FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# Zadání bakalářské práce

| | |
|---|---|
| **Název:** | Systém pro sledování zásilek v reálném čase |
| **Student:** | Anton Korolov |
| **Vedoucí:** | Ing. Marek Suchánek |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Webové a softwarové inženýrství, zaměření Softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce letního semestru 2023/2024 |

## Pokyny pro vypracování

Rozvážkové a doručovací služby jsou poskytovány již řadu let, ale v poslední době zažívají např. kvůli pandemii výrazný nárůst v oblibě. Častým problémem ale je časová synchronizace mezi doručovatelem a příjemcem, kdy příjemce typicky musí netrpělivě čekat v dlouhém časovém okně dle oznámení o odhadovaném doručení. Cílem této práce je poskytnout řešení, kde by mohl příjemce sledovat pohyb zásilky/kurýra a tím získat cennou informaci o předpokládaném doručení, navíc aktualizovanou v reálném čase.

- Analyzujte doménu doručování zásilek s využitím konceptuálního modelování.
- Proveďte stručnou rešerši existujících klíčových řešení v této oblasti.
- Na základě analýzy a rešerše stanovte požadavky na vlastní řešení.
- Navrhněte vlastní řešení jako webovou aplikaci, která bude naplňovat klíčové požadavky. Zohledněte při návrhu možnosti budoucího rozvoje i použití aplikace v reálném prostředí.
- Implementujte řešení dle návrhu s využitím technologií Spring Boot (Java) a React (TypeScript).
- Výslednou implementaci řádně otestujte.
- Zhodnoťte vlastní řešení v porovnání s již existujícími a popište možnosti reálného použití.

---

*Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 8. září 2022 v Praze.*

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Systém pro sledování zásilek v reálném čase

***Anton Korolov***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Marek Suchánek

8. května 2023

# Poděkování

Chtěl bych poděkovat svému vedoucímu, panu Ing. Markovi Suchánkovi, za odborné vedení, pomoc, cenné rady a znalosti, které mi věnoval při zpracování této bakalářské práce. Dále bych chtěl poděkovat vyučujícím předmětů BI-PJV a BI-TJV za vědomosti, jež mi byly poskytnuty během výuky, a také své rodině a přátelům za podporu při celé době mého studia.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelum. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 8. května 2023 . . . . . .

### Odkaz na tuto práci

Korolov, Anton. *Systém pro sledování zásilek v reálném čase*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

# Abstrakt

Tato bakalařská práce se zabývá analýzou existujících doručovacích systémů, návrhem a implementací nového řešení, jež umožňuje sledovat doručování zásilky v reálném čase. V práci se postupuje v souladu s metodou „Vodopád“ softwarového inženýrství. Provedeným výzkumem bylo zjištěno, že z existujících řešení jenom jedno má možnost sledování v reálném čase, z tohoto důvodu bylo v práci implementováno vlastní řešení, v němž je tato možnost realizována. Vytvořené řešení poskytuje možnost zadávat plán cesty kurýra, sledovat aktuální počet adres v reálném čase a zanechávat zpětnou vazbu o kurýrovi, který tuto zásilku doručoval. Na závěr práce je uvedeno zhodnocení výsledného řešení a porovnání s již existujícími.

**Klíčová slova** webová aplikace, implementace aplikace, doručovací systém, sledování kurýra, sledování zásilky, sledování v reálném čase, Java, React

# Abstract

This bachelor's thesis deals with the analysis of existing delivery systems, the design and implementation of a new solution that allows tracking the delivery of a shipment in real time. The work proceeds in accordance with the "Waterfall" method of software engineering. Through the conducted research, it was found that only one of the existing solutions has the possibility of real time tracking, for this reason, an own solution was implemented in the work, in which this option is realized. The created solution provides the ability to enter the courier's travel plan, monitor the current number of addresses in real time, leave feedback about the courier who delivered this shipment. At the end of the work, an evaluation of the resulting solution and a comparison with already existing ones is presented.

**Keywords** web application, application implementation, delivery system, courier tracking, shipment tracking, real-time tracking, Java, React

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Žijeme v době, kdy lidé potřebují a očekávají rychlé vyřízení všech problémů. Z toho vyplývá, že ve všem mají málo trpělivosti. To je jeden z hlavních důvodů, proč požadují rychlé doručení svých zásilek a balíků. Někdy doručování zásilky může trvat velmi dlouho. Adresátovi, i v rámci stejného města, může obyčejný dopis jít několik dní a také se stává, že se cestou může ztratit. Někdy je potřeba balík ihned poslat a není čas navštěvovat nejbližší pobočku.

Kurýrní doručení je služba, která umožňuje rychle doručit důležitý balík, ať už se jedná o dokumenty, dárky, cenné nebo rozbitné předměty, kytice atd. Výhodou této služby je, že ji lze využít jak pro pracovní, tak pro každodenní účely kdykoliv je potřeba.

Kurýr doručí balík osobně příjemci během několika hodin oproti podpisu. Důsledkem je to, že v současné době kurýrní společnosti řeší množství problémů, jako jsou zpoždění doručení, ztracení zásilky, neuspokojivá komunikace atd. Toto lze vyřešit použitím vhodného softwaru, díky němuž bude doručování na poslední míli nejméně komplikované.

Téma jsem si zvolil, neboť problém nebyl uspokojivě vyřešen a také protože implementace podobné webové aplikace mi umožní aplikovat znalosti, získané na univerzitě, v praxi.

Výsledná aplikace bude prospěšná pro společnosti, kterým záleží nejenom na spokojenosti svých zákazníků, ale i na optimalizaci a usnadnění práce svých zaměstnanců. Zákazník bude moct sledovat pohyb kurýra na mapě, aby mohl v klidu plánovat svůj den. Díky aplikaci, předání balíku bude rychlejší, protože kurýrovi zbyde zásilku jenom dostavit a o notifikaci zákazníka a hodnocení se postará software.

V úvodní části popisuji, jaký význam v dnešní době mají kurýrní společnosti, čím důležitý je software a také je v ní shrnuta struktura práce. V první kapitole stanovuji všechny cíle, které chci při řešení práce postupně splnit. Druhá kapitola je věnována analýze. Zde jsou identifikovány klíčové procesy, vytvořen doménový model, analyzovány již existující řešení, stanoveny funkční

a nefunkční požadavky na systém, které jsou dále rozdělené do čtyř kategorií podle metody „MoSCoW“. Třetí kapitola je věnována návrhu systému. Tato část obsahuje popis použitých technologií, případy užití, návrh architektury a uživatelského rozhraní. Následuje ji čtvrtá kapitola, v niž je popsána implementační část aplikace. V páté kapitole je představen průběh testování serverové a klientské části. Poslední kapitola popisuje zhodnocení přínosů použití informačního systému a je v ní navržen další budoucí rozvoj aplikace. V závěru je stručně shrnuto dosažení stanovených cílů práce, které byly prodělány.

Kapitola 1

# Cíl práce

Cílem práce je analyzovat, navrhnout, implementovat, otestovat a zhodnotit webovou aplikaci, která umožní uživateli v reálném čase sledovat pohyb svoje zásilky na mapě a dostávat aktuální informaci o počtu adres, které zbývají od kurýra do zákazníka. Při analýze, návrhu a implementaci se postupuje v souladu s metodou „Vodopád“ softwarového inženýrství.

V rámci analýzy bude provedena rešerše domény doručování zásilky a rešerše již existujících řešení pro sledování zásilky v reálném čase při doručení na poslední míli. Z této analýzy vyplývají funkční a nefunkční požadavky, jež se použijí při návrhu.

Dílčím úkolem je navrhnout, implementovat a otestovat aplikaci, která by měla být snadno rozšířitelná. V rámci návrhu je potřeba zohlednit možnosti budoucího rozvoje a použití aplikace v reálném prostředí.

Na závěr práce bude zhodnoceno řešení a přínosy použití vytvořené aplikace a také budou porovnány s řešeními, jež byly řádně analyzovány.

Kapitola 2

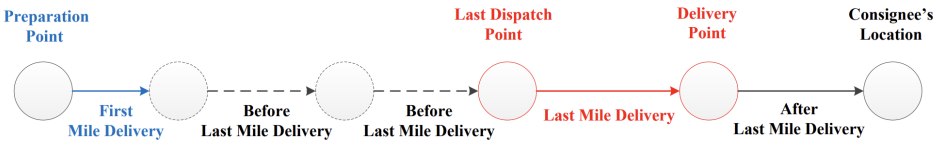
# Analýza

Analýza je jednou z prvních fází při vývoji softwaru. Tato kapitola se zabývá analýzou domény doručování zásilky a existujících řešení. Na závěr jsou sestaveny funkční a nefunkční požadavky.

## 2.1 Analýza domény doručování zásilky

Kurýrní služby jsou nutností pro mnoho lidí a organizací. Potřebu kurýrních služeb pociťují nejen firmy, ale i běžní občané. Společnosti poskytující kurýrní služby mají všechny výhody pošty. Mají však některé výhodné rozdíly. Například, kurýr může doručit cenný náklad ve stanovenou hodinu přímo do rukou adresáta. Právě tato kvalita je pro mnoho zákazníků prioritou (viz obrázek 2.1).

Doručování zásilky je větší proces, který se skládá z několika podprocesů. V dnešní době kurýrské služby doručují zásilky nejenom uvnitř jednoho města, ale také i mezi státy a kontinenty. Sám proces doručování zásilky kurýrem od distribučního centra do zákazníka se jmenuje „Doručení na poslední míli“. [2]



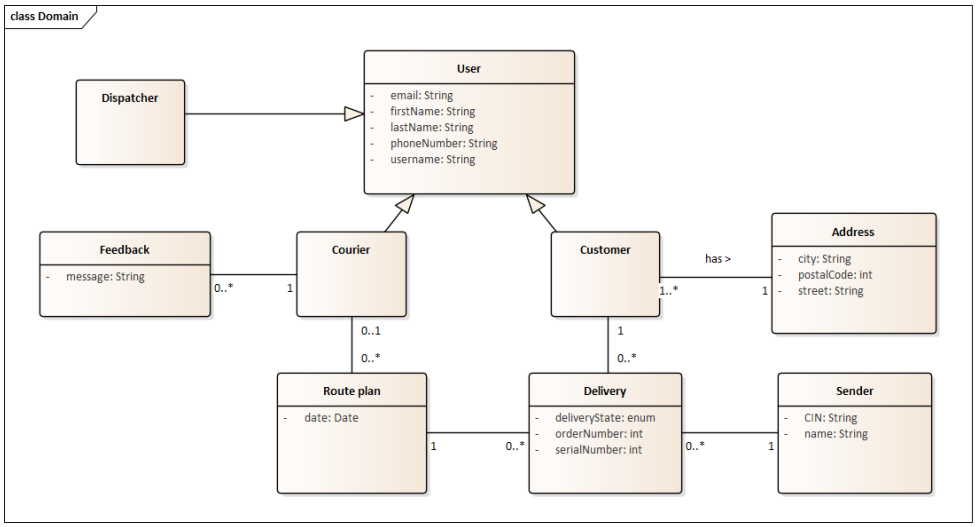

Obrázek 2.1: Stavy procesu doručování zásilky [1]

### 2.1.1 Doménový model doručování zásilky

Doména doručování zásilky je znázorněna pomocí doménového modelu (viz obrázek 2.2), sloužícího pro reprezentaci smysluplných konceptů reálného světa

souvisejících s doménou, které je třeba modelovat v softwaru. Koncepty zahrnují data z byznysu a pravidla, která se na ně vztahují. [3]

Byl použit jazyk UML (Unified Modeling Language), který je specifikaci, definující grafický jazyk pro vizualizaci, specifikaci, konstrukci a dokumentaci artefaktů systémů distribuovaných objektů. [4]

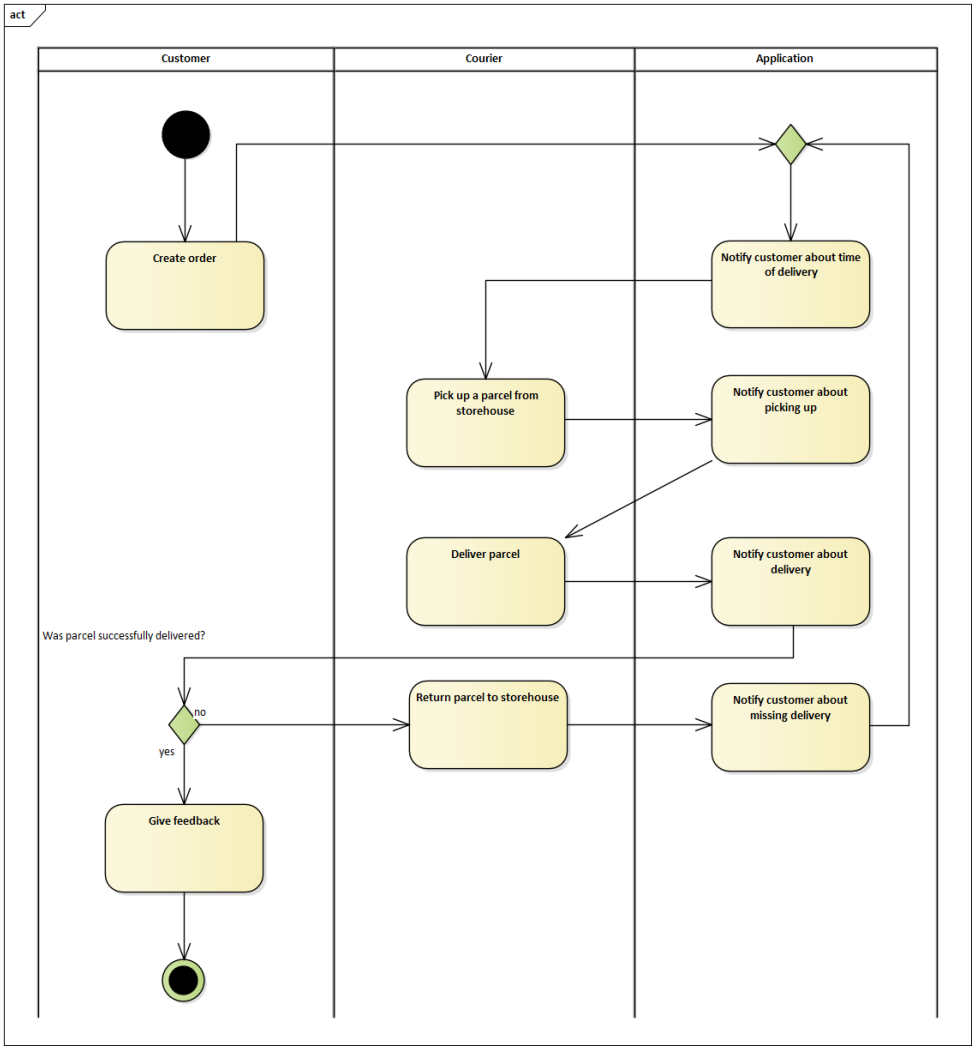



Obrázek 2.2: Doménový model

- **User**
  Představuje skutečnou osobu. Každá osoba má uživatelské jméno, email, křestní jméno, příjmení a telefonní číslo.
- **Courier**
  Osoba, která zásilku doručuje.
- **Customer**
  Osoba, která čeká na doručení zásilky.
- **Delivery**
  Představuje pokus o doručení zásilky ke zákazníkovi.
- **Route plan**
  Plán adres, které kurýr musí navštívit.
- **Feedback**
  Představuje zpětnou vazbu od zákazníka. Může obsahovat zprávu.
- **Sender**
  Představuje odesílatele zásilky.
- **Dispatcher**
  Osoba, která je dispečerem.

