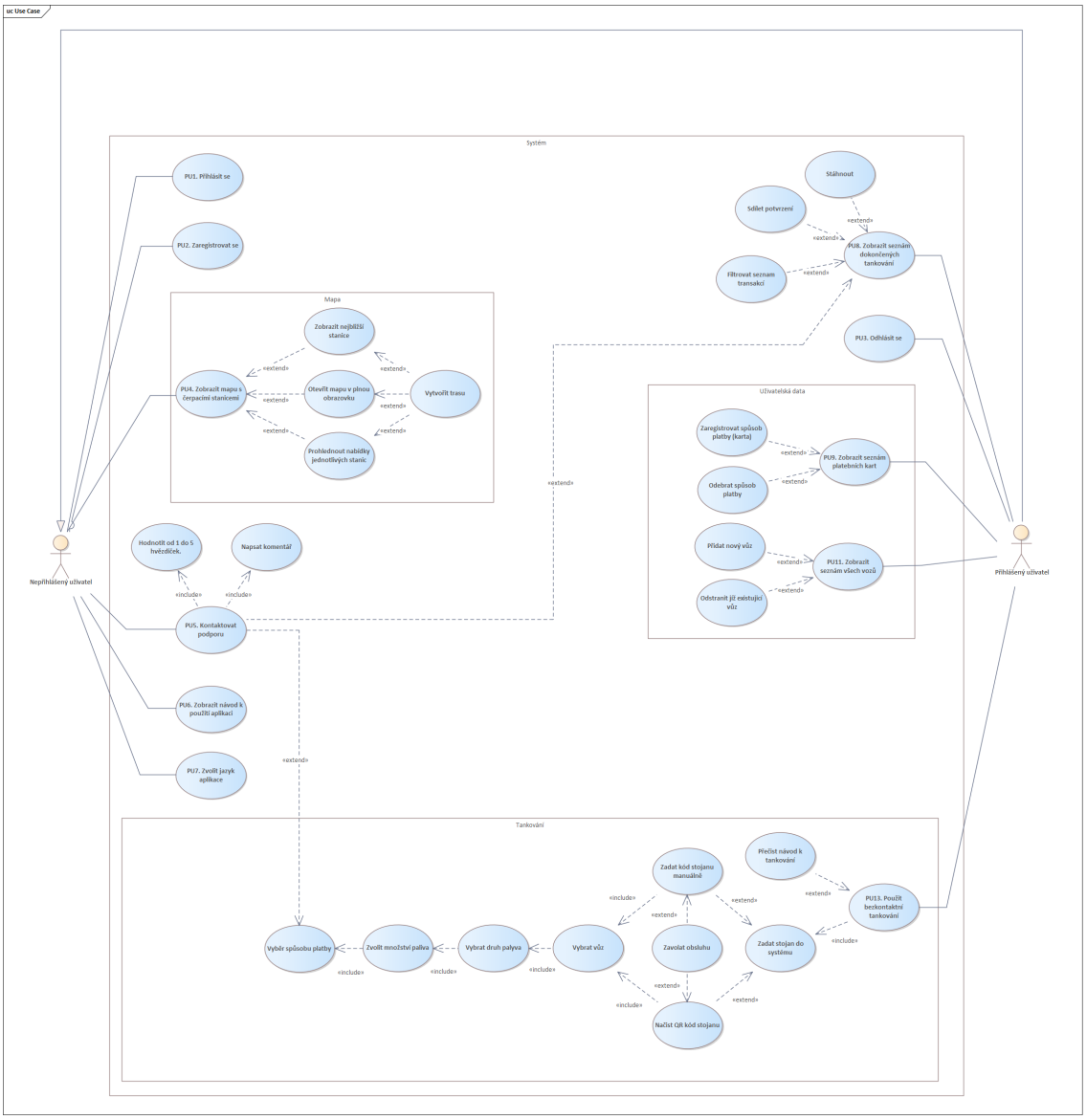
build-a-flutter-pdf-viewer/

[26] Adding Google Maps to a Flutter app | Google Codelabs. In: Codelabs.developers.google.com [online]. 2023 [cit. 2023-05-18]. Dostupné z: https://codelabs.developers.google.com/codelabs/google-maps-in-flutter#4

