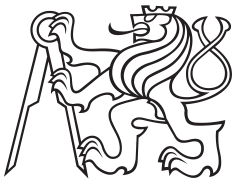
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Informační systém pro správu autoservisu

Dmitriy Shevchenko

Vedoucí: Ing. Božena Mannová, Ph.D.
Květen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Shevchenko** Jméno: **Dmitriy** Osobní číslo: **499359**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Informační systém pro správu autoservisu

Název bakalářské práce anglicky:

Information system for car service management

Pokyny pro vypracování:

Cílem práce je analyzovat stávající řešení autoservisu a vytvořit informační systém, který pomůže malým a středním autoservisům. Tato aplikace by měla zjednodušit interakci mezi zákazníky a pracovníky servisu, urychlit zpracování a vyřízení objednávek a umožnit evidenci všech objednávek, zákazníků, jejich vozidel a spotřebního materiálu (např. náhradních dílů). Výsledkem práce bude funkční prototyp aplikace s implementovanou serverovou a klientskou částí.

1. Seznamte se s problematikou provozu autoservisu.
2. Proveďte analýzu dostupných existujících řešení, proveďte jejich porovnání a vyhodnocení.
3. Na základě provedené analýzy navrhněte základní funkcionality navrhované aplikace.
4. Zvolte architekturu aplikace a vyberte nejvhodnější technologie pro implementaci. Výběr technologií zdůvodněte.
5. Aplikaci implementujte a otestujte.
6. Zhodnoťte výsledky a navrhněte případné další funkcionality nebo jiná zlepšení.
7. Při řešení využívejte vhodných prostředků softwarového inženýrství.

Seznam doporučené literatury:

[1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach , ISBN-10: 9780078022128

[2] Ing. Pavel Náplava In: Úvod do předmětu a problematiky Informačních systémů [online] 2019. [cit. 2020-11-30] Dostupné z https://moodle. fel.cvut.cz/mod/page/view.php?id=111142.

[3] jednickanatrhu In: Program pro autoservisy [online]. [cit. 2020-11-30] Dostupné z https://jednickanatrhu.cz/program-pro-autoservis.

[4] Mechanic [online]. [cit. 2020-12-1] Dostupné z https://www.nextis.cz/ mechanic/.

[5] carsys In: Software pro prodejce a servisy automobilů [online]. [cit. 2020- 12-1] Dostupné z https://www.carsys.cz/carsystem/.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Božena Mannová, Ph.D. kabinet výuky informatiky FEL

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

Datum zadání bakalářské práce: **13.02.2023** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Božena Mannová, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Rád bych poděkoval všem učitelům, kteří nám sdíleli své znalosti během těchto let. Také bych chtěl poděkovat svému vedoucímu práce, paní Ing. Boženě Mannové, za její přínos a spolupráci při psaní mé bakalářské práce. Nakonec bych chtěl poděkovat rodině a přátelům za podporu po celou dobu mého bakalářského studia.

Prohlášení

I hereby declare that this bachelor's thesis represents my own work, which has been prepared on the basis of knowledge acquired during my studies, nor has this work has been previously included in any other publication, diploma, or bachelor's thesis submitted to this or any other university. Nor have I been assisted in writing this thesis by anyone other than my thesis supervisor.

Prague, května 23, 2023

Tímto prohlašuji, že tato bakalářská práce představuje mou vlastní práci, která byla vypracována na základě znalostí získaných během mého studia, ani tato práce nebyla dříve zařazena do žádné jiné publikace, diplomové nebo bakalářské práci odevzdané na této nebo jiné vysoké škole. Rovněž mi při psaní této práce nepomáhal nikdo jiný než můj vedoucí práce.

V Praze, 23. May 2023

Abstrakt

Tato práce se zabývá analýzou stávajících řešení pro autoservisy a implementací informačního systému, který pomůže malým a středním podnikům v oblasti autoservisů. Hlavním účelem aplikace je usnadnit interakci mezi zákazníky a pracovníky autoservisu pomocí systému pro rezervaci a ukládání zakázek do systému. Na začátku byla provedena analýza stávajících řešení, na jejímž základě byly vyzdviženy klady a zápory jednotlivých řešení, a na základě analýzy byly shromážděny funkční a nefunkční požadavky na aplikaci, které by byly relevantní v dnešní době. V následujících kapitolách byla popsána celková koncepce aplikace a technologie a architektura použitá k přímé implementaci aplikace. Na závěr byly popsány nápady na vylepšení aplikace a její možný budoucí stav.

Klíčová slova: Webová aplikace, Informační systém, Autoservis, Java, ReactJS, REST

Vedoucí: Ing. Božena Mannová, Ph.D. Praha 2, Karlovo náměstí 13, E-430

Abstract

This thesis deals with the analysis of existing solutions for car repair services and the implementation of an information system that will help small and medium-sized businesses in the field of car repair services. The main purpose of the application is to facilitate the interaction between customers and employees by using a system for booking and storing orders in the system. In the beginning, an analysis of existing solutions was conducted to highlight the pros and cons of each solution, and based on the analysis, functional and non-functional requirements for the application were gathered that would be relevant today. The following chapters described the overall application design and the technology and architecture used to directly implement the application. Finally, ideas for improving the application and its possible future state were described.

Keywords: Web application, Information system, Autoservice, Java, ReactJS, REST

Title translation: Information system for car service management

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Ve své bakalářské práci se zabývám vývojem prototypu informačního systému pro autoservisy. Téma mi navrhl můj vedoucí práce a zaujalo mě svou aktuálností a zajímavostí, kterou pro mě představuje. Vývoj takového informačního systému speciálně přizpůsobeného pro autoservisy může výrazně zjednodušit řízení obchodních procesů, interakci mezi zákazníky a zaměstnanci a zkrátit čekací doby. Proto bylo rozhodnuto analyzovat a představit prototyp řešení pro malé a střední podniky.

1.1 Motivace

Automobilový průmysl se v naší době neustále rozvíjí, přibývá lidí, kteří vlastní auta, a samotná auta jsou technologicky vyspělejší. Získat konkrétní čísla je poněkud obtížné, protože za sčítání aut jsou zodpovědné různé orgány, ale podle nejlepších odhadů se v roce 2016 jednalo přibližně o 1,32 miliardy osobních a nákladních automobilů a autobusů a jejich počet každoročně roste [2]. Poptávka po opravách a údržbě těchto vozidel proto roste. Proto se autoservisy vyvíjejí stejně jako samotný automobilový průmysl, jsou stále aktuální a žádané.

V dnešních autoservisech je třeba pracovat s velkým množstvím informací, jako jsou údaje o zákaznících, vozidlech, servisních pracích, náhradních dílech atd. Dílny také potřebují efektivně řídit své zdroje, včetně personálu a zásob. Z vlastní zkušenosti jsem však bohužel viděl, že mnoho moderních autoservisů nákup softwaru pro řízení podniku a interakci mezi zákazníky a zaměstnanci zanedbává.

Proto chci vytvořit takový prototyp aplikace, který pomůže malým a středním autoservisům.

1.2 Cíle

Cílem bakalářské prace je analyzovat existující řešení, navrhnout strukturu a implementaci informačního systému pro autoservisy a připravit podklady pro zdokonalení aplikace v budoucnu.

Aplikace by měla zjednodušit interakci mezi zákazníky a pracovníky autoservisu, urychlit zpracování a realizaci zakázek a umožnit evidenci všech zakázek, zákazníků, jejich vozidel a spotřebního materiálu (např. náhradních dílů).

Výsledkem bakalářské práce bude funkční prototyp aplikace s implementovanou serverovou a klientskou částí.

Kapitola 2

Analýza existujících řešení

V této kapitole budou analyzována existující řešení, a to jak česká, tak zahraniční. Budou identifikovány jejich hlavní nedostatky a na jejich základě budou v další kapitole sestaveny funkční a nefunkční požadavky na informační systém.

2.1 Česká řešení

2.1.1 Carsys

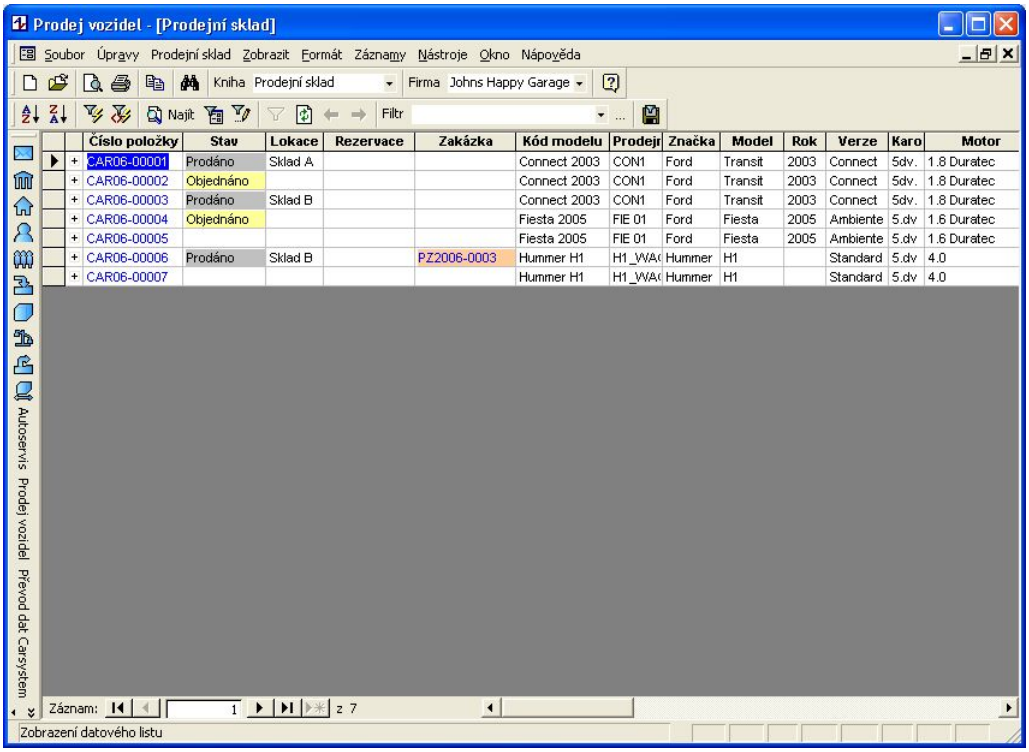
Carsys nabízí aplikaci pro prodejce automobilů. Jedná se o informační systém, který se specializuje na samotnou autoservisní dílnu a její prostředí, např. zpracovává informace o zákaznících, skladech, v nichž jsou k dispozici díly nebo o dokumentech. Na výběr je několik verzí, které se liší dalšími funkcemi a cenou [3].

Výhody:

- Systém umožňuje spravovat a vytvářet různé dokumenty, jako jsou smlouvy, nabídky a další

Nevýhody:

- Vysoké nároky na hardware
- Klient je považován pouze za firmu



Obrázek 2.1: Příklad informačního systému Carsys.

2.1.2 AutoFenix

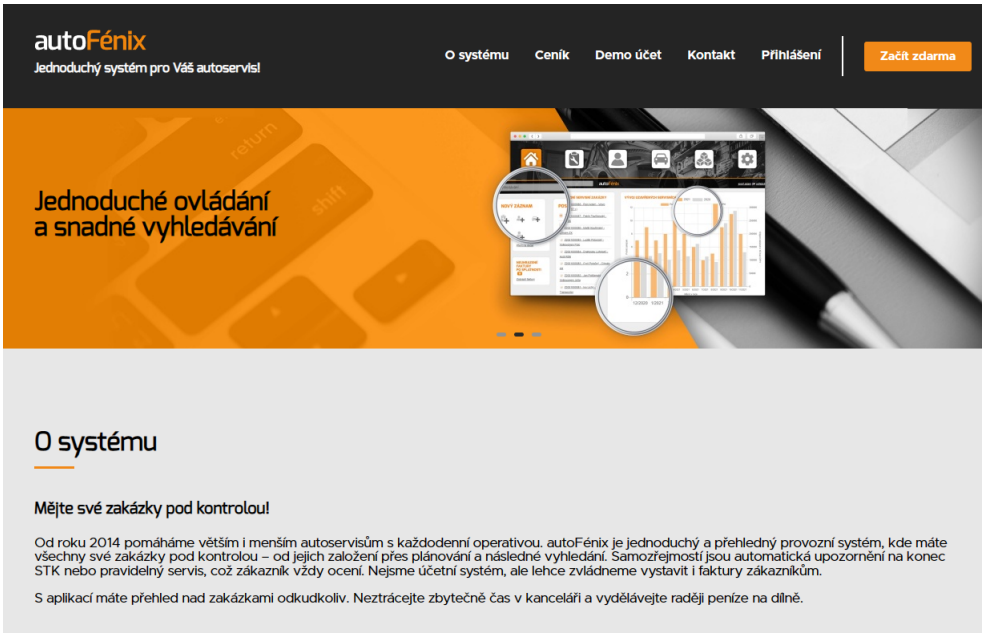
Jedná se o informační systém, který je rovněž reprezentován formou webové aplikace. Nabízí správu zakázek, zákazníků, vozidel, tvorbu faktury, vyhledávání, kalendář, neomezený počet uživatelů, automatické zálohování a možnost vyzkoušení softwaru zadarmo. Má dvě verze, základní a prémiovou [4].

Výhody:

- Neomezený počet uživatelů
- Zkušební verze zdarma

Nevýhody:

- Klient je považován pouze za firmu



Obrázek 2.2: Příklad informačního systému Autofenix [4].

2.1.3 Mechanic

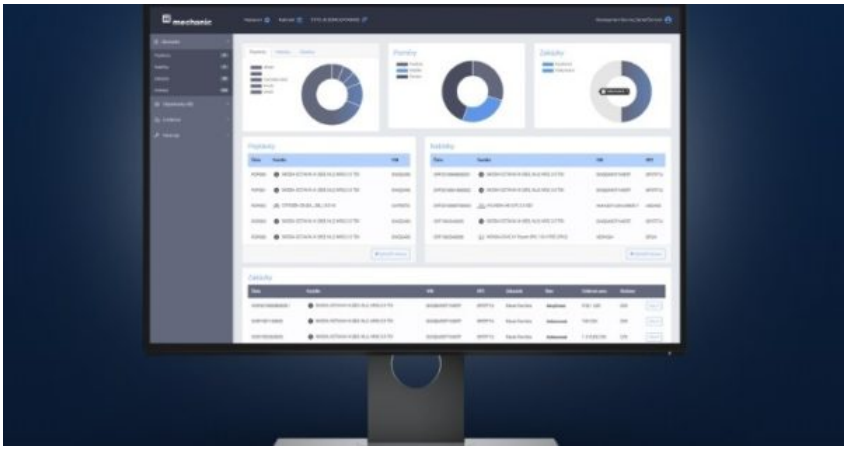
Mechanic je softwarové řešení společnosti Nextis, které vzniklo díky vysoké poptávce po automobilovém softwaru. Mechanic je určen pro malé nebo středně velké servisy a jedná se o velmi jednoduchý a intuitivní program, který umožňuje pohodlné a přesné vedení agendy servisu. Podle informací z oficiálních webových stránek tento systém obsahuje: možnost on-line příjmu zboží od podporovaných dodavatelů, evidence uskladněných pneumatik, jednoduchou skladovou evidenci, evidence zákazníků a jejich vozidel [5].

Výhody:

- Jednoduché a intuitivní uživatelské rozhraní

Nevýhody:

- Kritické chyby v systému (pokusil jsem se zaregistrovat v jejich systému, ale registrace nefungovala a nemohl jsem se dostat dál)



Obrázek 2.3: Příklad informačního systému Mechanic [5].

