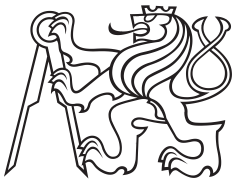
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

IS pro správu půjčovny

Jan Cháb

Vedoucí: Ing. Božena Mannová, Ph.D.
Studijní program: Softwarové inženýrství a technologie
Květen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Cháb** Jméno: **Jan** Osobní číslo: **492205**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

IS pro správu půjčovny

Název bakalářské práce anglicky:

Rental office management IS

Pokyny pro vypracování:

Seznamte se s problematikou provozu půjčovny a dostupnými aplikacemi pro správu půjčovny.
Na základě provedené analýzy specifikujte základní požadavky na funkcionality systému. Systém bude sloužit jak k provozu půjčovny, tak bude přístupný i zákazníkům, kterým usnadní vytváření objednávek a jejich přehled a historii. Systém nebude provozovat finanční agendu. Navrhněte případy užití a architekturu systému.
Seznamte se s technologiemi potřebnými pro vytvoření aplikace. Porovnejte tyto technologie a vyberte ty, které mohou být pro aplikaci použity. Svá rozhodnutí zdůvodněte.
Vhodně zvolenými nástroji aplikaci implementujte.
Otestujte aplikaci včetně uživatelských testů a výsledky vyhodnoťte.
Při zpracování využívejte prostředky Softwarového inženýrství.

Seznam doporučené literatury:

[1] Roger S. Pressmann Bruce Maxim:Software Engineering: A Practitioner's Approach , ISBN-10: 9780078022128
[2] Functional vs non-functional requirements [updated 2021], Enkonix.Com,https://enkonix.com/blog/functional-requirements-vs-non-functional/ , 2022 [3] Layered architecture, Cs.Uwaterloo.Ca, https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Layered.pdf , 2022.

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: **15.09.2022** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **19.02.2024**

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 vedoucí mé práce, Ing. Boženě Mannové, Ph.D, za cenné rady, věcné připomínky a vstřícnost při konzultacích a při vypracování této práce.

Také bych chtěl poděkovat mé rodině a přátelům, kteří mě po celou dobu studia podporovali.

Prohlášení

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

V Praze, 26. května 2023

.....

Jan Cháb

Abstrakt

Tato práce se zabývá analýzou, návrhem a implementací informačního systému pro správu půjčovny.

Půjčovna pomocí tohoto softwaru bude moci sledovat veškeré objednávky, stav a vytížení vybavení a plánovat své pracovní směny. Zákazníkům půjčovny bude díky systému k dispozici jednoduchý a přehledný nástroj pro objednávání půjčovnou poskytovaného vybavení.

Během práce jsem využíval znalosti a dovednosti týkající se vývoje softwaru nabyté během studia na ČVUT v Praze. Některé znalosti byly doplněny samostudiem.

Klíčová slova: IS půjčovny, analýza a návrh IS, vývoj softwaru, PostgreSQL, Java, Spring

Vedoucí: Ing. Božena Mannová, Ph.D.

Abstract

This thesis deals with the analysis, design and implementation of an information system for rental management.

Using this software, the rental company will be able to monitor all orders, equipment status and utilization, and schedule its work shifts. Thanks to the system, the rental customers will have at their disposal a simple and clear tool for ordering the equipment provided by the rental company.

During my work, I used the knowledge and skills related to software development gained during my studies at CTU in Prague. Some knowledge was supplemented by self-study.

Keywords: Rental office management IS, analysis and design of IS, software development, PostgreSQL, Java, Spring

Title translation: Rental office management IS

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Tato práce se zabývá analýzou, návrhem a implementací uživatelsky přívětivého systému pro usnadnění provozu půjčovny. Systém bude sloužit jak k vnitřnímu provozu půjčovny, tak bude i přístupný navenek zákazníkům, kterým usnadní vytváření objednávek a jejich přehled a historii. Finance jsou mimo rozsah této práce.

V práci popisuji, po rozhovoru s majitelkou půjčovny lodí, fungování její půjčovny. Dále analyzuji existující řešení pro systémy půjčovny a specifikuji požadavky na systém. Poté navrhuji případy užití, architekturu systému a technologie, které využiji při implementaci řešení. Na konec návrhu jsou přiloženy doménový model a diagram stavů objednávky.

V poslední kapitole popisuji implementaci a testování systému.

1.1 Motivace

Při mé letní brigádě v půjčovně lodí se používal informační systém dost omezeně, prakticky šlo pouze o databázi objednávek. Chtěl bych tedy navrhnout řešení, které by sjednotilo správu objednávek, zákazníků a také umožnilo půjčovně plánování směn, které je dodnes děláno pomocí vybarvování buněk v Google Tabulkách.

1.2 Cíl práce

Cílem této bakalářeké práce je analyzovat, navrhnout, implementovat a otestovat informační systém pro správu půjčovny, který budou moci využívat jak zaměstnanci půjčovny, tak její zákazníci. Systém bude umožňovat správu objednávek, půjčovaného vybavení, zákazníků a zaměstnanců. Součástí systému bude také plánování směn zaměstnanců půjčovny.

Část I

Analýza

Kapitola 2

Procesy půjčovny

Za účelem podrobné analýzy jsem se sešel s majitelkou půjčovny lodí, která operuje na řece Sázavě a kde jsem v minulosti několik let pracoval v rámci letních brigád. Prodiskutovali jsme fungování její půjčovny a její případné požadavky, které by na informační systém měla.

2.1 Hlavní proces půjčovny lodí

Hlavní proces se skládá ze 3 částí:

- **vytvoření objednávky**
- vyzvednutí vybavení (**začátek výletu**)
- vrácení vybavení (**konec výletu**)

Vytvoření objednávky
Proces začíná rozhodnutím zákazníka vypůjčit si vybavení. Může tedy sjednat objednávku pár dní dopředu dvěma způsoby: zavolá na telefonické centrum půjčovny, kde s operátorem objednávku vytvoří, nebo vyplní formulář na webových stránkách půjčovny. Také může ale na jednu z poboček přijít přímo v den výletu, kde mu vybavení, které na ten den není zamluvené, půjčí.

Začátek výletu
V den výletu zákazník v dopoledních hodinách přijde na pobočku půjčovny přebrat objednané vybavení a podepisuje smlouvu o výletu. Objednané vybavení samozřejmě musí být pracovníky půjčovny na odpovídající pobočce připraveno, nejčastěji, pokud se nejedná o pobočku fungující také jako sklad vybavení, je sem před začátkem výdejních hodin dovezeno autem ze skladu vybavení.

Konec výletu
Výlet končí příjezdem zákazníka na místo určené k vrácení vybavení, tedy nejčastěji některou z poboček dále po proudu řeky. Toto odevzdání půjčeného vybavení probíhá v odpoledních hodinách, kdy na místě čeká pracovník půjčovny, vybavení přebírá a objednávky označuje jako ukončené.

Později odpoledne je veškeré vybavení svezeno na jednu ze základen půjčovny, které působí také jako sklady vybavení. Poté je vybavení připraveno pro objednávky na další den.

2.2 Další procesy a požadavky na systém

Vedle hlavního procesu – zajištění výletu pro zákazníka (a na něj napojených funkcionalit systému) – jsme probrali i další procesy půjčovny, jejichž fungování může informační systém znatelně usnadnit.

2.2.1 Inventura vybavení

Paní majitelka by chtěla, aby měl zaměstnanec možnost kdykoliv zobrazit soupis veškerého vybavení, které se na libovolné pobočce nachází, a kolik vybavení tam v daný den musí být připraveno k vydání objednaným zákazníkům. Tak budou moci zaměstnanci potřebné vybavení připravit a na místo dovézt z jiné pobočky, kde není na další den potřeba. Přesun vybavení by se pak jen zadal v systému.

2.2.2 Vyřazení vybavení kvůli údržbě

Vybavení často není nejmladší a ani není nerozbitné – byla tedy také vyjádřena potřeba mít možnost vyřadit vybavení z provozu, ať už trvale, nebo na určitý počet dní během nichž se provede potřebná údržba.

2.2.3 Plánování směn

Půjčovna má minimum stálých zaměstnanců a v letní sezóně spoléhá na velký počet brigádníků, kteří ale nejsou vždy k dispozici. Dnes je plánování směn děláno pomocí Google Tabulek, kam brigádník každý měsíc označuje dny, kdy pracovat může a kdy ne. Pověřený seniorní zaměstnanec pak s předstihem v tabulce označí, kdo v určité dny do práce půjde - musí při tomto plánování vzít v potaz odlišné schopnosti a dovednosti brigádníků (např. vlastnictví řidičského průkazu, schopnost opravy rozbitých lodí, atd.) a také předpokládanou vytíženost půjčovny.

Plánování směn by se díky systému, kde by například ihned bylo vidět role zaměstnanců podle dovedností, mohlo velice usnadnit.