### 2.1.2 Proces doručení na poslední míli

Proces doručení na poslední míli je znázorněn pomocí UML diagramu aktivit (viz obrázek 2.3).




Obrázek 2.3: Proces doručení na poslední míli

### 2.1.3 Proces přeplánování doručení

Proces přeplánování doručení začíná, když zákazník chce změnit termín doručení zásilky. Když zákazník v aplikaci přejde na stránku, na níž může termín změnit, vybere jednu z nabízených možností a svůj výběr potvrdí. Dále začíná proces doručení zásilky.

## 2.2 Analýza existujících řešení doručování zásilky

Pro správné určení funkčních a nefunkčních požadavků byla provedena analýza již existujících řešení. Společností, které se zabývají doručováním balíků, je velmi mnoho. Řešení byla vybrána na základě popularity a hodnocení lidí na několika prvních webových stránkách s hledaným textem: „Nejlepší kurýrní služba“. Mezi ty nejlepší a nevýznamnější patří: Zásilkovna, DPD, DHL, GLS, TNT(FedEx), PPL, Česká pošta. Po zkoumání jejich webových stránek byla sestavena tabulka (viz tabulka 2.1).

| **Funkce** / **Společnost** | **Sledování zásilky** | **Možnost poslat zásilku P2P** | **Zrušení / Přeplánování doručení** | **Zanechání zpětné vazby*** | **Přehlednost a srozumitelnost webu** |
|---|---|---|---|---|---|
| Zásilkovna | - | + | - | + | + |
| DPD | + | + | + | + | + |
| DHL | - | + | + | - | - |
| GLS | - | + | + | - | + |
| TNT(FedEx) | - | + | + | - | + |
| PPL(partner DHL) | - | + | + | - | + |
| Česká pošta | - | + | + | - | + |
| Souhrn | 14 % | 100 % | 86 % | 29 % | 86 % |

\+ – funkce je
\- – funkce není
* – pokud na oficiálních stránkách společnosti nebo v popisu aplikace v Google Play a App Store nebyla žádná zmínka o této funkci, bylo považováno, že implementována není.

Tabulka 2.1: Funkce v kurýrních společnostech

Z analýzy vyplývá, že z nejvýznamnějších a nejvíce používaných kurýrních

služeb, funkce sledování zásilky v reálném čase, která je nejdůležitějším parametrem v této práci, existuje pouze u společnosti DPD. Proto, kromě DPD, byla analyzovány řešení, která, možná, nejsou významná, ale tuto funkci mají.

Celkem bylo vybráno šest řešení, která jsou dostatečně populární, lehce dohledatelná na internetu a mají vlastní web, obsahující informace ohledně možností a vlastností těchto služeb. Analýza byla rozdělena tak, že nejprve byly zanalyzovány 3 kurýrské služby (DPD, Liftago, Shadowfax), a dále byly zanalyzovány 3 řešení (Locate2u, Fixlastmile, Detrack), jež implementují online sledování.

### 2.2.1 DPD

DPD Česká republika [5] je součástí mezinárodní přepravní sítě DPDgroup, která patří na trhu zásilkových a expresních služeb k absolutním špičkám a je největší doručovací sítí v rámci balíkové přepravy v Evropě. Díky kombinaci moderních technologií s vysokou úrovní místních znalostí dokáže každý den po světě přepravit 7,5 milionů zásilek ve více než 230 zemích světa a se sítí 58 000 odběrných míst Pickup. Ukázka webové aplikace DPD Kurýr (viz obrázek 2.4). Ukázka mobilní aplikace DPD Kurýr (viz obrázek 2.5).

#### Zhodnocení jednotlivých funkcí aplikace

**Klady:**

- sledování kurýra na mapě v reálném čase,
- je napsáno kolik adres kurýru zbývá do zákazníka,
- možnost poslat zprávu kurýrovi,
- možnost zrušit / přeplánovat doručení.

**Zápory:**

- nemění se očekávaný čas v závislosti na poloze kurýra,
- není možnost zanechat zpětnou vazbu.

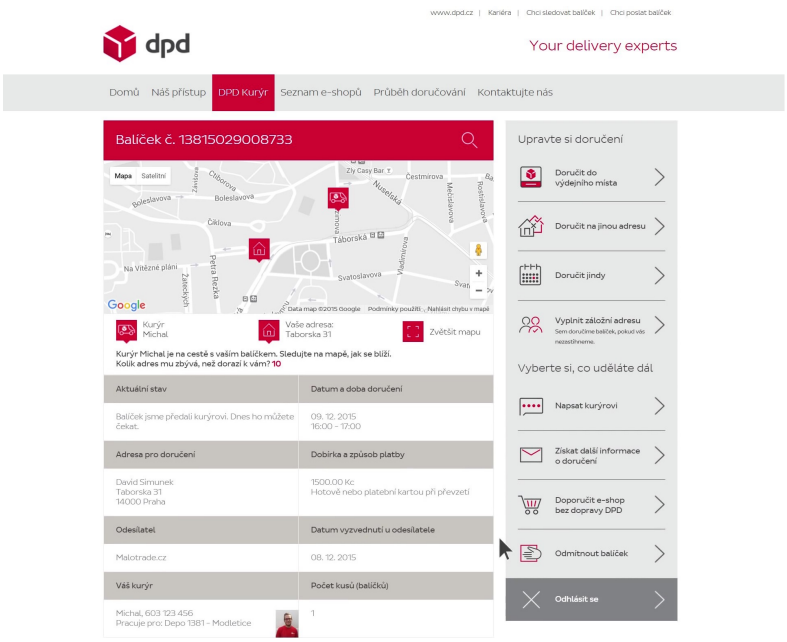
#### Zhodnocení vzhledu aplikace

**Klady:**

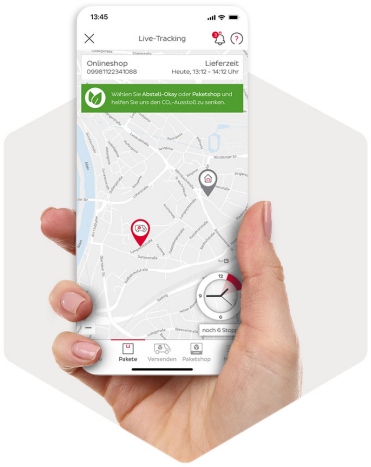
- přehlednost,
- zobrazení na mobilním zařízení.

**Zápory:**

- UX práce s mapou. Žádná možnost použití mapy, kromě sledování.




Obrázek 2.4: Webová aplikace DPD kurýr [5]

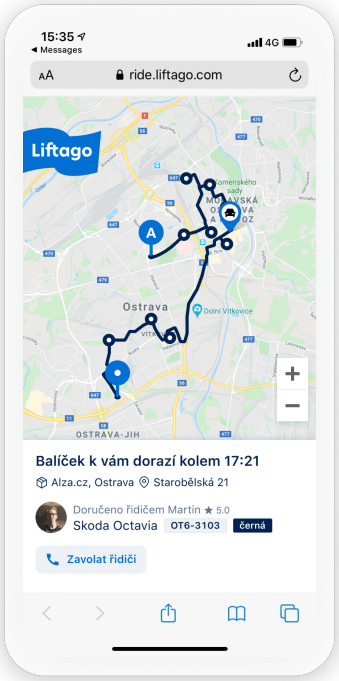



Obrázek 2.5: Mobilní aplikace DPD kurýr [5]

### 2.2.2 Liftago

Liftago [6] je český startup, který vytváří technologickou platformu pro optimalizaci městské dopravy v České republice a v zahraničí. Cílem společnosti jsou tzv. Chytrá města (Smart Cities), kde lidé ztratí závislost na vlastních automobilech a ubude problémů s parkováním. Ve velkých městech se kvůli fragmentaci trhu měsíčně najezdí miliony kilometrů prázdnými taxíky. Cílem

Liftago je vytvořit volný trh, do kterého se mohou zapojit současní poskytovatelé přepravy, ale i budoucí technologie. Ukázka mobilní aplikace Liftago (viz obrázek 2.6).




Obrázek 2.6: Ukázka doručování prostřednictvím služby Liftago [6]

### Zhodnocení jednotlivých funkcí aplikace

**Klady:**

- sledování kurýra na mapě v reálném čase,
- mění se očekávaný čas v závislosti na poloze kurýra,
- možnost zanechat zpětnou vazbu.

**Zápory:**

- není napsáno kolik adres kurýru zbývá do zákazníka,
- není možnost poslat zprávu kurýrovi (pouze zavolat),
- není možnost zrušit / přeplánovat doručení.

### Zhodnocení vzhledu aplikace

**Klady:**

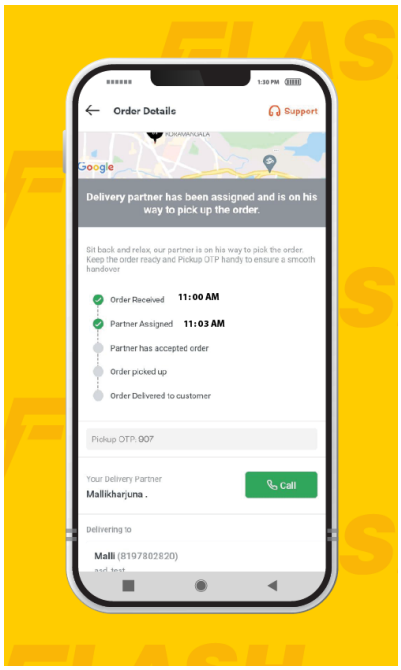
– přehlednost,
– zobrazení na mobilním zařízení.

**Zápory:**

– UX práce s mapou. Žádná možnost použití mapy, kromě sledování.

### 2.2.3 Shadowfax

Shadowfax [7] je přední indická platforma okamžité logistiky s podporou technologií na vyžádání. Shadowfax je jediná indická platforma B2B2C se schopnostmi obsluhovat více kategorií koncových zákazníků – včetně rozvozu potravin, rychlého obchodu, plnění elektronického obchodu ve více městech a zpětné logistiky prostřednictvím jediné platformy. Na základě rychle rostoucí základny, více než 100 000 partnerů, Shadowfax obsluhuje více než 170 podnikových klientů ve více než 700 městech a každý den dodává přes 1 milion objednávek. Ukázka mobilní aplikace Shadowfax (viz obrázek 2.7).




Obrázek 2.7: Ukázka doručování prostřednictvím služby Shadowfax [7]

#### Zhodnocení jednotlivých funkcí aplikace

**Klady:**

– sledování kurýra na mapě v reálném čase.

**Zápory:**

- není napsáno kolik adres kurýru zbývá do zákazníka,
- nemění se očekávaný čas v závislosti na poloze kurýra,
- není možnost poslat zprávu kurýrovi (pouze zavolat),
- není možnost zanechat zpětnou vazbu,
- není možnost zrušit / přeplánovat doručení.

#### Zhodnocení vzhledu aplikace

**Klady:**

- přehlednost,
- zobrazení na mobilním zařízení.

**Zápory:**

- UX práce s mapou. Žádná možnost použití mapy, kromě sledování.

### 2.2.4 Locate2u

Locate2u [8] je softwarová platforma, navržena pro jakýkoli podnik s dodávkami nebo službami. Toto řešení pomáhá těmto podnikům zlepšit efektivitu jejich tras, zlepšit zkušenosti zákazníků s doručováním a zvýšit produktivitu, a to vše při zkrácení času potřebného k plánování tras. Ukázka mobilní aplikace Locate2u (viz obrázek 2.8).

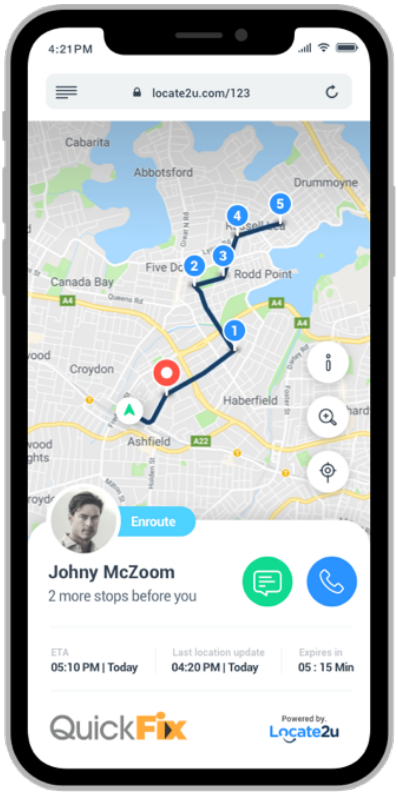
#### Zhodnocení jednotlivých funkcí aplikace

**Klady:**

- sledování kurýra na mapě v reálném čase,
- je napsáno kolik adres kurýru zbývá do zákazníka,
- možnost poslat zprávu kurýrovi (Jde to ale jenom jako SMS),
- možnost zanechat zpětnou vazbu.

**Zápory:**

- nemění se očekávaný čas v závislosti na poloze kurýra,
- není možnost zrušit / přeplánovat doručení.




Obrázek 2.8: Ukázka doručování s využitím řešení Locate2u [8]

**Zhodnocení vzhledu aplikace**

**Klady:**

- přehlednost,
- zobrazení na mobilním zařízení.

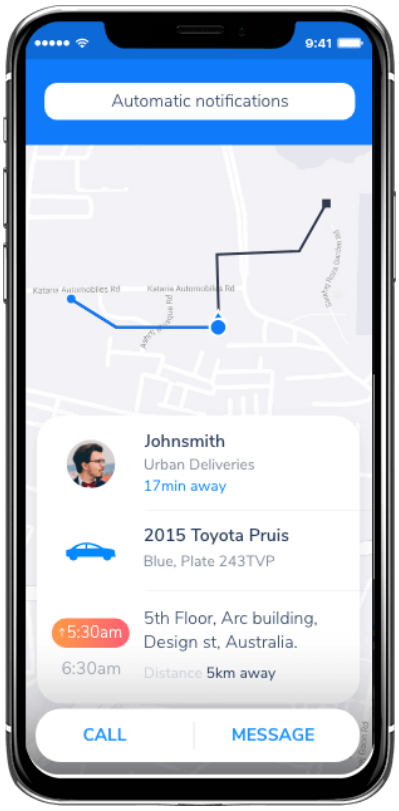
**Zápory:**

- UX práce s mapou. Žádná možnost použití mapy, kromě sledování.

### 2.2.5 Fixlastmile

Fixlastmile [9] je software pro doručování na poslední míli, jehož prostřednictvím lze sledovat objednávky v reálném čase, automatizovat plánování, optimalizovat trasy dodávek a efektivně škálovat operace. Uživatelé mohou rychle dodávat jídlo pomocí pokročilého plánování trasy v softwaru, dodávat věci s optimalizací trasy a vizualizovat míru úspěšnosti na webovém panelu. Je možné provádět expedici léků v reálném čase, řídit pracovní sílu, dodávat FMCG zboží a spravovat prádlo. Software zjednodušuje operace pomocí

automatizovaného odesílání, sledování v reálném čase a zajišťuje uspokojující zážitek pro zákazníky. Integruje se také s ERP, online objednávkovým systémem prostřednictvím uživatelsky přívětivého API. Software přichází se snadnou optimalizací trasy, zaměřuje se na všechny hlavní priority a automatické odesílání pro snížení mzdových nákladů a služeb. Pomocí výkonných a analytických funkcí softwaru lze zvýšit produktivitu a zajistit odpovědnost. Aby byla zajištěna spokojenost zákazníků, software přichází s integrací zákaznické komunikace, sběrem zpětné vazby a sledováním řidičů v reálném čase. Ukázka mobilní aplikace Fixlastmile (viz obrázek 2.9).