2.2 Zahraniční řešení

2.2.1 ShopMonkey

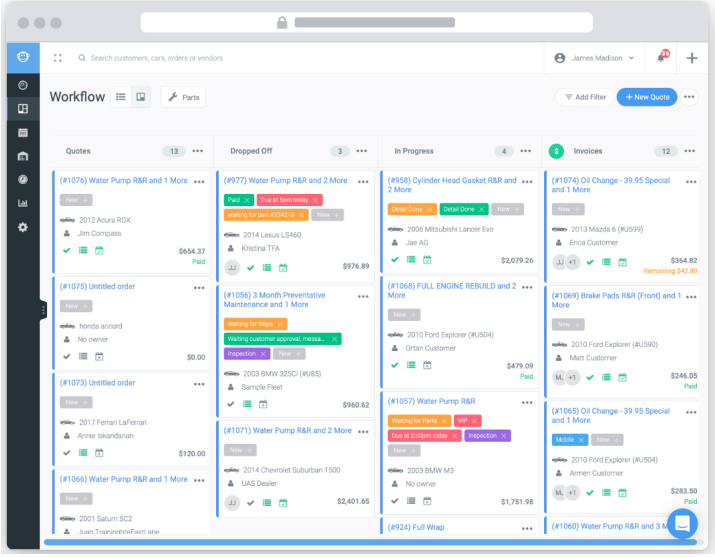
Jedná se o společnost, která vyvíjí informační systémy mimo Českou republiku (především v USA) pro malé a střední autoservisy, opravny lodí a prodejny náhradních dílů. Tento informační systém nabízí mnoho funkcí, jako je autorizace přes Google, docházkový systém pro zaměstnance, historie objednávek, online platby, podpora 24/7 či mobilní aplikace. K dispozici jsou také 4 verze s různými funkcemi a cenou [6].

Výhody:

- Integrace s Kalendářem Google
- Online podpora 24/7
- Mobilní aplikace

Nevýhody:

- Základní verze informačního systému má velmi omezené funkce;
- Existence různých verzí systému může způsob it komplikace při výběru nejvhodnější možnosti



Obrázek 2.4: Příklad informačního systému Monkeyshop [6].

2.3 Souhrn

Na základě analýzy a porovnání hlavních funkcí existujících systémů uvedených ve následující srovnávací tabulce se plánuje vývoj nového informačního systému pro autoservis. Tabulka bude výchozím bodem, který nastíní klíčové funkce a vlastnosti, jež lze do aplikace implementovat.

Funkce / systém	ShopMonkey	Mechanic	AutoFenix	Carsys
Autorizace prostřednictvím služby Google	Ano	N/A	N/A	N/A
Časová evidence	Ano	N/A	N/A	Ano
Řízení objednávek	Ano	Ano	Ano	Ano
Správa zákazníků a jejich vozidel	N/A	Ano	Ano	Ano
Zkušební období zdarma	Ne	Ano	Ano	Ne
Historie objednávek	Ano	Ano	Ano	Ano
Neomezený počet uživatelů	N/A	Ano	Ano	Ne

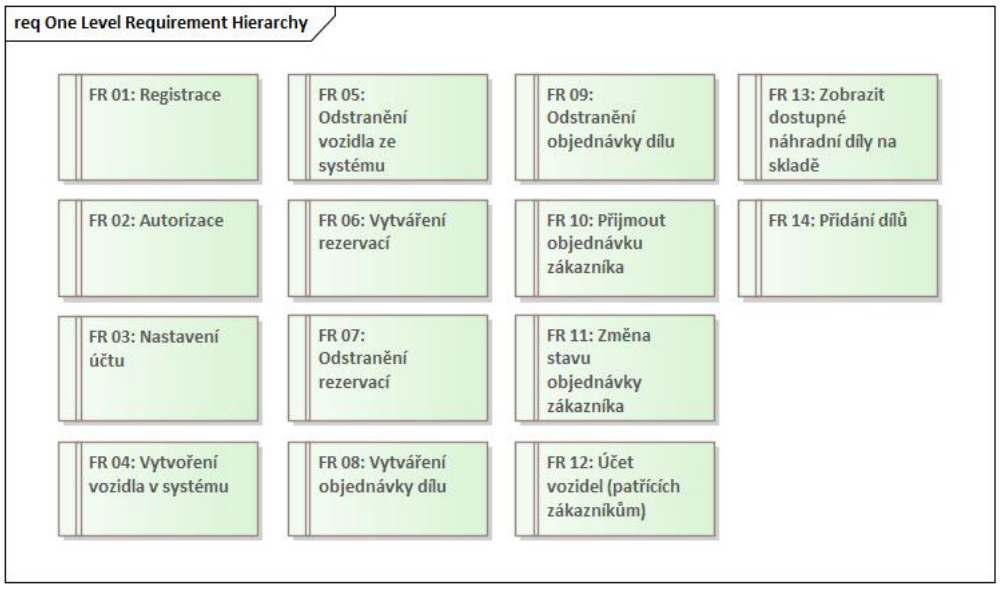
Tabulka 2.1: Výhody a nevýhody existujících řešení

Kapitola 3

Požadavky

3.1 Funkční požadavky

Funkční požadavky popisují, co systém má dělat - jaké služby má nabídnout, jak má reagovat na různé druhy vstupů a jak se má chovat v různých situacích [7]. Tyto požadavky byly vytvořeny na základě analýzy existujících řešení, vlastních zkušeností a zkušeností vycházejících ze zpětné vazby od aktivních uživatelů podobných autoservisů.



Obrázek 3.1: Funkční požadavky aplikace.

1. FR 01: Registrace

   Uživatel se bude moci zaregistrovat.

2. FR 02: Autorizace

   Na základě účtu v systému se uživatel bude moci přihlásit do našeho systému.

3. FR 03: Nastavení účtu

   V případě potřeby bude moci uživatel změnit různá nastavení svého profilu, například popis profilu.

4. FR 04: Vytvoření vozidla v systému

   Zákazníci budou moci zadat své vozidlo do systému.

5. FR 05: Odstranění vozidla ze systému

   Zákazníci budou moci své vozidlo ze systému vyřadit.

6. FR 06: Vytváření rezervací

   Zákazníci servisu budou moci vytvářet objednávky oprav pro své vozy.

7. FR 07: Odstranění rezervací

   V případě potřeby budou moci zákazníci své objednávky na opravu vozu smazat.

8. FR 08: Vytváření objednávky dílu

   Pracovníci autoservisů budou moci vytvářet objednávky náhradních dílů pro automobily.

9. FR 09: Odstranění objednávky dílu

   V případě potřeby budou moci pracovníci autoservisů smazat své objednávky autodílů.

10. FR 10: Přijmout objednávku zákazníka

    Zaměstnanec může přijmout jakoukoli objednávku zákazníka z fondu objednávek.

11. FR 11: Změna stavu objednávky zákazníka

    Zaměstnanec bude moci změnit stav objednávky zákazníka, např. dokončit ji nebo zrušit.

12. FR 12: Účet vozidel (patřících zákazníkům)

    Možnost zobrazit všechna vozidla, která se v autoservisu opravují.

13. FR 13: Zobrazit dostupné náhradní díly na skladě

    Možnost pracovníků zobrazit všechny dostupné náhradní díly pro automobily.

14. FR 14: Přidání dílů

    Možnost pracovníků přidávat náhradní díly do skladu.

3.2 Nefunkční požadavky

Nefunkční požadavky jsou omezení týkající se služeb nebo funkcí nabízených systémem. [7] Na základě analýzy také definujeme tyto požadavky na aplikaci.

Obrázek 3.2: Nefunkční požadavky aplikace.

1. NFR 01:

   Webová aplikace musí fungovat stejně dobře ve všech současných prohlížečích, jako je Google Chrome, Opera, Mozilla, Safari.

2. NFR 02:

   Minimální doba odezvy.

3. NFR 03:

   Aplikace by měla poskytovat určité funkce v závislosti na aktuální roli autorizovaného uživatele.

3.3 Uživatelské role

V aplikaci by mělo být více rolí, což umožní rozdělit mezi ně funkce. Hlavními aktéry, komunikujícími v informačním systému, jsou správce, zaměstnanec a zákazník. V tomto informačním systému budou konkrétně tyto role: ADMIN, EMPLOYEE, CUSTOMER.

3.3.1 Správce systému

Správce systému(ADMIN) - Bude mít plný přístup k aplikaci, bude moci prohlížet a odstraňovat uživatelské účty a vozy zákazníků. Bude mít také možnost přidávat do systému nově příchozí autodíly.

3.3.2 Zaměstnanec

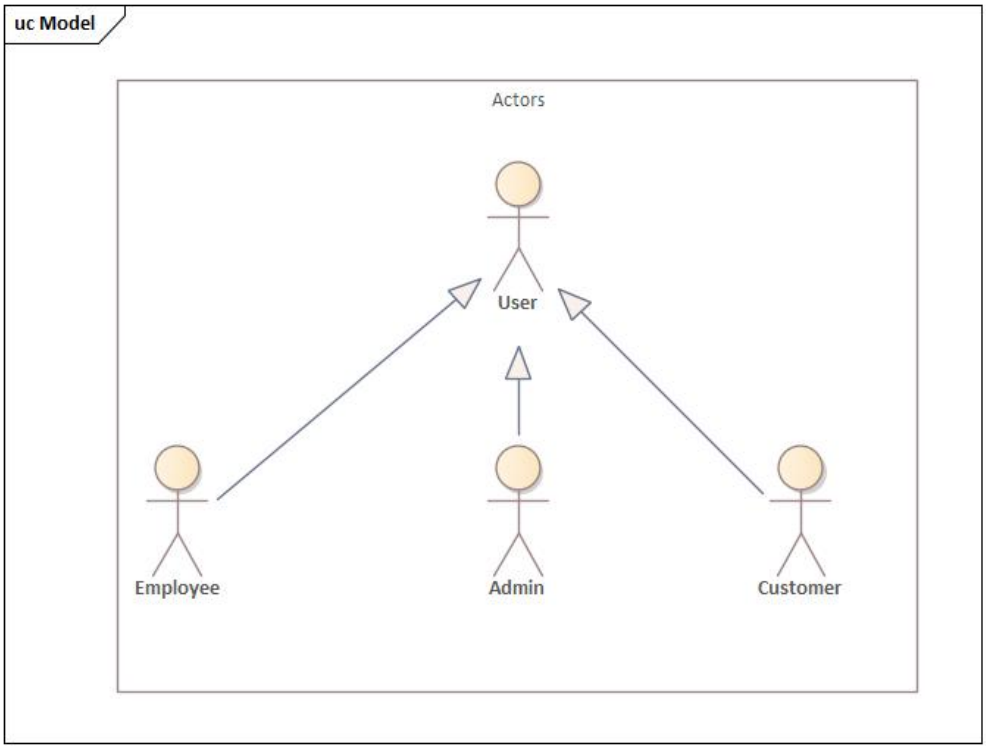
Autorizovaný uživatel, Zaměstnanec(EMPLOYEE) - Uživatel s touto rolí má přístup do aplikace, může vytvářet a mazat objednávky dílu, aktualizovat jejich stav, provádět různé objednávky dílu.

3.3.3 Zákazník

Autorizovaný uživatel, Zákazník(CUSTOMER) - uživatel s touto rolí může vytvářet a spravovat objednávky a může také sledovat aktuální stav objednávky.

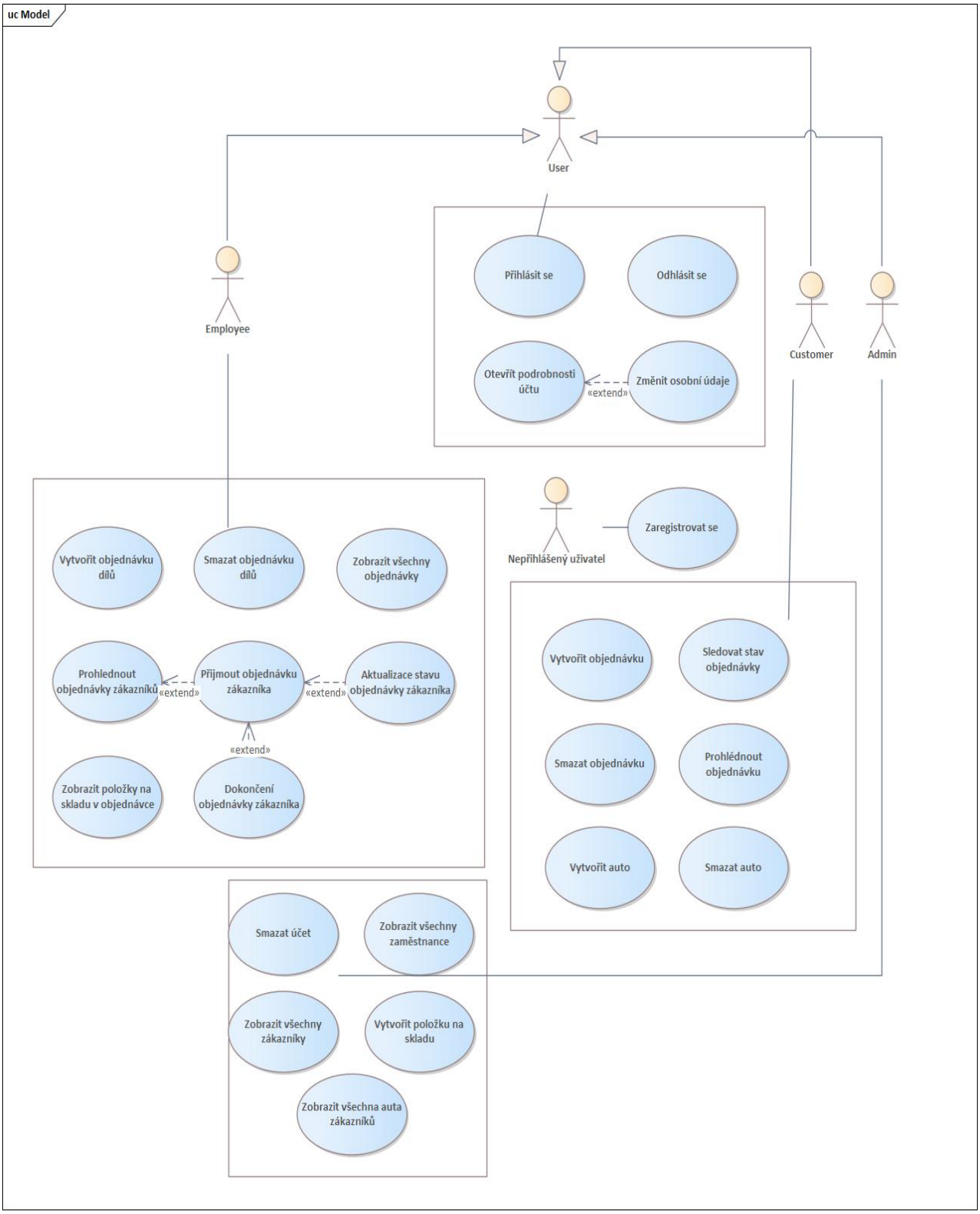
3.4 Diagram užití

V našem systému budou figurovat několik aktérů: administrátor, zaměstnanec a klient, přičemž všichni budou odvozeni od abstraktní třídy USER.



Obrázek 3.3: Aktéři v diagramu užití.

Následující schéma zobrazuje všechny možné případy užití v systému, a to na základě přidělení rolí. Aktér User má společný případ použití s ostatními uživateli.



Obrázek 3.4: Diagram užití aplikace.

Kapitola 4
Analýza technologií

Tato kapitola pojednává o různých technologiích, porovnává je mezi sebou a uvádí jejich výhody a nevýhody. Na základě této analýzy budou vybrány technologie, pomocí kterých bude systém realizován.

4.1 Databaze

Databáze je systém pro ukládání a organizaci velkého množství informací, jako jsou údaje o zákaznících, produktech nebo transakcích. Umožňuje snadno vyhledávat a získávat potřebné informace a také je analyzovat. Databáze může být uložena v počítači nebo na serveru a může k ní přistupovat více uživatelů současně [8].

4.1.1 Microsoft SQL

Microsoft SQL Server je relační databázový a analytický systém, který poskytuje esenciální nástroje pro efektivní správu a analýzu dat pro elektronické obchody, byznys a řešení datových skladů vyvinutý společností Microsoft [9]. Je velmi přehledný, s jednoduchým a intuitivním rozhraním. Jedinou podmínkou pro použití, a tedy možnou nevýhodou, je, že operačním systémem musí být Windows.

4.1.2 Oracle

Oracle Databaze je vysoce výkonný a snadno škálovatelný systém pro správu databází, který vytvořila společnost Oracle Corporation. Tento univerzální databázový systém podporuje mnoho technologií a jazyků, včetně imperativního programovacího jazyka PL/SQL, objektových databází a hierarchických datových modelů, jako jsou XML a XSQL databáze. Vzhledem k velkému množství podporovaných technologií a jazyků však bude pro nováčky velkou výzvou rychle se naučit a začít používat [10, 11].