Také by v případě nedostatku přihlášených zaměstnanců v daný den mohl být vypsán požadavek na určitou roli a uživatele s touto rolí by systém upozornil, že tento typ zaměstnance na směně zatím chybí a mohli by se k ní přihlásit.

Kapitola 3

Existující řešení

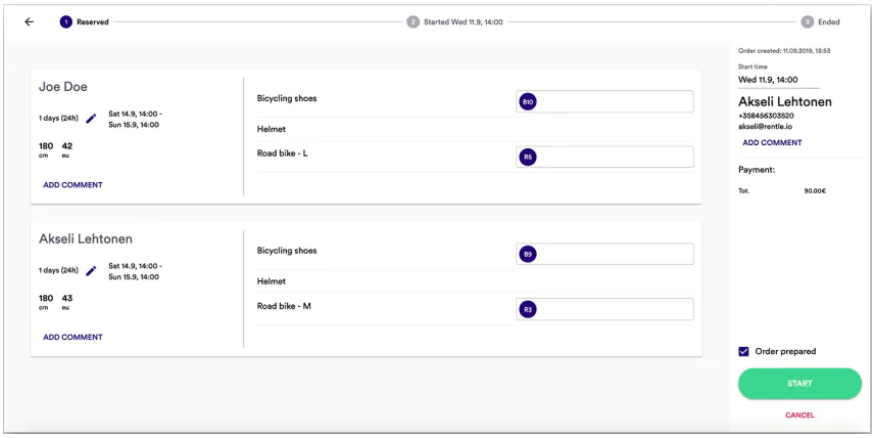
V této kapitole jsou popsány některé z nejčastěji využívaných informačních systémů pro půjčovny vybavení podle serveru Capterra [cap23] a jejich porovnání.

3.1 Popis existujících řešení

3.1.1 Rentle

Řešení společnosti Rentle poskytují nástroje a podporu pro zahájení, správu a růst podnikání v oblasti pronájmu a prodeje a velmi jednoduše se napojí na již existující webové stránky provozovatele. Řešení cílí právě na malé podniky a jednotlivce. [ren23]

Řešení sice nepřichází s vlastní mobilní aplikací, ale webová aplikace je pro mobilní uživatele uzpůsobena. Na to, že cílí na malé podniky, mně osobně přijde, s cenovkou $39/měsíc, lehce dražší.

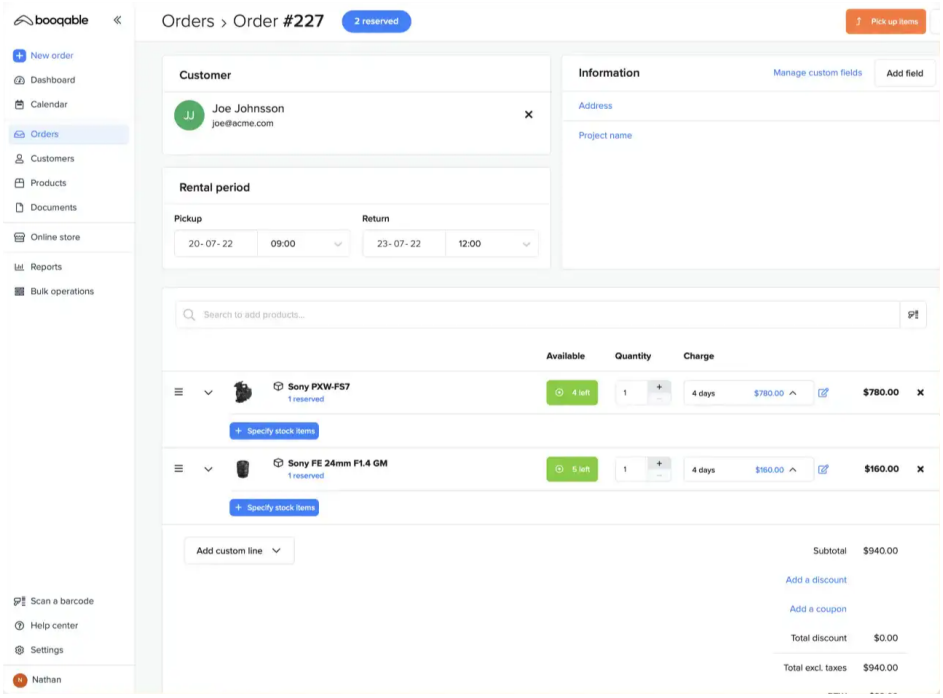


Obrázek 3.1: Existující řešení, Rentle, zdroj: https://www.rentle.io/

3.1.2 Booqable

Booqable nabízí řešení pro malé a střední půjčovny. Mezi nejčastější typy půjčoven využívající tento software patří půjčovny kol, fotovybavení, vybavení a dekorace na akce, nářadí, outdoorové vybavení, lyže, lodě a přívěsy.

Mezi hlavní funkce systému, mimo tvoření objednávek, patří také seznam a monitorování veškerého inventáře, kalendář s rozvrhem a databáze zákazníků společně s CRM. Za zmínku také stojí správa dokumentů. Řešení přichází s iOS aplikaci. [boo23]



Obrázek 3.2: Existující řešení, Booqable, zdroj: https://booqable.com/

3.1.3 Asset Panda

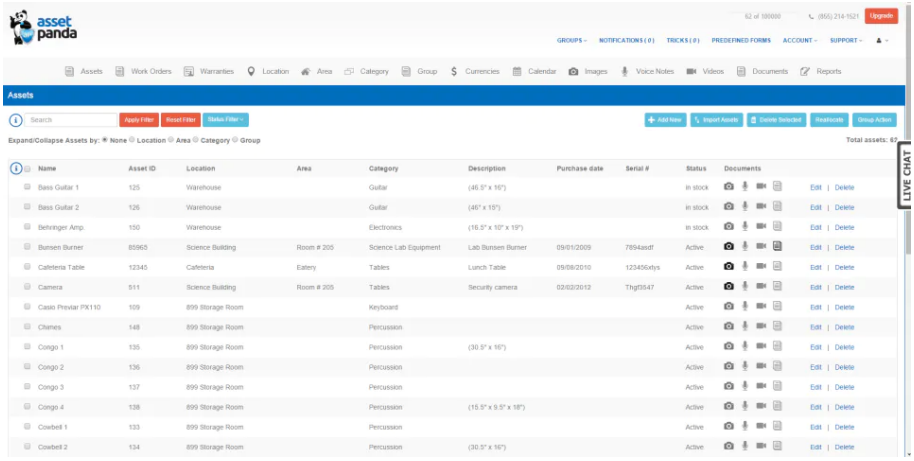
Asset Panda *(obrázek 3.3)* nabízí jednoduché a přehledné řešení pro správu a tracking veškerého vybavení a management objednávek. Není však přístupný navenek samotným zákazníkům, kteří tedy sami objednávky vytvářet nemohou. [ass23]

Cena řešení závisí na množství spravovaného vybavení.

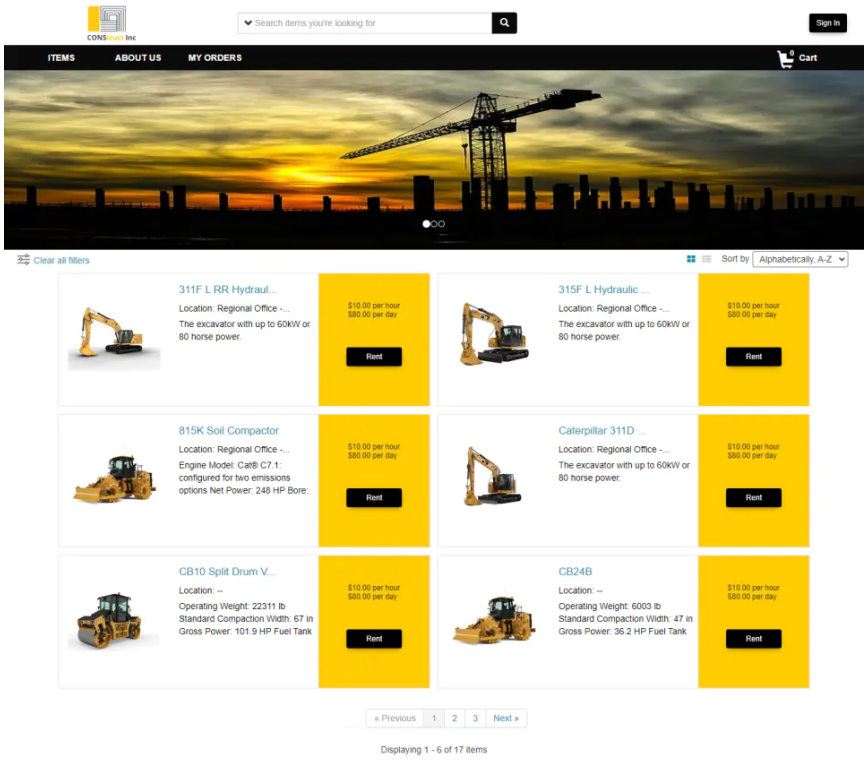
3.1.4 EZRentOut

EZRentOut *(obrázek 3.4)* se pyšní kvalitním sledováním stavu každého kusu půjčeného vybavení. U každého kusu také sleduje uptime pro maximalizaci ROI a může plánovat údržbu podle využívání vybavení. Také nabízí reporting výkonu zaměstnanců. [ezr]

Pro uživatele nabízí přehledný katalog vybavení, možnost zakládat objednávky a nastavovat jim pravidelné opakování.