Obrázek 2.9: Ukázka doručování s využitím řešení Fixlastmile [9]

### Zhodnocení jednotlivých funkcí aplikace

**Klady:**

- sledování kurýra na mapě v reálném čase,
- mění se očekávaný čas v závislosti na poloze kurýra,
- možnost poslat zprávu kurýrovi,
- možnost zanechat zpětnou vazbu.

**Zápory:**

- není napsáno kolik adres kurýru zbývá do zákazníka,
- není možnost zrušit / přeplánovat doručení.

**Zhodnocení vzhledu aplikace**

**Klady:**

- přehlednost,
- zobrazení na mobilním zařízení.

**Zápory:**

- UX práce s mapou. Žádná možnost použití mapy, kromě sledování.

### 2.2.6 Detrack

Detrack [10], který nabízí prvního řidiče navždy zdarma, je výkonný software pro správu dodávek, který umožňuje sledovat vozidla živě na mapě a zachycovat elektronický doklad o doručení v reálném čase pouze pomocí mobilní. Tato aplikace byla přeložena do 26 jazyků a denně se používá po celém světě k dokončení milionů dodávek. Ukázka mobilní aplikace Detrack (viz obrázek 2.10).

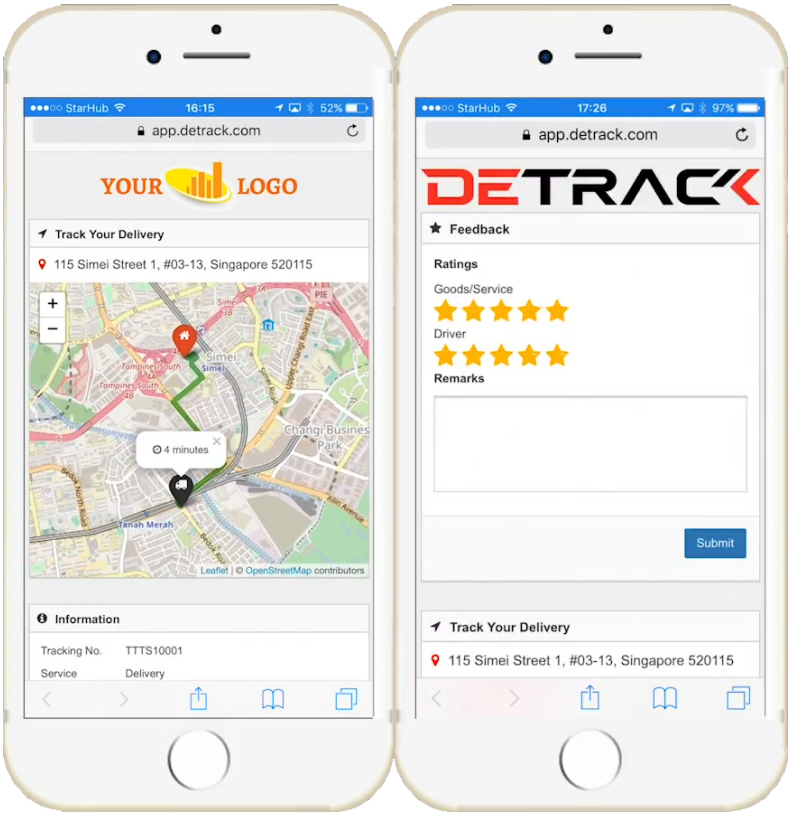
**Zhodnocení jednotlivých funkcí aplikace**

**Klady:**

- sledování kurýra na mapě v reálném čase,
- mění se očekávaný čas v závislosti na poloze kurýra,
- možnost zanechat zpětnou vazbu.

**Zápory:**

- není napsáno kolik adres kurýru zbývá do zákazníka,
- není možnost poslat zprávu kurýrovi,
- není možnost zrušit / přeplánovat doručení.




Obrázek 2.10: Ukázka doručování s využitím řešení Detrack [10]

### Zhodnocení vzhledu aplikace

#### Klady:

- přehlednost.

#### Zápory:

- zobrazení na mobilním zařízení,
- UX práce s mapou. Žádná možnost použití mapy, kromě sledování.

### Shrnutí

Všechny funkce existujících řešení byly shrnuty v tabulce (viz tabulka 2.2).

| Funkce / Společnost | Sledování kurýra | Dynamický počet adres | Dynamický čas doručení | Zpráva kurýrovi | Zanechání zpětné vazbu | Přeplánování doručení |
|---|---|---|---|---|---|---|
| DPD | + | + | - | + | - | + |
| Liftago | + | - | + | - | + | - |
| Shadowfax | + | - | - | - | - | - |
| Locate2u | + | + | - | + | + | - |
| Fixlastmile | + | - | + | + | + | - |
| Detrack | + | - | + | - | + | - |
| Souhrn | 100 % | 33 % | 50 % | 50 % | 50 % | 17 % |

\+ – funkce je
\- – funkce není

Tabulka 2.2: Funkce v kurýrních společnostech s možností sledování v reálném čase

## 2.3 Analýza požadavků

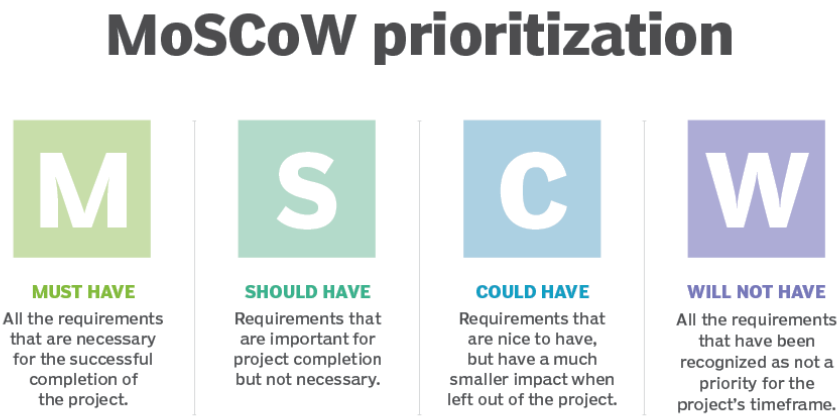
Z analýzy je vidět, že pouze 17 % zkoumaných řešení má funkci přeplánování nebo změny termínu doručení.

Jenom ve 33 % existuje možnost v reálném čase sledovat počet adres, který kurýrovi zbývá do zákazníka.

Dynamická změna času, možnost poslat zprávu kurýrovi nebo zanechat zpětnou vazbu vyskytuje v 50 %.

V žádném z řešení nejde nic dělat s mapou, kromě sledování kurýra, což by šlo změnit tak, aby se mapa používala na doplňující funkce.

Na základě této rešerše a zadání práce lze odvodit požadavky na aplikaci. Jednotlivé požadavky rozdělím na funkční, nefunkční a dále také podle metody MoSCoW [11] do čtyř kategorií: must have, should have, could have, will not have (viz obrázek 2.11).




Obrázek 2.11: MoSCoW Prioritization [11]

### 2.3.1 Funkční požadavky

Funkční požadavky jsou vlastnosti nebo funkce produktu, které musí vývojáři implementovat, aby uživatelům umožnili provádět jejich úkoly. Funkční požadavky jsou shrnuty v tabulce (viz tabulka 2.3).

| № | Funkční požadavek | Priorita |
|---|---|---|
| F1 | Sledování kurýra na mapě v reálném čase | M |
| F2 | Možnost zadávat plán cesty kurýra | M |
| F3 | Možnost zanechání zpětné vazby | S |
| F4 | Možnost sledování aktuálního počtu zbývajících adres | S |
| F5 | Dynamická změna zbývajícího času do doručení | C |
| F6 | Možnost přeplánování nebo změny termínu doručení | W |

Tabulka 2.3: Funkční požadavky

#### [F1] Sledování kurýra na mapě v reálném čase

Požadavek vyplývá ze zadání práce, aplikace umožní uživateli v reálném čase sledovat polohu kurýra, který zásilku doručuje.

#### [F2] Možnost zadávat plán cesty kurýra

Aplikace umožní zadávat plán cesty, který následovně bude použit pro konstruování optimální cesty přes všechny adresy, jež plán obsahuje.

**[F3] Možnost zanechání zpětné vazby**

Aplikace umožní uživateli zanechat zpětnou vazbu o kurýrovi.

**[F4] Možnost sledování aktuálního počtu zbývajících adres**

Aplikace bude dynamicky měnit počet adres, který aktuálně zbývá od kurýra do zákazníka.

**[F5] Dynamická změna času, který zbývá kurýrovi do doručení zásilky**

Aplikace bude každou 1 minutu aktualizovat přibližný odhad času doručení.

**[F6] Možnost přeplánování nebo změny termínu doručení**

Aplikace umožní uživateli přeplánovat nebo změnit termín doručení zásilky.

### 2.3.2 Nefunkční požadavky

Nefunkční požadavky nesouvisejí s funkčností systému, definují, jak by měl systém fungovat. Nefunkční požadavky jsou shrnuty v tabulce (viz tabulka 2.4).

| № | Nefunkční požadavek | Priorita |
|---|---|---|
| N1 | Implementace backendu s využitím Spring Boot (Java) | M |
| N2 | Implementace frontendu s využitím React (TypeScript) | M |
| N3 | Webová aplikace | M |
| N4 | Rozšiřitelnost | M |
| N5 | Uživatelská přívětivost | S |
| N6 | Intuitivní rozhraní | S |
| N7 | Lokalizace | C |
| N8 | Mobilní aplikace | W |

Tabulka 2.4: Nefunkční požadavky

**[N1] Implementace backendu s využitím Spring Boot (Java)**

Požadavek vyplývá ze zadání práce. Serverová část aplikace bude napsána s použitím frameworku Spring Boot v programovacím jazyce Java.

**[N2] Implementace frontendu s využitím React (TypeScript)**

Požadavek vyplývá ze zadání práce. Klientská část aplikace bude napsána s použitím knihovny React v programovacím jazyce TypeScript.

**[N3] Webová aplikace**

Aplikace bude implementována formou webu.

**[N4] Rozšiřitelnost**

Aplikace bude snadno horizontálně a vertikálně rozšiřitelná pro budoucí vývoj. Systém bude rozdělen podle funkčností na jednotlivé moduly.

**[N5] Uživatelská přívětivost**

Design aplikace bude navrhnut tak, aby uživatel měl dobrý dojem při interakci se systémem. Design všech stránek bude minimalistický a podobný. Rozvržení všech stránek bude vypadat celistvě a bude také mířit na uživatelé telefonů a tabletů.

**[N6] Intuitivní rozhraní**

Aplikace bude navrhnuta tak, aby uživatel mohl rychle a jednoduše najít přesně to, co potřebuje.

**[N7] Lokalizace**

Aplikace umožní překládat obsah stránek do jiných jazyků pomocí standardních nástrojů pro vybrané technologie.

**[N8] Mobilní aplikace**

Bude vytvořena mobilní verzi, kterou půjde snadno stáhnout, nainstalovat a spustit.

Kapitola 3

# Návrh

Tato kapitola je věnována návrhu softwarového produktu pro sledování zásilky v reálném čase. Kapitola obsahuje návrh případů užití, výběr technologií a architektury systému, návrh databázového modelu, API a grafického uživatelského rozhraní. Ze zadání práce vyplývá, že systém bude realizován formou webové aplikace.

## 3.1 Případy užití

Případy užití jsou důležitou částí návrhu aplikace v softwarovém inženýrství. Případ užití je scénář, který popisuje interakci uživatele se softwarovým produktem za účelem dosažení konkrétního cíle. Typicky případy užití plynou z funkčních požadavků.

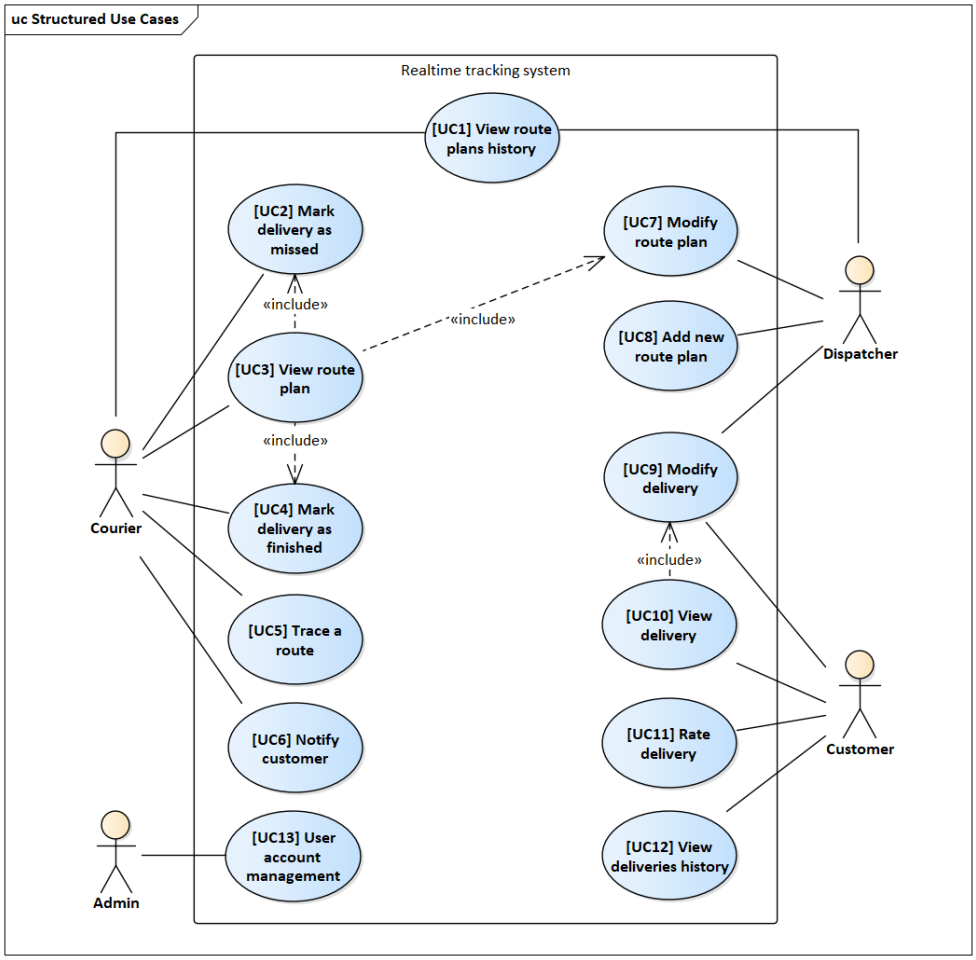
### 3.1.1 Seznam rolí

Uživatelé se dělí na jednotlivé role podle oprávnění k jednotlivým případům užití. Uživatelské role:

- **Dispečer** bude moci zadávat a spravovat plán cesty kurýra.
- **Zákazník** bude moci sledovat na mapě kurýra, doručujícího zásilku.
- **Kurýr** bude moci notifikovat zákazníka o příjezdu a spravovat doručení.
- **Admin** bude moci spravovat účty, jež byly registrovány v systému.