4.1.3 MySQL

MySQL, v současné době ve vlastnictví společnosti Oracle, je ideální pro malé a středně velké aplikace. Díky své schopnosti fungovat na různých platformách, včetně Windows, představuje alternativu k Microsoft SQL. Avšak při manipulaci s rozsáhlými databázemi obsahujícími miliony řádků nebo při migraci z jedné relační databáze do druhé mohou nastat komplikace [12].

4.1.4 PostgreSQL

PostgreSQL nebo Postgres je open-source systém pro správu relačních databází. Je založený na objektově-relačním modelu a přináší řadu rozmanitých funkcí. Poskytuje podporu pro komplexní struktury a velké spektrum datových typů [13].

4.1.5 MongoDB

MongoDB je multiplatformní databáze dokumentů. Jedná se o databázi NoSQL, která dokáže zpracovávat obrovské množství rychle se měnících nestrukturovaných dat jiným způsobem než relační databáze, což usnadňuje a urychluje vytváření a integraci dat pro aplikace. Jedná se o software s otevřeným zdrojovým kódem [14]. Výhodami NoSQL jsou rychlost zpracování dat, škálovatelnost, ale na druhou stranu databáze sql mají integritu a jsou strukturované na rozdíl od databází nosql.

Databáze	Výhody	Nevýhody
Microsoft SQL	- Přehledné a intuitivní rozhraní	- Omezen na operační systém Windows
Oracle	- Vysoký výkon a škálovatelnost	- Složitější pro nováčky
MySQL	- Alternativa k Microsoft SQL	- Možné problémy s velkými databázemi
PostgreSQL	- Open-source a široká škála funkcí	- Vyžaduje znalost objektově-relačního modelu
MongoDB	- Rychlost zpracování a škálovatelnost	- Nestrukturovaná data, menší integrita

Tabulka 4.1: Výhody a nevýhody databází

4.2 Serverová část

Serverová část je hardwarová a softwarová část služby, která je uložena na serveru, nikoli v počítači nebo prohlížeči uživatele, a která zpracovává přijatá data a odesílá odpověď.

4.2.1 C#

C# je objektově orientovaný programovací jazyk, který byl vyvinut v roce 1998. Patří do skupiny jazyků podobných jazyku C, jako je Java, C++. Jazyk C# zdědil mnohé z výše uvedených jazyků, ale zároveň odstranil některé problémy, které se ukázaly být škodlivé pro vývoj aplikací, například C# nepodporuje vícenásobnou dědičnost tříd na rozdíl od C++ [15].

K vývoji webových aplikací v jazyce C# se používá framework APS.NET, vyvinutý společností Microsoft, který je určen pro běh v systému Windows, ale v roce 2016 byla vytvořena modulární platforma .NET Core, která je kompatibilní s různými operačními systémy [18]. Jinými slovy, je multiplatformní. .NET funguje na architektuře MVC(model-view-controller), kde model poskytuje data a reaguje na příkazy controlleru, view je grafické zobrazení dat, controller reaguje na události a zajišťuje změny v modelu.

4.2.2 Python

Python je vysokoúrovňový programovací jazyk s dynamickým typováním a snadno naučitelnou syntaxí. Používá se k vývoji webových aplikací, vědeckým výpočtům, analýze dat a dalším činnostem. Python má poměrně jednoduchou syntaxi, snadno se učí, obsahuje velké množství knihoven pro různé úlohy a podporuje mnoho operačních systémů [16].

K vývoji webových aplikací v jazyce Python slouží open source framework Django. Používá se k vytváření webových aplikací pomocí architektury MVC (model-view-controller). Django poskytuje širokou škálu nástrojů pro vývoj webových aplikací, například správce modelů, systém šablon, mechanismy směrování a další. Django je velmi populární díky své síle a flexibilitě a díky své architektuře, která umožňuje vývojářům rychle a snadno vytvářet a vyvíjet webové aplikace [17].

4.2.3 Java

Java je objektově orientovaný programovací jazyk vytvořený společností Sun Microsystems (nyní součástí Oracle). Umožňuje vývoj multiplatformního softwaru, protože kód napsaný v jazyce Java lze spustit na libovolné platformě s virtuálním strojem Java. Tento jazyk je oblíbený také pro vývoj mobilních aplikací, síťových aplikací a webových stránek [19].

Spring je framework pro vývoj aplikací v jazyce Java, který poskytuje mnoho nástrojů a knihoven pro vytváření kvalitního kódu a správu závislostí. Spring Boot je modul Springu, který poskytuje rychlý a snadný způsob vytváření nových aplikací v prostředí Spring. Automaticky konfiguruje konfiguraci a závislosti, což vývojářům umožňuje rychleji a snadněji začít pracovat na obchodní logice aplikace.

Jazyk/Framework	Výhody	Nevýhody
C#/.NET	- Jednoduchá syntaxe	- Omezen na platformu Windows
	- Objektově orientovaný programovací jazyk	
	- Výhodný pro vývoj webových aplikací	
	- Velká komunita a podpora ze strany Microsoft	
Python/Django	- Snadno naučitelná syntaxe	- Menší výkon oproti některým jiným jazykům
	- Silná podpora knihoven a nástrojů	- Nízká paralelní zpracování
	- Dobrá volba pro vědecké výpočty	- Větší paměťová náročnost
Java/Spring	- Multiplatformní podpora	- Delší čas potřebný pro vývojový cyklus
	- Velká komunita a mnoho dostupných knihoven	- Vyšší paměťová náročnost
	- Podpora enterprise aplikací	- Složitější syntaxe pro začátečníky
	- Vysoký výkon a škálovatelnost	

Tabulka 4.2: Výhody a nevýhody programovacích jazyků na straně serveru a jejich frameworků

4.3 Klientská část

Klientská část je to, co uživatel vidí na obrazovce počítače nebo mobilního zařízení, když navštíví webové stránky nebo používá webovou aplikaci. Zahrnuje to design, grafiku a interaktivitu webových stránek. Frontendoví vývojáři používají programovací jazyky, jako jsou HTML, CSS a většinou JavaScript, k vytvoření vzhledu webových stránek.

4.3.1 JavaScript

JavaScript je programovací jazyk používaný k vytváření interaktivních webových stránek a interakci s uživatelem. Lze jej použít k vytváření efektů stránek, ověřování formulářů a manipulaci s daty na straně klienta. JavaScript lze také použít k vytváření kompletních webových aplikací pomocí knihoven a frameworků třetích stran, jako jsou React a Angular.

ReactJS

React je knihovna jazyka JavaScript pro vytváření uživatelských rozhraní. Byla vyvinuta a je spravována společností Facebook a je široce používána pro vytváření webových aplikací i mobilních aplikací pomocí React Native. React používá koncept zvaný "komponenty", který vývojářům umožňuje vytvářet prvky uživatelského rozhraní pomocí opakovaně použitelného a modulárního kódu. Hlavní výhodou Reactu je jeho schopnost efektivně aktualizovat a vykreslovat komponenty v reakci na změny dat, které jsou známé jako virtuální DOM. Výsledkem je rychlé a citlivé uživatelské rozhraní [20].

Angular

Angular je populární open-source framework JavaScriptu pro vytváření webových aplikací. Je vyvíjen a spravován společností Google a je široce používán pro vytváření jednostránkových aplikací (SPA) i dynamických webových aplikací. Angular využívá architekturu založenou na komponentách a výkonný jazyk šablon, což vývojářům usnadňuje vytváření a údržbu rozsáhlých aplikací s čistým, opakovaně použitelným kódem. Mezi klíčové funkce jazyka Angular patří obousměrné vázání dat, vstřikování závislostí a vestavěný systém modulů [20].

Jazyk/Framework/Knihovna	Výhody	Nevýhody
JavaScript	- Používá se pro interaktivní webové stránky a interakci s uživatelem	- Náchylnost k chybám a nekonzistenci mezi různými prohlížeči
	- Široká podpora a rozšířenost	- Nižší výkon ve srovnání s některými kompilovanými jazyky
React	- Efektivní aktualizace a vykreslování komponentů	- Vyžaduje správu stavu a dat
	- Komponentní architektura umožňuje znovupoužitelný kód	- Naučení se syntaktiky a konceptů Reactu může být na začátku obtížné
Angular	- Plně vybavený framework pro vytváření webových aplikací	- Větší složitost a náročnost na učení se
	- Obousměrné vázání dat a vestavěný systém modulů	- Větší velikost souborů a pomalejší inicializace než u jiných frameworků

Tabulka 4.3: Výhody a nevýhody programovacích jazyků na straně klienta a jejich frameworků

4.4 Souhrn

Po analýze dostupných technologií a programovacích jazyků bylo nakonec rozhodnuto, že Java bude použita jako jazyk na straně serveru, JavaScript na straně klienta a PostgreSQL jako databáze jsou nejlepšími možnostmi pro vývoj tohoto projektu. Toto rozhodnutí bylo založeno na skutečnosti, že tyto technologie dobře splňují požadavky projektu a nabízejí spolehlivou a stabilní vývojovou platformu. Kromě toho byl výběr těchto technologií ovlivněn také předchozími zkušenostmi a odbornými znalostmi v těchto technologiích získanými během bakalářského studia, které mohou zajistit efektivnější a účinnější proces vývoje.

Kapitola 5

Návrh

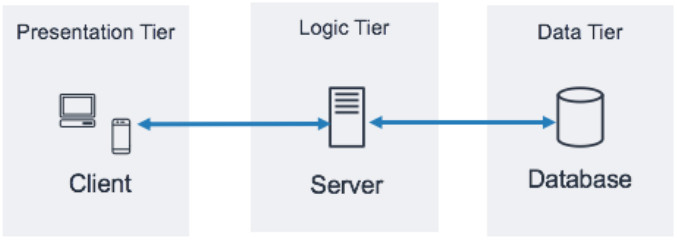
5.1 Architektura

Architektura aplikace je celkové uspořádání a struktura kódu aplikace, která definuje, jakým způsobem spolu komponenty komunikují a jak se mezi nimi přenášejí data. Existují různé typy aplikačních architektur, z nichž každá má své silné a slabé stránky.

5.1.1 Třívrstvá architektura

Třívrstvá architektura je návrhový vzor softwaru, který rozděluje aplikaci na tři logické vrstvy: prezentační vrstvu, vrstvu aplikační logiky a vrstvu ukládání dat. Každá vrstva má specifickou odpovědnost a komunikuje s ostatními vrstvami prostřednictvím přesně definovaných rozhraní. Třívrstvá architektura je oblíbená, protože poskytuje jasné oddělení zájmů, což usnadňuje pochopení, údržbu a rozšiřování aplikace. Kromě toho umožňuje provádět změny v jedné vrstvě, aniž by byly ovlivněny ostatní vrstvy, což usnadňuje aktualizaci a škálování aplikace.

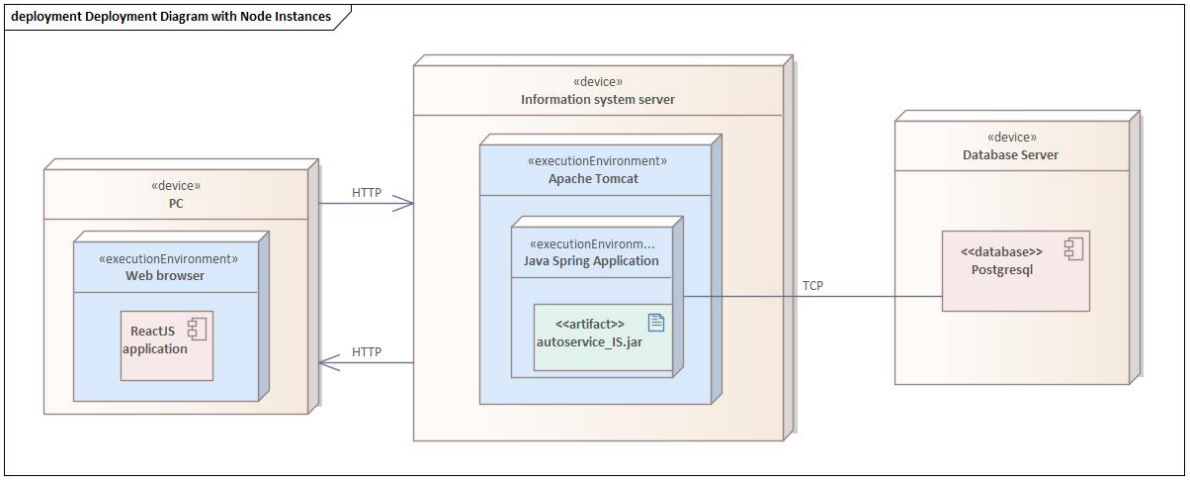
- Prezentační vrstva: Řeší uživatelské rozhraní a interakci s uživatelem.
- Vrstva obchodní logiky: Řeší obchodní pravidla a funkce, včetně interakce s databází.
- Datová vrstva: Odpovídá za ukládání, obnovu a aktualizaci dat aplikace.



Obrázek 5.1: Zobrazení 3 vrstev v třívrstvé architektuře

5.2 Diagram nasazení

Diagram nasazení je typ strukturálního diagramu, který je určen k zobrazení architektury systému popisem rozmístění fyzických artefaktů v uzlech. Ukazuje, jak všechny části aplikace spolupracují a jak souvisejí s hardwarem nebo softwarem, na kterém běží.



Obrázek 5.2: Diagram nasazení aplikace.

První uzel v diagramu představuje klientskou část aplikace. Obsahuje uživatelské rozhraní a komunikuje se serverem za účelem přijímání a odesílání dat.

Druhý uzel představuje serverovou část, jádro naší aplikace. Jedná se o hlavní zpracovatel dat, který přijímá požadavky z klientské vrstvy, zpracovává je a vrací požadovaná data.

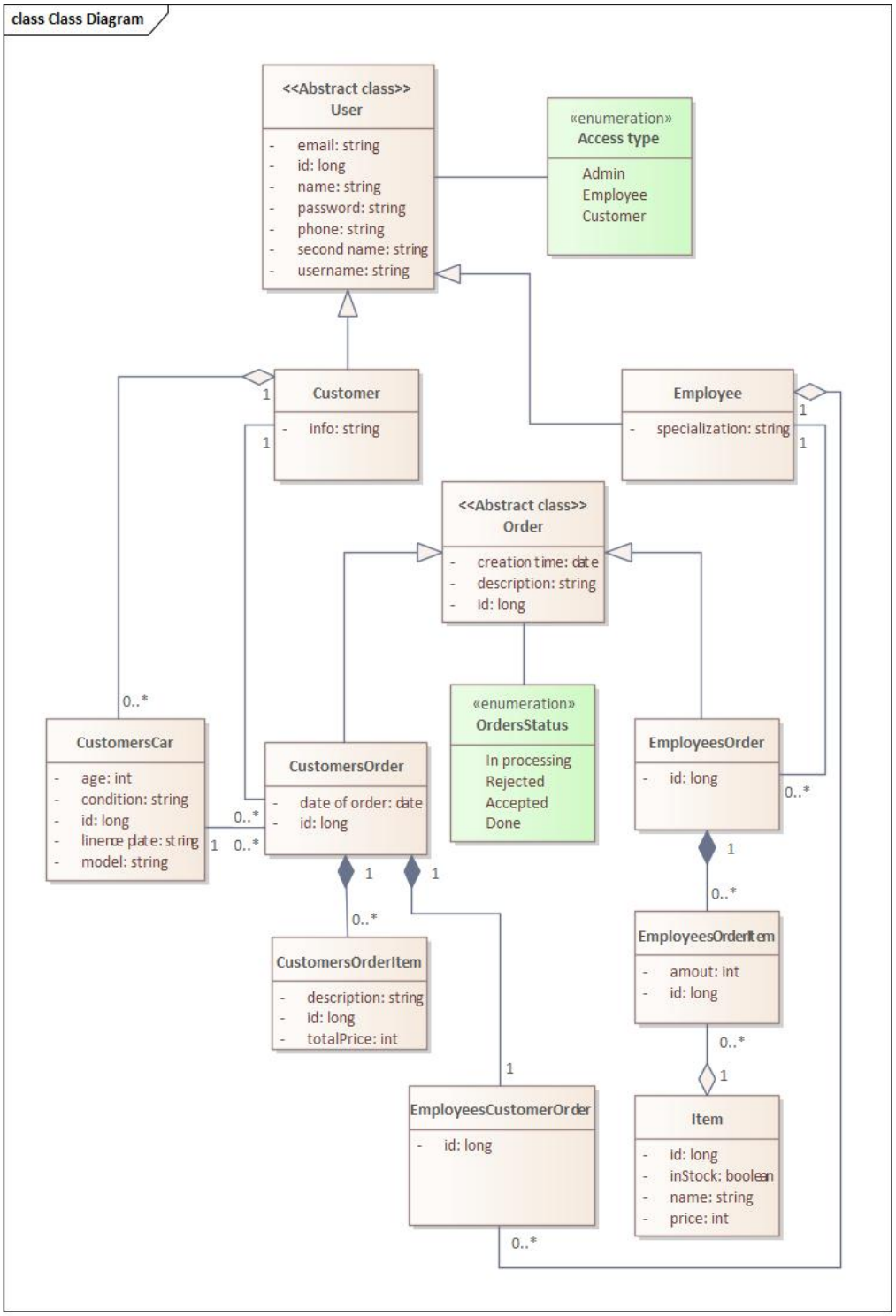
A konečně třetí uzel představuje databázi. Jedná se o úložiště všech dat, se kterými aplikace pracuje. Server přistupuje k tomuto uzlu, aby načítal a ukládal data v reakci na požadavky klientské vrstvy.