Obrázek 3.3: Existující řešení, Asset Panda, zdroj: https://www.assetpanda.com/



Obrázek 3.4: Existující řešení, EZRentOut, zdroj: https://www.ezrentout.com/

3.2 Porovnání existujících řešení

Podle informací na samotných stránkách produktů a jejich hodnocení na serveru Capterra [cap23] bylo možné sestavit tabulku porovnávající řešení v plnění některých ze základních funkcionalit požadovaných po systému pro půjčovnu vybavení *(obrázek 3.5)*.

Všechny zmíněné aplikace jsou webové, tedy je lze používat bez nutnosti instalace a konfigurace. Znatelnou nevýhodou produktu *Asset Panda* v tomto projektu je, že se jedná "pouze" o správu vybavení a není přístupný pro zákazníky. Lehkou nevýhodou *Booqable* je absence mobilní aplikace pro Android, v případě *Rentle* chybí aplikace i pro iOS. Obě řešení tento nedostatek ale dohánějí webovou aplikací, která na mobilních zařízeních funguje a vypadá dobře. V rámci správy zaměstnanců nabízí detailnější funkce jen *Booqable*, ostatní uvádějí pouze změny rolí uživatelů.

	Tvorba objednávek zákazníky	Management zákazníků	Management objednávek / výpůjček	Historie jednotlivých položek	Management zaměstnanců	Mobilní aplikace
Rentle	✓	✓	✓	✗	✗	✗
Booqable	✓	✓	✓	✓	✓	Pouze iOS
Asset Panda	✗	✗	✓	✓	✗	✓
EZRentOut	✓	✓	✓	✓	✗	✓

Obrázek 3.5: Existující řešení, Porovnání řešení a základních funkcionalit, zdroj: Autor

Kapitola 4
Specifikace požadavků

Na základě analýzy procesů půjčovny, vyjádřených požadavků paní majitelky půjčovny lodí a funkcionalit existujících řešení jsou v následující kapitole specifikovány role v systému a funkční a nefunkční požadavky na systém.

4.1 Role v systému

V systému budeme rozlišovat role přihlášený uživatel (**User, Customer**), který bude výchozí rolí systému pro zákazníka a bude sjednocovat přístup k základním funkcionalitám. Dále zde bude zaměstnanec (**Employee**) a správce půjčovny (**Manager, Admin**). Správce půjčovny bude mít přístup ke všem funkcionalitám systému.

4.2 Funkční požadavky

Funkční požadavky definují, co bude systém uživatelům umožňovat. Požadavky jsou rozděleny podle uživatelských rolí v systému. [fun23] [Kom16]

Nepřihlášený uživatel

FR 001 – Registrace
Jako nepřihlášený uživatel budu mít možnost založit si nový uživatelský účet.
FR 002 – Přihlášení
Jako nepřihlášený uživatel budu mít možnost se přihlásit svými přihlašovacími údaji.

Přihlášený uživatel

FR 003 – Zobrazení seznamu vybavení
Jako přihlášený uživatel budu mít možnost zobrazit seznam vybavení poskytovaného půjčovnou.
FR 004 – Zobrazení detailu vybavení

Jako přihlášený uživatel budu mít možnost zobrazit detail vybavení půjčovny.

FR 005 – Zobrazení seznamu poboček půjčovny
Jako přihlášený uživatel budu mít možnost zobrazit seznam poboček půjčovny.

FR 006 – Zobrazení detailu pobočky
Jako přihlášený uživatel budu mít možnost zobrazit detail pobočky půjčovny.

FR 007 – Zobrazení uživatelského profilu
Jako přihlášený uživatel budu mít možnost zobrazit svůj profil s údaji.

FR 008 – Upravení uživatelského profilu
Jako přihlášený uživatel budu mít možnost upravit své údaje.

FR 009 – Vytvoření nové objednávky
Jako přihlášený uživatel budu mít možnost vytvořit novou objednávku.

FR 010 – Zobrazení seznamu objednávek
Jako přihlášený uživatel budu mít možnost zobrazit seznam všech svých objednávek.

FR 011 – Zobrazení detailu objednávky
Jako přihlášený uživatel budu mít možnost zobrazit detail objednávky.

FR 012 – Upravení vlastní objednávky
Jako přihlášený uživatel budu mít možnost upravit vlastní objednávku.

FR 013 – Storno vlastní objednávky
Jako přihlášený uživatel budu mít možnost stornovat vlastní objednávku.

Zaměstnanec

FR 014 – Zobrazení seznamu uživatelů
Jako zaměstnanec budu mít možnost zobrazit seznam uživatelů.

FR 015 – Zobrazení detailu uživatele
Jako zaměstnanec budu mít možnost zobrazit detail libovolného uživatele.

FR 016 – Upravení uživatele
Jako zaměstnanec budu mít možnost upravit údaje libovolného uživatele.

FR 017 – Zobrazení seznamu objednávek
Jako zaměstnanec budu mít možnost zobrazit seznam objednávek.

FR 018 – Filtrace a vyhledávání objednávek
Jako zaměstnanec budu mít možnost filtrovat a vyhledávat objednávky podle jejich atributů.

FR 019 – Upravení objednávky
Jako zaměstnanec budu mít možnost upravit libovolnou objednávku.

FR 020 – Změna stavu objednávky
Jako zaměstnanec budu mít možnost změnit stav objednávky.

FR 021 – Zobrazení kalendáře směn
Jako zaměstnanec budu mít možnost zobrazit kalendář s přehledem směn.

FR 022 – Přihlášení ke směně
Jako zaměstnanec budu mít možnost přihlásit se ke směně nebo k požadavku vypsanému zaměstnavatelem v kalendáři půjčovny.

FR 023 – Zrušení přihlášení ke směně
Jako zaměstnanec budu mít možnost zrušit své přihlášení ke směně.

Správce půjčovny

FR 024 – Změna role uživatele
Jako správce půjčovny budu mít možnost změnit roli uživatelům.
FR 025 – Vytvoření a přiřazení dovednosti zaměstnanci
Jako správce půjčovny budu mít možnost vytvořit dovednost a přiřadit ji zaměstnanci.
FR 026 – Vytvoření nové směny
Jako správce půjčovny budu mít možnost vytvořit novou směnu v určitý den, ke které se zaměstnanec s danou rolí bude moci přihlásit.
FR 027 – Vytvoření pobočky půjčovny
Jako správce půjčovny budu mít možnost vytvořit novou pobočku půjčovny.
FR 028 – Upravení pobočky
Jako správce půjčovny budu mít možnost upravit údaje pobočky půjčovny.
FR 029 – Vytvoření skladu půjčovny
Jako správce půjčovny budu mít možnost vytvořit nový sklad půjčovny.
FR 030 – Upravení skladu
Jako správce půjčovny budu mít možnost upravit údaje skladu půjčovny.
FR 031 – Vytvoření vybavení
Jako správce půjčovny budu mít možnost vytvořit položku vybavení půjčovny.
FR 032 – Upravení vybavení
Jako správce půjčovny budu mít možnost upravit vybavení a jeho kapacity ve skladech půjčovny.

4.3 Nefunkční požadavky

Nefunkční, neboli raději kvalitativní, požadavky definují očekávané vlastnosti softwaru, ale nesouvisí s funkčním aspektem softwaru.

NFR 01 – Autorizace
Systém povolí přístup k funkcionalitě pouze uživatelům s rolí s odpovídajícími právy.
NFR 02 – Responzivita
Webová aplikace bude responzivní.
NFR 03 – Uživatelská přívětivost
Webová aplikace nebude náročná na používání pro uživatele s minimální technickou zdatností.
NFR 04 – Funkčnost na moderních prohlížečích
Webová aplikace bude funkční na moderních prohlížečích:
- Chrome
- Firefox
- Edge

- Safari
- Opera

NFR 05 – Jazyková lokalizace

Frontend bude prozatím v anglickém jazyce. Při budoucím rozvoji systému je možné rozšíření o českou verzi.