### 3.1.2 Diagram případů užití

Případy užití jsou znázorněny pomoci UML diagramu, který je na obrázku 3.1. Následují jednotlivé případy užití včetně základních scénářů.




Obrázek 3.1: Diagram případů užití

### UC1 – View history of route plans

- Kurýr nebo dispečer otevře hlavní obrazovku.
- Přejde do své sekce a uvidí seznam plánů.

### UC2 – Mark delivery as missed

- Kurýr otevře aktivní plán cesty.
- Na aktuálním doručení klikne na křížek.

### UC3 – View route plan

- Kurýr otevře hlavní obrazovku.

- Přejde do svoje sekce.
- Na hlavní obrazovce svoje sekce rozklikne aktivní plán cesty.

**UC4 – Mark delivery as finished**

- Kurýr otevře aktivní plán cesty.
- Na aktuálním doručení klikne na dýmku.

**UC5 – Trace a route**

- Kurýr otevře aktivní plán cesty.
- Na aktuálním doručení zkopíruje adresu.
- Pomocí vedlejší aplikace vyhledá cestu do zákazníka.

**UC6 – Notify customer**

- Kurýr otevře aktivní plán cesty.
- Na aktuálním doručení klikne na telefonní knihu.
- Na vyskakovacím okně uvidí kontakt na zákazníka a kontaktuje jej.

**UC7 – Modify route plan**

- Dispečer otevře hlavní obrazovku.
- Přejde so svoje sekce.
- Ze seznamu plánů vybere ten, který potřebuje.
- Rozklikne vybraný plán.
- Upraví položky.

**UC8 – Add new route plan**

- Dispečer otevře hlavní obrazovku.
- Přejde so svoje sekce.
- Vedle seznamu plánů klikne na tlačítko „Add new“.
- Vyplní všechna políčka formuláře.
- Stiskne tlačítko „Add“ pod formulářem.

**UC9 – Modify delivery**

- Dispečer nebo zákazník otevře hlavní obrazovku.
- Dispečer vybere a rozklikne existující plán cesty.
- Ze seznamu doručení vybere to, které potřebuje.
- Rozklikne vybrané doručení, zobrazí se mu formulář.
- Upraví doručení.
- Stiskne tlačítko „Save“ pod formulářem.

**UC10 – View delivery**

- Zákazník otevře hlavní obrazovku.
- Ze seznamu doručení vybere to, které potřebuje.
- Rozklikne vybrané doručení.

**UC11 – Rate delivery**

- Zákazník otevře hlavní obrazovku.
- Ze seznamu doručení vybere to, které potřebuje.
- Rozklikne vybrané doručení.
- Ve formuláři hodnocení, který se automaticky otevře po úspěšném doručení, ohodnotí kurýra.

**UC12 – View deliveries history**

- Zákazník otevře hlavní obrazovku.
- Pod seznamem aktuálních doručení uvidí historii doručení ve formě seznamu.

**UC13 – User account management**

- Admin otevře hlavní obrazovku.
- Na hlavní obrazovce ze seznamu registrovaných uživatelů vybere účet, který potřebuje.
- Rozklikne uživatelský účet a upraví jej.

## 3.2 Technologie

Systém lze rozdělit na tří dílčí části. První částí je frontend aplikace. Tato část se stará o uživatelské rozhraní. To je vše, co uživatel vidí, když otevře webovou stránku a s čím interaguje. Frontend také neustále komunikuje se serverem, neboli backendem aplikace. Druhou částí je backend, jež se stará o logiku aplikace a je zodpovědný za interakci s interními daty. Poslední částí je databáze, do níž se ukládají data, jež je třeba uchovávat.

### 3.2.1 Frontend

Táto sekce se zaměřuje na popis jednotlivých technologií, které budou použity při implementaci frontendové části systému. Použití knihovny React s programovacím jazykem TypeScript vyplývá ze zadání práce.

#### 3.2.1.1 TypeScript

TypeScript je silně typovaný programovací jazyk, který staví na JavaScriptu a poskytuje lepší nástroje v jakémkoli měřítku. [20]

Typ `any` se však často používá jako typ v jazyce TypeScript, což je nadmnožina jazyka JavaScript, která do jazyka přidává volitelné statické typování. V jazyce TypeScript se typ `any` používá k označení proměnné nebo hodnoty, která může mít libovolný typ, podobně jako se v jazyce JavaScript používá klíčové slovo `var`.

**Hlavní výhody:**

- Implementuje volitelné statické typování.
- Běží kdekoli běží JavaScript díky transpilaci.
- Podporuje těsnější integraci s editorem.

#### 3.2.1.2 React

React je deklarativní, efektivní a flexibilní JavaScriptová knihovna pro vytváření uživatelských rozhraní. Umožňuje skládat složitá uživatelská rozhraní z malých a izolovaných částí kódu nazývaných „komponenty“. [21]

**Hlavní výhody:**

- JSX je kombinací HTML a JavaScriptu. Do prvků HTML můžete vložit objekty JavaScriptu.
- Možnost vykreslovat na serveru pomocí Node.js a pohánět mobilní aplikace pomocí React Native.
- Virtual DOM.

### 3.2.2 Backend

Táto sekce se zaměřuje na popis jednotlivých technologií, které budou použity při implementaci backendové části systému. Použití frameworku Spring Boot, který je nadstavbou nad Spring Framework, vyplývá ze zadání práce. Jelikož Spring Security a Spring Data patří do ekosystému Spring, této technologie jsou primární volbou pro implementaci zabezpečení a pro přistup k datům. Dále následuje popis důležitých technologií, které budou použity při implementaci systému.

#### 3.2.2.1 Spring Framework

Spring Framework poskytuje komplexní programovací a konfigurační model pro moderní enterprise aplikace založené na Javě – na jakémkoli druhu platformy nasazení. [12]

Klíčovým prvkem Springu je podpora infrastruktury na aplikační úrovni: Spring se zaměřuje na „instalaci“ enterprise aplikací, aby se týmy mohly soustředit na byznys logiku na aplikační úrovni, bez zbytečných vazeb na konkrétní prostředí nasazení. [12]

#### 3.2.2.2 Spring Boot

Spring Boot je nadstavba nad Spring Framework, která automatizuje proces konfigurace, urychluje a usnadňuje proces vytváření a nasazení Springových aplikací. [13]

**Hlavní výhody:**

- Snadné řízení závislostí.
- Automatická konfigurace všech komponent pro Spring aplikaci na produkční úrovní.
- Nativní podpora aplikačního serveru.
- Jedna z nejmodernějších a nejpoužívanějších technologií pro řízení závislostí.

#### 3.2.2.3 Spring Security

Spring Security je výkonný a vysoce přizpůsobitelný framework pro autentizaci a řízení přístupu. Je to de facto standard pro zabezpečení aplikací na bázi Spring. [14]

Spring Security je framework, který se zaměřuje na poskytování autentizace a autorizace Java aplikacím. Stejně jako všechny projekty Spring spočívá skutečná síla Spring Security v tom, jak snadno jej lze rozšířit tak, aby splňoval vlastní požadavky. [14]

**Hlavní výhody:**

- Komplexní a rozšiřitelná podpora pro autentizaci i autorizaci.
- Ochrana proti útokům, jako je session fixation, clickjacking, cross site request forgery atd.
- Volitelná integrace se Spring Web MVC.

#### 3.2.2.4 Spring Data

Hlavním cílem Spring Data je poskytnout známý a konzistentní springový programovací model pro přístup k datům a přitom si zachovat speciální vlastností základního datového úložiště. [15]

Usnadňuje použití technologií pro přístup k datům, relačních a nerelačních databází, map-reduce frameworků a cloudových datových služeb. Jedná se o zastřešující projekt, který obsahuje mnoho dílčích pod-projektů, které jsou specifické pro danou databázi. Projekty jsou vyvíjeny ve spolupráci s mnoha společnostmi a vývojáři, kteří stojí za těmito technologiemi. [15]

**Hlavní výhody:**

- Výkonné úložiště a vlastní abstrakce mapování objektů.
- Dynamické odvozování dotazů z názvů metod úložiště.
- Podpora transparentního auditování (vytvořeno, naposledy změněno).
- Možnost integrace vlastního kódu úložiště.
- Snadná integrace se Spring prostřednictvím JavaConfig a vlastních jmenných prostorů XML.

#### 3.2.2.5 Gradle

Gradle je open-source nástroj pro automatizaci sestavování zaměřený na flexibilitu a výkon. Skripty sestavení Gradle jsou psány pomocí Groovy nebo Kotlin DSL. [16]

Jelikož Gradle má větší výkon, nepotřebuje kompilaci Java kódu a je přizpůsobitelnější než Maven [17], proto bude použit v tomto projektu.

**Hlavní výhody:**

- Vysoce přizpůsobitelné – Gradle je modelován způsobem, který je přizpůsobitelný a rozšiřitelný těmi nejzákladnějšími způsoby.

- Rychlý – Gradle dokončuje úlohy rychle opakovaným použitím výstupů z předchozích spuštění, zpracováním pouze změněných vstupů a paralelním prováděním úloh.

- Výkonný – Gradle je oficiální nástroj pro sestavení pro Android a přichází s podporou mnoha oblíbených jazyků a technologií jako Java, C++, Kotlin atd.

#### 3.2.2.6 OpenAPI

Specifikace OpenAPI (OAS) definuje standardní, jazykově agnostické rozhraní pro HTTP APIs, které umožňuje lidem i počítačům objevovat a chápat schopnosti služby bez přístupu ke zdrojovému kódu, dokumentaci nebo prostřednictvím kontroly síťového provozu. Když je správně definován, může spotřebitel rozumět vzdálené službě a komunikovat s ní s minimálním množstvím implementační logiky. [18]

OpenAPI definici pak mohou používat nástroje pro generování dokumentace k zobrazení API, nástroje pro generování kódu serverů a klientů v různých programovacích jazycích, testovací nástroje a mnoho dalších případů použití. [18]

Především z důvodu zkušeností a znalostí autora byl vybrán právě tento nástroj. Existující alternativy, například jako je RAML [19], nenabízejí oproti OpenAPI nic navíc.

**Hlavní výhody:**

- Zjednodušení procesu sestavování rozhraní API.

- Generování kódu serverů a klientů v mnoha programovacích jazycích.

### 3.2.3 Databáze

Databáze je místo pro ukládání dat. Jelikož kurýrské služby evidují dostatečné množství různých dat, je lepší použit samostatnou databázi, v níž budou tato data uchovávána. Existuje mnoho různých typů databází, které se liší způsobem ukládání, zpracování a indexování dat. K těm nejvýznamnějším patří SQL a NoSQL databáze.

Jelikož v tomto systému databázové schéma se nebude často měnit a data budou mezi sebou provázaná, bude použita relační (SQL) databáze místo nerelační (NoSQL), z toho důvodu, že relační databáze mají pevné, statické nebo předdefinované schéma a jsou nejvhodnější pro složité dotazy, ačkoliv i jsou pouze vertikálně rozšiřitelné. [22]

Byla vybrána relační databáze PostreSQL, která patří mezi nejvíce používané a často se používá v produkci webových aplikací.

#### 3.2.3.1 PostgreSQL

PostgreSQL je výkonný objektově relační databázový systém s otevřeným zdrojovým kódem, který využívá a rozšiřuje jazyk SQL v kombinaci s mnoha funkcemi, které bezpečně ukládají a škálují ty nejsložitější datové zátěže. Počátky PostgreSQL se datují do roku 1986 jako součást projektu POSTGRES na University of California v Berkeley a má za sebou více než 35 let aktivního vývoje na základní platformě. [23]

PostgreSQL si získal silnou reputaci pro svou osvědčenou architekturu, spolehlivost, integritu dat, robustní sadu funkcí, rozšiřitelnost a odhodlání komunity s otevřeným zdrojovým kódem za softwarem trvale poskytovat výkonná a inovativní řešení. PostgreSQL běží na všech hlavních operačních systémech, je kompatibilní s ACID od roku 2001 a má výkonné doplňky, jako je populární rozšiřovač geoprostorových databází PostGIS. Není žádným překvapením, že PostgreSQL se stal open-source relační databází pro mnoho lidí a organizací. [23]

## 3.3 Architektura

Architektura aplikace definuje strukturu a popisuje techniky používané pro návrh a sestavení systému, představuje rozložení všech komponent a jejich vzájemnou interakci. [26]

V této aplikaci bude použita architektura klient-server. Komunikace mezi částmi bude probíhat pomoci REST rozhraní přes HTTP protokol.

### 3.3.1 Klient

Jelikož rychlost zobrazení informace při sledování zásilky hraje velmi důležitou roli a rendering na stráně klienta, neboli Client Side Rendering (CSR), oproti renderingu na stráně serveru, neboli Server Side Rendering (SSR), má výhody v rychlosti dynamické aktualizace dat bude použit rendering na stráně klienta.

Existuje dva významných návrhových vzory webových aplikací: Single-Page Application (SPA) a Multi-Page Application (MPA). V následujících paragrafech jsou tyto vzory rozebrány.

#### 3.3.1.1 Single-Page Application (SPA)

Jednostránková aplikace, také známá jako SPA, je typ webové aplikace, která umožňuje pracovat na jedné HTML stránce. Díky technologii AJAX je možné zobrazovat obsah bez opětovného načítání stránky, když ji člověk používá. Při použití tento kód, který obvykle závisí na JavaScript frameworku, zaručuje vysokou rychlost SPA. [24]

#### 3.3.1.2 Multi-Page Application (MPA)

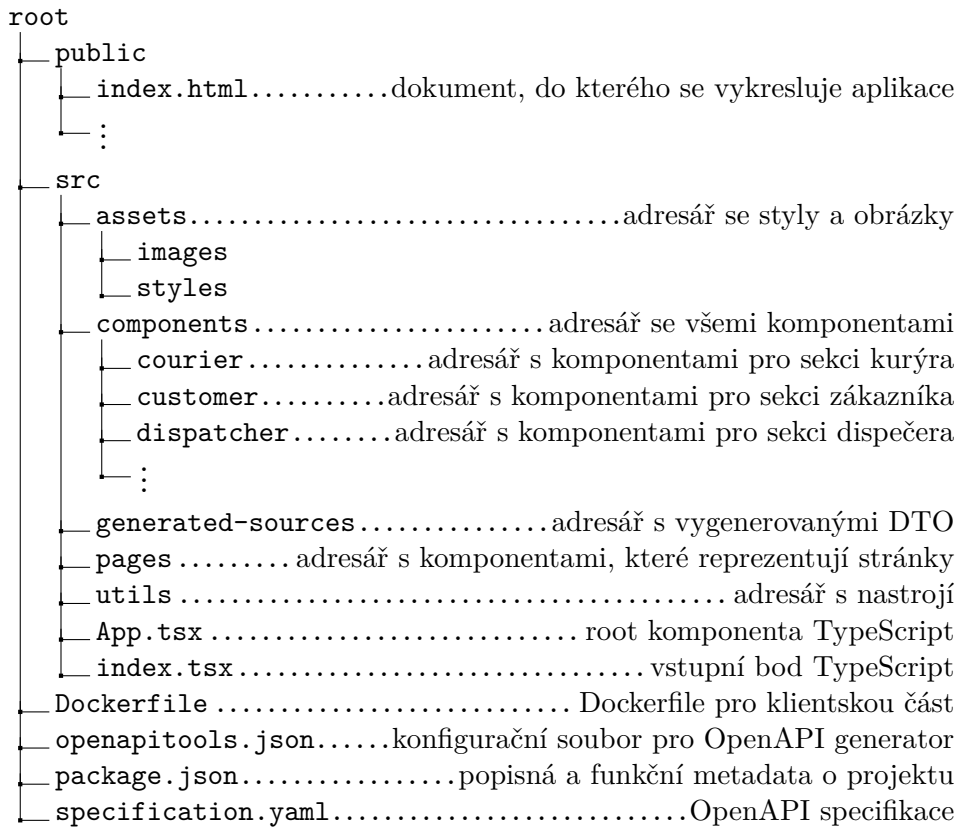
Jak název napovídá, vícestránková aplikace (nebo MPA) obsahuje mnoho webových stránek stažených při přístupu uživatelů k různým oblastem webových stránek. Toto je standardní přístup k vývoji webových aplikací pro weby, které potřebují zpracovávat velké objemy obsahu. [24]

#### 3.3.1.3 Výběr návrhového vzoru

Jelikož aplikace nepotřebuje zobrazovat velké množství dat, ale je důležité, aby stránka byla co nejvíce interaktivní, byla vybrána modernější a rychlejší varianta Single-Page Application (SPA).