5.3 Diagram tříd

Diagram tříd je grafické znázornění, které se používá v objektově orientovaném návrhu k zobrazení struktury tříd a jejich vztahů. Zobrazuje třídy, jejich vlastnosti a metody a vztahy mezi třídami, jako je dědičnost, agregace a kompozice.

Níže je uveden diagram tříd, který ukazuje, které entity bude náš systém obsahovat, přibližně tak, jak budou existovat v databázi, s výjimkou entit typu enum, které budou v databázi reprezentovány jako atributy entity USER nebo ORDER.

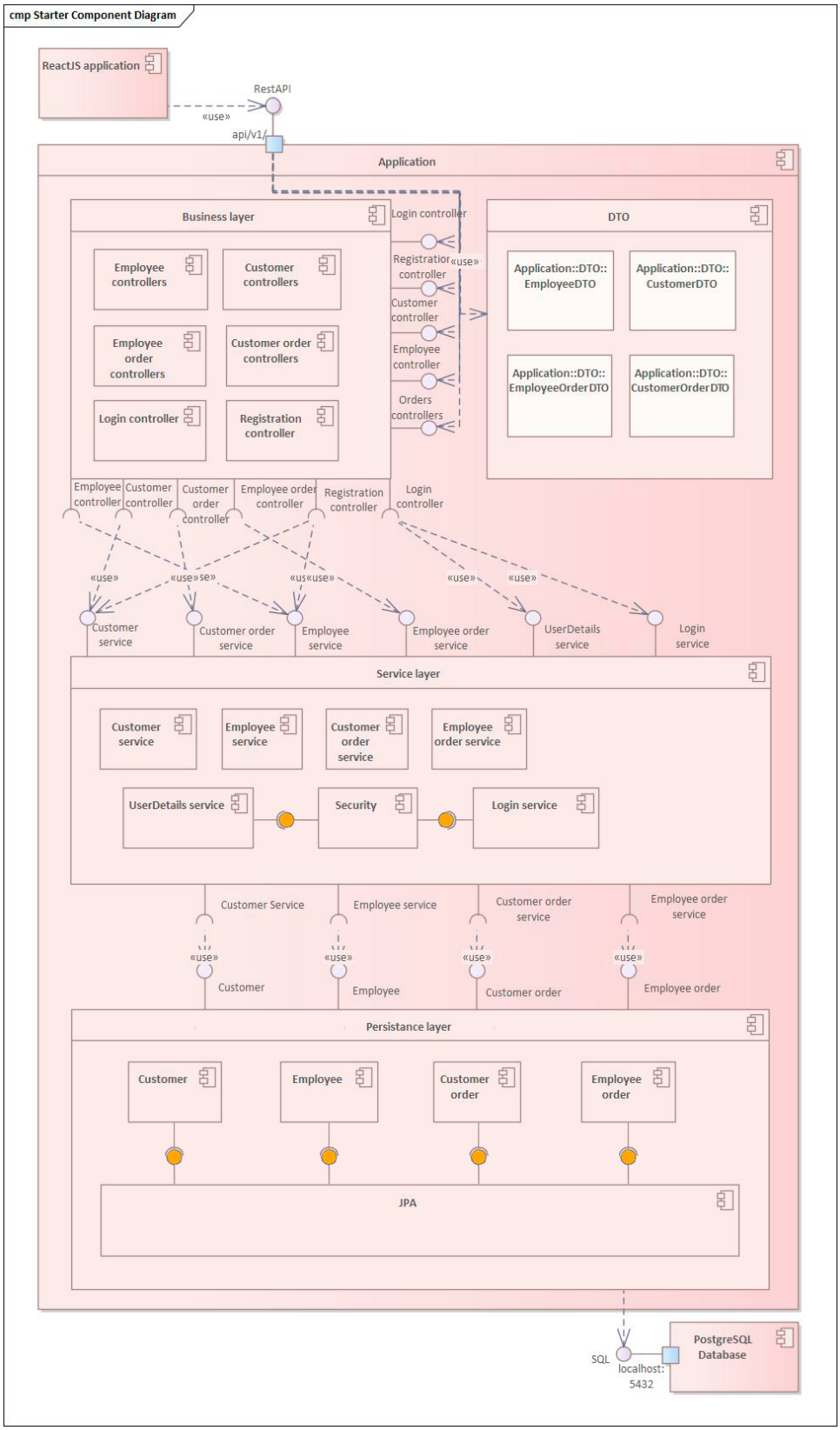
Aplikace bude rozdělena na 2 typy uživatelů: Zákazník, Zaměstnanec. Každý z nich bude mít své typy objednávek, které se liší funkčností, ve schématu je také entita EmployeesCustomerOrder, které umožňuje zaměstnanci přijímat objednávky zákazníků k provedení. Také každý z nich má svůj vlastní typ přístupu, tento enum je určen k vytvoření zabezpečení v aplikaci.



Obrázek 5.3: Diagram tříd aplikace.

5.4 Serverová část

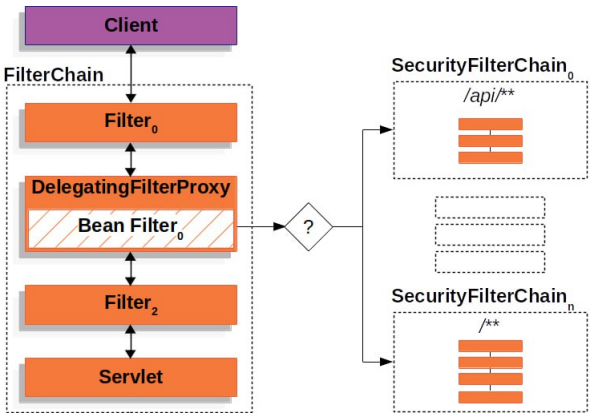
Serverová část aplikace bude implementována v programovacím jazyce Java s využitím frameworku Spring. Spring poskytuje rozsáhlou sadu nástrojů pro zjednodušení vývoje a zvýšení efektivity serveru. Spring v sobě obsahuje aplikační server Apache Tomcat, který dále zjednodušuje proces vývoje tím, že nevyžaduje samostatnou instalaci a konfiguraci serveru. Apache Tomcat bude použit k nasazení naší aplikace na server, což zajistí její stabilní a efektivní provoz. Níže je uveden diagram nasazení, ze kterého bude vytvořena serverová část aplikace.



Obrázek 5.4: Diagram komponent aplikace.

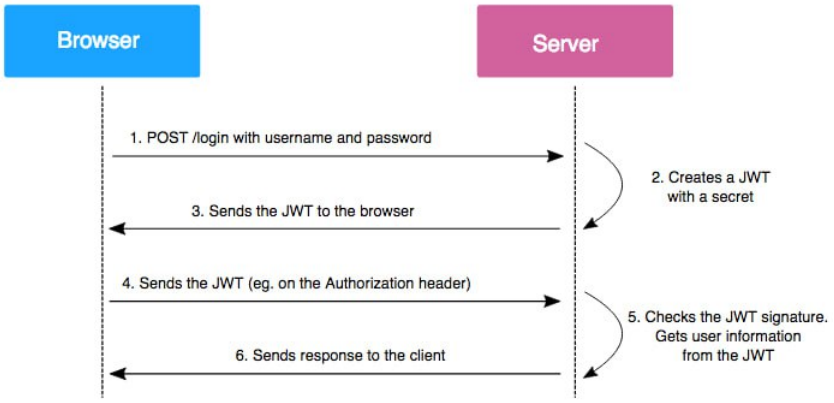
5.5 Zabezpečení

Bezpečnost je jedním z nejdůležitějších aspektů aplikace. V naší aplikaci zajišťuje bezpečnost Spring Security. Spring Security je výkonný nástroj pro zajištění bezpečnosti v aplikacích postavených na frameworku Spring. Jedním z důležitých aspektů zabezpečení je ověřování uživatelů, která je zajištěna prostřednictvím správy rolí a oprávnění. V rámci ověřování a autorizace umožňuje Spring Security používat JWT (JSON Web Token) jako mechanismus pro správu a přenos identity uživatele.



Obrázek 5.5: Princip spring security filter chain [30].

JWT je formát tokenu, který obsahuje informace o uživateli a jeho oprávnění. Používá se k ověřování identity uživatele a zajišťuje, že pouze oprávněné osoby mají přístup k určitým zdrojům v aplikaci. JWT je podepsán digitálním klíčem a může být předáván mezi klientem a serverem prostřednictvím HTTP hlavičky nebo v URL. Po ověření uživatele vygeneruje server JWT, který pak klient použije k ověření své identity pro další požadavky. Každý JWT obsahuje tvrzení, která může server ověřit, aniž by musel nahlížet do databáze. To výrazně zlepšuje výkonnost a škálovatelnost aplikace.



Obrázek 5.6: Princip fungování JWT [31].

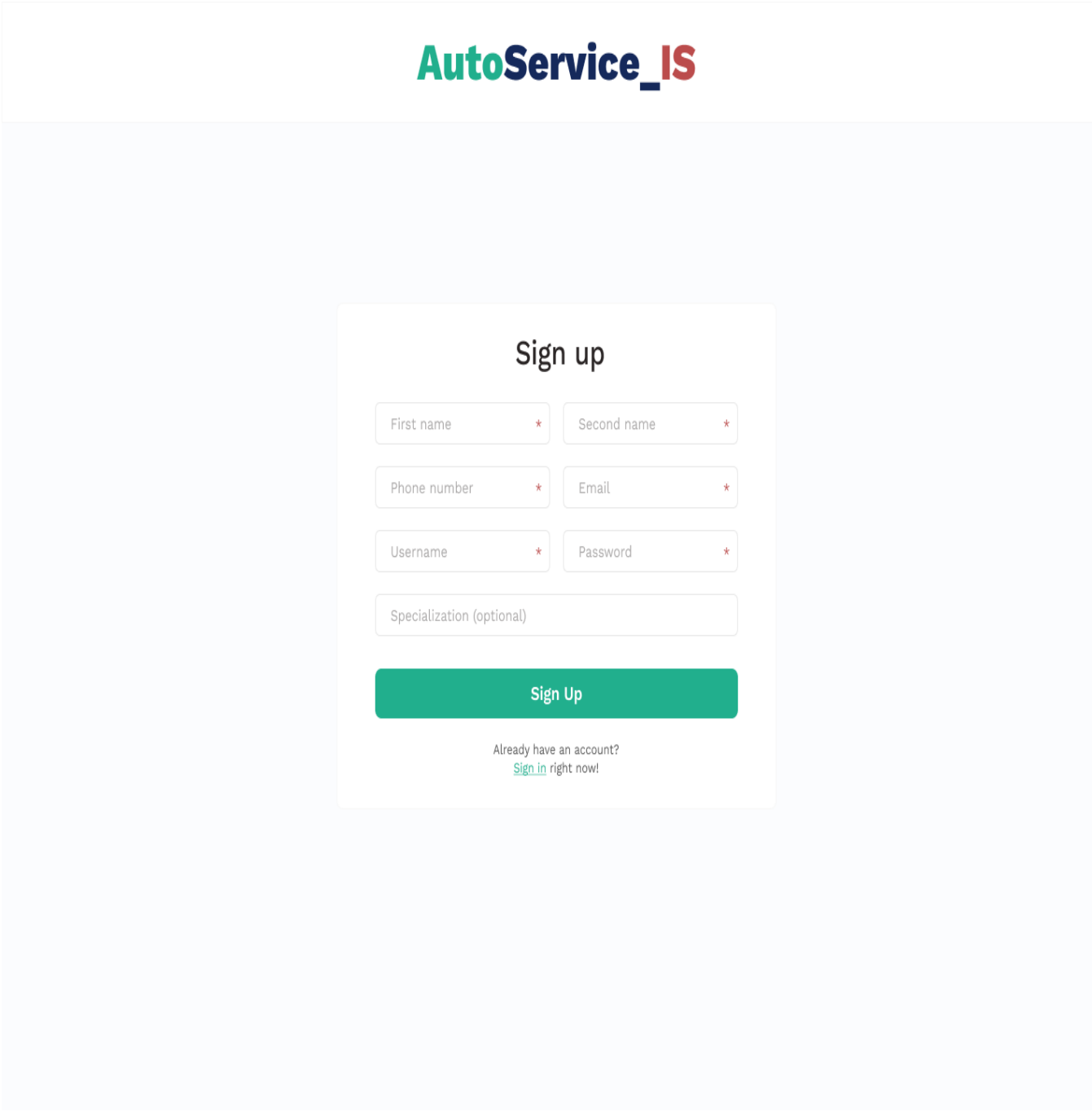
Spring Security s JWT poskytuje efektivní a bezpečné řešení pro ověřování a autorizaci v aplikaci a chrání data a funkce před neoprávněným přístupem. Spring Security umožňuje pomocí anotace @PreAuthorize definovat podmínky přístupu těsně před spuštěním metody. Tato anotace se používá k omezení přístupu k určitým metodám na základě uživatelských rolí nebo oprávnění. Pokud je podmínka anotace vyhodnocena jako „true“, metoda se provede. Pokud je podmínka vyhodnocena jako „false“, přístup k metodě je odepřen. Pomocí anotace @PreAuthorize("hasRole('ADMIN')") lze tedy například omezit přístup k metodě pouze na správce.