Část II

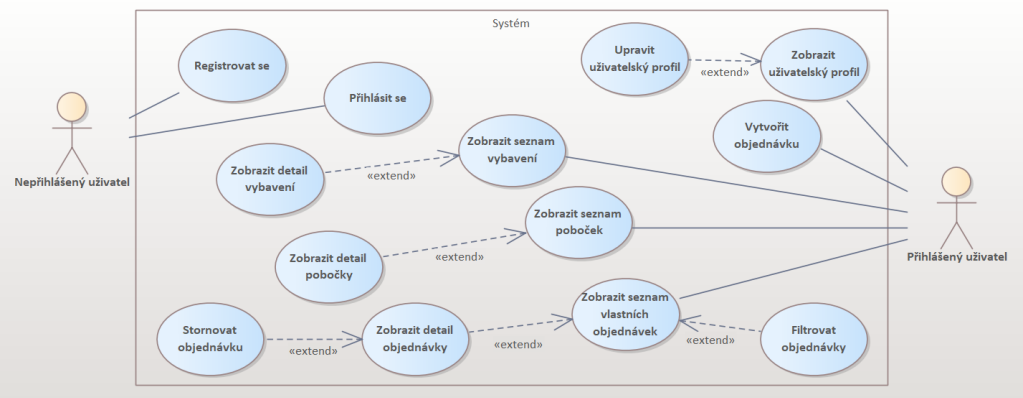
Návrh

Kapitola 5

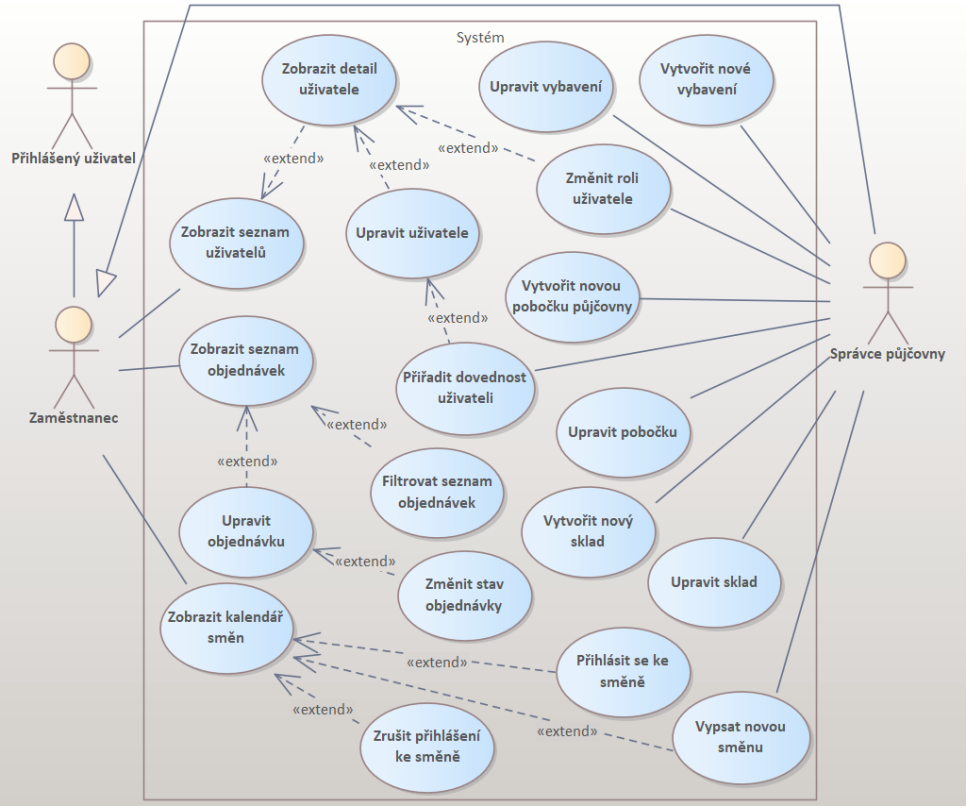
Případy užití

Modelování případů užití je způsob zachycení funkčních požadavků. Model případů užití vyjadřuje, kdo (**aktéři**) bude jakým způsobem (**případy užití**) systém používat a jeho hranice. [Kom19] Na základě dříve specifikovaných funkčních požadavků byl tedy sestaven model případů užití, pro přehlednost byl rozdělen do dvou příloh (*obrázek 5.1, obrázek 5.2*).

5.1 Model případů užití



Obrázek 5.1: Případy užití: Přihlášený a nepřihlášený uživatel, zdroj: Autor



Obrázek 5.2: Případy užití: Zaměstnanec a Správce půjčovny, zdroj: Autor

Kapitola 6
Architektura

Pro architekturu systému jsem vybral architekturu vrstevnatou *(popsána v sekci 6.1)*, převážně z důvodu největší zkušenosti z předmětů Enterprise architektury a Návrh softwarových systémů.

V sekci 6.2 poté krátce popisuji další typy architektur, které byly pro systém uvažovány, z výběru však byly vyřazeny, ať už pro jejich omezenost, nebo naopak přebytečnou komplexitu.

6.1 Vrstevnatá architektura

Vrstevnatá architektura využívá principu Separation of Concerns *(SoC, oddělení zodpovědností)* – spočívá v rozdělení modulů a komponent s podobnou funkcí do horizontálních vrstev. Každá vrstva pak zastává v systému specifickou roli a za účelem jeho propojení poskytuje API *(Application Programming Interface, rozhraní pro programování aplikací)* vrstvě vyšší, která pak přes něj může nižší vrstvu volat. [lay23b] Členění do vrstev také zjednodušuje následnou údržbu a rozvoj systému a rozložení pracovní zátěže při implementaci v týmu.[lay23a]

Navrhovaný systém půjčovny je rozdělen do tří základních vrstev: **prezentační** vrstva (klientská aplikace), **aplikační/business** vrstva (aplikační server) a **perzistentní/databázová** vrstva *(obrázek 6.1)*.

6.1.1 Prezentační vrstva

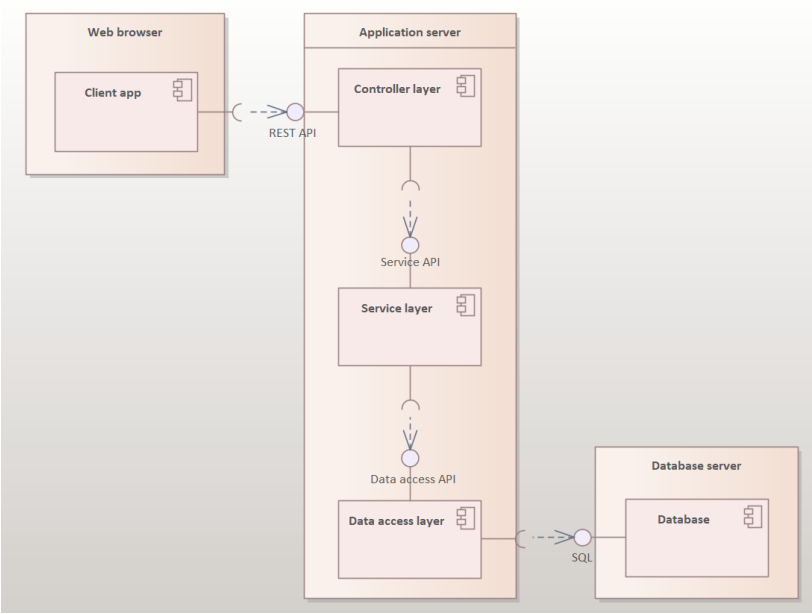
Prezentační vrstvu systému tvoří klientská webová aplikace. Ta komunikuje s aplikačním serverem zasíláním požadavků na REST API poskytované controllery.

6.1.2 Aplikační vrstva

Aplikační server zajišťuje rozhraní pro komunikaci s prezentační vrstvou. Před zpracováním požadavku aplikační server provádí autentizaci a autorizaci, potom požadavky zpracovává a propaguje do dalších vrstev aplikace, aby byly obslouženy.

6.1.3 Perzistentní vrstva

Perzistentní vrstva má na starosti uchovávání dat, tvoří ji tedy databázový server s relační databází, který poskytuje rozhraní pro manipulaci s uloženými daty. S tímto rozhraním pak komunikuje aplikační server skrz DAO.