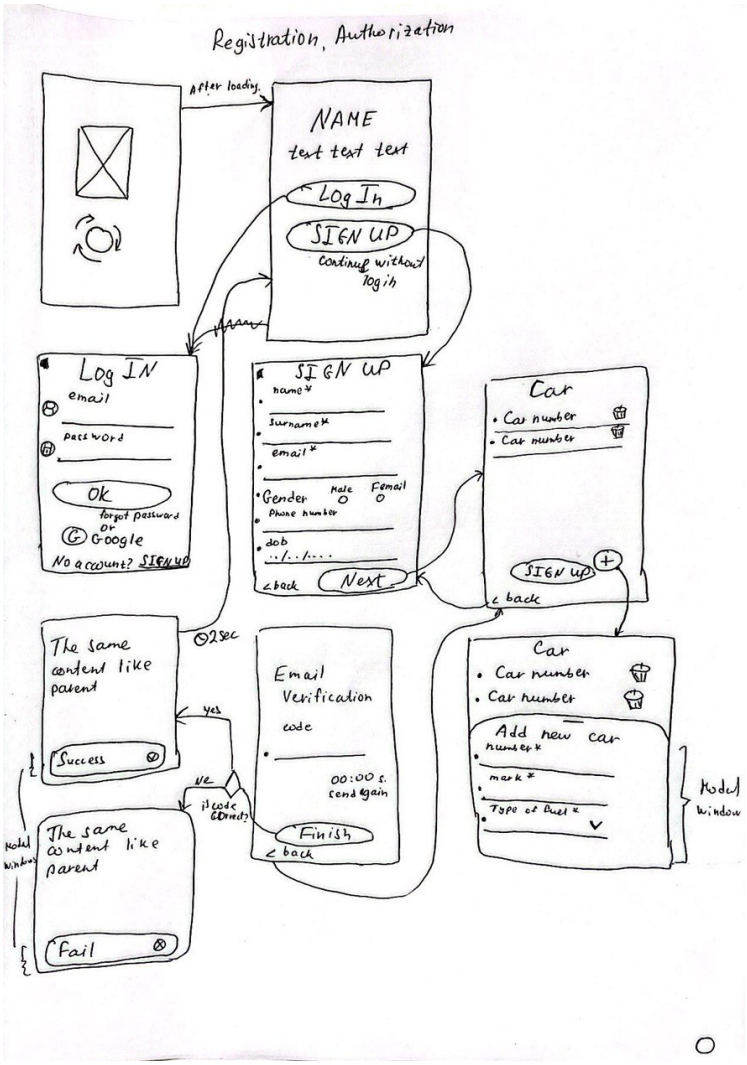
Příloha A
Seznam použitých zkratek

ČR	Česká Republika
QR	Quick Response (QR kód)
KČ	Koruny České
FP	Funkční Požadavek
NFP	Nefunkční Požadavek
PCI DSS	Payment Card Data Security Standard
REST	Representational State Transfer
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
PU	Případ Užití
MVP	Minimal Viable Product
XML	Extensible Markup Language
DI	Dependency Injection
TCP	Transmission Control Protocol
IP	Internet Protocol
SSL	Secure Sockets Layer
SDK	Software Development Kit
UI	User Interface
LO-FI	Low-Fidelity Prototyping
HI-FI	High-Fidelity Prototyping
PDF	Portable Document Format
GIT	Distributed version control system
PaaS	Platform as a Service
ORM	Object-Relational Mapping
JPA	Java Persistence API
SQL	Structured Query Language
JDBC	Java Database Connectivity
DAO	Data Access Object
URL	Uniform Resource Locator
EAR	Enterprise architecture
RAM	Random- ccess Memory
ATM	Automated Teller Machine
GS	Gas Station
CVV	Card Security Code