5.6 Klientská část

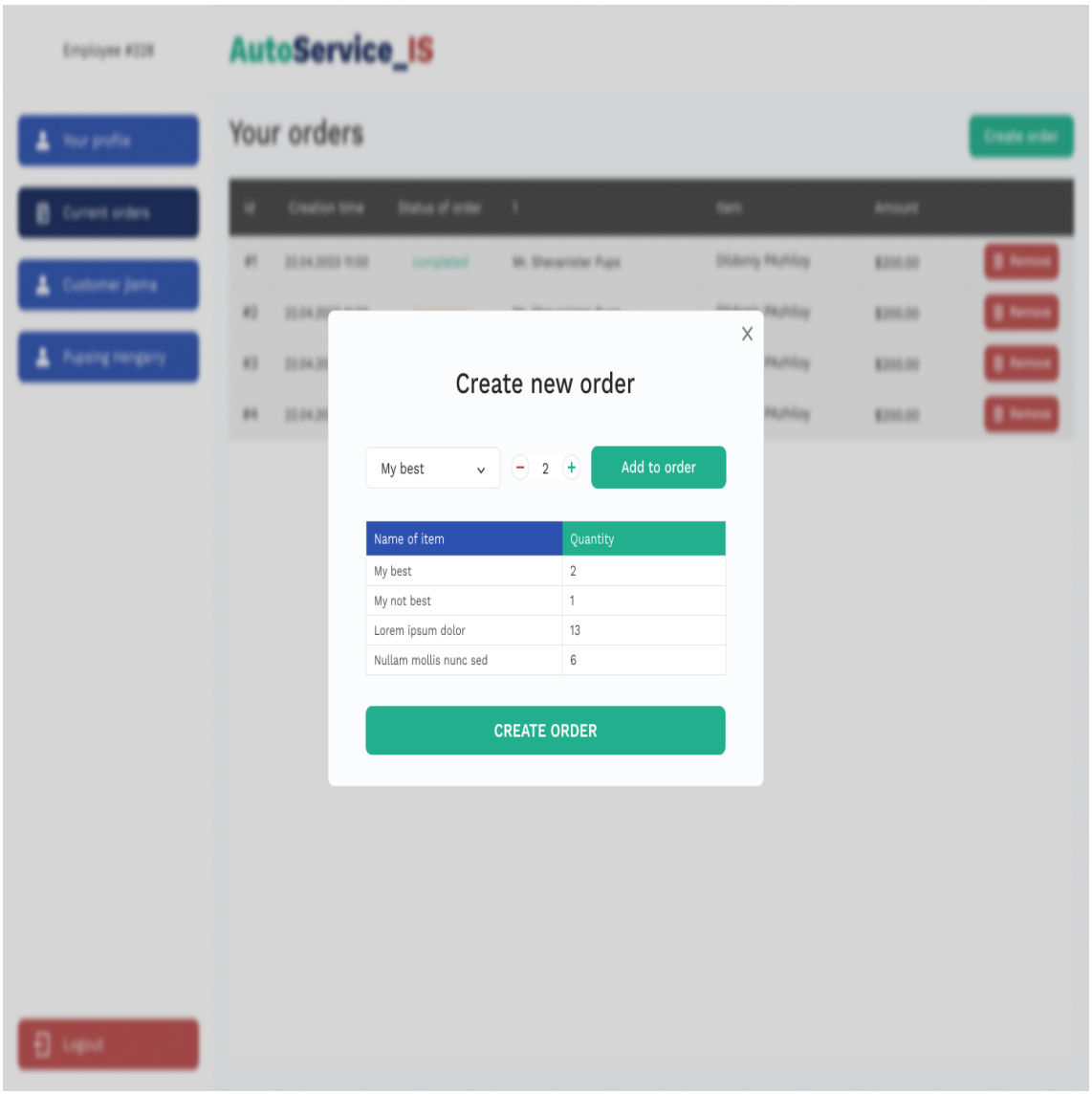
V této podkapitole se zaměříme na návrh klientské části naší aplikace. Pomocí jazyka JavaScript v kombinaci s knihovnou ReactJS vytvoříme interaktivní uživatelské rozhraní, které zajistí efektivní a pohodlnou interakci uživatele s aplikací. Jazyk JavaScript byl zvolen pro svou flexibilitu a výkonnost, zatímco ReactJS pro svůj komponentový přístup, který zjednodušuje vývoj složitých rozhraní. K vytvoření struktury webových stránek používáme jazyk HTML a k definování vzhledu prvků rozhraní používáme jazyk CSS. Tyto technologie jsou standardem ve vývoji webových stránek a umožňují nám vytvářet rozhraní, která jsou vizuálně přitažlivá a snadno použitelná.

5.6.1 Návrh uživatelského rozhraní

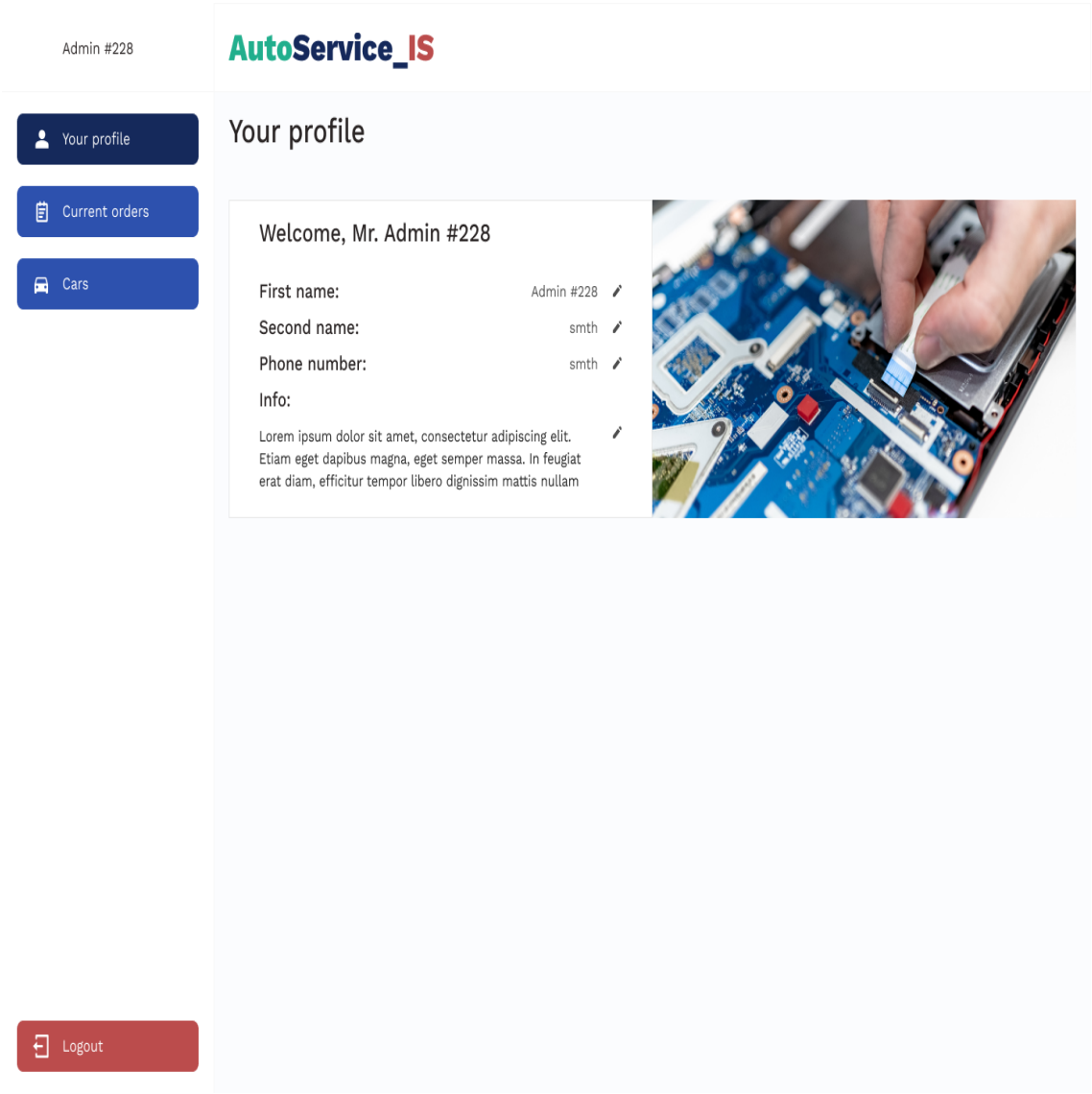
Vývoj uživatelského rozhraní pro tento projekt prošel několika klíčovými kroky, aby vše bylo organizované a efektivní. Nejprve byly pomocí návrhového nástroje Figma vytvořeny podrobné prototypy rozhraní.



Obrázek 5.7: Prototyp přihlašování.



Obrázek 5.8: Prototyp stránky pro vytvoření objednávky.



Obrázek 5.9: Prototyp profilové stránky.

5.7 Databaze

Při vývoji aplikace byla zvolena databáze PostgreSQL na základě předchozích zkušeností s tímto systémem. PostgreSQL nabízí spolehlivost, flexibilitu a kompatibilitu s ACID, což přispívá k efektivní správě dat. Pro interakci s databází byl použit nástroj pgAdmin, který umožňuje snadnou správu a monitorování databáze.

```
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.hibernate.show-sql=true
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=username
spring.datasource.password=password
```

Obrázek 5.10: Konfigurace aplikace pro přístup k databázi.

5.8 Hibernate

Hibernate je nástroj pro Javu, který usnadňuje práci s databázemi. Převádí informace z databáze na objekty jazyka Java, které můžete použít ve svém kódu. Tomu se říká objektově-relační mapování (ORM).

Díky Hibernate není nutné psát spoustu složitého kódu, aby bylo možné získat data z databáze nebo je uložit zpět do databáze. Místo toho stačí pracovat s objekty jazyka Java jako obvykle. Hibernate je také zprostředkovatelem JPA, což znamená, že implementuje rozhraní Java Persistence API (JPA) [22].

5.9 JPA

Java Persistence API (JPA) je specifikace jazyka Java, která představuje standardizovaný rámec pro mapování objektů v jazyce Java na data v databázi. JPA je založena na konceptu ORM (Object-Relational Mapping), který umožňuje vývojářům pracovat s databází na úrovni objektů, nikoliv pomocí dotazů SQL. JPA poskytuje sadu anotací a rozhraní API, které lze použít k definování způsobu mapování tříd jazyka Java a jejich instancí (objektů) na tabulky a záznamy v databázi [21].

5.10 Maven

Maven je nástroj, který pomáhá automatizovat a zjednodušit mnoho aspektů vývoje projektů Java. Pomáhá řídit všechny části projektu, včetně sledování závislostí, sestavování projektu, testování, nasazení a tvorby dokumentace. Maven pomáhá při automatizaci procesu sestavování projektu. To znamená,

že automaticky zkompiluje kód, zabalí jej do spustitelného souboru a připraví jej k nasazení. Je důležité poznamenat, že Maven hraje velkou roli při správě životního cyklu projektu. Výsledkem je nástroj, který lze nyní použít pro tvorbu a správu jakéhokoli projektu v jazyce Java [23].

5.11 REST

REST, což je zkratka pro Representational State Transfer, je architektonický styl, který se používá v aplikacích pro komunikaci mezi serverovou a klientskou stranou. REST nám umožňuje komunikovat se serverem prostřednictvím požadavků HTTP, jako jsou GET, POST, PUT a DELETE, které lze použít k manipulaci s daty. Aplikace definuje sadu RESTful endpointů na straně serveru, z nichž každý představuje adresu URL spojenou s konkrétní metodou HTTP. Klientská strana pak na tyto endpointy provádí požadavky HTTP a provádí s daty různé operace.

```
no usages  Dmitriy Shevchenko +1
@PostMapping
@PreAuthorize("hasAnyAuthority('CUSTOMER_ACCESS', 'ADMIN_ACCESS')")
public ResponseEntity<?> saveCustomer(CustomerDto dto) {
    CustomerDto customerDto = customerService.addCustomer(dto);
    return ResponseEntity.ok(customerDto);
}
```

Obrázek 5.11: Endpoint v kódu na straně serveru

```
const API_URL = "http://localhost:8080/api/v1/registration";
1 usage   Dmitriy Shevchenko
const handleSubmit = async (event) => {
    event.preventDefault();
    try {
        const response = await axios.post(API_URL, data: {
            firstName: form.firstName,
            secondName: form.secondName,
            username: form.username,
            accessType: form.accessType,
            email: form.email,
            phone: form.phone,
            password: form.password,
            info: form.info
        })
        window.location.href="/login";
    } catch (error) {
        console.error(error);
    }
}
```

Obrázek 5.12: Endpoint v kódu na straně klienta

5.12 Souhrn

Tato kapitola popisuje proces vývoje aplikace od návrhu po implementaci. Podrobně jsou popsány všechny použité technologie, které byly vybrány pro úspěšný vývoj aplikace a dosažení jejích cílů.

Kapitola 6
Implementace

V této kapitole budou podrobně rozebrány implementační detaily aplikace. Zároveň budou stručně uvedeny základní pojmy související s jednotlivými technologiemi použitými při vývoji aplikace.

6.1 Implementace serverové části

V této podkapitole přejdeme k vývoji serverové části naší aplikace. Strana serveru neboli back-end hraje klíčovou roli v každé webové aplikaci nebo službě. Řeší obchodní logiku, spravuje databázi a propojuje klientskou část aplikace s potřebnými zdroji a službami. Blíže rozebereme technologie a nástroje, které byly k vytvoření serverové části použity.

6.1.1 Apache Tomcat

Apache Tomcat je otevřený aplikační server Java, který poskytuje čisté prostředí Java pro provoz webových aplikací. To znamená, že Tomcat je schopen zpracovávat kód jazyka Java používaný k vytváření webových aplikací a obsluze webových požadavků. Apache Tomcat má výhody flexibility, snadného použití a správy, otevřeného kódu a velké podpory ze strany komunity [26].

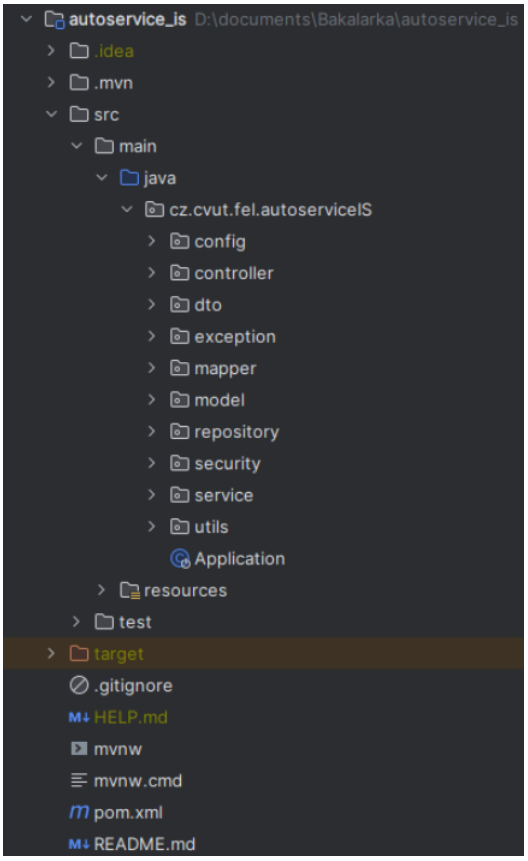
6.1.2 Spring Boot

Spring Boot je projekt s otevřeným zdrojovým kódem, jehož cílem je usnadnit vytváření aplikací tím, že poskytuje sadu výchozích nastavení, která umožňují začít rychleji a s menším úsilím. To znamená, že Spring Boot výrazně zjednodušuje proces vývoje tím, že poskytuje širokou škálu nástrojů a přizpůsobení. Odstraňuje potřebu definice šablonového kódu a konfigurace XML [24, 25]. Spring Boot je tedy vynikající volbou pro vývoj moderních webových aplikací v jazyce Java, zjednodušuje a zrychluje proces vývoje.

6.1.3 Struktura serverové části

Kód serverové části se nachází ve složce

/autoservice_is/src/main/java/cz/cvut/fel/autoserviceIS/. Při dalším vývoji strukturujeme naši aplikaci tak, že ji rozdělíme na klíčové vrstvy: modely, repository a servisy. Tato struktura nám pomáhá efektivněji organizovat kód a poskytuje jasné vazby mezi jednotlivými složkami systému.



Obrázek 6.1: Struktura kódu serverové strany.

- /config/ - adresář obsahuje všechny potřebné konfigurační soubory, které slouží k nastavení parametrů naší aplikace.
- /controller/ - adresář obsahuje kontroléry REST, což jsou rozhraní API pro komunikaci s klientskou částí aplikace.
- /dto/ - DTO neboli Data Transfer Object je návrhový vzor, který se používá k přenosu dat mezi subsystémy aplikace. V kontextu naší aplikace obsahuje adresář "dto"třídy DTO, které se používají k přenosu dat mezi vrstvami aplikace, zejména mezi řadiči a službami. Tyto objekty zjednodušují přenos dat, protože obsahují pouze nezbytné údaje bez zbytečných informací souvisejících s aplikační logikou.
- /exception/ - tento adresář obsahuje nestandardní výjimky.
- /mapper/ - adresář v naší aplikaci zpracovává transformaci objektů DTO (Data Transfer Objects) na entity a naopak. Třídy mapperu v tomto adresáři jsou zodpovědné za převod mezi objekty DTO a entitami. To je obvykle nutné, když data přicházejí z externího zdroje (např. požadavek klienta) ve formě DTO a je třeba je převést na entity pro další zpracování a uložení v databázi.

- /model/ - adresář hraje v naší aplikaci klíčovou roli, protože obsahuje modelové třídy, které představují entity naší obchodní logiky. Modely jsou třídy, které popisují strukturu dat a chování a interakce těchto dat v aplikaci.
- /repository/ - adresář obsahuje úložiště, která hrají důležitou roli v architektuře naší aplikace tím, že propojují datovou vrstvu a vrstvu obchodní logiky.
- /security/ - adresář má ve struktuře naší aplikace zvláštní místo, protože obsahuje kód zodpovědný za zajištění bezpečnosti aplikace.
- /service/ - adresář obsahuje třídy služeb, které jsou základem business logiky naší aplikace.
- /utils/ - obsahuje různé pomocné soubory pro aplikaci.
- /test/ - všechny testy jsou v tomto adresáři.

6.2 Implementace klientské části

V této podkapitole se zaměříme na vývoj klientské části naší aplikace. Klient neboli front-end je rozhraní, se kterým uživatel komunikuje. Zahrnuje návrh, uživatelské rozhraní a interakci se serverovou částí aplikace. Podíváme se na technologie a nástroje používané k vytvoření klientské strany

6.2.1 React a React Routes

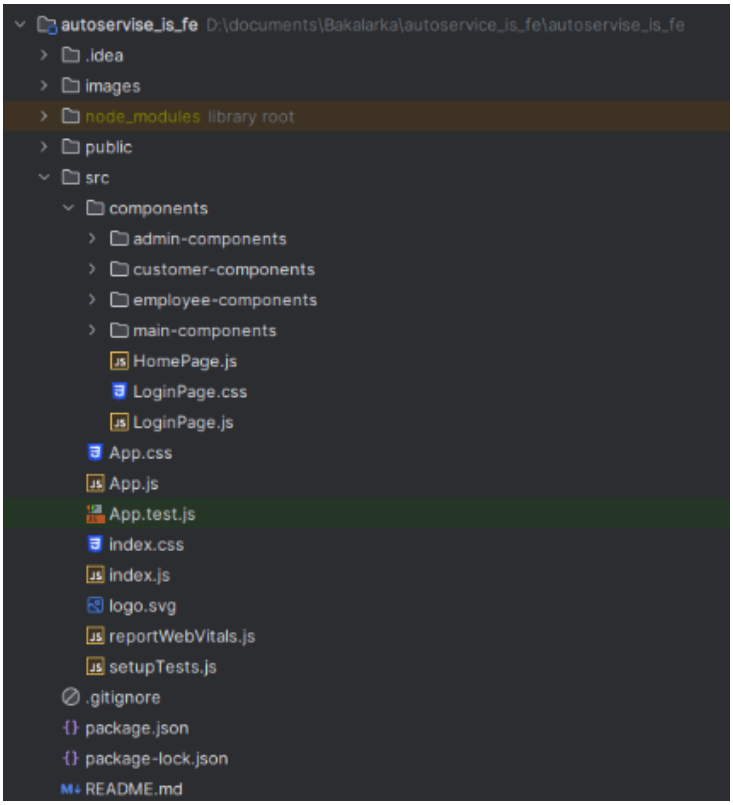
React se díky použití virtuálního DOM vyznačuje vysokým výkonem. Jedná se o efektivní abstrakci nad skutečným DOM, která umožňuje Reactu optimalizovat vykreslování komponent a minimalizovat aktualizace skutečného DOM. React Router je doplněk pro React, který poskytuje výkonné možnosti směrování. Umožňuje propojit adresy URL s komponentami React, což zajišťuje plynulou navigaci bez nutnosti načítat celou stránku. To zlepšuje výkon a uživatelský zážitek tím, že jsou přechody mezi stránkami rychlé a plynulé [27].

```
5+ usages  Dmitriy Shevchenko +1
function App() {
  return (
      <BrowserRouter>
          <Routes>
              <Route exact path ="" element={<HomePage/>}/>
              <Route path="/login" element={<LoginPage />}/>
              <Route path="/registrationEmp" element={<EmployeeRegistrationPage />}/>
              <Route path="/employeeOrder" element={<EmployeeOrderPage/>}/>
              <Route path="/profile" element={<ProfilePage/>}/>
              <Route path="/customerOffers" element={<EmployeeCustomerOrders/>}/>
              <Route path="/current-offers" element={<EmployeeCurrentOffers/>}/>
              <Route path="/employeeOffers" element={<EmployeeOffersTable/>}/>
              <Route path="/registrationCus" element={<CustomerRegistrarionPage/>}/>
              <Route path="/customerOrder" element={<CustomerOrderPage/>}/>
              <Route path="/customerCars" element={<CustomerCarsPage/>}/>
              <Route path="/employees" element={<AdminEmployeesPage/>}/>
              <Route path="/customers" element={<AdminCustomersPage/>}/>
              <Route path="/customersCars" element={<AdminCarsPage/>}/>
              <Route path="/itemHub" element={<AdminItemsPage/>}/>
          </Routes>
      </BrowserRouter>
  );
}
```

Obrázek 6.2: React routy v aplikaci.

6.2.2 Struktura klientské části

Kód klientské části aplikace se nachází ve složce /autoservice_is_fe/src/



Obrázek 6.3: Struktura kódu klientské části.

- /components/ - adresář obsahuje reaktivní komponenty aplikace, z nichž každá je samostatnou, opakovaně použitelnou jednotkou uživatelského rozhraní.
- /components/admin-components/ - adresář pro komponenty správce.
- /components/customer-components/ - adresář zákaznických komponent.
- /components/employee-components/ - adresář pro zaměstnanecké komponenty.
- /components/main-components/ - adresář hlavních komponent.

6.3 Vývojové prostředí

6.3.1 JetBrains IntelliJ IDEA

IntelliJ IDEA je integrované vývojové prostředí (IDE) vyvinuté společností JetBrains, které poskytuje výkonné nástroje a funkce pro vývoj softwaru. Je určeno k vývoji aplikací v různých programovacích jazycích včetně jazyků Java, Kotlin, Python, JavaScript a dalších.

6.3.2 Postman

Postman je nástroj pro testování a vývoj rozhraní API (Application Programming Interface). Poskytuje uživatelsky přívětivé rozhraní, které umožňuje odesílat požadavky HTTP na rozhraní API a také zobrazovat a analyzovat přijaté odpovědi. Během vývoje byl intenzivně používán k testování endpointů.

6.3.3 Figma

Figma je nástroj pro návrh a prototypování rozhraní, který umožňuje vytvářet a upravovat návrhy uživatelského rozhraní pro webové a mobilní aplikace. Pomocí Figmy je možné přizpůsobovat prvky rozhraní, přidávat vizuální styly, definovat přechody mezi obrazovkami a vytvářet interaktivní prototypy.

6.3.4 Microsoft Edge

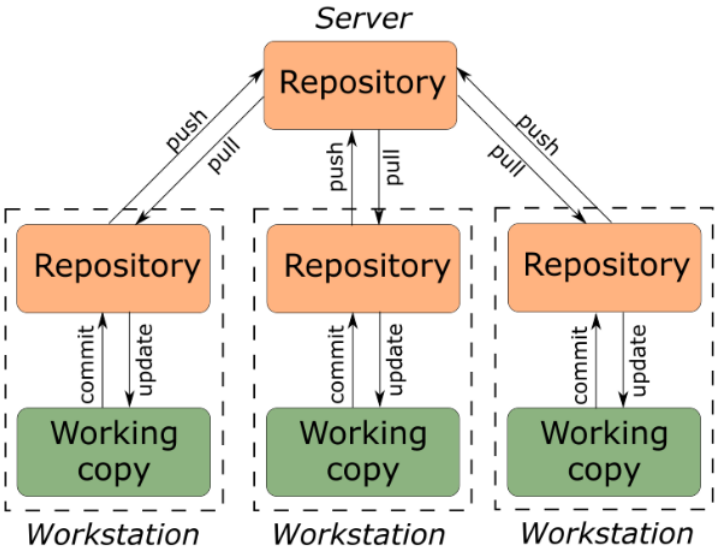
Během vývoje aplikace byl použit také prohlížeč Microsoft Edge. Tento prohlížeč byl vybrán pro svou kompatibilitu s operačním systémem Windows, podporu moderních webových standardů a dostupnost vývojářských nástrojů, které pomohly odhalit a opravit případné chyby. To zajistilo, že aplikace funguje správně a poskytuje příjemný uživatelský zážitek.

6.3.5 PgAdmin

PgAdmin je nástroj pro správu databází PostgreSQL. Poskytuje uživatelsky přívětivé a intuitivní rozhraní, které umožňuje snadné sledování a správu databáze. Poskytuje celou řadu funkcí pro správu databáze, včetně vytváření tabulek, spouštění dotazů, konfigurace oprávnění a mnoha dalších.

6.4 Verzování

Správa verzí umožňuje více vývojářům pracovat na stejném projektu současně bez obav z přepsání práce někoho jiného. Každý vývojář pracuje se svou vlastní kopií kódu a poté své změny sloučí s hlavní verzí projektu. U konkrétních systémů verzování můžeme dále rozlišit lokální a pracovní kopii [28].



Obrázek 6.4: Princip fungování systému správy verzí [33].

6.4.1 Gitlab

GitLab je úložiště kódu s otevřeným zdrojovým kódem a platforma pro společný vývoj softwaru pro velké projekty. GitLab je pro jednotlivce zdarma. Během vývoje svého projektu jsem aktivně používal GitLab jako systém pro správu verzí. To mi umožnilo efektivně sledovat a spravovat změny kódu v průběhu projektu. Možnost vytvářet větve v systému GitLab mi poskytla svobodu experimentovat s novými funkcemi a opravami bez obav, že bych zničil základní kód projektu.

Repozitář se zdrojovými kódy serverové části je k dispozici zde: `https://gitlab.fel.cvut.cz/shevcdmi/autoservice_is`

Repozitář se zdrojovými kódy klientské části je k dispozici zde: `https://gitlab.fel.cvut.cz/shevcdmi/autoservice_is_fe`

6.5 Souhrn

V této kapitole byla podrobně popsána implementace vyvinutého systému. Byly popsány klíčové aspekty procesu implementace, včetně výběru vhodných technologií a metodik a struktury jednotlivých částí aplikace. V následujících kapitolách bude na základě výsledků této kapitoly pojednáno o testování aplikace, popisu její funkčnosti a námětech na její vylepšení.

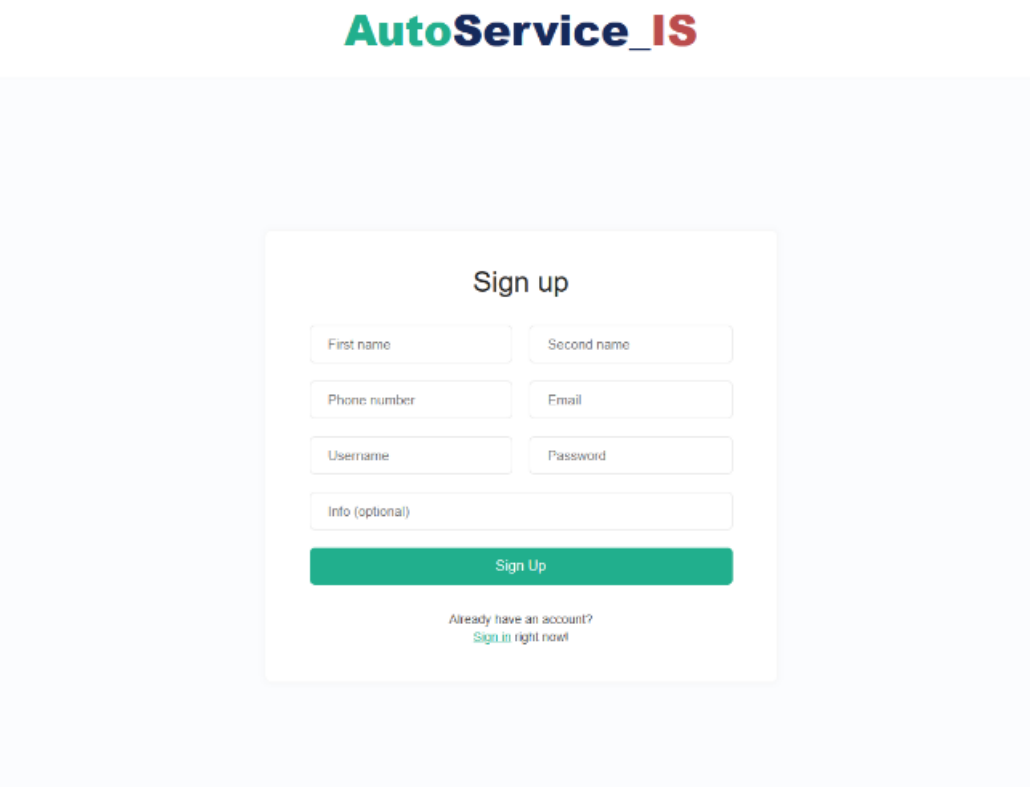
Kapitola 7

Popis aplikace

V této kapitole se zaměříme na popis funkčnosti již implementovaného prototypu aplikace a na její schopnosti. Prototyp aplikace slouží k demonstraci základních funkcí a možností aplikace. Níže naleznete několik scénářů použití aplikace

7.1 Registrace

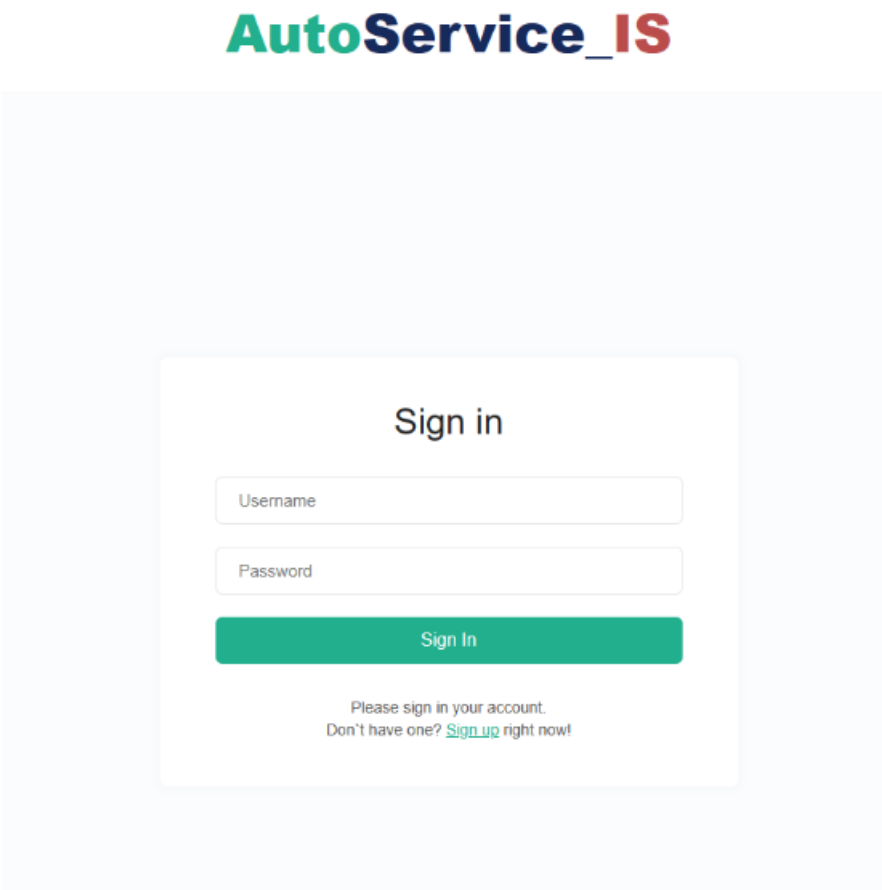
Na stránce pro registraci zákazníka bude uživatel vyzván k vyplnění několika povinných polí, kterými jsou jméno, příjmení, e-mailová adresa a heslo atd. Pro zaměstnance bude k registraci určen samostatný odkaz, který je přesměruje na registrační stránku určenou speciálně pro ně.



Obrázek 7.1: Stránka pro registraci zákazníků.

7.2 Přihlášení

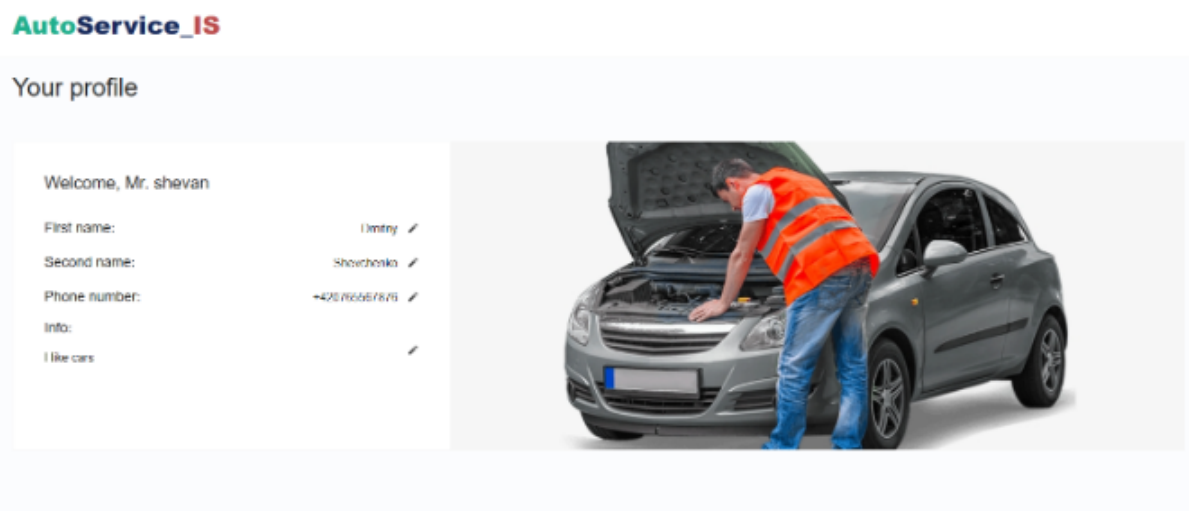
Autorizace v aplikaci je společná pro všechny uživatele a provádí se přímo v systému. Poté, co uživatel projde procesem registrace, bude přesměrován na přihlašovací stránku, kde bude vyzván k zadání uživatelského jména a hesla pro přístup ke svému účtu. To zajišťuje bezpečný přístup do systému a umožňuje každému uživateli individuální zkušenost s používáním aplikace.



Obrázek 7.2: Přihlašovací stránka.

7.3 Profil uživatele

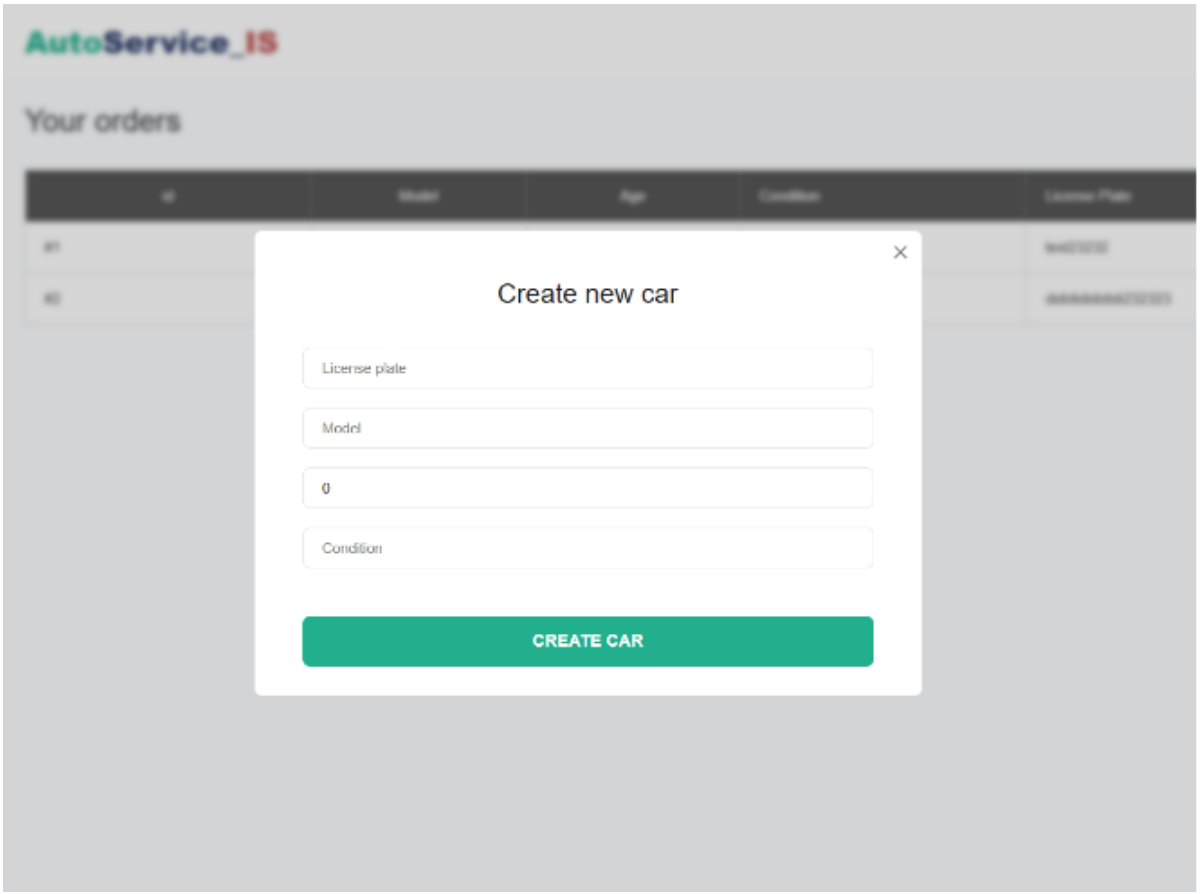
Uživatelský profil obsahuje základní informace o uživateli, včetně jeho uživatelského jména, jména a příjmení, telefonního čísla a dalších údajů, které závisí na typu uživatele. V případě zákazníka mohou informace o něm zahrnovat údaje týkající se jeho preferencí, kontaktních údajů, adresy a dalších relevantních informací, které pomáhají zlepšit jeho zkušenosti s používáním aplikace. V případě zaměstnance mohou informace o jeho profilu obsahovat jeho specializaci a další profesní údaje, které mohou být důležité pro poskytování služeb a zajištění kvality.



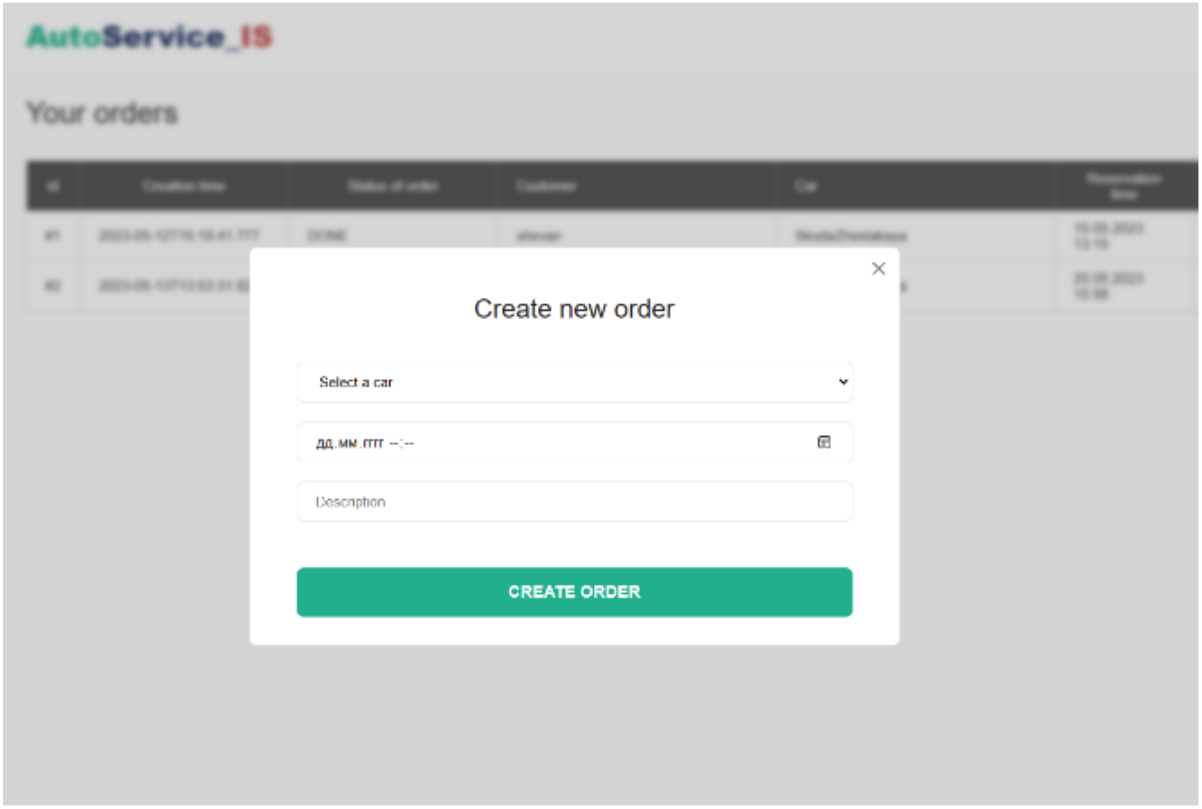
Obrázek 7.3: Profil uživatele.

7.4 Zákazníci

Zákazníci budou moci vytvářet požadavky na opravu svého vozidla tak, že jej nejprve přidají do systému. Následně budou moci zadat preferované datum a čas, kdy jim vyhovuje do servisu přijet. Vytvořená objednávka a informace o vozidle budou viditelné v příslušných záložkách aplikace. Po vytvoření objednávky budou moci zákazníci sledovat její stav.



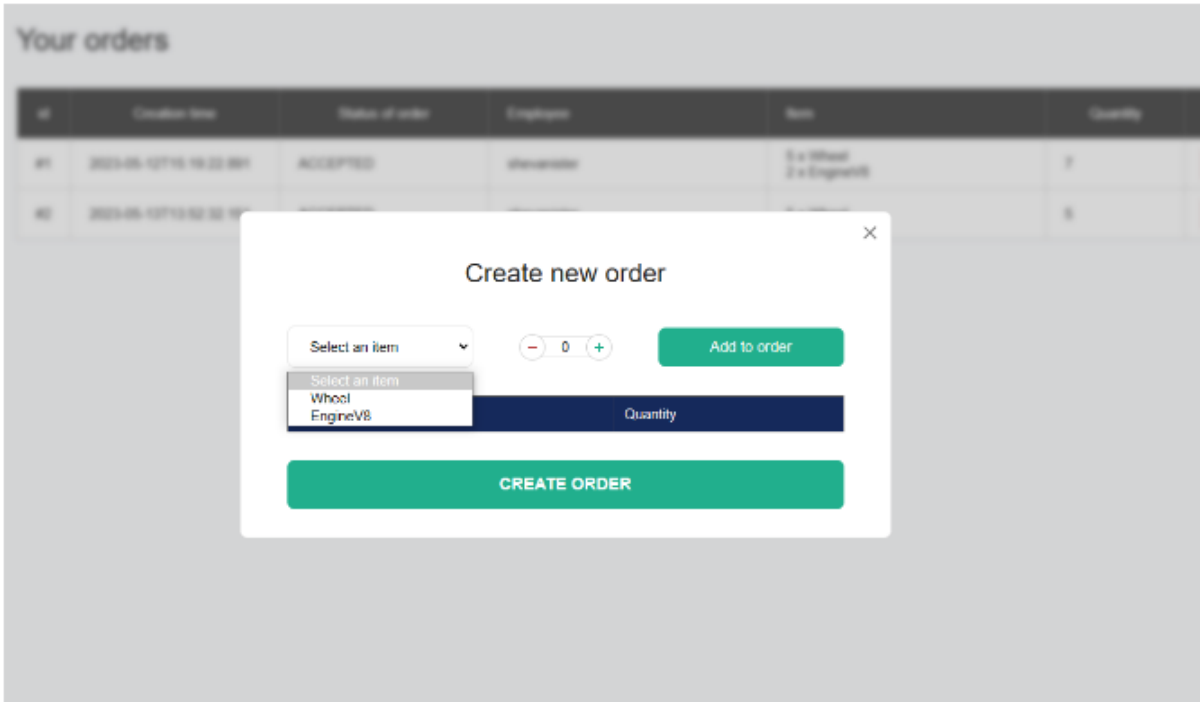
Obrázek 7.4: Vytvoření nového vozu.



Obrázek 7.5: Vytvoření nové objednávky zákazníka.

7.5 Zaměstnanci

Zaměstnanci mohou v systému vytvářet objednávky náhradních dílů pro vozidla. Tyto objednávky jsou určeny k doplnění zásob nebo k opravám.

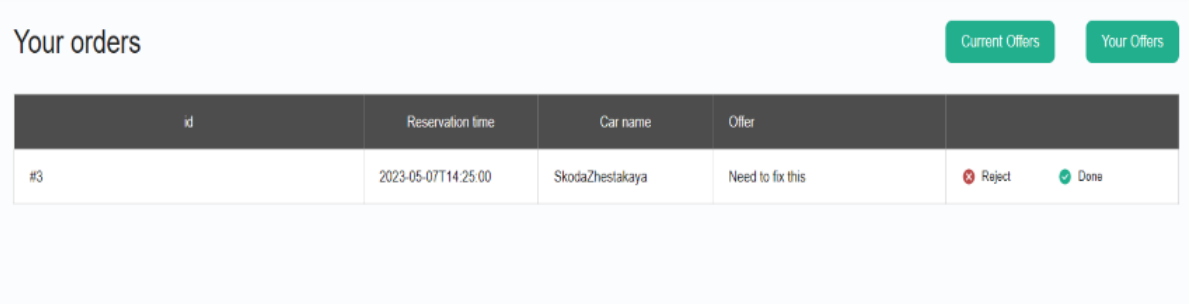


Obrázek 7.6: Vytvoření nové objednávky zaměstnance.

Ve společném fondu zákaznických zakázek mohou zaměstnanci procházet dostupné zakázky a vybírat ty, které jim nejvíce vyhovují nebo na které se specializují. Po výběru zakázky se pracovník stává odpovědným za její vyřízení. V tomto okamžiku je zakázka považována za „převzatou“ pracovníkem.

Obrázek 7.7: Fond objednávek zákazníků.

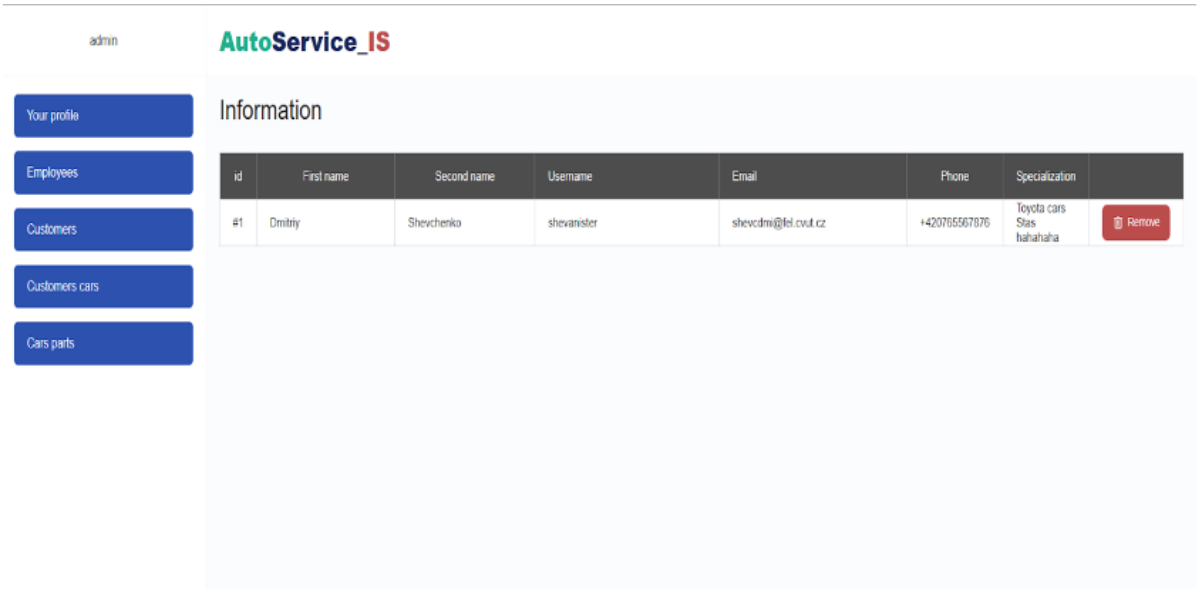
Pracovník má možnost zakázku dokončit, nebo zrušit v závislosti na své dostupnosti, dovednostech a zdrojích. Pokud je zakázka dokončena, je takto označena i v systému.