#### 3.3.1.4 Struktura frontendu

Přehlednost kódu je jedna z nejdůležitějších vlastností aplikace, kterou je snadné udržovat a rozšiřovat. Struktura frontendové části aplikace je znázorněna na obrázku 3.2.



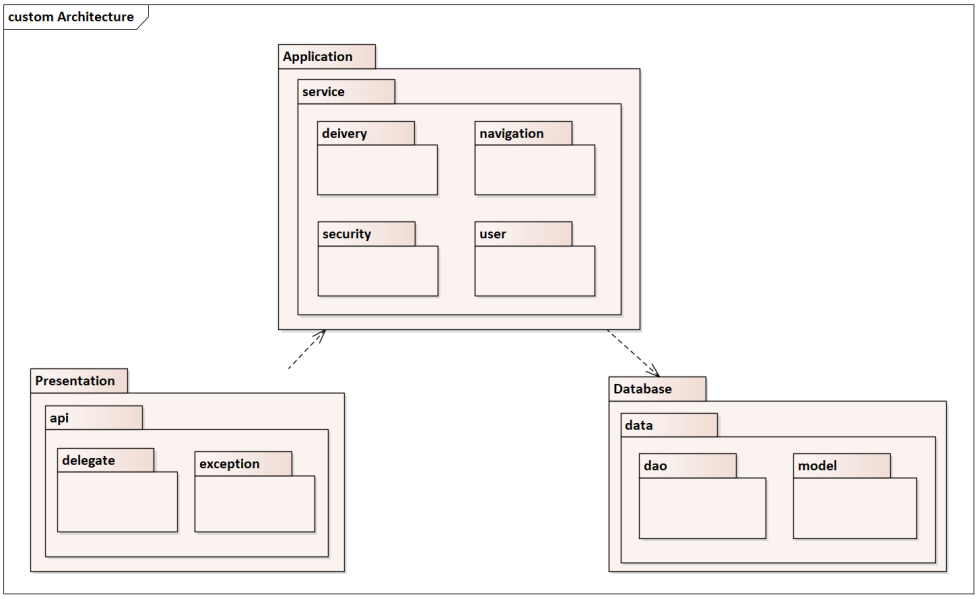

Obrázek 3.2: Popis struktury frontendu

### 3.3.2 Server

V tomto systému bude vhodné použit třívrstvou architekturu, která oproti dvouvrstvé a jednovrstvé má zásadní výhody a je nejběžnějším typem vícevrstvé architektury v distribuovaných systémech. [25]

Hlavní výhodou tří vrstev v architektuře klient-server je to, že tyto vrstvy jsou vyvíjeny a udržovány nezávisle a v případě jakékoli úpravy by to nemělo vliv na ostatní vrstvy. Toto umožňuje lepší výkon a lze dosáhnout ještě větší škálovatelnosti architektury, protože s rostoucí poptávkou lze přidat více serverů. [25]

Diagram třívrstvé architektury je znázorněn na obrázku 3.3.




Obrázek 3.3: Třívrstvá architektura

#### 3.3.2.1 Prezentační vrstva

Prezentační vrstva je uživatelské rozhraní a komunikační část aplikace, přes kterou koncový uživatel komunikuje s aplikací. Hlavním účelem je zobrazovat a shromažďovat informace pro uživatele. Tato vrstva nejvyšší úrovně může běžet například ve webovém prohlížeči, jako desktopová aplikace nebo jako grafické uživatelské rozhraní (GUI). [26]

#### 3.3.2.2 Aplikační vrstva

Aplikační vrstva, známá také jako logická vrstva nebo střední vrstva, je srdcem aplikace. V této vrstvě se informace shromážděné v prezentační vrstvě zpracovávají – někdy proti jiným informacím v datové vrstvě – pomocí ob-

chodní logiky a specifické sady obchodních pravidel. Aplikační vrstva může také přidávat, odstraňovat nebo upravovat data v datové vrstvě. [26]

#### 3.3.2.3 Datová vrstva

Datová vrstva, někdy ještě nazývaná databázová vrstva nebo vrstva pro přístup k datům, je místem, kde jsou uloženy a spravovány informace zpracovávané aplikace. [26]

## 3.4 Databázový model

Diagram tříd ukazuje logickou strukturu databáze, včetně vztahů a omezení, která určují, jak lze data ukládat a přistupovat k nim. Databázový model je znázorněn na obrázku 3.4.

## 3.5 API

V této sekci jsou popsány jednotlivé API endpointy, které se používají nejvíce nebo jsou specifické z hlediska implementace.

### 3.5.1 `/login`

Endpoint slouží pro login uživatele. Vyžaduje uživatelské jméno a heslo. Vrací tokeny pro přístup a jejich obnovu. Vrací role tohoto uživatele.

### 3.5.2 `/users/token/refresh`

Endpoint slouží pro obnovu tokenů. Vrací obnovené tokeny.

### 3.5.3 `/couriers/{username}/position`

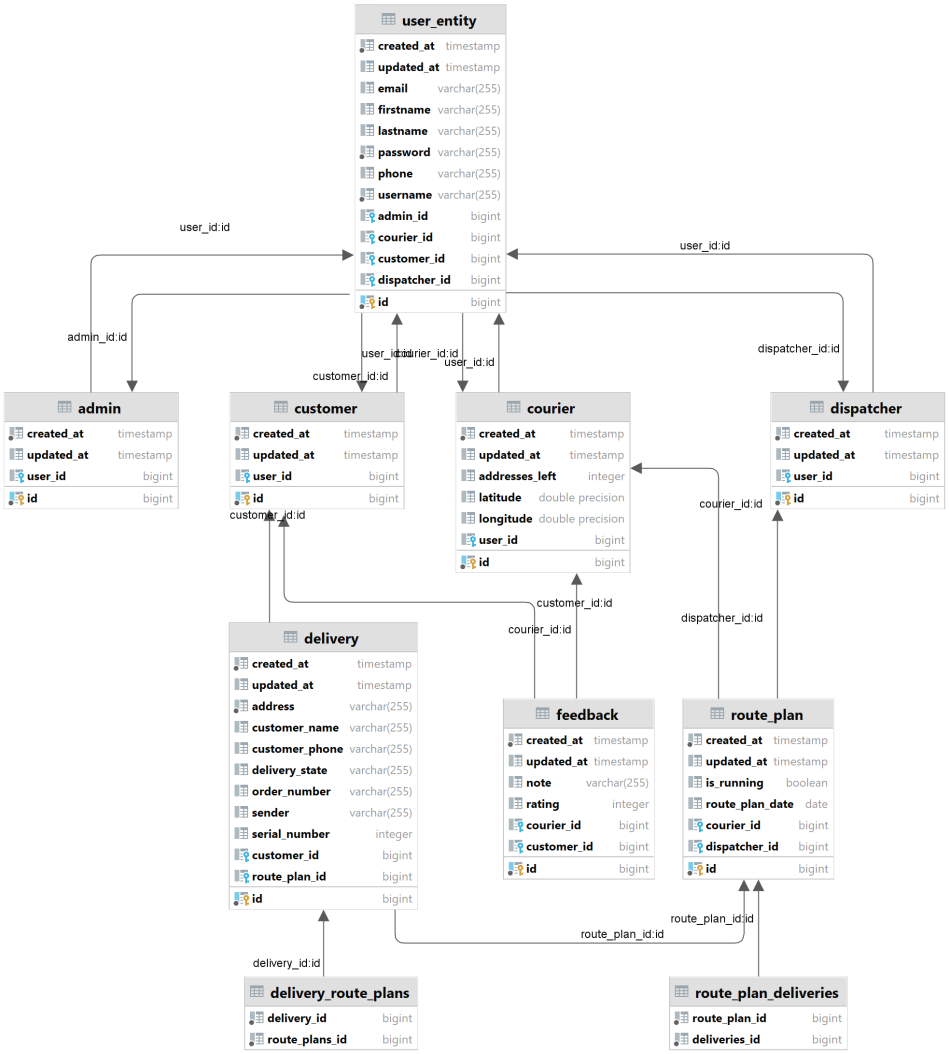
Endpoint slouží pro obnovu polohy kurýra. Vyžaduje nové souřadnice polohy kurýra. Vrací informace o tomto kurýrovi s obnovenou polohou.

### 3.5.4 `/couriers/{username}/info`

Endpoint vyžaduje uživatelské jméno kurýra. Vrací stručné informace o tomto kurýrovi pro zákazníka.

### 3.5.5 `/dispatchers/{username}/info`

Endpoint vyžaduje uživatelské jméno dispečera. Vrací stručné informace o tomto dispečeru pro kurýra.




Obrázek 3.4: Databázový model

### 3.5.6 `/couriers/id/{id}/info`

Endpoint je podobný `/dispatchers/id/{id}/info`, tyto endpointy očekávají na vstupu `id`. Jelikož v URL `/username` a `/id` není možné rozlišit, bylo vyřešeno přidat `/id` do URL, který na vstupu vyžaduje `id`. Vracejí stejné informace jako endpointy 3.5.4 a 3.5.5 odpovědně.

### 3.5.7 `/customers/{username}/deliveries/active`

Endpoint podobný `/customers/{username}/deliveries/history`, vyžaduje uživatelské jméno zákazníka na vstupu. Vrací aktivní, případně, historická doručení tohoto zákazníka.

### 3.5.8 `/routePlans/{routePlanId}/start`

Endpoint slouží pro zahájení plánu cesty. Vyžaduje na vstupu `id` plánu cesty.

### 3.5.9 `/navigation/route`

Endpoint slouží pro hledání cesty mezi jednotlivými adresami. Na vstupu vyžaduje `origin` - souřadnice první adresy, `destination` - souřadnice poslední adresy a `waypoints` - seznam souřadnic adres mezi první a poslední. Vrací informace pro zobrazení cesty na mapě. Endpoint byl připraven pro další rozšíření systému.

## 3.6 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (GUI) je důležitá část aplikace, kterou vidí uživatel a s čím může interagovat. V této sekci byl navržen vzhled nejdůležitějších stránek s ohledem na to, aby rozhraní bylo intuitivní, uživatelsky přívětivé a snahou autora bylo dodržet jednotný vizuální styl. Pro návrh wireframů byl použit nástroj Balsamiq Mockups.

### 3.6.1 Registrace a přihlášení

Je to úvodní stránka, na kterou se dostane každý nepřihlášený uživatel. Pokud návštěvník již má účet, tak se může do něj přihlásit, v opačném případě by mohl přejít na stránku registrace nového účtu pomocí tlačítka „Create an account“. Návrh této stránky lze vidět na obrázku 3.5.

### 3.6.2 Hlavní obrazovka pro kurýra

Je to hlavní obrazovka, na kterou se po přihlášení dostane uživatel, jež má kurýrský účet. Na této stránce by kurýr mohl vidět aktivní plán cesty, pokud

nějaký má, a také historii plánů. Po rozkliknutí nějakého plánu se kurýr dostane na stránku, jež obsahuje adresy doručení. Návrh této stránky lze vidět na obrázku 3.6.

### 3.6.3 Doručení zásilky

Po rozkliknutí aktivního plánu se kurýr dostává na stránku doručení zásilky. Na této stránce může vidět adresu, na kterou by měl další zásilku doručit, a zbývající. Vedle aktivní adresy jsou tlačítka, která slouží pro sestavení cesty a přechodu do navigátoru, zavolání zákazníkovi, označení, že zásilka byla úspěšně doručena nebo zameškaná. Návrh této stránky lze vidět na obrázku 3.7.

### 3.6.4 Historie adres

Po rozkliknutí nějakého plánu z historie se kurýrovi zobrazí seznam adres, na něž zásilky během tohoto plánu doručoval. Vedle adresy je vidět jestli zásilka byla doručena nebo zameškaná. Návrh této stránky lze vidět na obrázku 3.8.

### 3.6.5 Hlavní obrazovka pro zákazníka

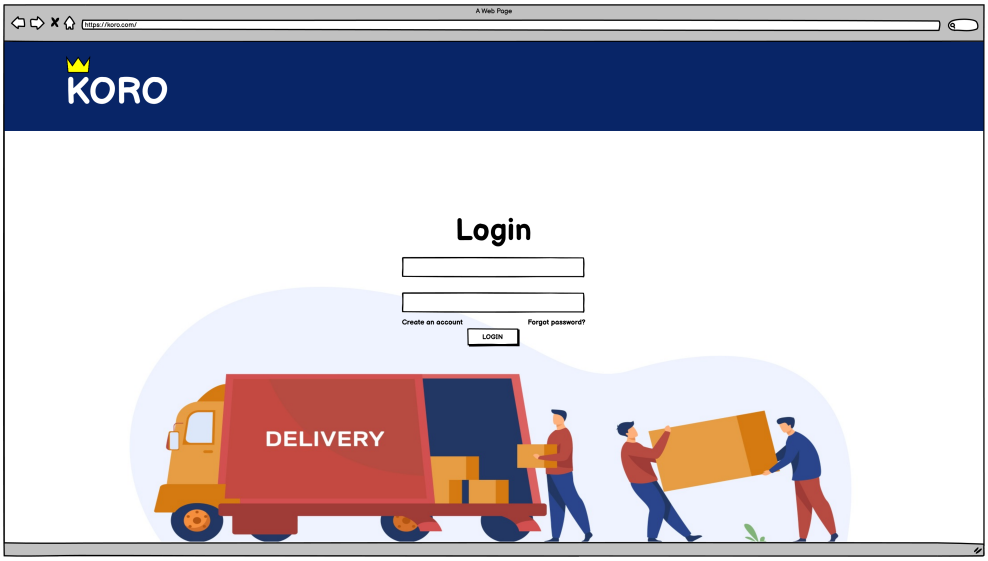
Je to hlavní obrazovka, na kterou se po přihlášení dostane uživatel, jež má zákaznický účet. Na stránce může vidět aktivní doručení, na která čeká, a jejich historii. Návrh této stránky lze vidět na obrázku 3.9.