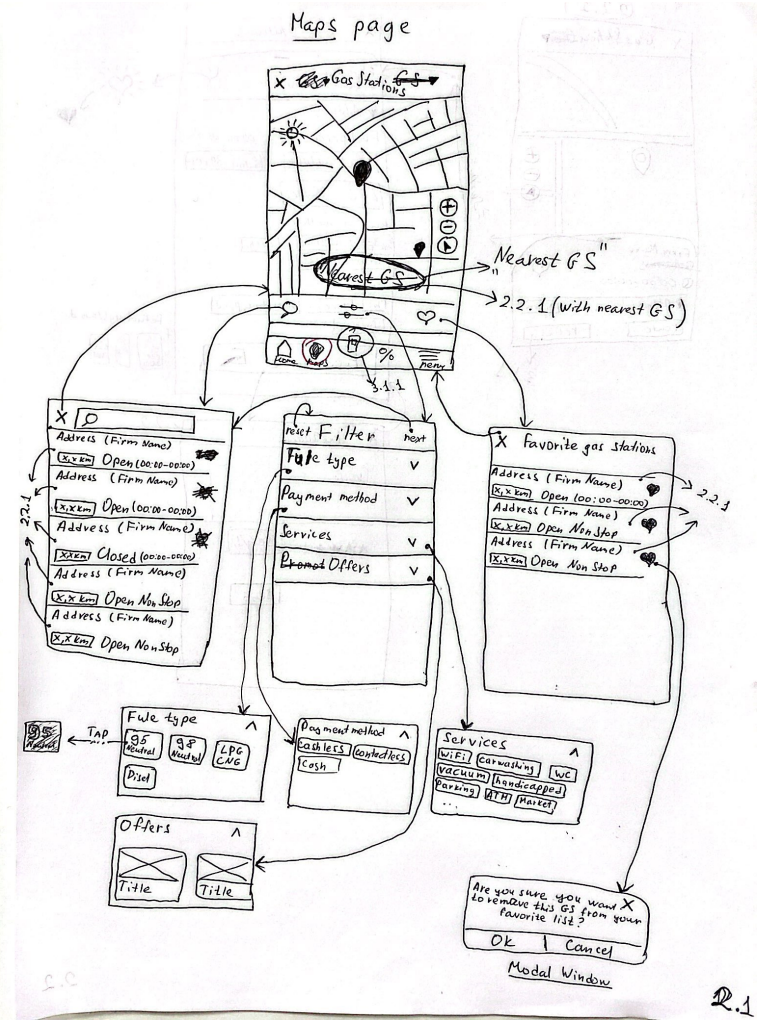
Příloha B
Kompletní diagram případů užití



Příloha C
Příklad LO-FI prototypu

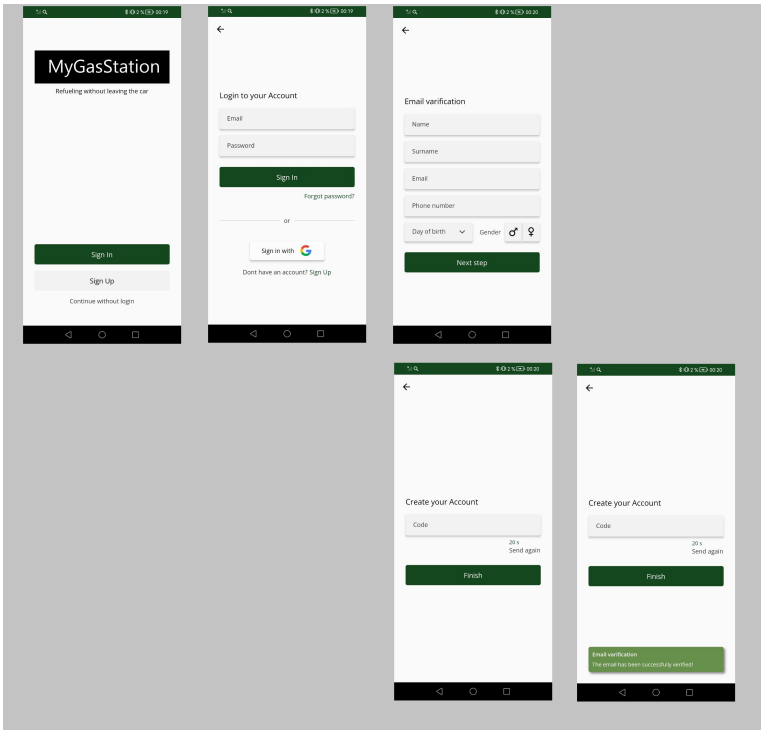


Obrázek C.1. Sekce autorizace a registrace (LO-FI prototype).