Obrázek 7.8: Objednávky prováděné pracovníkem.

7.6 Správce

Správce systému má oprávnění k zobrazení seznamů všech zaměstnanců, zákazníků a informací o jejich vozidlech. Může získat přehled o všech registrovaných uživatelích a jejich údajích. Prohlížení seznamu zaměstnanců umožňuje správci zobrazit informace o každém zaměstnanci v rozsahu jméno, kontaktní údaje a specializace. Podobně může správce zobrazit seznam zákazníků a informace o jejich vozidlech. To zahrnuje údaje o zákaznících, jako jsou jména, kontaktní údaje a další údaje a také informace o registrovaných vozidlech, jako například model či registrační značka. Tato funkcionalita správci pomáhá sledovat databázi zákazníků a vozidel.



Obrázek 7.9: Seznam uživatelů.

Správce může do systému zadávat také díly vozidel. To správci umožňuje spravovat seznam dostupných náhradních dílů včetně informací o nich, jako je název, cena a dostupnost. Všechny tyto údaje se pak zobrazí v databázi.

Kapitola 8
Testovaní

V následující kapitole se podrobněji zabývám procesem testování aplikace, který byl klíčovou součástí mé bakalářské práce. Testování je pro ověření funkčnosti a použitelnosti aplikace nezbytné a je považováno za jednu z nejdůležitějších fází vývoje softwaru. Cílem testování bylo zajistit, že aplikace splňuje všechny stanovené požadavky a že je bez chyb, které by mohly omezit její výkon nebo použitelnost. V této kapitole popisuji různé metody a techniky testování, které jsem použil, včetně jednotkových testů či integračních testů. Také se věnuji výsledkům těchto testů a diskutuji o jejich dopadech na konečný produkt.

8.1 Testování serverové části aplikace

Tato podkapitola je věnována testování serverové části aplikace. Pro zajištění robustnosti a spolehlivosti našeho systému jsme implementovali jak jednotkové (unit), tak integrační testy. Jednotkové testy jsou zaměřeny na ověření funkčnosti jednotlivých komponent naší aplikace, zatímco integrační testy kontrolují, jak tyto komponenty společně pracují.

8.1.1 JUnit

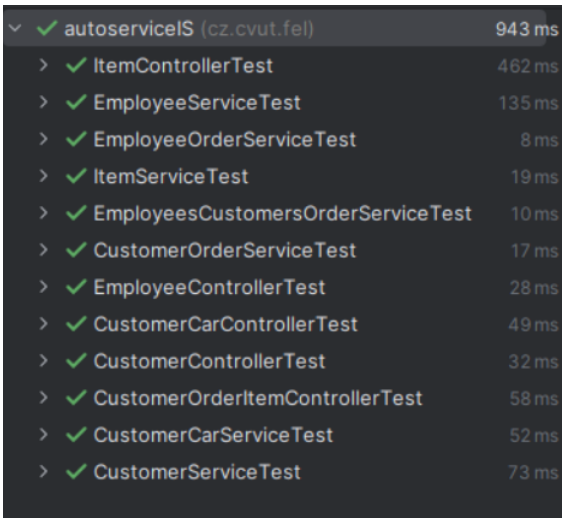
JUnit je open-source testovací framework, který se používá pro psaní a spouštění opakovatelných testů v jazyce Java. Poskytuje sadu anotací, které umožňují definovat testy a testové sady. Také poskytuje aserci pro kontrolu očekávaných výsledků. JUnit je ideální pro jednotkové testy, což jsou testy, které ověřují funkčnost jednotlivých částí kódu, jako jsou metody nebo třídy, nezávisle na ostatních částech systému.

8.1.2 Mockito

Pro napsání těchto testů jsem použil nástroj nazývaný Mockito. Mockito je populární knihovna pro jazyk Java, která umožňuje vytvářet a používat tzv. mock objekty. Mock objekty jsou v podstatě falešné objekty, které napodobují chování skutečných objektů v kontrolovaném prostředí [29]. Používají se v unit testech, kde je potřeba izolovat kód, který je testován, od ostatních částí

systému. Mockito tedy umožňuje vytvářet simulované prostředí, ve kterém je možné kontrolovat, jak se testovaný kód chová v reakci na různé podmínky a vstupy.

Pomocí výše popsaných technik byly implementovány unit testy a integrační testy. Tyto testy odhalily řadu chyb, které byly okamžitě opraveny. Výsledkem je, že testy v aplikaci vypadají následovně:



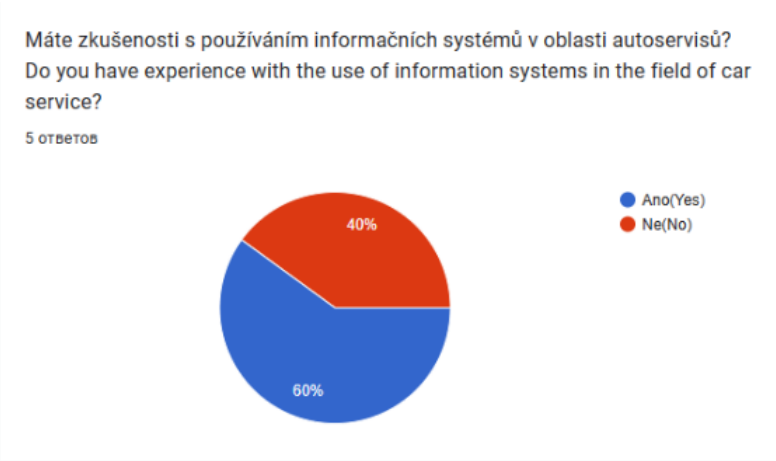
Obrázek 8.1: Testy v aplikaci.

8.2 Výzkumná část a testování klientské části

Byl vytvořen dotazník pro sběr zpětné vazby k uživatelskému rozhraní, díky kterému byly shromážděny údaje o dojmech jednotlivých dotazovaných uživatelů z aplikace a o jejich celkové uživatelské zkušenosti. Odkaz na dotazník: `https://forms.gle/Xr3Y9f4H63kXDr7aA`

Příklad otázek v dotazníku:

1. Využívali jste někdy služby autoservisu?
2. Máte zkušenosti s používáním informačních systémů v oblasti autoservisů?
3. Líbí se vám design uživatelského rozhraní?
4. Je pro vás design aplikace intuitivní, tedy pokud jste schopni jej pochopit bez cizí pomoci?
5. Pokud něčemu nerozumíte, můžete prosím popsat, čemu přesně?
6. Myslíte si, že je funkčnost aplikace dostatečná?
7. Pokud je vaše odpověď záporná, popište, co podle vás chybí?



Obrázek 8.2: Příklad otázky z dotazníku.

Dotazník vyplnilo pět lidí. Podle výsledků dotazníku využilo 60 % respondentů služeb autoservisů a pracovalo s jejich informačními systémy. Respondentům se design aplikace spíše líbil než nelíbil, v tomto bodě nebyla žádná zvláštní zpětná vazba. Během uživatelského testování se také ukázalo, že některým respondentům není zcela jasné, jak objednávku vytvořit a která pole jsou povinná. Dále byly zkoumány odpovědi týkající se přehlednosti rozhraní a byla získána zpětná vazba, konkrétně následující odpovědi:

1. V části Vaše objednávky je sloupec pro položku a není jasné, jaká položka tam bude, protože v objednávce je mnoho položek.

Nejvíce zpětné vazby jsem dostal k funkčnosti aplikace, hlavní body jsem z odpovědí vyzdvihl:

1. Aplikace by se mohla stát platformou pro další autoopravárenské služby, což by mohlo umožnit škálování projektu.
2. Chybějící funkce pro kontrolu položek ve skladu.
3. Bylo by dobré dostávat e-mailové oznámení o aktualizaci stavu objednávky.

Kapitola 9

Budoucí stav aplikace

Výsledkem je, že vyvinutá aplikace úspěšně splňuje všechny funkční i nefunkční požadavky zadané na začátku procesu vývoje. Je však třeba poznamenat, že tato aplikace je pouze prototypem a možnosti dalšího zlepšování a vývoje jsou vždy otevřené.

Podle mého názoru by bylo možné zlepšit/opravit/doplnit následující aspekty:

- Vytvoření tabulky časových přídělů pro zaměstnance.
- Přidání chybějících funkcí, např. změna stavu objednávky náhradních dílů ze skladu.
- Zlepšení systému registrace zaměstnanců autoservisů. Bylo by možné vytvořit jedinečné odkazy, které by okamžitě generovaly jedinečné ID uživatele.
- Multijazyčnost, konkrétně přítomnost češtiny v systému.
- Přidání funkcí, které z různých důvodů chybí a obecně rozšíření funkcí správce.
- Použití nástroje Docker jako hostitele aplikace.

Zlepšení s ohledem na dotazník zpětné vazby v kapitole č.8.2:

- Integrace zasílání oznámení na e-maily uživatelů.
- Vylepšení uživatelského rozhraní, například změna písma nebo zviditelnění aktuální karty, na které se uživatel právě nachází.

Kapitola 10

Závěr

V rámci bakalářské práce byl proveden podrobný výzkum a analýza existujících řešení a technologií, což umožnilo vytvoření detailních diagramů tříd, případů užití a nasazení a také definování funkčních a nefunkčních požadavků. Tato analýza byla nezbytná pro hluboké pochopení požadavků a omezení projektu, což následně umožnilo určení nejvhodnějších technologií a návrhových vzorů pro jeho realizaci. Výsledkem bakalářské práce je návrh a implementace informačního systému pro autoservisy. Systém byl otestován, aby byla zajištěna jeho správná funkčnost a použitelnost.

Seznam použitých zkratek

ACID Atomicity, Consistency, Isolation, and Durability.

API Application Programming Interface.

CSS Cascading Style Sheets.

DOM Document Object Model.

DTO Data Transfer Object.

HTML The HyperText Markup Language.

HTTP HyperText Transfer Protocol.

IDE Integrated development environment.

JPA Java Persistence API.

JSON JavaScript Object Notation.

JWT JSON Web Token.

MVC Model View Controller.

NoSQL Not only SQL.

ORM Object Relational Mapping.

REST Representational State Transfer.

SPA Single Page Application.

URL Uniform Resource Locator.

XML Extensible Markup Language.

Literatura

[1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach , ISBN-10: 9780078022128

[2] Andrew Chesterton *How many cars are there in the world?*, [online]. [Cit. 15.04.2023] Dostupné z `https://www.carsguide.com.au/car-advice/how-many-cars-are-there-in-the-world-70629`

[3] Carsys *Software pro prodejce a servisy automobilů*, [online]. [Cit 15.04.2023] Dostupné z `https://www.carsys.cz/carsystem/`

[4] AutoFenix *Jednoduchý systém pro Váš autoservis*, [online]. [Cit 16.04.2023] Dostupné z `https://autofenix.cz/`

[5] Mechanic, [online]. [Cit 16.04.2023] Dostupné z `https://www.nextis.cz/mechanic/`

[6] ShopMonkey *The Auto Repair Software*, [online]. [Cit 16.04.2023] Dostupné z `https://www.shopmonkey.io/auto-repair-software`

[7] Ian Sommerville *Software Engineering*, [Cit 25.04.2023] Dostupné z `https://mycourses.aalto.fi/pluginfile.php/1177979/mod_resource/content/1/Sommerville-Software-Engineering-10ed.pdf`

[8] Oracle *What Is a Database?*, [online]. [Cit 27.04.2023] Dostupné z `https://www.oracle.com/database/what-is-database/`

[9] Sergio Darias Perez *WHAT IS MICROSOFT SQL SERVER AND WHAT IS IT FOR?*, [online]. [Cit 29.04.2023] Dostupné z `https://intelequia.com/en/blog/post/what-is-microsoft-sql-server-and-what-is-it-for`

[10] *What Is Oracle Database*, [online]. [Cit 19.04.2023] Dostupné z `https://www.oracletutorial.com/getting-started/what-is-oracle-database/`

[11] *What is Oracle?*, [online]. [Cit 19.04.2023] Dostupné z `https://www.javatpoint.com/what-is-oracle`

[12] Oracle *What is MySQL?*, [online]. [Cit 27.04.2023] Dostupné z https://www.oracle.com/mysql/what-is-mysql/

[13] PostgreSQL *What Is PostgreSQL?*, [online]. [Cit 28.04.2023] Dostupné z https://www.postgresql.org/docs/15/intro-whatis.html

[14] IBM *What is MongoDB?*, [online]. [Cit 28.04.2023] Dostupné z https://www.ibm.com/topics/mongodb

[15] Microsoft *A tour of the C# language*, [online]. [Cit 29.04.2023] Dostupné z https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[16] Python, [online]. [Cit 29.04.2023] Dostupné z https://docs.python.org/3/tutorial/index.html

[17] *Django introduction*, [online]. [Cit 30.04.2023] Dostupné z https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[18] Microsoft *What is ASP.NET?*, [online]. [Cit 30.04.2023] Dostupné z https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet

[19] *What is Java?*, [online]. [Cit 30.04.2023] Dostupné z https://www.java.com/en/download/help/whatis_java.html

[20] *Introduction to client-side frameworks*, [online]. [Cit 30.04.2023] Dostupné z https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction

[21] Matthew Tyson *What is JPA?*, [online]. [Cit 02.05.2023] Dostupné z https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

[22] Martin Mois *Hibernate Tutorial The Ultimate Guide*, [Cit 03.05.2023] Dostupné z https://enos.itcollege.ee/~jpoial/allalaadimised/reading/Hibernate-Tutorial.pdf

[23] Apache Maven Project, [online]. [Cit 06.05.2023] Dostupné z https://maven.apache.org/what-is-maven.html

[24] Spring Boot, [online]. [Cit 07.05.2023] Dostupné z https://spring.io/projects/spring-boot

[25] Craig Walls *Spring Boot in action*, [Cit 07.05.2023] Dostupné z https://doc.lagout.org/programmation/Spring%20Boot%20in%20Action.pdf

[26] Apache Tomcat, [online]. [Cit 10.05.2023] Dostupné z https://tomcat.apache.org/

[27] Chinmayee Deshpande *The Best Guide to Know What Is React*, [online]. [Cit 12.05.2023] Dostupné z https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

[28] Jan Faigl *Úvod do verzovacích systémů(informativní)*, [online]. [Cit 14.05.2023] Dostupné z https://cw.fel.cvut.cz/old/_media/courses/a0b36pr2/lectures/lecture11-slides.pdf

[29] Mockito, [online]. [Cit 20.05.2023] Dostupné z https://github.com/mockito/mockito/wiki/FAQ

[30] Spring Security Filter Chain, [online]. [Cit 18.05.2023] Dostupné z https://docs.spring.io/spring-security/reference/servlet/architecture.html#servlet-securityfilterchain

[31] *JWT tokens and security – working principles and use cases*, [online]. [Cit 18.05.2023] Dostupné z https://www.vaadata.com/blog/jwt-tokens-and-security-working-principles-and-use-cases/

[32] David Přáda *Informační systém pro správu autoservisu* Dostupné z https://dspace.cvut.cz/bitstream/handle/10467/94654/F3-BP-2021-Prada-David-Informacni%20system%20pro%20spravu%20autoservisu.pdf

[33] Reshma Ahmed *What Is Git ? – Explore A Distributed Version Control Tool*, [online]. [Cit 20.05.2023] Dostupné z https://www.edureka.co/blog/what-is-git/