### 3.6.6 Sledování zásilky

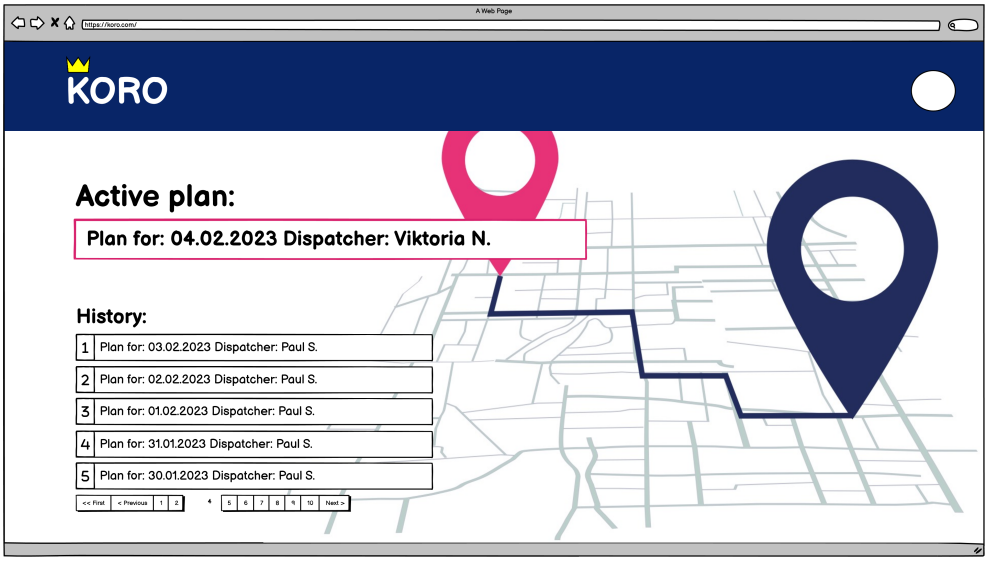
Po rozkliknutí nějakého aktivního doručení se zákazníkovi zobrazí mapa, na které může sledovat polohu kurýra v reálném čase, počet adres, který zbývá od kurýra do zákazníka, předpokládaný čas doručení a také informace o zásilce a kurýrovi, který tuto zásilku doručuje. Návrh této stránky lze vidět na obrázku 3.10.

### 3.6.7 Hodnocení doručení

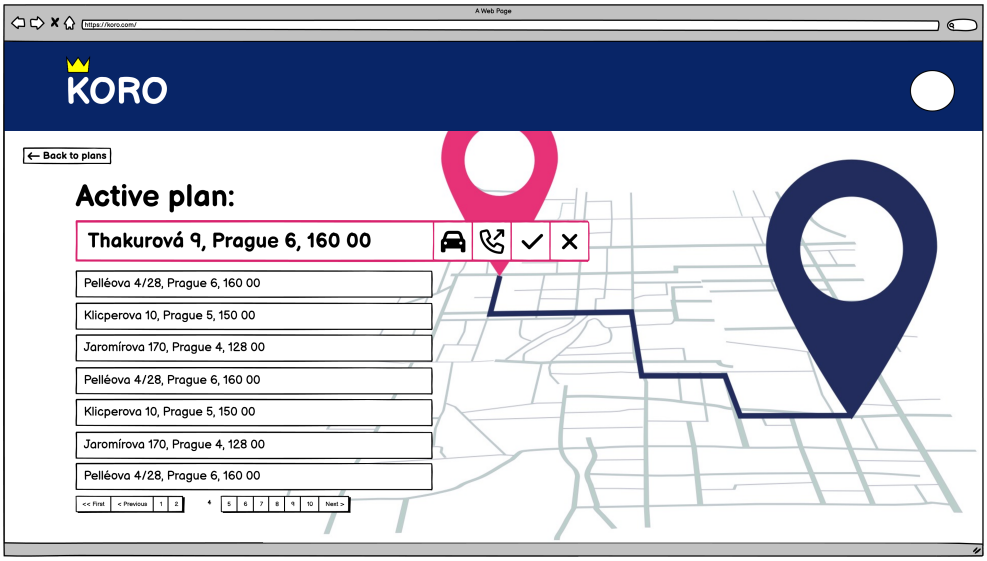
Po úspěšném doručení se zákazníkovi zobrazí stránka, na které může ohodnotit aplikaci. Návrh této stránky lze vidět na obrázku 3.11.



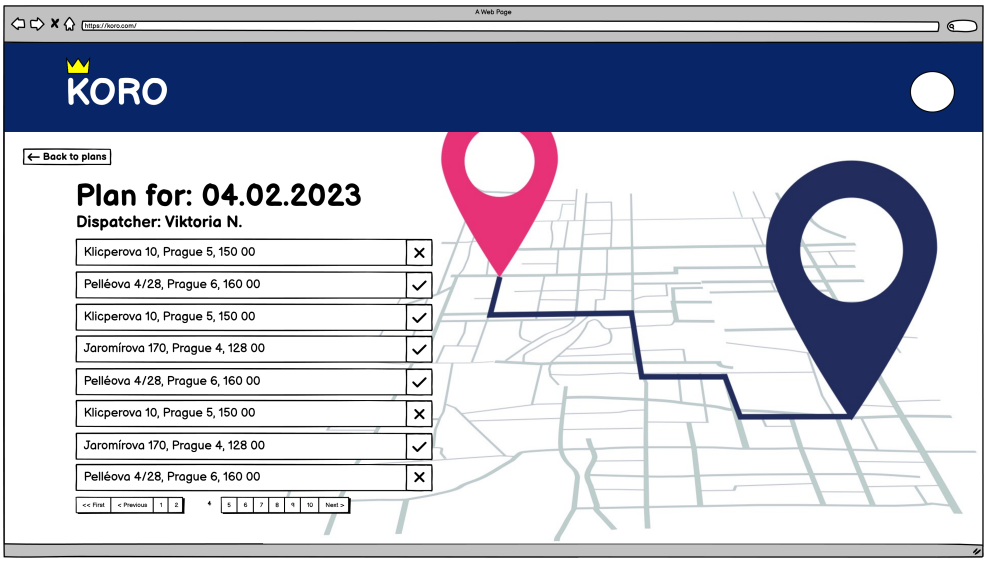

Obrázek 3.5: Registrace a přihlášení



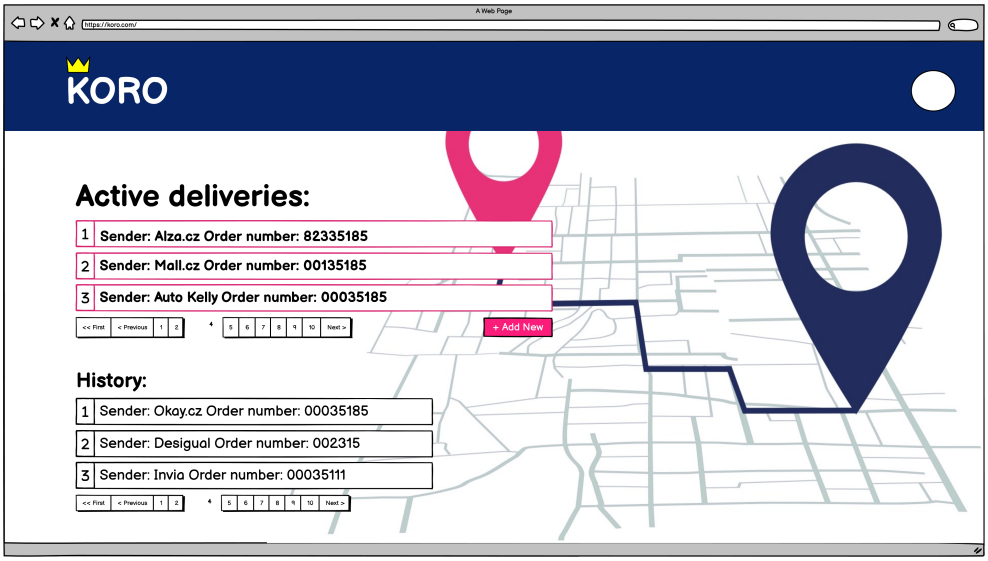

Obrázek 3.6: Hlavní obrazovka pro kurýra



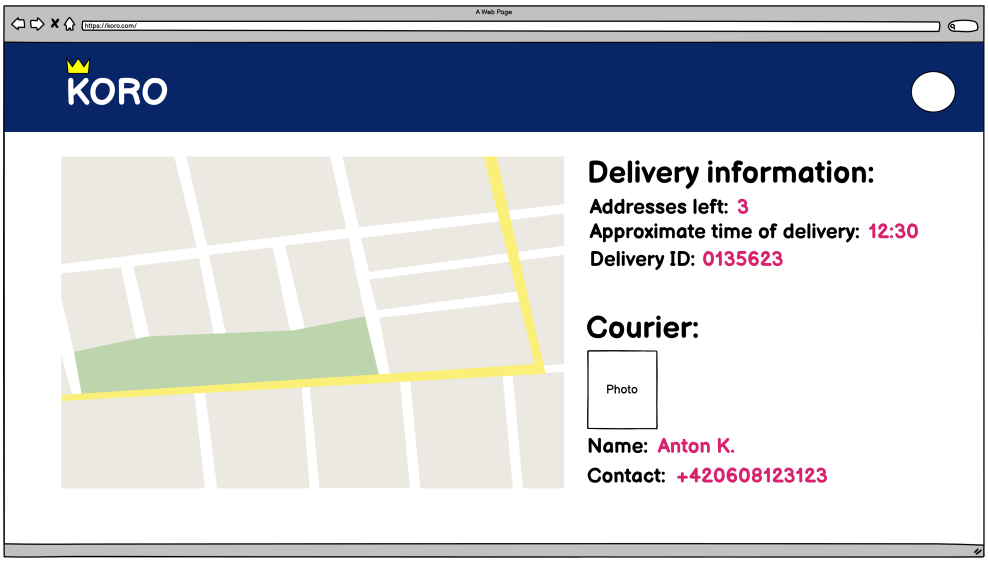

Obrázek 3.7: Doručení zásilky



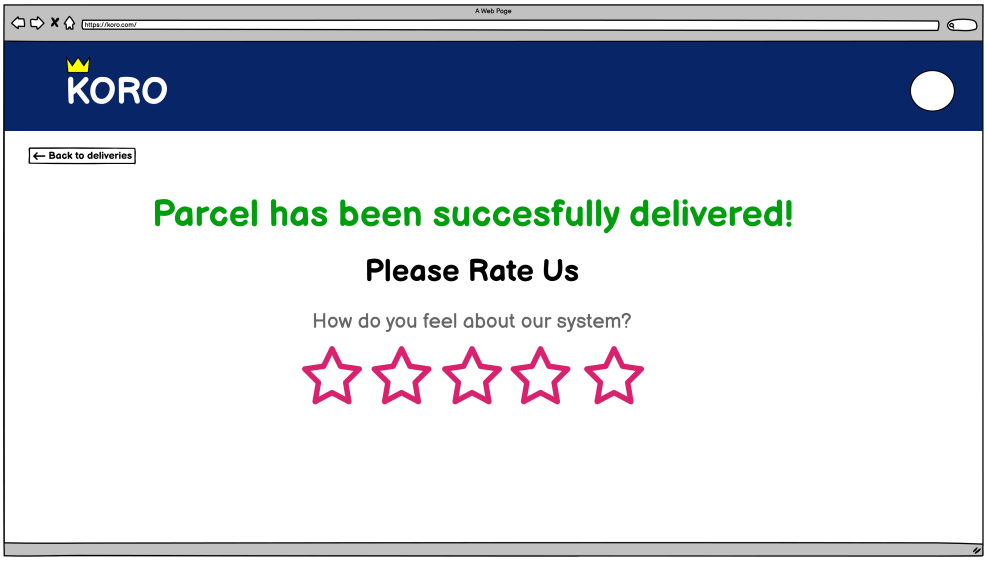

Obrázek 3.8: Historie adres




Obrázek 3.9: Hlavní obrazovka pro zákazníka




Obrázek 3.10: Sledování zásilky




Obrázek 3.11: Hodnocení doručení

Kapitola 4

# Implementace

Po úspěšné analýze a návrhu následuje implementační fáze. Tato kapitola je věnována implementaci navrhovaného systému s použitím vybraných technologií a podle stanovených požadavků.

## 4.1 API pro sledování v reálném čase

Existuje velké množství různých API, které nabízejí možnost sledování na mapě v reálném čase.

Nejvýznamnější z nich je Google Maps API [27], které má největší počet uživatelů a rozsáhlou dokumentaci s velkým množství příkladů, ale toto API nebylo vybráno pro použití v této práci z toho důvodu, že je drahé a není žádná možnost použití tohoto API zdarma.

Jedno z dalších nejvýznamnějších a nejvíce používaných API pro práci s mapou a geocoding je Mapbox API [28], které oproti Google Maps API nabízí možnost použití zdarma. Sice je nutné v účtu zadat údaje platební karty, aby bylo možné dostat token, a počet requestů je omezený na 50000 za jeden měsíc, možnosti tohoto API jsou postačující pro splnění účelu této bakalářské práce.