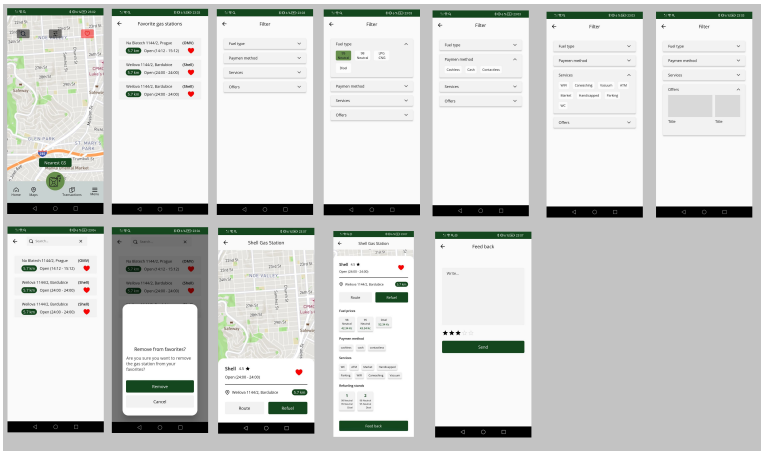


Obrázek C.2. Sekce maps (LO-FI prototype).

Příloha D
Ukázka výsledného pohledu aplikace



Obrázek D.3. Sekce autorizace a registrace (Výsledná aplikace).



Obrázek D.4. Sekce maps (Výsledná aplikace).

Příloha E
Instalační návod

Tato část bude popisovat, jak spustit aplikaci, která vám umožní testovat. Protože se aplikace skládá ze serverové a klientské části, popíšu každou část zvlášť.

Klientská část aplikace

Pokud budete postupovat podle pokynů ze zdroje `https://medium.com/flutter-community/flutter-getting-started-f07df7d4cce2`:
1. Stáhněte si Flutter pro váš operační systém `https://docs.flutter.dev/get-started/install`
2. Zkontrolujte, zda je Flutter správně nainstalován, zadáním následujícího příkazu na příkazový řádek: flutter doctor.
3. Použijte jedno z následujících vývojových prostředí: Android Studio nebo VS Code.
4. Spuštění klienta:

- **Emulátor** - Tento návod použijte k nastavení emulátoru `https://docs.flutter.dev/get-started/install/windows`
- **Device** - Chcete-li spustit klientskou aplikaci na skutečném zařízení Android, postupujte takto.
  a. Připojte zařízení Android k počítači nebo notebooku pomocí kabelu USB. b. Aktivujte možnost vývojáře na svém zařízení Android. c. Nainstalujte aplikaci Mirroring pro zrcadlení vašeho zařízení Android ve vašem počítači
  Celý tutoriál - `https://ferilukmansyah.medium.com/easy-way-to-setup-your-android-device-to-run-flutter-project-28bddf0fa7f1`
- **Příkaz** - Aplikaci lze spustit následujícím příkazem - flutter run –no-sound-null-safety

Serverová část aplikace

1. Stáhněte si klienta Docker pro pohodlí - `https://www.docker.com/products/docker-desktop/`
2. Použijte vývojové prostředí IntelliJ IDEA
3. Nainstalujte plugin Docker: File -> Settings -> Plugins -> Add Docker
4. Otevřete soubor docker-compose.yml v kořenové složce
5. Spusťte proces stavby kontejneru kliknutím na zelenou šipku na levém panelu
6. Poté přejděte na Docker Desktop, spusťte image mygs
7. Provoz na serverové straně aplikace můžete zkontrolovat pomocí již napsaných skriptů umístěných v kořenové složce `requests`.

Pro správnou funkci aplikace je nutné nastavit pro odesílání emailů údaje jako spring.mail.username, spring.mail.password; bing.api.key pro výpočty rozsahu.
Tato data musíte zadat ručně v `\src\main\resources\application.properties`.
Údaje si můžete vyžádat písemně na email: `kurbarub@fel.cvut.cz`.