Obrázek 6.1: Architektura, Vrstevnatá architektura, zdroj: Autor

6.2 Další možné architektury

- **Monolitická architektura**

  V monolitické architektuře jsou všechny součásti systému pevně spojeny do jediné spustitelné nebo nasaditelné jednotky. To znamená, že celý systém je nasazen jako jediný balíček a všechny komponenty sdílejí stejnou codebase (kódovou základnu) a datová úložiště. Tato architektura se sice snadno vyvíjí, ale může být také náročná na škálování a údržbu při růstu systému.[mon23]

- **Architektura klient-server**

  V architektuře klient-server je systém rozdělen, jak název napovídá, na dvě hlavní části: klienta a server. Klient je zodpovědný za prezentaci uživatelského rozhraní uživateli a odesílání uživatelských vstupů na server. Server má na starosti zpracování vstupu, provedení potřebné business logiky a vrácení výsledků klientovi.[cli20]

- **Microservices (Architektura mikroslužeb)**

  V architektuře mikroslužeb je systém rozdělen na malé nezávislé služby, které lze vyvíjet, nasazovat a škálovat samostatně. Každá služba je

zodpovědná za specifickou sadu funkcí a komunikuje s ostatními službami pomocí odlehčených rozhraní API. Každá služba má obvykle vlastní datové úložiště a lze ji horizontálně škálovat přidáváním dalších instancí služby podle potřeby. Ve srovnání s vrstvenou architekturou však může vést k vyšší míře složitosti.[LF14]

Kapitola 7
Technologie

V této kapitole popisuji technologie, které budou využity při implementaci řešení, a důvody jejich výběru.

7.1 PostgreSQL

PostgreSQL je open-source objektově-relační databázový systém, který využívá a rozšiřuje jazyk SQL, a je schopný ukládat i velmi komplikovaná data. Je schopný běžět na všech velkých operačních systémech a je v souladu s ACID vlastnostmi transakcí *(atomicita, konzistence, izolovanost, trvalost)*. [pos23]

7.1.1 Porovnání s MySQL

MySQL je open-source relační databázový systém, který se zaměřuje na jednoduchost a flexibilitu a je jedním z nejpoužívanějších DB systémů.

- PostgreSQL je obecně považován za rychlejší a efektivnější než MySQL pro složité dotazy a velké množství dat.
- PostgreSQL má komplexnější syntaxi jazyka SQL a širší škálu vestavěných funkcí a datových typů, takže je flexibilnější a dokáže zpracovávat širší škálu dat.
- PostgreSQL je znám svými pokročilými bezpečnostními funkcemi, jako je zabezpečení na úrovni řádků, a svým důrazem na integritu dat, což z něj činí bezpečnější volbu pro aplikace, které vyžadují vysokou úroveň zabezpečení.

Kvůli předchozím zkušenostem a uvedeným výhodám oproti MySQL jsem pro systém vybral PostgreSQL.

7.2 Java

Na základě mých zkušeností a dovedností nabytých během studia jsem se rozhodl backend systému implementovat v jazyce Java.

Java byla postavena téměř výhradně jako objektově orientovaný jazyk. Velkou výhodu Javy je její přenositelnost a platformní nezávislost. Java je jazykem interpretovaným, takže místo skutečného strojového kódu se při kompilaci vytváří pouze takzvaný bytecode. Výhodou toho je, že tento bytecode je spouštěn na JVM *(Java Virtal Machine, virtuálním stroji Javy)*. Program tedy lze spouštět všude tam, kde je JVM nainstalován. [Fal07]

7.2.1 Spring

Spring je framework pro programovací jazyk Java. Návrhové vzory Inversion of Control (IoC) a Dependency Injection (DI) využívané frameworkem Spring poskytují základ pro širokou škálu funkcí, které pak usnadňují vývoj enterprise aplikací v Javě. [spr23]

7.2.2 Python a framework Django

Alternativou ke Springu by mohl být pythonovský framework Django. Při vývoji monolitických systémů je sice oblíbenější a obecně se považuje za jednodušší než Spring, ten však vítězí ve schopnosti zpracovat více dotazů najednou. A není postaven na Pythonu.

7.3 REST API

REST API je rozhraní, které splňuje principy architektonického stylu REST *(representational state transfer, přenos reprezentativního stavu)*.[Fie00] Tyto principy jsou:

- **Jednotné rozhraní**
- **Oddělení klienta od serveru**
- **Bezstavovost** – Každý požadavek musí obsahovat všechny informace potřebné pro jeho zpracování. Serverové aplikace nesmějí ukládat žádná data související s požadavkem klienta.
- **Možnost ukládání do mezipaměti** – Pokud je to možné, měly by být prostředky na straně klienta nebo serveru uloženy v mezipaměti. Cílem je zlepčit výkon.
- **Vrstevnatá architektura systému**
- **Kód na vyžádání (volitelný)** – Rozhraní REST API obvykle odesílají statické zdroje, ale v některých případech mohou odpovědi obsahovat také spustitelný kód (například applety Java). V těchto případech by se měl kód spouštět pouze na vyžádání.

REST API komunikuje prostřednictvím požadavků HTTP a provádí standardní databázové funkce, jako je vytváření, čtení, aktualizace a mazání záznamů (známé také jako CRUD operace – *create, read, update, delete*) v

rámci zdroje. Rozhraní REST API například používá požadavek GET pro načtení záznamu, POST pro jeho vytvoření, PUT pro aktualizaci záznamu a DELETE pro jeho odstranění.[res23]

7.4 Frontend

I když frontend systému není součástí zadání, rozhodl jsem se alespoň jeho většinu napsat v JavaScriptovém frameworku **React**.

React je v praxi velmi oblíbenou a často využívanou knihovnu, kterou jsem si již pár let chtěl vyzkoušet a naučit se.

React se používá k vytváření uživatelských rozhraní specificky pro jednostránkové aplikace. Také nám umožňuje vytvářet opakovaně použitelné komponenty uživatelského rozhraní. [rea23]

7.5 Mobilní aplikace

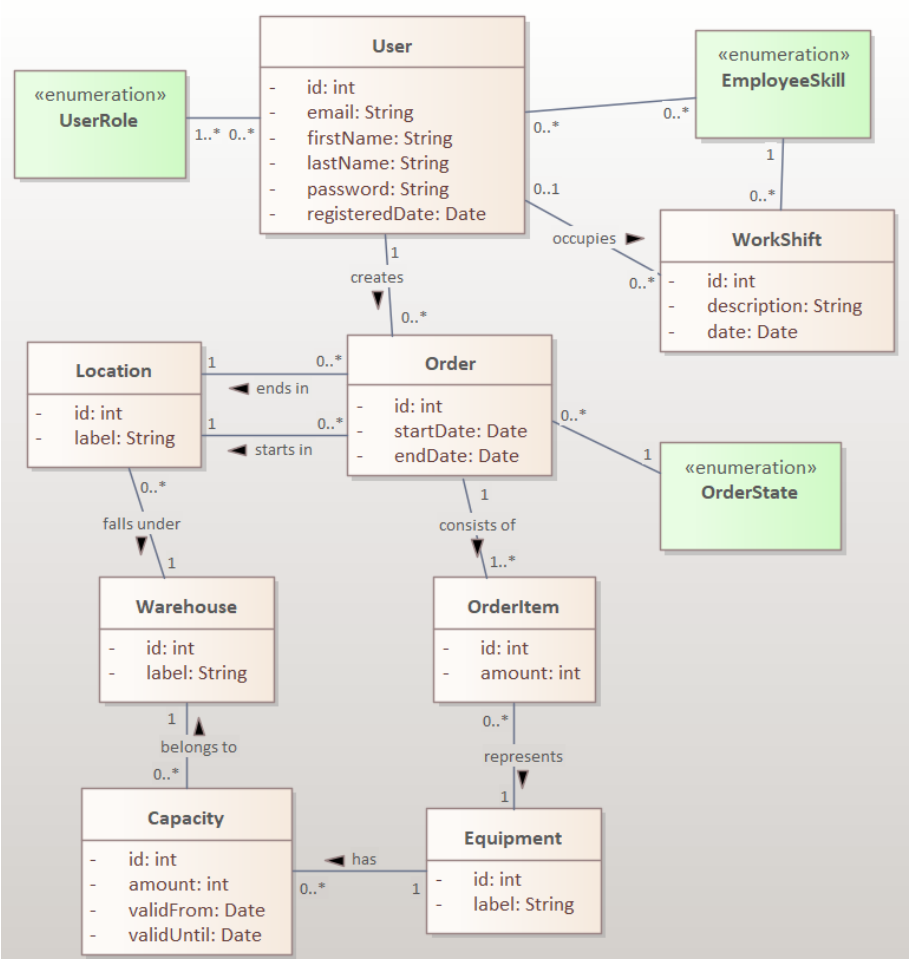
Při návrhu systému jsem přemýšlel i nad potřebou implementace mobilní aplikace.

V případě jejího psaní bych zvolil buď vývoj v jazyce **Kotlin**, se kterým mám zkušenost z předmětu Principy tvorby mobilních aplikací, a nebo, v souladu s výše zmíněnou implementací frontendu v Reactu, framework **React Native**.

Kapitola 8

Diagram tříd

Diagram tříd (*class diagram*) popisuje jednotlivé entity, jejich atributy a vztahy mezi nimi.



Obrázek 8.1: Doménový model, zdroj: Autor

Z diagramu lze vidět, že:

- **Uživatel** má přiřazeny své **role**, pokud je jedna z nich zaměstnanec, mohou mu být přiřazeny **zaměstnanecké dovednosti**, na které jsou vypisovány **pracovní směny**.
- Směny jsou vytvářeny **správcem půjčovny** na určité dny a pokud zaměstnanec dovednost "umí" a daný den zatím nemá přihlášenou jinou směnu, může se přihlásit k této.
- Uživatel vytváří **objednávky**. Ty mají **počáteční datum a místo** vypůjčení vybavení a **koncové datum a místo** vrácení.
- Objednávka se dále skládá z **položek objednávky**. Ty obsahují typ a počet vybavení a jsou při vytváření kontrolovány proti **kapacitě skladu**, kterému patří **místo vypůjčení** vybavení.
- **Kapacita** se nastavuje pro vybavení na určité časové rozmezí. Reprezentuje počet vybavení, které je v rozmezí ze skladu celkem možné půjčit.
- Objednávka má také svůj **stav**. Stavy objednávky popisuje následující kapitola.

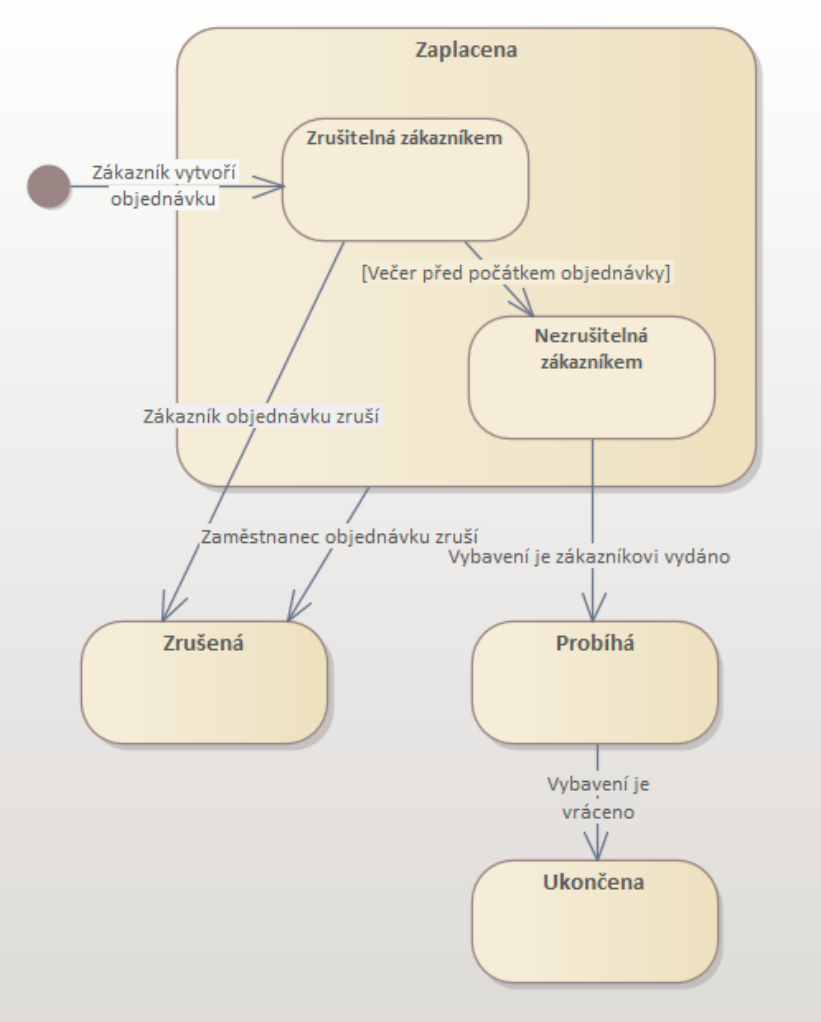
Kapitola 9
Diagram stavů

Diagram stavů slouží k modelování životního cyklu entit. Obsahuje počátek, jednotlivé stavy entity, přechody mezi nimi a koncový stav. V práci proto modeluji diagram stavů objednávky.

9.1 Diagram stavů objednávky

S ohledem na zadání, že systém nebude provozovat finanční agendu, odpadla nutnost v diagramu *(obrázek 9.1)* modelovat a systémem řešit placení objednávek. Objednávce je tedy ihned po vytvoření přiřazen stav **Zaplacena**. Tehdy **je objednávku možné zrušit** samotným zákazníkem i zaměstnanci půjčovny. Pokud se však zákazník snaží zrušit svou objednávku večer přede dnem, který je uveden jako počátek objednávky, systémem mu to není dovoleno.

Ve chvíli, kdy je zákazníkovi objednané vybavení vydáno, přepíná pověřený zaměstnanec stav objednávky do stavu **Probíhá**. Po vrácení vybavení je přepnuta do stavu **Ukončena**.



Obrázek 9.1: Diagram stavů, Diagram stavů objednávky, zdroj: Autor

Část III

Implementace a testování

Kapitola 10
Implementace

Tato kapitola popisuje implementaci systému, která vychází z předchozí části Návrh. Nachází se zde nástroje použité při implementaci, její postup a uvádím i změny, které byly během fáze implementace zapracovány.

10.1 Nástroje

Při výběru nástrojů jsem dával přednost, stejně jako u technologií, mně známému softwaru.

10.1.1 Intellij IDEA

Pro vývoj backendu i frontendu jsem vybral "Ideu" od české firmy JetBrains, k jejichž produktům mají studenti během studia na ČVUT v Praze přístup. Toto vývojové prostředí poskytuje vše potřebné k psaní kódu, nabízí nástroje k práci s databází i s verzovacím systémem a je k němu k dispozici velké množství pluginů dále usnadňujících práci.

10.1.2 Git

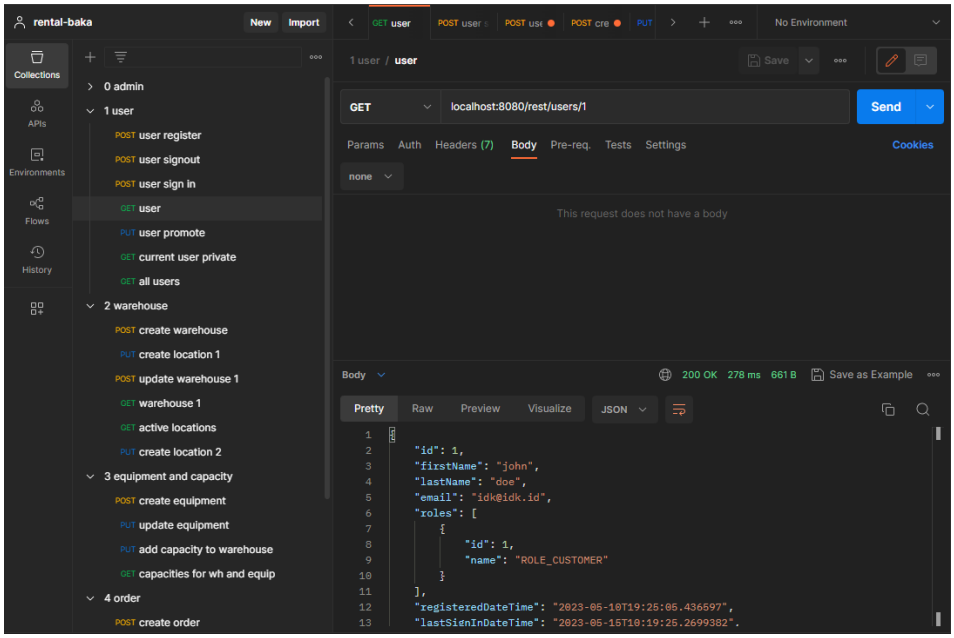
Git je verzovací systém a při práci v týmu slouží k jednoduchému sdílení aktuální verze kódu a možnosti pracovat na něm ve více lidech současně. Při vývoji jednotlivce je využit k zálohování souborů. Pro gitové repozitáře pro backend a frontend byla využívána služba Github.

10.1.3 Postman

Postman je nástroj na vývoj a využívání API a skvěle se hodí k testování REST API endpointů *(koncových bodů rozhraní)*. Postman nechává uživatele vytvářet celé kolekce HTTP dotazů, sdílet je a navzájem je na sebe řetězit v pipelinách.

10.1.4 Enterprise Architect

Toto profesionální prostředí bylo využito ke tvorbě všech UML diagramů, které se v této práci nachází.



Obrázek 10.1: Postman, Ukázka kolekcí a HTTP požadavku na RESTový endpoint, zdroj: Autor

10.1.5 PgAdmin

PgAdmin je open-source nástroj pro správu PostgreSQL. Poskytuje grafické uživatelské rozhraní, které uživatelům umožňuje snadnou interakci s databázemi PostgreSQL.[Pag23]

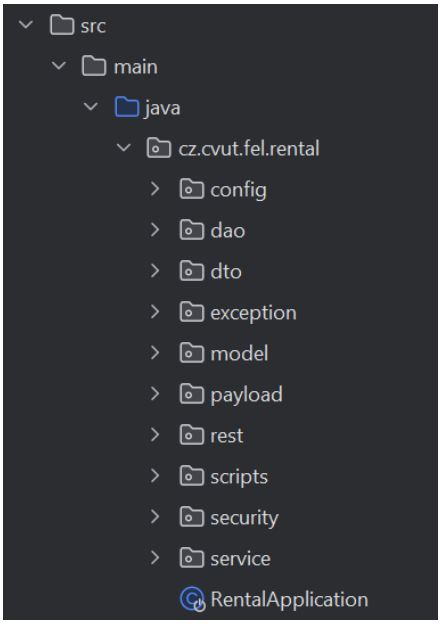
10.2 Zajímavosti z implementace

10.2.1 Postup

Začal jsem implementací backendu. Nejprve třídami složek **model** a **dao**, které obsahují samotné databázové entity a rozhraní pro jejich manipulaci v databázi. Poté jsem střídavě pracoval na třídách složek **service** a **rest**, kde přes REST rozhraní přichází požadavky na systém. Tyto požadavky jsou poté zpracovány v service třídách, popřípadě, pokud pracují s databázovými objekty, jsou předávány dál do dao vrstvy.

Lokálně potom byla zprovozněna PostgreSQL databáze a vyřešena autentizaci a autorizaci uživatelů přes JWT (JSON Web Token) pomocí tříd ve složce **security**. Složka **exception** obsahuje vlastní výjimky. Třídy složek **dto** a **payload** potom slouží k zapouzdření a přenosu dat mezi vrstvou controllerů a prezentační vrstvou systému.

Ve chvíli, kdy endpointy rozhraní fungovaly, jak měly, a prošly testováním v Postmanu, bylo na čase se přesunout na implementaci klientské části. Té muselo předcházet několikadenní studování Reactu, jelikož má dosavadní zkušenost s ním byla téměř nulová. React má ale bohatou dokumentaci a



Obrázek 10.2: Struktura backendu, zdroj: Autor

velmi aktivní komunitu, takže se základy problém nebyl.

K interakci s rozhraními API a k získávání dat ze serverů je použita JS knihovna Axios.

Jelikož v lokálním prostředí frontend jede na jiném portu *(3000)* než backend *(8080)*, tudíž se jedná o dvě různé domény, bylo potřeba vyřešit CORS *(Cross-Origin Resource Sharing, sdílení zdrojů napříč doménami)*. To je v systému řešeno proxy middlewarem:

```
const { createProxyMiddleware }
                    = require("http-proxy-middleware");
module.exports = function (app) {
    app.use(
        "/rest",
        createProxyMiddleware({
            target: "http://localhost:8080/",
            changeOrigin: true,
        })
    );
};
```

Soubory Axios služeb potom vypadají tímto způsobem (příklad z *auth.service.js*):

```
import axios from "axios";

const API_URL = "http://localhost:3000/rest/auth";
const login = (email, password) => {
    return axios
        .post(API_URL + "/signin", {
```

```
            email,
            password,
        }.then(
            ...
        );
}
```

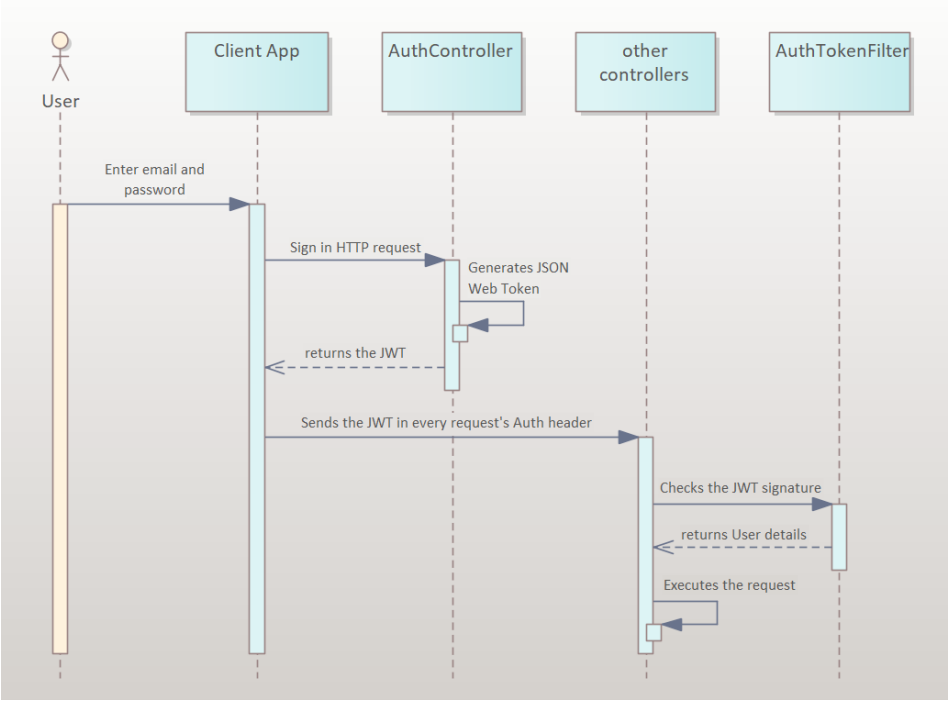
Můžeme tedy přes Axios posílat požadavky na port frontendu a jsou poté díky proxy middleware přesměrovány bez porušení CORS politik.

10.2.2 JWT Autentizace

JWT je JSON objekt, který se skládá z hlavičky (header), dat (payload) a podpisu (signature). Cílem JWT je možnost ověření autenticity dat – je důkazem, že data nebyla cestou změněna.[jwt23]

1. Uživatel se přihlásí uživatelským jménem (v tomto případě emailem) a heslem.
2. Autentizační server ověří identitu uživatele a vytvoří JWT, který vrátí uživateli.
3. Uživatel potom posílá JWT v hlavičce při každém volání API rozhrání

Toto je také vidět na vloženém sekvenčím diagramu *(obrázek 10.3)*



Obrázek 10.3: JWT Autentizace, Sekvenční diagram JWT Autentizace, zdroj: Autor

10.3 Změny během implementace

10.3.1 Uzavření systému neautorizovaným uživatelům

V minulé verzi analýzy a návrhu role nepřihlášeného uživatele obsahovala požadavky a na nich postavené případy užití:

- Zobrazit seznam vybavení
- Zobrazit seznam poboček

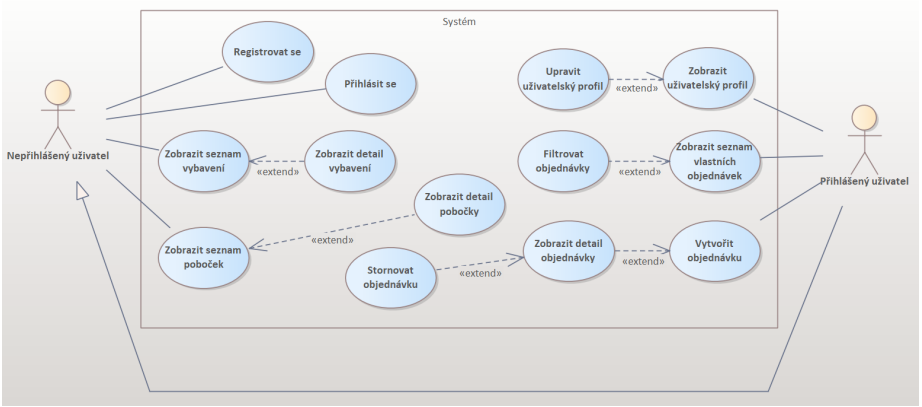
a jejich rozšiřující use-casy *(viz obrázek 10.4)*.

Během implementace bylo rozhodnuto tyto use-casy uzavřít za přihlášení do systému *(viz obrázek 5.1 v kapitole 5 Případy užití)*. Vybavení a pobočky by mohl nepřihlášený uživatel nalézt například na oddělených webových stránkách půjčovny a do našeho systému proto nemusí.

10.3.2 Částečný frontend

Implementovaný frontend konzumuje koncové body API spojené se správou skladů a poboček půjčovny, vybavení a uživatelů a je možné spravovat celý životní cyklus objednávky, postrádá však, pro jeho komplikovanost a rozsah, zpracování koncových bodů API obsluhujících plánování směn zaměstnanců. Tyto endpointy byly nicméně důkladně protestovány v nástroji Postman.

Frontend má také rezervy ve zpracování různých vyjímek, které jsou na backendu zachyceny.



Obrázek 10.4: Změny během implementace: Staré případy užití, zdroj: Autor

Kapitola 11

Testování

Testování je nedílnou součástí vývoje softwaru. Jde při něm o odhalování chyb v systému, zda funguje tak, jak se po něm chce, a zda se uživateli dobře používá.

11.1 Postman

Hlavní část testování probíhala v nástroji Postman. Průběžně v něm byly všechny funkcionality protestovány posíláním HTTP požadavků na koncové body RESTového rozhraní.

11.2 Uživatelské testování

Uživatelské testování mělo za cíl ověřit, že zaměstnanci i zákazníci budou schopni systém používat, vzhledem k neúplnému frontendu se však primárně jednalo o ověření, že systém bude dělat to, co po něm uživatel požaduje.

Testování proběhlo se 3 uživateli, 2 z nich v minulosti v půjčovně pracovali. Spolu se zařízením, na kterém systém lokálně běžel, jim byl předán testovací scénář obsahující všechny případy užití, které frontend obsahoval implementované.

11.2.1 Testovací scénář

1. Přihlaste se k poskytnutému účtu správce půjčovny
2. Založte půjčovnu – to zahrnuje:
   - Vytvořte sklad vybavení
   - Vytvořte vybavení
   - Založte ve vytvořeném skladu kapacitu vytvořeného vybavení pro tento měsíc
   - Vytvořte dvě lokace, mezi kterými půjde půjčovat vybavení
3. Odhlaste se

4. Založte si uživatelský účet pro zákazníka a přihlaste se k němu
5. Vytvořte novou objednávku
6. Zrušte ji a vytvořte novou

7. Odhlaste se a přihlaste znovu k účtu správce
8. Povyšte váš zákaznický účet na zaměstnance

9. Odhlaste se a přihlaste znovu k vašemu právě povýšenému účtu.
10. Jako zaměstnanec půjčovny najděte v denním výpisu Vaši objednávku a označte ji jako probíhající

11.3 Vyhodnocení testování

Průběžné testování v Postmanu odhalovalo většinu funkcionálních nedostatků, které byly zpravidla hned opraveny. V době uživatelského testování tedy backend fungoval tak, jak měl. Uživatelé neměli sice s používáním aplikace žádné větší problémy, frontend však není tak rozsáhlý a nedává uživateli mnoho možností se ztratit.

Kapitola 12

Závěr

Cílem práce bylo navrhnout a implementovat informační systém pro správu půjčovny. Byl proveden průzkum fungování půjčovny lodí a analýza nejčastěji volených řešení pro tento systém půjčovny.

Na základě této analýzy byly specifikovány požadavky na systém a případy užití. Zaměstnancům se díky systému zpřehlední průběh a správa objednávek, dále se zjednoduší správa zákazníků a vybavení. Příjemným rozšířením ve fungování půjčovny bude i plánování směn. Zákazníkům se díky systému zpřístupní přehledný nástroj pro půjčování vybavení.

Dále byla navržena architektura a technologie, které byly využity k implementaci systému.

Jako architekturu systému jsem zvolil architekturu vrstevnatou, převážně kvůli zkušenostem nabytých během studia, ale také pro možnost jednoduchého škálování během implementace nebo po nasazení.

Jako programovací jazyk backendu jsem vybral Javu s frameworkem Spring, který zjednodušuje vývoj enterprise aplikací. Pro databázi byl využit PostgreSQL.

Frontend jsem se rozhodl psát v knihovně React JS kvůli jejímu hojnému využití v praxi.

Za použití vhodných nástrojů byla realizována implementace. Také bylo provedeno testování včetně uživatelského testování.

Budoucí vývoj

Přestože informační systém byl implementován a otestován, bude v budoucnu třeba dalších úprav a rozšíření. Během implementace se neustále objevovaly nápady na nové funkcionality, ale vypracování projektu takového rozsahu by vyžadovalo mnoho času, který překračoval časové možnosti jednoho semestru.

- V 10.3.2 zmiňuji částečný frontend, kde nejsou napojeny všechny koncové body rozhraní.
- Napsaný frontend by si v budoucnu zasloužil propracovanější zpracování.
- Není implementováno vyhledávání a filtrování objednávek a uživatelů, které by v reálu jistě byly kritickými požadavky.
- Rozšíření o detailnější plánování směn.
- Rozšíření o finanční agendu - platba objednávek, mzdy zaměstnanců podle odpracovaných směn.
- Rozšíření frontendu o další jazyky.

Přílohy

Příloha A

Literatura

[ass23] *Construction equipment and tool tracking software - asset panda*, Asset Panda, https://www.assetpanda.com/solutions/construction/, 2023, Online; zhlédnuto 29. prosince 2022.

[boo23] *Manage your rental business - orders, inventory, payments - booqable rental software*, Booqable.Com, https://booqable.com/backend, 2023, Online; zhlédnuto 30. prosince 2022.

[cap23] *Top software at capterra | software software reviews for business nonprofit*, Capterra.Com, https://www.capterra.com, 2023, Online; zhlédnuto 27. prosince 2022.

[cli20] *What is client/server architecture? - definition from techopedia*, Techopedia.Com, https://www.techopedia.com/definition/438/clientserver-architecture, 2020, Online; zhlédnuto 9. dubna 2023.

[ezr] *Equipment rental software - ezrentout*, Ezrentout.Com, https://www.ezrentout.com/, Online; zhlédnuto 29. prosince 2022.

[Fal07] Lukáš Faltýnek, *Java – dnes při šálku dobré kávy*, Linuxexpres.Cz, https://www.linuxexpres.cz/praxe/java-dnes-pri-salku-dobre-kavy, 2007, Online; zhlédnuto 10. února 2023.

[Fie00] Roy Thomas Fielding, *Fielding dissertation: Chapter 5: Representational state transfer (rest)*, Ics.Uci.Edu, https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 2000, Online; zhlédnuto 17. dubna 2023.

[fun23] *Functional vs non-functional requirements [updated 2021]*, Enkonix.Com, https://enkonix.com/blog/functional-requirements-vs-non-functional/, 2023, Online; zhlédnuto 29. ledna 2023.

[jwt23] *Spring boot security + jwt*, Javainuse.Com, https://www.javainuse.com/spring/boot-jwt, 2023, Online; zhlédnuto 17. dubna 2023.

[Kom16] Ing. Martin Komárek, *Analýza a dokumentace požadavků, [přednáška]*, moodle.fel.cvut.cz, https://moodle.fel.cvut.cz/pluginfile.php/42038/mod_resource/content/3/ANALYZA_A%20DOKUMENTACE%20POZADAVKU_3PREDNASKA.pdf, 2016, Online; zhlédnuto 21. ledna 2023.

[Kom19] ———, *Uml diagramy případů užití, [přednáška]*, moodle.fel.cvut.cz, https://moodle.fel.cvut.cz/pluginfile.php/233850/mod_resource/content/5/P%C5%99edn%C3%A1%C5%A1ka4a5_PripadyUzitiKomplet.pdf, 2019, Online; zhlédnuto 22. ledna 2023.

[lay23a] *Layered architecture*, Cs.Uwaterloo.Ca, https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Layered.pdf, 2023, Online; zhlédnuto 10. března 2023.

[lay23b] *Software architecture patterns*, O'Reilly Online Learning, https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html, 2023, Online; zhlédnuto 10. března 2023.

[LF14] James Lewis and Martin Fowler, *Microservices*, Martinfowler.Com, https://martinfowler.com/articles/microservices.html, 2014, Online; zhlédnuto 9. dubna 2023.

[mon23] *Microservices vs. monolithic architecture | atlassian*, Atlassian.Com, https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith, 2023, Online; zhlédnuto 9. dubna 2023.

[Pag23] Dave Page, *Pgadmin - postgresql tools*, Pgadmin.Org, https://www.pgadmin.org/, 2023, Online; zhlédnuto 15. března 2023.

[PM14] Roger S. Pressmann and Bruce Maxim, *Software engineering: A practitioner's approach*, McGraw Hill, 2014, ISBN-10: 9780078022128.

[pos23] *Postgresql: About*, Postgresql.Org, https://www.postgresql.org/about/, 2023, Online; zhlédnuto 13. března 2023.

[rea23] *What and why react.js*, C-Sharpcorner.Com, https://www.c-sharpcorner.com/article/what-and-why-reactjs/, 2023, Online; zhlédnuto 2. dubna 2023.

[ren23] *The platform to sell and rent anything, anywhere | rentle*, Rentle.Io, https://www.rentle.io/?hsLang=en, 2023, Online; zhlédnuto 30. prosince 2022.

[res23] *What is a rest api? | ibm*, Ibm.Com, https://www.ibm.com/topics/rest-apis, 2023, Online; zhlédnuto 17. dubna 2023.

[spr23] *Why spring?*, Spring.Io, `https://spring.io/why-spring`, 2023, Online; zhlédnuto 12. února 2023.