## 4.2 Implementační poznámky

### 4.2.1 Sledování kurýra v reálném čase

Na ukázce kódu 4.1 je znázorněna ukázka inicializace mapy v klientské části aplikace. Je důležité zajistit, aby mapa byla inicializována pouze jednou a do ní určitě byl přidán `GeoJSONSource` s `id`: `courier`.

```
useEffect(() => {
  // map initialization - have to be initialized only once
  if (map.current) return;
```

```
    map.current = new mapboxgl.Map({
       container: mapContainer.current || '',
       style: 'mapbox://styles/mapbox/streets-v12',
       center: [courierDefaultLng, courierDefaultLat], // Prague
           center
       zoom: 10
    });

    // adding courier's default position
    map.current.on('load', async () => {
       map.current!.addSource('courier', {
          type: 'geojson',
          data:
          {
             'type': 'FeatureCollection',
             'features': [
             {
                'type': 'Feature',
                "properties": {
                   "name": "courier"
                },
                'geometry': {
                   'type': 'Point',
                   'coordinates': [courierDefaultLng,
                       courierDefaultLat]
                }
             }
             ]
          }
       });

       // adding courier symbol on the map as a layer
       map.current!.addLayer({
          'id': 'courier',
          'type': 'symbol',
          'source': 'courier',
          'layout': {
             'icon-image': 'rocket',
             'icon-size': 1.5
          }
       });
    });

 }, []);
```

Ukázka kódu 4.1: Inicializace mapy

Na ukázce kódu 4.2 je znázorněna ukázka časovače, který se používá pro aktualizaci polohy kurýra. Časovač je nastaven na 5 vteřin.

```
useEffect(() => {
   const interval = setInterval(() => {
      setTime(Date.now())
   }, 5000);
   return () => {
      clearInterval(interval);
   };
}, [])
```

Ukázka kódu 4.2: Časovač

Na ukázce kódu 4.3 je znázorněna ukázka aktualizace polohy kurýra na mapě. Nejdříve se ověří jestli aktuální doručení ještě probíhá a následovně se zavolá funkce `getCourierLocation()`, která vrátí polohu kurýra ve formátu `GeoJSON`. Pokud tato funkce vrátí validní data, aktualizuje se zatím `GeoJSONSource` s `id`: `courier`, které slouží pro zobrazení objektu na správném místě mapy, pomocí metody `setData(GeoJSONSource source)` z knihovny `mapboxgl`.

```
useEffect(() => {
   if (courierInit && delivery?.deliveryState != "FINISHED" &&
       delivery?.deliveryState != "MISSED") {
      const courierLocationData: GeoJSON.FeatureCollection |
          undefined = getCourierLocation(); // get courier
          location in GeoJSON format
      if (courierLocationData) {
         const source = map.current!.getSource('courier') as
             mapboxgl.GeoJSONSource;
         if (source) {
            source.setData(courierLocationData)
         }
      }
   }

   // update delivery, check if finished
   if (isConfReady(conf) && !isEvaluatedDelivery) {
      deliveryAPI.getDelivery({deliveryId: deliveryId})
      .then((delivery) => {
         setDelivery(delivery);
         if (delivery?.deliveryState === "FINISHED") {
            setIsFinishedDelivery(true);
         }
      })
   }
}, [time]);
```

Ukázka kódu 4.3: Aktualizace polohy kurýra na mapě pro zákazníka

Na ukázce kódu 4.4 je znázorněna ukázka zjištění polohy kurýra ze serveru pomocí funkce `getCourierLocation()`. Pokud kurýr ma nastaveno souřadnice `longitude` a `latitude`, funkce vrátí validní data ve formátu `GeoJSON`, v opačném případě vrátí `undefined`.

```
function getCourierLocation(): GeoJSON.FeatureCollection |
    undefined {
  if (courierInit && isConfReady(conf)) {
    courierAPI.getCourierInfo({username: courierUsername,
        deliveryId: deliveryId})
    .then((courier) => {
      setCourier(courier);
    })
    .catch((error) => {
      console.log(error)
    })
  }

  // return courier location in GeoJSON
  if (courier?.longitude && courier.latitude) {
    return {
      'type': 'FeatureCollection',
      'features': [
      {
        'type': 'Feature',
        "properties": {
          "name": "courier"
        },
        'geometry': {
          'type': 'Point',
          'coordinates': [courier.longitude!,
              courier.latitude!]
        }
      }
      ]
    };
  } else {
    return undefined;
  }
}
```

Ukázka kódu 4.4: Zjištění polohy kurýra

Na ukázce kódu 4.5 je znázorněna ukázka aktualizace polohy kurýra ze strany kurýra. Poloha kurýra se posílá na server jednou za 5 vteřin pomocí časovače, na serveru se ukládá do databáze, z níž potom v zákaznické části lze polohu kurýra dostat.

```
useEffect(() => {
```

```
    if (isConfReady(conf)) {
      navigator.geolocation.getCurrentPosition(function
          (position) {
        courierAPI.updateCourierPosition({
          username: localStorage.getItem("username")!,
          longitude: position.coords.longitude,
          latitude: position.coords.latitude
        })
        .then()
        .catch((error) => {
          console.log(error);
        })
      });
    }
  }, [time])
```

Ukázka kódu 4.5: Aktualizace polohy kurýra ze strany kurýra

### 4.2.2 Cyklické závislosti

V serverové části tohoto systému se používá Spring Boot verze 2.7.5. Od Spring Boot verze 2.6.0 jsou cyklické závislosti zakázané. Z toho důvodu, že pravidla třívrstvé architektury nejsou porušeny, a kvůli vzniklým cyklickým závislostem bylo vyřešeno použit `spring.main.allow-circular-references=true` v `application.properties`.

### 4.2.3 Použití BaseEntity

Výhoda třídy `BaseEntity` je v tom, že poskytuje všem rozšiřujícím třídám – jak název napovídá – základní funkcionalitu a nemusíte vytvářet id pro každou třídu. [29] Ukázka třídy `BaseEntity` je na ukázce kódu 4.6.

```
@MappedSuperclass
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class BaseEntity {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  @NotNull
  @Column(name = "id", nullable = false, updatable = false)
  private Long id;

  @CreationTimestamp
  @Column(name = "created_at", nullable = false, updatable =
      false)
```

```
    private Timestamp createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at")
    private Timestamp updatedAt;
}
```

Ukázka kódu 4.6: Třída BaseEntity

Každá třída v serverové části tohoto systému dědí ze třídy `BaseEntity`. Ukázku použití této třídy na příkladě třídy `Feedback` lze vidět na ukázce kódu 4.7.

```
@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "feedback")
public class Feedback extends BaseEntity {

    @Column
    private Integer rating;

    @Column
    private String note;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "customer_id")
    private Customer customer;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "courier_id")
    private Courier courier;
}
```

Ukázka kódu 4.7: Použití třídy BaseEntity

## 4.3 Bezpečnost

Autentizace a autorizace jsou dvě slova používaná ve světě bezpečnosti. Mohou znít podobně, ale jsou od sebe zcela odlišné.

**Autentizace** je akt ověření, že uživatelé jsou tím, za koho se vydávají. Toto je první krok v jakémkoli bezpečnostním procesu. [30]

**Autorizace** v zabezpečení systému je proces, který dává uživateli oprávnění k přístupu ke konkrétnímu zdroji nebo funkci. [30]

V tomto systému byla použita autorizace pomoci JWT tokenu místo páru username a heslo, jako to je v Basic Authentication, z toho důvodu, že je bezpečnější a má další výhody:

- JWT jsou časově omezené a jejich platnost po nějakém intervalu vyprší.
- JWT mohou být vydány jednou službou a ověřeny jinou.
- JWT mohou udělovat omezená oprávnění.
- JWT mohou být bezpečné a rychle ověřitelné najednou.

Na ukázce kódu 4.8 lze vidět, že všechny adresy, kromě `/signup` a `/login` vyžadují, aby uživatel byl přihlášen.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity httpSecurity)
    throws Exception {
  httpSecurity.csrf().disable()
  .cors(withDefaults())
  .sessionManagement()
  .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
  .and()
  .authorizeRequests()
  .antMatchers("/signup", "/login").permitAll()
  .anyRequest().authenticated()
  .and()
  .exceptionHandling()
  .authenticationEntryPoint(authenticationEntryPoint);

  httpSecurity.addFilter(new AppAuthenticationFilter(
  authenticationManager(httpSecurity.getSharedObject(
  AuthenticationConfiguration.class)), jwtTokenService));
  httpSecurity.authenticationProvider(authenticationProvider());
  httpSecurity.addFilterBefore(new AppAuthorizationFilter(
  jwtTokenService), UsernamePasswordAuthenticationFilter.class);

  return httpSecurity.build();
}
```

Ukázka kódu 4.8: Bezpečnost systému

V serverové části tohoto systému autorizace podle rolí probíhá pomocí anotace `@PreAuthorize`. Anotace `@PreAuthorize` se ve Spring Bootu používá k určení pravidel řízení přístupu pro metody v controlleru Spring MVC. Tuto anotaci lze použít k vynucení bezpečnostních omezení metod na základě rolí nebo oprávnění ověřovaného uživatele. Příklad zabezpečení je na ukázce kódu 4.9.

```
@Override
@PreAuthorize("hasAnyRole('ADMIN', 'DISPATCHER')")
```

```
public ResponseEntity<List<DeliverySlimDTO>> getAllDeliveries() {
   return ResponseEntity.ok(deliveryMapper.toSlimDTOs(
   deliveryService.readAll()));
}

@Override
@PreAuthorize("hasAnyRole('ADMIN', 'DISPATCHER', 'COURIER') ||
    @deliveryService.isCustomerDelivery(#deliveryId)")
public ResponseEntity<DeliveryDTO> getDelivery(Long deliveryId) {
   return ResponseEntity.ok(deliveryMapper.toDTO(
   deliveryService.readById(deliveryId)));
}

@Override
@PreAuthorize("hasAnyRole('ADMIN', 'DISPATCHER') ||
    @deliveryService.isCourierDelivery(#deliveryId)")
public ResponseEntity<DeliveryDTO> changeDeliveryState(Long
    deliveryId, String newState) {
   DeliveryDTO deliveryDTO = deliveryMapper.toDTO(
   deliveryService.changeDeliveryState(deliveryId, newState));
   return ResponseEntity.ok(deliveryDTO);
}
```

Ukázka kódu 4.9: Zapezpečení pomocí anotací

Kapitola 5

# Testování

Po implementační části následuje fáze testování. Testování je nedílnou části vývoje softwaru. V této kapitole budou popsány typy testu, který byly použity při testování výsledného systému, průběh samotného testování a jeho výsledky.

## 5.1 Testování

Testování softwaru lze charakterizovat jako proces ověřování a validace, zda je software nebo aplikace bez chyb, zda splňuje technické požadavky vyplývající z jeho návrhu a vývoje a zda účinně a efektivně splňuje požadavky uživatelů tím, že řeší všechny výjimečné a mezní případy. [31]

Cílem procesu testování softwaru je nejen najít chyby ve stávajícím softwaru, ale také najít opatření ke zlepšení softwaru z hlediska efektivity, přesnosti a použitelnosti. Jeho hlavním cílem je měření specifikace, funkčnosti a výkonnosti softwarového programu nebo aplikace. [31]

## 5.2 Typy testů

Testování softwaru lze obecně rozdělit na dva typy: manuální a automatické.

### Manuální testování

Manuální testování zahrnuje ruční testování softwaru, tj. bez použití automatizačního nástroje nebo skriptu. Při tomto typu testování přebírá tester roli koncového uživatele a testuje software s cílem identifikovat jakékoli neočekávané chování nebo chybu. Existují různé fáze manuálního testování, jako je unit testování, integrační testování, systémové testování a uživatelské akceptační testování. [31]

Testeři používají k testování softwaru testovací plány, testovací případy nebo testovací scénáře, aby zajistili úplnost testování. Součástí manuálního

testování je také průzkumné testování, kdy testeři zkoumají software, aby v něm identifikovali chyby. [31]

#### Automatické testování

Automatické testování, které je také známé jako automatizace testování, spočívá v tom, že tester napíše scénáře a použije k testování produktu jiný software. Tento proces zahrnuje automatizaci manuálního procesu. Automatizované testování slouží k rychlému a opakovanému procházení testovacích scénářů, které byly provedeny ručně při manuálním testování. [31]

Kromě regresního testování se automatické testování používá také k testování aplikace z hlediska zátěže, výkonu a stresu. Ve srovnání s manuálním testováním zvyšuje pokrytí testů, zlepšuje přesnost a šetří čas i peníze. [31]

## 5.3 Uživatelské testování

Uživatelské testování je proces, při kterém jsou rozhraní a funkce webové stránky, aplikace, produktu nebo služby testovány skutečnými uživateli, kteří provádějí konkrétní úkoly v reálných podmínkách. Účelem tohoto procesu je vyhodnotit použitelnost dané webové stránky nebo aplikace a rozhodnout, zda je produkt připraven ke spuštění pro skutečné uživatele. Pro dosažení relevantních výsledků by testující neměli být příliš usměrňováni a mělo by jim být umožněno přirozeně komunikovat s webovou stránkou nebo aplikací, aby se zjistilo, zda je systém dostatečně intuitivní a pohodlný pro používání lidmi, kteří s ním ještě nejsou obeznámeni. [32]

### 5.3.1 Průběh testování

Uživatelského testování se zúčastnilo 5 testerů. Testování probíhalo v několika prohlížečích: Google Chrome verze 112.0.5615.138 a Microsoft Edge verze 112.0.1722.58.

V prohlížeči Google Chrome také probíhalo i testování responzivního chování pomocí nástrojů Chrome DevTools, který umožňuje v prohlížeči měnit velikost obrazovky pro zobrazení webových stránek.

### 5.3.2 Scénáře

Testovací scénáře jsou seznamy jednotlivých instrukcí, podle kterých by měl tester ověřovat aplikaci. Scénáře často vychází z případů užití a požadavku na systém. Scénáře byly rozděleny to čtyř kategorií podle rolí, jimž je tato funkcionalita dostupná: společné, zákazník, dispečer, kurýr a administrátor. Jednotlivé testovací scénáře jsou představeny níže.

#### 5.3.2.1 Společné

Společné testovací scénáře slouží pro ověření částí aplikace, které jsou stejné pro všechny role.

**Registrace**

- Přejít na úvodní obrazovku nepřihlášeného uživatele.
- Zaregistrovat se do aplikace.

**Přihlášení**

- Přejít na úvodní obrazovku nepřihlášeného uživatele.
- Přihlásit se do aplikace.

**Změna osobních údajů**

- Po přihlášení do aplikace rozkliknout profil uživatele.
- Změnit jednotlivé položky svého profilu a uložit je.

#### 5.3.2.2 Zákazník

Sice každý nově registrovaný uživatel je zákazník, bylo vyřešeno tyto scénáře popsat v sekci pro zákazníka.

**Přidání nového doručení**

- Po přihlášení najít na hlavní obrazovce možnost přidání nového doručení.
- Přidat nové doručení, použit kód, který byl obdržen od doručovací služby.

**Sledování kurýra**

- Po přihlášení najít doručení, které uživatel chce sledovat.
- Otevřít toto doručení a obdržet informace nutné pro přijetí zásilky.

#### 5.3.2.3 Dispečer

Scénáře, které ověřují správnost části aplikace, jež je přístupná uživatelům s rolí dispečer.

**Vyhledání existujícího plánu cesty**

- Po přihlášení do aplikace přejít do sekce dispečera.
- Najít plán cesty podle data.

**Vyhledání existujícího doručení**

- Po přihlášení do aplikace přejít do sekce dispečera.
- Otevřít plán cesty, ve kterém by toto doručení mohlo být.
- Najít doručení podle identifikátoru.

**Přidání nového plánu cesty**

- Po přihlášení do aplikace přejít do sekce dispečera.
- Vytvořit nový plán cesty a uložit jej.

**Přidání doručená do plánu cesty**

- Po přihlášení do aplikace přejít do sekce dispečera.
- Otevřít plán cesty, do kterého by toto doručení mělo být přidáno.
- V plánu cesty vytvořit nové doručení a uložit jej.

**Změna plánu cesty**

- Po přihlášení do aplikace přejít do sekce dispečera.
- Otevřít plán cesty, který je třeba editovat.
- Změnit potřebné položky a uložit změny.

#### 5.3.2.4 Kurýr

Scénáře, které ověřují správnost části aplikace, jež je přístupná uživatelům s rolí kurýr.

**Vyhledání aktuálního plánu cesty**

- Po přihlášení do aplikace přejít do sekce kurýra.
- Pokud nějaký aktuální plán cesty existuje, otevřít jej.

**Doručení zásilky**

- Po přihlášení do aplikace přejít do sekce kurýra.
- Rozkliknout aktuální plán cesty.
- Doručovat zásilky na jednotlivé adresy.

## 5.4 Výsledky testování

Při testování systém fungoval podle stanovených požadavků a očekávání. Menší chyby, které vyskytly během testování, byly opraveny. Bylo odhaleno, že při zobrazení na menších obrazovkách některé informace nebyly dostatečně přehledné. Tento nedostatek byl zmíněn v možnostech dalšího rozšíření aplikace a bude řešen v budoucích verzích systému. Celkový dojem testerů od použití této webové aplikace byl kladný.

Kapitola 6

# Zhodnocení

V první části této kapitoly jsou popsány možností reálného použití výsledného systému. Dále následuje porovnání s již existujícími řešeními, které byly analyzovány v první kapitole této práce. V poslední části jsou uvedeny možností budoucího rozšíření systému.

## 6.1 Možnosti reálného použití

Výsledný systém byl navržen, implementován a řádně otestován s ohledem na všechny požadavky a s použitím moderních technologií tak, aby jej bylo možné lehce upravovat a rozšiřovat pod specifiky jednotlivých firem. Sice toto je pouze první verze systému, v ní bylo implementováno všechno, co je potřeba pro běžný a pohodlný provoz doručovací firmy. Tento systém může být použit v každé doručovací firmě, které záleží na spokojeností svých zákazníků a zaměstnanců. Pro jednoduché nasazení systému do běžného provozu by bylo možné použít nástroj Docker [33].

Docker [33] je kontejnerová platforma, která vývojářům umožňuje zabalovat, nasazovat a spouštět aplikace konzistentním a přenositelným způsobem v různých prostředích. Toho dosahuje tím, že poskytuje vrstvu abstrakce mezi aplikací a základní infrastrukturou, což vývojářům umožňuje vytvářet lehké, izolované kontejnery, které lze spustit kdekoli, od notebooku vývojáře až po produkční server. Jednotlivé Dockerfile již byly připraveny jak pro serverovou část této aplikace, tak i pro klientskou.

## 6.2 Porovnání s existujícími řešeními

Z analýzy je vidět, že sice možnost sledování kurýra v reálném čase zvětšuje pohodlnost doručování zásilek, ale tato možnost není implementována v mnoha existujících systémech. Výsledný systém oproti existujícím tuto možnost poskytuje a na mapě je vidět, kde se aktuálně nachází kurýr, který zásilku doru-

čuje. Navíc zákazník v reálném čase dostává doplňující informaci o aktuálním počtu adres, který zbývá od kurýra do zákazníka a po dokončení doručení je realizována možnost zanechání zpětné vazby o kurýrovi.

## 6.3 Možnosti rozšíření

Sice výsledná aplikace splňuje všechny stanovené požadavky a cíle, níže byly popsány možnosti rozšíření, které by ji doplnily nebo zlepšily, čim by tento systém udělaly ještě silnějším konkurentem oproti existujícím řešením.

### 6.3.1 Přesměrování kurýra do navigátoru

Přidání tlačítka na stránku aktuálního plánu cesty do políčka probíhajícího doručení, po stisknutí kterého by proběhlo přesměrování kurýra do nějakého známého navigátoru, jako je například Waze [34] nebo Google Maps [35].

### 6.3.2 Určení předpokládaného času doručení

Momentálně zákazník vidí aktuální polohu kurýra a počet adres zbývajících od kurýra do zákazníka. Přidání informaci o předpokládaném čase doručení by udělalo doručování zásilky ještě pohodlnějším, protože by zákazník přesněji věděl čas, v kolik by měl zásilky převzít.

### 6.3.3 Možnost zanechávaní spropitného kurýrovi

Jelikož práce kurýrem není z lehkých a často se stává, že lidé chtějí nějak poděkovat kurýrům, ale nemají hotovost, přidání možnosti zanechání spropitného kurýrovi po doručení pomocí online platby by vyřešilo tento problém.

### 6.3.4 Možnost přeplánování termínu doručení

Momentálně není žádná možnost měnit termín doručení pomocí webové aplikace. Přidání omezeného počtu možností přeplánování termínu doručení by mohlo být příjemným zlepšením pro zákazníky a dalším přivýdělkem pro firmy.

### 6.3.5 Možnost přesouvat pořadí doručení v plánu cesty

Momentálně systém nemá tuto možnost implementováno, ale přesouvání pořadí doručení by udělalo práci dispečera pohodlnější, protože nemusel by předem vědět správné pořadí adres, ale mohl by je měnit dle potřeby.

### 6.3.6 Možnost editování doručení dispečerem

Jelikož systém neposkytuje možnost změny položek existujícího doručení, přidání možnosti jeho editování by odstranilo tento nedostatek.

### 6.3.7 Zabezpečení na přidání zásilky

Momentálně zákazník může přidat libovolné doručení podle čísla. Toto číslo by měl dostat od doručovací firmy. Kvůli tomuto nedostatku informace o doručování zásilky by se mohla dostat k podvodníkům, což by mohlo nést negativní následky. Zabezpečení pro přidávání zásilky zákazníkem, například pomocí hesla nebo verifikace přes telefonní číslo, by vyřešilo tento problém.

### 6.3.8 Rozlišení pro menší menší obrazovky

Tato možnost rozšíření systému vychází z uživatelského testování. Přizpůsobení aplikace pro menší obrazovky by udělalo zobrazení na mobilních zařízeních mnohem hezčí a čitelnější.

### 6.3.9 Mobilní aplikace pro Android a IOS

Tato možnost vychází ze současných trendů. Mobilní aplikace by usnadnila použití systému kurýrům a zákazníkům.

### 6.3.10 Přidání GUI pro správu uživatelů

Momentálně není implementováno žádné grafické rozhraní pro správu uživatelů. V serverové části aplikace tato možnost už je implementována. Přidání jednoduchého a intuitivního rozhraní, kde by byla možnost správy rolí u jednotlivých uživatelů, by vyřešilo tento nedostatek.

### 6.3.11 Přidání GUI na změnu hesla

V sekci uživatelského profilu aktuálně není možnost měnit heslo. V serverové části aplikace tato možnost už je implementována. Přidání grafického rozhraní pro změnu hesla by zvýšilo bezpečnost uživatelů tohoto systému.

## 6.4 Shrnutí zhodnocení

Sice, jako každá první verze softwaru, tato aplikace má některá omezení a nedostatky, které by šlo vylepšit anebo změnit pod specifiky doručovací firmy. Tato webová aplikace splňuje všechny zadefinované cíle a stanovené na ni požadavky a, navíc, je v tom stavu, který by šlo nasadit do provozu a začít běžně používat.

# Závěr

Cílem práce bylo na základě analýzy navrhnout, implementovat, otestovat a zhodnotit webovou aplikaci, která umožní uživatelům sledovat doručení svoje zásilky v reálném čase.

Během analytické části byly analyzovány: doména doručování zásilky, existující řešení a požadavky, které byly rozděleny na funkční, nefunkční a dále také podle metody MoSCoW do čtyř kategorií.

Následovala fáze návrhu, ve které byly navrženy případy užití, vybrány vhodné technologie a architektura s ohledem na snadné rozšíření aplikace v budoucnosti. Dále bylo sestrojeno databázové schéma a byl vytvořen návrh grafického uživatelského prostředí, během něhož byl kladen důraz na snadnou orientaci a přehlednost. Tento návrh byl realizován jako webová aplikace během implementační fáze.

V následující fázi bylo popsáno samotné testování. Jednotlivé části aplikace byly řádně otestovány uživatelským testováním podle připravených scénářů.

V závěrečné kapitole byly zhodnoceny výsledky celé práce a poté byly popsány možnosti reálného použití.

Všechny požadavky vyplývající ze zadání práce, rešerše domény a existujících řešení byly úspěšně splněny. Vytvořená aplikace umožňuje v reálném čase sledovat kurýra, dostávat aktuální informaci o počtu zbývajících adres od kurýra do zákazníka, zadávat plán cesty a zanechávat zpětnou vazbu o kurýrovi.

V budoucnosti, jak už bylo zmíněno v poslední kapitole, by bylo možné aplikaci rozšířit o mobilní verzi pro iOS a Android, což by udělalo ji pohodlnější pro uživatele.

# Literatura


[1] MOTAVALLIAN, J. *Last mile delivery in the retail sector in an urban context.* Melbourne: RMIT University, 2019 [cit. 2022-11-28]. Disertační práce. RMIT University, Graduate School of Business and Law.

[2] WIKIMEDIA FOUNDATION. Poslední míle (doprava) - Last mile (transportation). *Encyclopedia* [online]. 2021 [cit. 2022-11-28]. Dostupné z: `https://wikijii.com/wiki/Last_mile_(transportation)`

[3] EVANS, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software* [online]. Boston: Addison-Wesley, 2003 [cit. 2022-11-28]. ISBN: 0321125215.

[4] OBJECT MANAGEMENT GROUP. Unified Modeling Language. *Unified Modeling Language* [online]. 2022 [cit. 2022-11-28] Dostupné z: `https://www.omg.org/spec/UML/`

[5] DPD GROUP. DPD. *DPD* [online]. 2022 [cit. 2022-11-28]. Dostupné z: `https://www.dpd.com`

[6] LIFTAGO A.S. Liftago. *Liftago* [online]. 2022 [cit. 2022-11-28]. Dostupné z: `https://www.liftago.cz/`

[7] SHADOWFAX TECHNOLOGIES PVT LTD. Shadowfax We Deliver. *Shadowfax* [online]. 2022 [cit. 2022-11-28]. Dostupné z: `https://www.shadowfax.in/`

[8] LOCATE2U PTY. LTD. Locate2u. *Locate2u* [online]. 2022 [cit. 2022-11-28]. Dostupné z: `https://www.locate2u.com`

[9] YELOW SOFT INC. Fixlastmile. *Fixlastmile* [online]. 2021 [cit. 2022-11-28]. Dostupné z: `https://www.fixlastmile.com/`

[10] DETRACK SYSTEMS PTE LTD. Detrack. *Detrack* [online]. 2022 [cit. 2022-11-28]. Dostupné z: https://www.detrack.com/

[11] BRUSH, K. MoSCoW method. *TechTarget* [online]. 2020 [cit. 2022-11-28]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method

[12] PIVOTAL SOFTWARE. Spring Framework. *Spring Projects* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://spring.io/projects/spring-framework

[13] PIVOTAL SOFTWARE. Spring Boot. *Spring Projects* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://spring.io/projects/spring-boot

[14] PIVOTAL SOFTWARE. Spring Security. *Spring Projects* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://spring.io/projects/spring-security

[15] PIVOTAL SOFTWARE. Spring Data. *Spring Projects* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://spring.io/projects/spring-data

[16] GRADLE INC. Gradle Build Tool. *Gradle* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://gradle.org/

[17] APACHE SOFTWARE FOUNDATION. Apache Maven Project. *Maven* [online]. 2023 [cit. 2023-02-11]. Dostupné z: https://maven.apache.org/

[18] SMARTBEAR SOFTWARE. OpenAPI Specification. *Swagger* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://swagger.io/resources/open-api/

[19] MULESOFT, LLC. RAML. *The simplest way to model APIs* [online]. 2020 [cit. 2023-02-11]. Dostupné z: https://raml.org/

[20] MICROSOFT. TypeScript is JavaScript with syntax for types. *TypeScript* [online]. 2023 [cit. 2023-02-12]. Dostupné z: https://www.typescriptlang.org/

[21] META PLATFORMS, INC. React – A JavaScript library for building user interfaces. *React* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://reactjs.org/

[22] GEEKSFORGEEKS. Difference between SQL and NoSQL. *GeeksforGeeks* [online]. 2022 [cit. 2023-02-12]. Dostupné z: https://www.geeksforgeeks.org/difference-between-sql-and-nosql/

[23] THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL: The World's Most Advanced Open Source Relational Database. *PostgreSQL* [online]. 2023 [cit. 2023-01-29]. Dostupné z: https://www.postgresql.org/

[24] SIMICART. SPA vs. MPA: Pros, Cons and How To Make Final Choice. *SimiCart Blog* [online]. 2022 [cit. 2023-02-17]. Dostupné z: https://www.simicart.com/blog/spa-vs-mpa/#:~:text=Differences%20between%20SPA%20and%20MPA,-Speed%20is%20clearly&text=SPA%20has%20the%20benefit%20of,quality%20of%20security%20is%20poor.

[25] GEEKSFORGEEKS. Three-Tier Client Server Architecture in Distributed System? *GeeksforGeeks* [online]. 2022 [cit. 2023-02-06]. Dostupné z: https://www.geeksforgeeks.org/three-tier-client-server-architecture-in-distributed-system/

[26] IBM. What is three-tier architecture? *IBM* [online]. 2023 [cit. 2023-02-06]. Dostupné z: https://www.ibm.com/topics/three-tier-architecture

[27] GOOGLE. Google Maps Platform. *Google* [online]. 2023 [cit. 2023-04-23]. Dostupné z: https://mapsplatform.google.com/

[28] MAPBOX. Maps and location for developers. *Mapbox* [online]. 2023 [cit. 2023-04-23]. Dostupné z: https://www.mapbox.com/

[29] WOZNIAK Kamil. Java Persistence API: Creating a base entity. *smarterco.de* [online]. 2023 [cit. 2023-04-24]. Dostupné z: https://smarterco.de/jpa-creating-a-base-entity/#:~:text=The%20advantage%20of%20a%20BaseEntity,on%20each%20class%20for%20itself.

[30] OKTA. Authentication vs. Authorization *Okta* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://www.okta.com/identity-101/authentication-vs-authorization/

[31] GEEKSFORGEEKS. Software Testing | Basics *GeeksforGeeks* [online]. 2022 [cit. 2023-04-26]. Dostupné z: https://www.geeksforgeeks.org/software-testing-basics/

[32] OMNICONVERT. What is... User testing *Omniconvert* [online]. 2022 [cit. 2023-04-26]. Dostupné z: https://www.omniconvert.com/what-is/user-testing/

[33] DOCKER INC. Develop faster. Run anywhere. User testing *Docker* [online]. 2023 [cit. 2023-04-29]. Dostupné z: https://www.docker.com/

[34] WAZE MOBILE. We could all go for a lot less traffic in our lives. *Waze* [online]. 2023 [cit. 2023-05-01]. Dostupné z: `https://www.waze.com/`

[35] GOOGLE. Google Maps Platform. *Google* [online]. 2023 [cit. 2023-05-01]. Dostupné z: `https://www.google.com/maps/`

Příloha A

# Seznam použitých zkratek

**ACID** Atomicity, consistency, isolation, and durability

**AJAX** Asynchronous Javascript and XML

**API** Application Programming Interfac

**B2B2C** Business to Business to Consumer

**BI-PJV** Programování v Javě

**BI-TJV** Technologie Java

**CIN** Corporate Identification Number

**DOM** Document Object Model

**DTO** Data Transfer Object

**DSL** Domain-specific language

**ERP** Enterprise Resource Planning

**FMCG** Fast-moving consumer goods

**GUI** Graphical user interface

**HTML** HyperText Markup Language

**IOS** iPhone OS

**JSX** JavaScript Syntax Extension

**JWT** JSON Web Tokens

**LMD** Last-mile delivery

**MPA** Multiple-page application

**MVC** Model-View-Controller

**NoSQL** Not Only Structured Query Language

**OAS** OpenAPI Specification

**RAML** RESTful API Modeling Language

**REST** Representational State Transfer

**SPA** Single-page application

**SQL** Structured Query Language

**SSR** Server-Side Rendering

**UML** Unified Modeling Language

**URL** Uniform Resource Locator

**XML** Extensible Markup Language

Příloha B

# Seznam příloh

- `readme.txt` ..... stručný popis obsahu CD
- `examples` ..... ukázky aplikace
  - `screenshots` ..... obrazové ukázky aplikace
  - `videos` ..... video ukázky aplikace
- `src` ..... zdrojové soubory
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF