VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRODEJCE NÁTĚROVÝCH HMOT

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ KŘENEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Křenek Tomáš**
Program: Informační technologie
Název: **Informační systém prodejce nátěrových hmot**
Information System of Paints Seller
Kategorie: Informační systémy

Zadání:

1. Seznamte se s tvorbou webových aplikací a prostudujte potřebná vývojová prostředí.
2. Seznamte se s metodami pokročilé analýzy dat např. pomocí metod získávání znalostí.
3. Analyzujte požadavky na informační systém firmy prodávající barvy zahrnující kromě základní funkcionality i plánování objednávek s využitím Google Calendar API a různé možnosti statistické a pokročilé analýzy prodejních dat.
4. Na základě požadavků navrhněte informační systém.
5. Navržený systém implementujte a otestujte jeho funkčnost na vhodném vzorku dat.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 11. října 2021

Abstrakt

Cílem této práce bylo vytvořit návrh a implementovat informační systém pro prodejce nátěrových hmot. Implementace je provedena formou webové aplikace v jazyce PHP s využitím frameworku Laravel. Při práci byl kladen důraz zejména na jednoduchost a uživatelskou přívětivost, protože se jedná o firemní systém. Hlavní funkcí systému je správa objednávek, přidávání produktů do objedávek a výběr balení produktů. Zaměstnanci mají možnost zobrazit si termíny objednávek ve svém Google kalendáři. Součástí systému je také zobrazení prodejních statistik. Zákazníkům je doporučováno zboží na základě zpracování asociačních pravidel.

Abstract

The aim of the thesis was to create a design and to implement an information system for paint surface treatment sellers. Implementation is carried out in the form of a web application in PHP using the Laravel framework. Simplicity and user-friendly design were emphasized due the system being used in a corporate environment. The main function of the system is order management, adding products to respective orders and selecting product packaging. Employees have the ability to view order dates in their Google Calendar client. The system also includes reporting sales statistics. Customers are advised goods based on processing of association rules.

Klíčová slova

Informační systém, PHP, Laravel, Bootstrap, Apriori, Google Calendar API, MySQL, HTML, CSS, JavaScript, Zaměstnanec, Zákazník, Produkt, Objednávka

Keywords

Information sytem, PHP, Laravel, Bootstrap, Apriori, Google Calendar API, MySQL, HTML, CSS, JavaScript, Employee, Customer, Product, Order

Citace

KŘENEK, Tomáš. *Informační systém prodejce nátěrových hmot.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém prodejce nátěrových hmot

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Křenek
7. května 2022

Poděkování

Chtěl bych poděkovat panu Ing. Vladimíru Bartíkovi, Ph.D. za to, že odborně vedl moji bakalářskou práci a poskytoval mi cenné rady.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá návrhem a tvorbou informačního systému pro firmu, zabývající se prodejem nátěrových hmot. Cílem práce je vytvořit přehledný firemní systém, ve kterém bude jednoduchá orientace a žádné zbytečné prvky, které by zastiňovaly nejdůležitější prvky systému, které budou uživatelé nejčastěji používat. V rámci informačního systému by mělo být zákazníkům umožněno jednoduché vytvoření a následná správa jejich objednávek. Zaměstnanci firmy, kteří se systémem budou pracovat, budou mít přehled o všech objednávkách, které zákazníci vytvořili a budou moct jednoduše měnit stav, ve kterém se právě nachází a označovat provedenou práci. Zaměstnanci budou také mít přehled o všech objednávkách v rámci Google kalendáře. V rámci informačního systému budou k dispozici statistiky týkající se zejména prodeje produktů, ale i další. V informačím systému bude implementována také analýza asociačních pravidel, díky které zákazníkům budou doporučeny produkty na základě obsahu jejich objednávky.

Nejdříve ve své práci popisuji základní principy při tvorbě webových aplikací v rámci kapitoly 2. V této kapitole jsou popsány architektury, které se při tvorbě webových aplikací využívají. Dále jsou zde obecně popsány nejčastěji používané technologie při tvorbě informačních systémů. V kapitole 3 je popsána teorie k získávání znalostí dat. Nejdříve je popsáno, o co se vlastně jedná a proč dolování provádíme. Poté jsou uvedeny metody pro dolování dat shlukováním, predikcí, klasifikací a asociačními pravidly.

V kapitole 4 jsou popsány a analyzovány požadavky na výsledný systém od zákazníka. V rámci této kapitoly je popsán a navržen use case diagram. Na tuto kapitolu navazuje kapitola 5, ve které je popsán návrh systému. V první části je popsán ER-diagram a entity, které obsahuje. Ve druhé části této kapitoly popisuji MVC architekturu, kterou tento informační systém využívá. Následuje kapitola 6, která se zabývá konkrétními technologiemi, které byly při tvorbě systému využity. Nejdříve je popsán framework Laravel a porovnán s ostatními frameworky. Dále jsou popsány front-end technologie a také Google Calendar API.

Kapitola 7 popisuje strukturu souborů a také implementaci jednotlivých částí systému. Po této kapitole následuje poslední kapitola 8, která popisuje, jak probíhalo testování informačního systému.

Závěrečná kapitola 9 hodnotí celou práci. Je zde popsáno, co se povedlo a také, co mohlo být vyřešeno lépe. Dále jsou popsány možná vylepšení této práce do budoucna.

Kapitola 2

Architektura a tvorba informačních systémů

Úvodní kapitola, která vysvětluje základní pojmy jako je webová aplikace nebo informační systém a popisuje nejčatější architektury, nad kterými informační systémy pracují. Součástí této kapitoly je také popis nejčastěji využívaných technologií v této problematice.

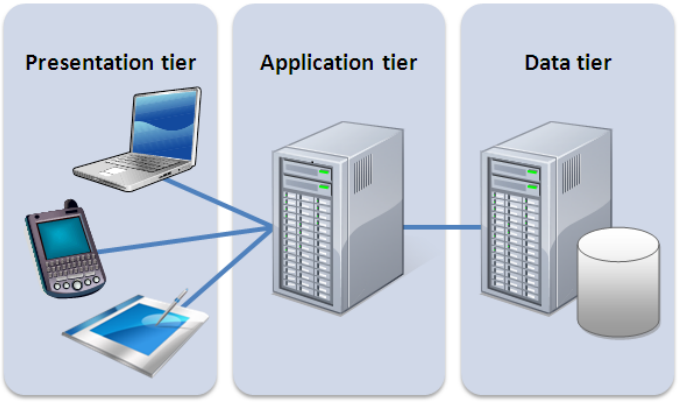
2.1 Webová aplikace

Webová aplikace je aplikace, která je uložena na vzdáleném webovém serveru a je poskytována uživatelům v rámci internetu nebo v rámci podniku pomocí sítě intranet. Popularita webových aplikací je zapříčiněna především snadným přístupem za pomoci webového prohlížeče, kterým může být například Mozilla Firefox nebo Google Chrome. [44] Dříve se využívala architektura klient-server, ta je ale v dnešní době poměrně zastaralá, proto většina moderních webových aplikací využívá tzv. třívrstvou architekturu.

Třívrstvá architektura

Třívrstvá architektura je v dnešní době nejpoužívanější architekturou v rámci informačních systémů. Architektura je rozdělena do tří základních vrstev: [41]

- Nejnižší vrstvou architektury je vrstva datová (databázová), kterou tvoří databáze. S daty dále pracuje a provádí nad nimi operace, které zajišťují ukládání, agregaci nebo výběr dat a další.
- Aplikační vrstva (funkční) je prostřední vrstva zajišťující operace a výpočty, které jsou prováděny mezi daty a vstupně-výstupními požadavky s daty.
- Nejvyšší vrstva se nazývá prezentační. Tato vrstva je pro uživatele viditelná, zajišťuje vstup uživatelských požadavků a následnou prezentaci výsledků. Vrstva je reprezentována webovým prohlížečem.

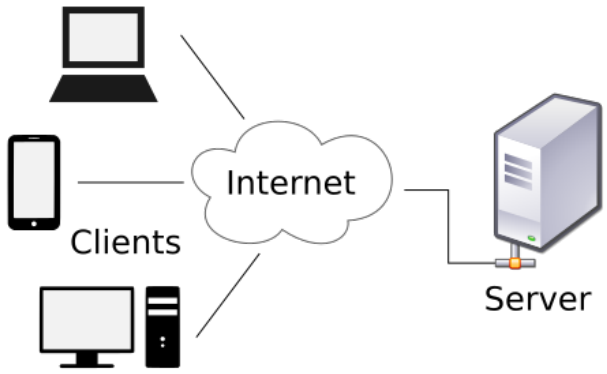


Obrázek 2.1: Třívrstvá architektura. Převzato z: [41].

Architektura klient-server

Tato architektura se v dnešní době už tolik nevyužívá, protože bývá nahrazována již zmíněnou třívrstvou architekturou. Uvádím ji zde pouze pro představu, jaké jsou mezi nimi odlišnosti. Tato architektura se oproti třívrstvé architektuře skládá pouze ze dvou částí: klienta a serveru. Pokud chce klient vyžívat nějakou službu, kterou server nabízí, připojí se k serveru a následně mezi sebou vzájemně komunikují a posílají si data. Klient posílá požadavky a server následně odpovídá, pro každého klienta vytvoří server relaci. V dnešní době je tato architektura nahrazena právě proto, že klient obsahuje většinu aplikační logiky a aplikace jsou velmi složité, proto rostou nároky na počítače a zařízení obecně. Také kvůli bezpečnosti není v dnešní době architektura vhodná. [10]

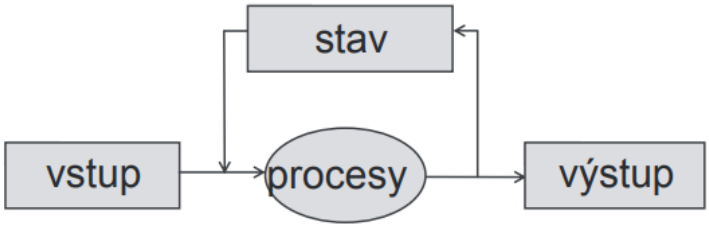
Na serveru je dostupná relační databáze, nad kterou server provádí zpracování dotazů. Klient obsahuje aplikační logiku a uživatelské rozhraní pro uživatele. Uživatel posílá požadavky, které klient přeloží do podoby, která je pro server srozumitelná. Poté dostane odpověď od serveru, kterou opět přeloží, tak aby byla srozumitelná pro uživatele a zobrazí mu výsledek v přijatelné podobě. [10]



Obrázek 2.2: Architektura klient-server. Převzato z: [9].

2.2 Informační systém

Informační systém lze chápat jako vzájemně propojené informace a s nimi pracující procesy. Můžeme říci, že informace jsou data, pomocí kterých se v systému řídíme a rozhodujeme. Procesy se dají chápat jako funkce, které transformují vstupní údaje na výstupní. Procesy tedy zabezpečují sběr, uložení, přenos a distribuci dat. [23]



Obrázek 2.3: Schéma informačního systému. Převzato z: [24].

2.3 Klasifikace informačních systémů

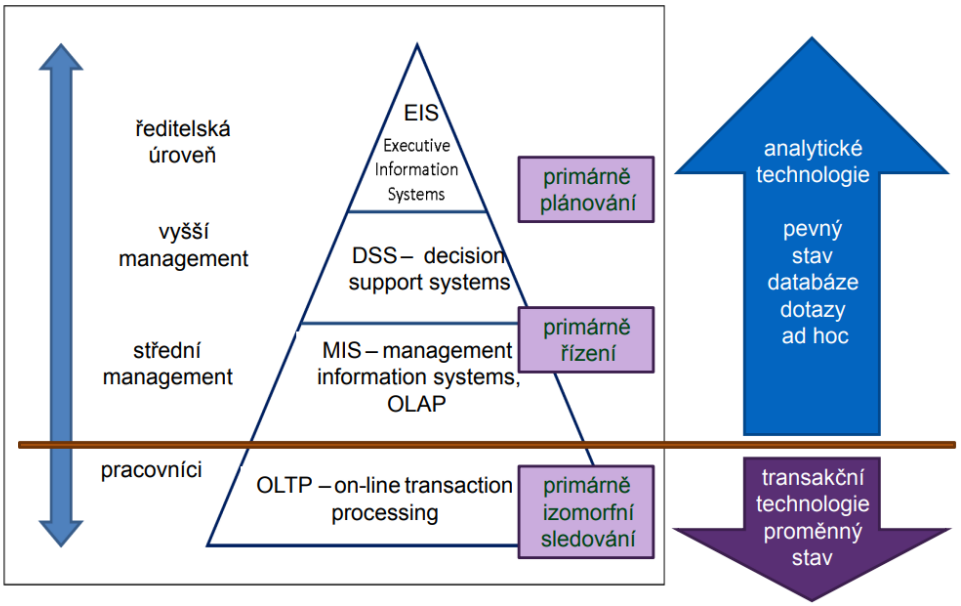
Informační systémy se dají rozdělit podle jejich klasifikace do dvou základních skupin, těmi jsou OLTP a OLAP systémy.

OLTP systémy

Pomocí OLTP (Online Transaction Processing) systémů může uživatel databázového serveru provádět mnoho trasnsakcí online. Mezi tyto transakce patří například vkládání, mazání, dotazování nebo provádění jednoduchých analýz nad daty v databázi. V praxi si tyto transakce můžeme představit například jako internetové bankovnictví nebo nákupy v e-shopech. Jsou to tedy systémy, které využívají uživatelé v běžném životě. Mezi výhody OLTP patří zejména to, že více uživatelů přistupuje ke stejným datům a provádí nad nimi velké množství transakcí a zároveň zajišťuje integritu dat. Také poskytuje indexované soubory dat. [35]

OLAP systémy

Systémy OLAP (Online Analytical Processing) jsou systémy určené pro pokročilé analyzování velkého množství dat. Díky těmto analýzám vznikají reporty a souhrny dat, na jejichž základě se rozhodují manažeři v oblasti řízení firmy, řízení technologických nebo ekonomických procesů. Pro získání analýz pomocí OLAP je zapotřebí vykonat velké množství různých operací, a rozhodně se nejedná o triviální proces. [28]



Obrázek 2.4: Klasifikace informačních systémů na základě úrovně rozhodování. Převzato z: [24].

2.4 Back-end

Jedná se o administrační sekci webu, sloužící k administraci obsahu stránek. Tato sekce není běžným uživatelům přístupná. Pro přístup k této části aplikace je totiž vyžadována autorizace. [3]

PHP

PHP (Hypertext Preprocessor)[1] je multiplatformní programovací jazyk pracující na straně serveru. Byl vyvinut Rasmusem Lerdorfem pro vývoj webů. PHP je multiplatformní jazyk, lze ho tedy používat na většině operačních systémů. Zejména se používá pro tvorbu dynamických webových stránek, ale lze ho použít také pro vývoj desktopových aplikací. PHP není příliš náročné pro uživatelská zařízení, proto je v dnešní době velice rozšířené a oblíbené. [36]

PHP má mnoho frameworků pro vývoj webových aplikací, mezi které patří například Laravel, Nette nebo Symfony.

Java

Java[2] je programovací jazyk, který byl vyvinut firmou Sun Microsystems v roce 1995. Java je vysokoúrovňový programovací jazyk, který podporuje objektově orientované programování. Mezi největší výhody Javy patří to, že je multiplatformní a vysoce zabezpečený. [25] Java má také mnoho frameworků, mezi ty patří například Spring, JSF nebo GWT.

[1] Dostupné z: https://www.php.net/.
[2] Dostupné z: https://www.java.com/en/.

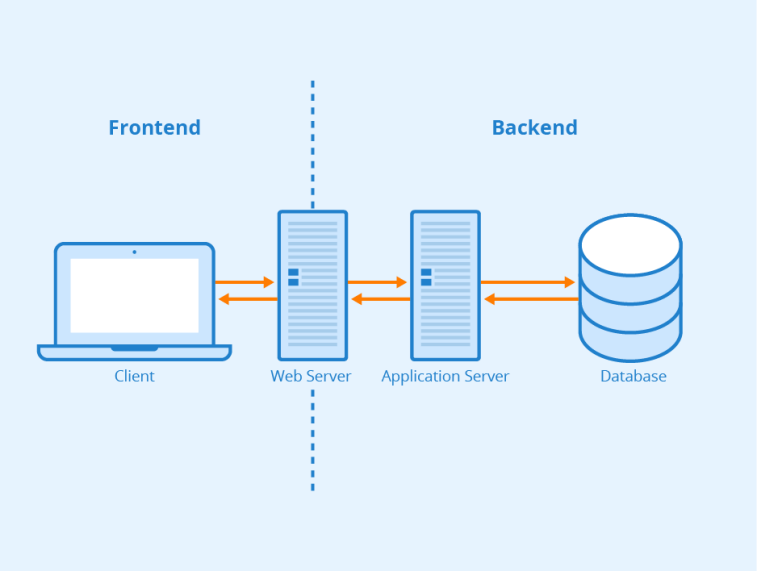
Python

Python[3] je vysokoúrovňový, objektově orientovaný a interpretovatelný programovací jazyk, který je v dnešní době velice populární. Jeho největší výhodou je určitě to, že je velice jednoduchý, přehledný a obsahuje méně syntaktických konstrukcí než jiné jazyky, proto je také vhodný pro začátečníky s programováním. Je také dynamicky typovaný a nabízí velké množství knihoven, které lze využít. [37]

Python také nabízí různé frameworky, těmi nejznámějšími jsou: Django, Pyramid nebo Flask.

2.5 Front-end

Front-end je grafický výstup webové aplikace. Jedná se tedy o sekci aplikace, kterou vidí uživatel. Proto by měl být zvláště kladen důraz na to, aby byl frontend dosatečně přehledný, uživatelsky přívětivý a jednoduchý.



Obrázek 2.5: Vztah back-endu a front-endu. Převzato z: [18].

HTML

HTML (HyperText Markp Laguage) je značkovací jazyk používaný k tvorbě obsahu webových stránek. Jeho strukturu tvoří html tagy, které mohou být párové nebo nepárové a mohou obsahovat další atributy. Tagy obalují text, který je pak formátován podle toho, který tag jej obklopuje. [22]

Celá struktura html dokumentu je mezi tagy `<html>` a `</html>`, dále musí obsahovat tagy `<head>` a `</head>`. Tyto tagy označují hlavičku dokumentu, ta obsahuje metadata, například titulek nebo odkaz na CSS soubor. Samotný obsah dokumentu se píše mezi tagy `<body>` a `</body>`. [22]

[3] Dostupné z: https://www.python.org/.

CSS

Jazyk CSS (Cascading Style Sheets) byl vytvořen v roce 1994. Je to jazyk sloužící pro popis vzhledu webové stránky pro jazyk HTML. Pomocí CSS můžeme například měnit barvu a styl písma nebo měnit zarovnání textu a rozložení celé stránky. [13]

CSS styl můžeme přidat do HTML dokumentu několika způsoby: [13]

- Zapsání pravidla přímo do HTML elementu.
- Přidání elementu `<style>` v elementu `<head>` v HTML dokumentu.
- Nejčastěji se využívá zadání odkazu na externí CSS soubor v elementu <head>.

Pro použití CSS musíme zadat pravidlo, které se skládá ze selektoru a deklarace. Pomocí selektoru se pozná na jaký HTML element se má pravidlo použít. Může se jednat přímo o html tag, třídu nebo identifikátor. Třídu vytvoříme tak, že do elementu přidáme atribut `class` (příklad: `class="trida"`) a jeho název. V případě, že se jedná o identifikátor, zadáme klíčové slovo `id` a název (příklad: `id="identifikator"`). [13]

Příklad zapsání pravidla CSS:

- Nastavení fontu pro html element p: `p {font-family: "Times New Roman";}`
- Nastavení červené barvy písma pro třídu: `.trida: {color: red;}`
- Nastavení velikosti písma 20 pro identifikátor: `#identifikator {font-size: 20;}`

Java-script

Javascript[4] je objektově orientovaný multiplatformní jazyk, který byl vytvořen v roce 1995. Syntaxe java-scriptu patří do stejné rodiny jako například Java nebo C. Je to dynamicky typovaný jazyk. Je využíván zejména proto, že pomocí něj můžeme měnit obsah webové stránky u uživatele. To nám otevírá možnost tvořit dynamické menu a další komponenty, které umožňují ušetřit místo na stránce, když jsou zavřené a po najetí myší se otevřou. [26] V této práci je využita také javascriptová knihovna jQuery[5], díky které je práce s javascriptem jednodušší.

Mezi největší výhody patří zejména to, že je java-script rychlý a rozšiřitelný. Na druhou stranu může kvůli němu dojít k ohrožení bezpečnosti klienta. [27]

2.6 Databáze

Databáze je softwarový systém, sloužící pro uchovávání dat při tvorbě webových aplikací a informačních systémů a následné správě těchto dat. Databáze běží na serveru a webové aplikace si z nich stahují data, které se poté zobrazují na webových stránkách prostřednictvím webového prohlížeče. Existuje několik typů databází, mezi ty patří například: relační, hierarchické, síťové nebo objektové databáze. [15] V této práci pracuji s relační databází.

SŘBD (Systém řízení báze dat) je univerzální označení pro software, který zprostředkovává komunikaci mezi aplikacemi a uloženými daty. SŘBD umožňuje provádět s daty následující operace: spouštění dotazů, ukládání, mazání a aktualizaci. [16]

[4] Dostupné z:https://www.javascript.com/.
[5] Dostupné z: https://jquery.com/.

Relační databáze

Relační databáze se skládá z tzv. relací, které reprezentují databázové tabulky. Tyto tabulky mohou být navzájem propojeny. Každá tabulka je definována záhlavím a tělem. Sloupce tabulky představují atributy a řádky tabulky reprezentují záznamy v tabulce. [38]

V relačních databázích máme primární a cizí klíče. Primární klíč je atribut jednoznačně identifikující záznamy a cizí klíč vytváří vztahy mezi tabulkami tím, že odkazuje z jedné tabulky na primární klíč jiné tabulky. [38]

MySQL

MySQL[6] bylo vytvořeno švédskou firmou MySQL AB v roce 1995. V dnešní době jej vlastní společnost Oracle Corporation, jedná se o jeden z nejpoužívanějších databázových systémů vůbec. MySQL uplatňuje relační databázový model, data tedy ukládá do tabulek, kde každá tabulka reprezentuje jednu entitu dat. Komunikace s databází probíhá pomocí dialektu jazyka SQL, který má určitá rozšíření oproti klasickému SQL. [33]

Jedná se o multiplatformní databázi, takže je přenositelná mezi více různými platformami. MySQL je poskytováno zdarma a je velice rychlé a jednoduché, nabízí však omezený počet funkcí. Kromě verze zdarma, existuje také placená verze, která poskytuje velké množství funkcí navíc. Pro práci s databází je v této práci využit velmi populární databázový řídící systém PHPMyAdmin, který umožňuje jednoduchou správu databáze prostřednictvím webového rozhraní. [34]

Další populární databázové systémy

Dalšími populárními databázovými systémy ve světě jsou [17]:

- Oracle database – multiplatformní databázový systém vyvíjen firmou Oracle, který umožňuje pokročilé zpracování dat a nabízí velmi vysoký výkon.
- Microsoft SQL Server – databázový systém vytvořen firmou Microsoft v roce 1989. Opět nabízí vysoký výkon.
- Postgres SQL – vydáno v roce 1996 firmou PostgreSQL Global Development Group.

[6] Dostupné z: https://www.mysql.com/.

Kapitola 3

Získávání znalostí dat

Tato kapitola pojednává o problematice získávání znalostí dat. Jsou zde popsány všeobecné základy, ale také informace zaměřující se na různé druhy dolovacích metod.

3.1 Úvod

Získávání znalostí dat (data-mining) je proces, při kterém získáváme zajímavé modely dat a vzorů z velkých objemů dat. Tyto modely a vzory reprezentují znalosti získané z dat. Získávání znalostí není triviální, znamená to tedy, že tyto znalosti nejsou na první pohled v databázi viditelné, nebo nestačí na jejich získání použít pouze SQL dotaz, ale je potřeba využít složité postupy, které zahrnují například použití netriviálních matematických vzorců. Získávání znalostí dat se rozmohlo zejména v dnešní době a několika posledních letech, kdy máme obrovský objem dat a je zapotřebí přeměnit tyto data na užitečné informace a znalosti. Ty je pak možné využívat v mnoha různých oblastech. [21, Kapitola 1]

Proces získávání znalostí

Při získávání znalostí se řídíme určitým postupem, který se skládá z několika kroků, těmi jsou: [21, Kapitola 1]

1. Čištění dat – pokud chybí nějaká data, musíme se se ztrátami vypořádat, také odstranit šum a řešit různé nekonzistence dat.

2. Integrace dat – integrujeme data, které pochází z několika různých zdrojů. Čištění i integrace jsou obvykle řešeny v jednom kroku, v takovém případě ukládáme data do datového skladu. Je to zapříčiněno tím, že data po čištění musíme rovnou někam ukládat. Data z různých zdrojů jsou také většinou nekonzistentní.

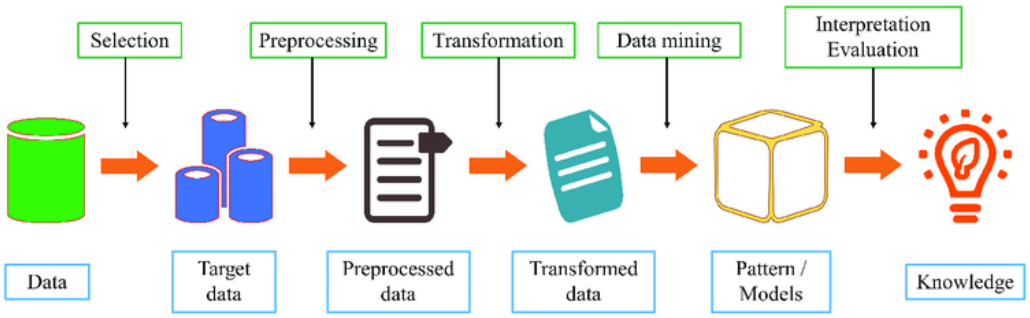
3. Výběr dat – v tomto kroku vybíráme data, která jsou relevantní. Vybíráme tedy z jedné tabulky určité sloupce (atributy).

4. Transformace dat – data musíme transformovat do podoby, ze které je možno provádět dolování. Jedná se například o agregaci nebo sumarizaci.

5. Dolování dat – jak již název napovídá, jedná se o nejdůležitější krok, při kterém dochází při aplikaci určité metody a konkrétního algoritmu k extrakci vzorů z dat.

6. Hodnocení modelů a vzorů – identifikujeme zajímavé vzory, které budou nejužitečnější.

7. Prezentace znalostí – poslední krok, při kterém prezentujeme získaná data uživateli.



Obrázek 3.1: Proces dolování dat. Převzato z: [14].

Typy dolovacích úloh

Typ dolovací úlohy určujeme podle toho, který model dat chceme po dolování získat. Dolovací úlohy můžeme rozdělit do dvou základních skupin, těmi jsou: [21, Kapitola 1]

1. Deskriptivní – jedná se o obecné vlastnosti analyzovaných dat. Například zjištění častých společných nákupů pomocí analýzy nákupního košíku.

2. Prediktivní – provádí dedukci pro předpověď budoucího chování na základě analýzy dat současných. Příkladem je klasifikace, pomocí které můžeme určit, zda je vhodné určité osobě poskytnout úvěr na základě jeho platu a dalších aspektů.

3.2 Předzpracování dat pro dolování

Data je potřeba nějakým způsobem předzpracovat, protože databáze standartně nejsou ve stavu, abychom z nich rovnou mohli dolovat. V databázích jsou často data, která jsou zašuměná, nekonzistentní, nebo dokonce určitá data chybí. Tato neočištěná data by nás mohla dovést při dolování ke zkresleným nebo chybným závěrům. Kvalita dat se všeobecně hodnotí podle několika kritérií. Data by měla co nejpřesněji odpovídat realitě, kterou modelují a musí být aktuální. Měla by být úplná, to znamená mít dostatečný počet hodnot (úplnost do hloubky) a zároveň dostatek potřebných atributů (úplnost do šířky). Data musí být také konzistentní a jejich věrohodnost musí být co nejvyšší. Také se řeší, jak snadno jsou data dostupná a hodnoty by měly být snadno interpretovatelné. [21, Kapitola 3]

Pro dolování platí již zmíněná kritéria také, ale nekvalita dat v souvislosti s předzpracováním se nejčastěji týká těchto problémů: [21, Kapitola 3]

- Nekompletní data – některé hodnoty v databázi mohou chybět, to může být zapříčiněno několika důvody. Například při zadávání dat byla položka označena jako nepovinná, položka mohla být zadána chybně nebo došlo k nějaké poruše. Hodnota také mohla být nekonzistentní s ostatními hodnotami a tak byla zrušena.

- Zašuměná data – atributy obsahují odlehlé nebo nesprávné hodnoty. Může se jednat například o hodnotu venkovní teploty o hodnotě 1000 stupňů Celsia. Příčinou může být například chyba při sběru dat.
- Nekonzistentní data – může dojít k redundanci dat, pokud použijeme data z několika různých zdrojů. Nekonzistence mohou být v pojmenování, kódování nebo formátech a dalších. Redundance ale může nastat i u dat z jednoho zdroje, například při špatném návrhu databáze.

Hlavní úlohy předzpracování: [21, Kapitola 3]

1. Čištění dat – při tomto kroku se snažíme odstranit odlehlé a chybějící hodnoty a řešit nekonzistence. Pokud bychom měli dolovat z nekvalitních dat, mohl by být ovlivněn dolovací algoritmus a my bychom mohli dostat nespolehlivé výsledky.
2. Integrace dat – integrujeme data z různých datových zdrojů.
3. Transformace dat – data je nutné transformovat do tvaru vhodného pro řešení dané dolovací úlohy. Jedná se o agregaci a normalizaci.
4. Redukce dat – v tomto kroku se snažíme zredukovat objem dat. I menší redukce je velice nápomocná a usnadní mnoho práce při dolování. Patří sem například redukce počtu hodnot a redukce dimenzionality.
5. Diskretizace dat – jedná se také o redukci dat, ale má zvláštní význam, a to takový, že se redukují počty hodnot atributů. Týká se hlavně spojitých dat.

3.3 Klasifikace a predikce

V této podkapitole jsou popsány základy klasifikace a predikce, také jsou uvedeny příklady jejich metod.

Klasifikace

Klasifikace je proces, díky kterému jsme schopni data přiřadit do určitých tříd na základě jejich vlastností. Jako příklad pro klasifikaci může být uvedena banka, do které zákazníci přijdou pro úvěr. Pomocí klasifikace jsme schopni rozdělit zákazníky do dvou skupin na základě jejich vlastností, například na základě jejich platu, hypotéky, majetku a další. První skupina budou zákazníci, kterým bude úvěr schválen a do druhé skupiny budou patřit zákazníci, u kterých by bylo riziko úvěru příliš vysoké a úvěr nedostanou. [21, Kapitola 8]

Fáze klasifikace

První fáze klasifikace se nazývá fáze učení. V této fázi jsou vybrány vzorky dat z databáze, množinu vybraných dat nazýváme trénovací množinou. Vzorky dat dále slouží jako vstup pro klasifikátor, u těchto vzorků musíme předem vědět do jaké třídy je zařadit. Klasifikátor poté zjistí, jaká jsou klasifikační pravidla, které poté slouží pro klasifikaci objektu do dané třídy. [21, Kapitola 8]

Druhá fáze je testovací. V této fázi vybereme vzorky dat, u kterých víme, do které třídy je zařadit a neměly by být vybrány z trénovací množiny. Tato množina dat se nazývá trénovací množina. Data se poté zařazují do tříd pomocí klasifikátoru, který už ví z předchozí

fáze, do jaké třídy data zařadit. Po zařazení dat se procentově určí, v kolika případech byla provedena správná klasifikace a poté se rozhodne, jestli může být klasifikátor běžně využíván v praxi. [21, Kapitola 8]

Metody Klasifikace

- Rozhodovací strom
- Bayesovská klasifikace
- Klasifikace pomocí neuronových sítí

Predikce

Predikce je proces, pomocí kterého lze přiřazovat hodnoty datům, které mají spojitý charakter. Jako příklad lze uvést určení platu pracovníka na základě jeho aktuálního platu. [21, Kapitola 8]

Metody predikce

- Lineární jednoduchá regrese
- Lineární vícenásobná regrese
- Nelineární regrese

3.4 Dolování asociačních pravidel a frekventovaných množin

Asociační pravidla nám ukazují vztahy mezi položkami ve vzorku dat. Můžeme s jejich pomocí například určit, jaké položky si v obchodě často kupují zákazníci společně a na základě tohoto zjištění může prodejce vytvářet různé akční nabídky nebo mít vystavené tyto produkty poblíž sebe.

Při získávání znalostí dat existují dvě metriky, těmi jsou podpora a spolehlivost. Podpora určuje v kolika procentech byly dvě položky koupeny společně v rámci jedné transakce. Druhá z nich je spolehlivost, která určuje procento zákazníků, kteří si někdy koupili položku a poté si v rámci jiné transakce koupili druhou.

Při získávání asociačních pravidel jsou důležité 2 kroky. Prvním z nich je nalezení frekventovaných množin, což jsou ty položky, které splňují podmínku minimální podpory. Druhým krokem je poté již generování silných asociačních pravidel z dříve nalezených frekventovaných množin. U těchto pravidel musí být splněna podmínka minimální podpory a také podmínka minimální spolehlivosti. [21, Kapitola 6]

Typy asociačních pravidel

Existuje několik různých typů asociačních pravidel. Samotná asociační pravidla se klasifikují podle několika kritérií: [21, Kapitola 6]

Podle typu hodnot v pravidlech – podle tohoto kritéria rozlišujeme booleovská a kvantitativní asociační pravidla. O booleovská asociační pravidla se jedná, pokud se zajímáme pouze o přítomnost nebo nepřítomnost položky, naopak o kvantitativní asociační pravidla se jedná pokud tyto pravidla popisují asociace mezi kvantitativními položkami nebo atributy. [21, Kapitola 6]

Podle dimenzí obsažených v pravidlech – asociační pravidla se dají rozdělit podle počtu obsažených dimenzí. Jednodimenzionální asociační pravidla jsou booleovská pravidla, protože obsahují pouze jednu dimenzi, konkrétně v již zmíněném příkladu dimenzi, která určuje, zda zboží bylo zakoupeno či nikoliv. Vícedimenzionální pravidla jsou pravidla kvantitativní, protože kromě dimenze, zda bylo zboží zakoupeno, obsahují také další dimenze.

Podle úrovní abstrakce – existují víceúrovňová asociační pravidla. Některé metody mohou získávat asociační pravidla nad těmito pravidly z různých úrovní abstrakce. [21, Kapitola 6]

Algoritmus Apriori

Apriori je nejjednodušší varianta získávání asociačních pravidel. Apriori doluje jednoúrovňová asociační pravidla, tedy zjišťuje pouze jednu dimenzi, konkrétně zda se položka vyskytuje v transakci nebo ne. Tento algoritmus využívá předchozí znalost o již získaných frekventovaných množinách. V každé iteraci jsou získané frekventované k-množiny použity k vygenerování (k+1) množin, je tedy nutné projít databázi v každé iteraci, z tohoto důvodu se nejedná o nejrychlejší metodu.

Pro vyšší efektivitu je využívána vlastnost Apriori, která říká, že každá podmnožina frekventované množiny musí být frekventovaná také. Vlastnost Apriori vychází z faktu, že přidání prvku do množiny nemůže způsobit vzrůst její podpory. [21, Kapitola 6]

Algoritmus Apriori pracuje následujícím způsobem: [21, Kapitola 6]

1. Při první iteraci projdeme celou databázi a spočítáme podporu pro každou položku. Množina C_1 bude obsahovat všechny tyto položky.
2. Určíme minimální podporu, poté nalezneme množinu L_1, ta bude tvořena položkami C_1, které splňují minimální podporu.
3. Dalším krokem je vygenerování C_2 při spojení množin z L_1.
4. Vypočítá se podpora kandidátů z C_2 a odstraní se kandidáti, kteří nesplňují minimální podporu. Výsledkem je množina L_2.
5. Vygeneruje se C_3 z množiny L_2 tím, že se vytvoří množiny po třech prvcích. Z C_3 můžeme odstranit podmnožiny, které nejsou frekventované v L_2.
6. Tímto způsobem pokračujeme dál a zvyšujeme počty prvků v množinách C_k. Jakmile je nějaká množina C_k prázdná, můžeme algoritmus ukončit a výsledkem je množina L_{k-1}.

Metoda FP-stromu

Algoritmus Apriori může mít problém s generováním kandidátů u velkých databází, protože obsahují velké množství dat. Tento problém se dá řešit metodou vzrůstu frekventovaných množin, u které se nejdříve databáze zkomprimuje do struktury nazvané strom frekventovaných množin (FP-strom). Poté se strom rozdělí do podmíněných FP-stromů, ty jsou vytvořeny pro všechny frekventované položky. Z těchto stromů poté získáme frekventované množiny. Efektivitu algoritmu Apriori mohou také zvýšit další úpravy, například: hashovací přístup, redukce transakcí, vzorkování nebo rozdělení dat. [21, Kapitola 6]

3.5 Shluková analýza

Shlukování rozděluje objekty na základě jejich podobnosti do tříd (clusterů). Výsledkem jsou třídy, které obsahují velice podobné objekty a jsou velice odlišné od objektů z jiných tříd. K určení podobnosti objektů se často využívají vzdálenostní funkce. [21, Kapitola 10]

Shlukovací metody mají mnoho různých vlastností, většinou jsou na ně kladeny tyto požadavky: [21, Kapitola 10]

- Škálovatelnost
- Schopnost zpracovávat různé typy atributů
- Tvorba shluků různého tvaru
- Vyrovnání se s šumem

Je mnoho druhů shlukovacích metod. Konkrétní shlukovací metoda se zvolí na základě typu analyzovaných dat a na konkrétním účelu aplikace. Shlukovací metody se dělí následovně: [21, Kapitola 10]

- Metody založené na rozdělování – metody, které rozdělují objekty do tříd, přičemž každá třída musí obsahovat alespoň jeden objekt a každý objekt může patřit pouze do jedné třídy.
- Hierarchické metody – metody vytvářející hierarchický rozklad dané množiny objektů, vzniká tak strom shluků.
- Metody založené na hustotě – metody, které považují za shluky oblasti s velkou hustotou objektů, které jsou odděleny oblastmi s malou hustotou výskytu objektů (šum) v datovém prostoru.
- Metody založené na mřížce – tyto metody využívají datovou strukturu v podobě víceúrovňové mřížky. Prostor objektů rozdělují na konečný počet buněk.
- Metody založené na modelech – tyto metody se snaží optimalizovat shodu mezi matematickým vzorcem a datovou množinou. Snaží se tedy nalézt shluky, které co nejvíce odpovídají danému modelu.

3.6 Datové sklady a OLAP technologie

V této podkapitole jsou stručně popsány datové sklady a jejich porovnání s relačními databázemi. Dále jsou zde popsány základy technologie OLAP, schémata OLAP a operace.

Datový sklad

Datový sklad je zvláštní typ databáze, který primárně slouží pro analýzu dat v oblasti, kde slouží jako podklady pro manažerské rozhodování [39]. Datový sklad může být zkonstruován poté, co jsou data očištěna a integrována. Jedná se o velmi důležitý krok v předzpracování, proto aby později mohlo být uskutečněno získávání znalostí. [21, Kapitola 4]

Rozdíl mezi běžnou relační databází a datovým skladem

V relační databázi se obvykle snažíme o co nejmenší redundanci ukládaných dat. Redundance dosahujeme vnitřním provázáním jednotlivých funkčních celků nebo normalizací. Datový sklad se na rozdíl od relační databáze snaží o jasnou vnitřní separaci jednotlivých funkčních celků, jejíž výsledkem je poté struktura, kterou mohou uživatelé lépe číst, avšak za cenu zvýšených nároků na paměťový prostor. Pokud popisujeme strukturu datového skladu, popisujeme ji jako multidimenzionální strukturu uložených dat. [39]

Multidimenzionální datový model

Data, nad kterými jsou prováděny OLAP operace jsou zobrazeny ve formě datové kostky, ta nám umožňuje data vnímat jako vícerozměrná. Datová kostka je definována dimenzemi a fakty.

Dimenze mohou být entity nebo pohledy. Záznamy v nějakém obchodě mohou být ukládány vzhledem k prodávanému zboží, místě prodeje, dodavateli a času, kdy byly prodány. Každé této dimenzi budou přiřazeny tabulky dimenzí, které tyto dimenze popisují. Například tabulka dimenze dodavatele může mít tyto atributy: jméno, adresa, telefon a další.

Dále jsou v multidimenzionálním datovém modelu tabulky faktů. Fakta jsou numerické měrné jednotky, představují množství, na jehož základě poté analyzujeme vztahy mezi dimenzemi. Kromě fakt obsahuje tabulka faktů také cizí klíče do dimenzí. [21, Kapitola 4]

Schémata představující dimenzionální databáze

Narozdíl od OLTP systémů, kde se pro jejich modelování využívají ER-diagramy se u datových skladů používají multidimenzionální modely. [21, Kapitola 4]

- Schéma hvězdy – jedná se o nejběžnější schéma. V tomto schématu se nachází tabulka faktů, což je centrální tabulka obsahující velké množství dat bez redundance. Kromě této centrální tabulky je v tomto schématu množina tabulek dimenzí a každá z nich uchovává v rámci atributů informace o jedné dimenzi.

- Schéma sněhové vločky – toto schéma je obdobné jako schéma hvězdy s tím rozdílem, že jsou některé tabulky dimenzí normalizovány, což znamená, že jsou vytvořeny nové tabulky. Největší výhodou tohoto schématu je velká úspora místa z důvodu snížení redundance. Avšak nejvíce prostoru zabere samotná tabulka faktů, což znamená, že celková úspora není tak velká, jak se původně mohlo zdát. Toto Schéma se příliš v praxi nevyužívá z důvodu náročnosti vykonávání dotazů. Mezi tabulkami je totiž nutné provést mnoho spojení.

- Schéma souhvězdí – toto schéma se využívá, pokud aplikace vyžaduje více než pouze jednu tabulku faktů, které sdílejí některé tabulky dimenzí. Tento model můžeme vnímat jako kolekci hvězd.

OLAP operace v multidimenzionálních datech

Na multidimenzionální data můžeme pohlížet hned několika různými pohledy, z důvodu, že každá dimenze obsahuje hned několik úrovní abstrakcí. Abychom těmito pohledy mohli na data nahlížet, nejběžnější operace jsou uvedeny zde: [21, Kapitola 4]

Roll-Up

Tato operace provede na datové kostce agregaci. Jinými slovy se v jedné z hierarchií posuneme o úroveň výše. Například data v kostce po provedení této operace nebudou seskupena podle čtvrtletí, ale podle celých let.

Drill-down

Tato operace je pravým opakem operace roll-up. Po provedení této operace se posuneme v jedné hierarchii o úroveň níže a dostaneme tak detailnější pohled.

Slice & Dice

Operace Slice & Dice provede nad jednou dimenzí selekci dat. Výsledkem této operace je podkostka s daty splňující nějakou podmínku.

Rotate

Operace Rotate (někdy nazývaná Pivot) je vizualizační operace. Mění datové osy za účelem jiné prezentace dat kostky.

Kapitola 4

Analýza a požadavky na systém

První krok, který je zapotřebí uskutečnit při tvorbě informačního systému je krok, který předchází návrhu systému. Jedná se o důkladnou specifikaci požadavků od zákazníka, pro kterého informační systém vytváříme. Musíme přesně zjistit, kdo všechno bude se systémem pracovat, a jaké akce bude chtít v systému vykonávat. Naším úkolem je tyto požadavky analyzovat a zjistit, zda je vůbec možné všem těmto požadavkům vyhovět. Pokud ne, je zapotřebí se zákazníkem najít určitý kompromis, který je možné implementovat.

V této kapitole jsou popsány požadavky na informační systém od zákazníka. Informační systém budou používat uživatelé, kteří budou mít jednu ze tří rolí: admin, zákazník, zaměstnanec.

4.1 Zákazník

Jako zákazník je chápána jakákoliv firma, která se zaregistruje do informačního systému. Firma se může zaregistrovat sama, nebo může požádat admina a ten firmu zaregistruje sám. Firma při registraci zadá svoje údaje do registračního formuláře, je nutné zadat login a heslo pomocí něhož se bude později do systému přihlašovat. Po založení účtu si může každá firma zadat do systému neomezený počet kontaktních osob, které mohou zaměstnanci nebo admin poté kontaktovat pokud bude třeba. Zákazník si poté může své kontaktní osoby zobrazit a také měnit jejich údaje, nebo je úplně odstranit. Kontaktní osobu může k firmě přidat také admin.

Zákazník si po registraci může libovolně měnit svoje údaje, které zadal při registraci a také si může změnit heslo. Tuto možnost má také admin, pokud by zákazník zapomněl svoje heslo, může požádat admina a ten mu heslo změní.

4.2 Zaměstnanec

Účet zaměstnance je vytvořen adminem poté, co je zaměstnanec přijat do firmy. Poté se zaměstnanec do systému přihlašuje pomocí svého emailu a hesla, které byly zadány při registraci. Zaměstnanec si pak může heslo a své ostatní údaje libovolně měnit, pokud je potřeba. Každý zaměstnanec má ve firmě konkrétní pozici, na kterou byl přijat. Podle této pozice je přidělen na oddělení a je mu přidělena funkce. V systému jsou zaměstnanci, kteří mohou mít jednu z následujících funkcí:

- Míchač

- Skladník
- Balič

Podle určité funkce, kterou zaměstnanec ve firmě zastává poté provádí práci, avšak v systému je umožněno, aby například míchač mohl naskladnit zboží v případě, že by zaměstnanec musel nahradit jiného zaměstnance z důvodu jeho absence.

4.3 Manipulace s objednávkami

Objednávku může vytvořit zákazník, nebo ji pro konkrétního zákazníka může vytvořit admin. Každá objednávka má termín, do kterého má být vyřízena. Po založení objednávky je automaticky termín nastaven na datum, které je za týden od data vytvoření objednávky. Tento termín poté zákazník může změnit. Termíny objednávek se automaticky objeví v Google kalendáři jako celodenní událost. Ta nese název, ve kterém je obsaženo ID objednávky. Pokud zákazník nebo admin tento termín změní, nebo je objednávka odstraněna, projeví se tato změna také v Google kalendáři. Pokud by zákazník zvolil termín, ve kterém není možné objednávku vyřídit, ozval by se mu zaměstnanec a dohodl se s ním na kompromisu. Objednávku může zákazník nebo admin měnit a také ji může úplně odstranit.

Každá objednávka je v nějakém stavu, objednávky v tomto systému mohou mít jeden z následujících stavů:

- Založeno
- Namícháno
- Zabaleno
- Vyřízeno

Po založení je objednávka automaticky ve stavu založeno. Stav objednávky dále mění zaměstnanci firmy, tento stav poté mohou sledovat zákazníci a mohou tak odhadnout, za jak dlouho objednávka bude přibližně připravena.

Admin a zaměstnanec si mohou zobrazit všechny objednávky, vidí jejich informace a po rozkliknutí vidí jejich obsah. Zákazník si může zobrazit pouze svoje objednávky, tzn. ty, které si sám vytvořil nebo mu ji vytvořil admin.

Práce na objednávce

U každé objednávky je umožněno zaměstnanci označit práci na objednávce, kterou provedl. Zaměstnanec může označit dva druhy práce na objednávce: míchání a zabalení. Zaměstnanec má po přihlášení přístup k přehledu všech svých vykonaných činností na objednávkách, ale nevidí provedenou práci ostatních, k tomu má přístup pouze admin.

4.4 Položky v objednávce

Každá objednávka se skládá z několika položek. Položky do objednávky může přidávat admin, nebo zákazník, který objednávku vytvořil. Položky lze různě upravovat, měnit jejich množství nebo je úplně odstranit. U každé položky je možné vybrat konkrétní balení.

4.5 Produkty

Položky, které se do objednávky přidávají jsou produkty. V systému existují dva základní druhy produktů, konkrétně originální a míchané. Každý produkt je identifikován jednoznačným kódem, pomocí jehož prvního písmena poznáme zda se jedná o originální produkt (kód začíná znakem „O“) nebo míchaný produkt (kód začíná znakem „M“). Produkty do systému přidává pouze admin a také jedině on je může upravovat nebo mazat. Zákazníci a zaměstnanci si mohou zobrazit jejich seznam. Produktů je na skladě logicky omezené množství, toto množství při přidání produktu do systému zadá v případě originálního produktu admin, poté ho mohou doplňovat zaměstnanci nebo také znovu admin. Míchané produkty se naskladní až po vložení produktu do systému, protože při každém naskladnění míchaného produktu jsou automaticky odečteny produkty, které jsou součástí receptu, pro namíchání. Každý produkt spadá do jednoho odvětví, v systému mohou produkty spadat do těchto odvětví:

- Barva
- Lak
- Tmel
- Mořidlo

Originální produkty

Originální produkty jsou produkty, které jsou dodány dodavatelem do firmy a jsou pouze dále přeprodávány ve stejné podobě jako byly doručeny. Tyto produkty do systému přidává admin, který je také potom může měnit, nebo úplně odstranit ze systému.

Každý originální produkt je dodán jedním dodavatelem. Firma odebírá od několika dodavatelů, o kterých jsou v systému vedeny informace, tyto dodavatele do systému opět přidává, mění a odstraňuje pouze admin.

Recepty míchaných produktů

Ve firmě se prodávají míchané produkty. To jsou produkty, které firma sama připravuje z originálních produktů, které jim dodal jiný dodavatel. Každý tento produkt má určený recept, podle kterého se produkt míchá. Tento recept zadává do systému pouze admin, také pouze on tento recept může upravovat. Zaměstnanci si tyto recepty zobrazí a podle nich míchají produkty.

Balení produktů

Každý produkt objednávky musí být nějak zabalen, aby byl bezpečně doručen. Pokud zákazník neurčí jak chce položku zabalit, zaměstnanec položku zabalí podle vlastního uvážení. Pokud chce zákazník zabalit pokožku podle svých konkrétních požadavků, vybere si v systému z nabízených nádob a také vybere jejich počet. Zákazníkovi je dokonce umožněno nechat si jednu položku zabalit do více různých nádob.

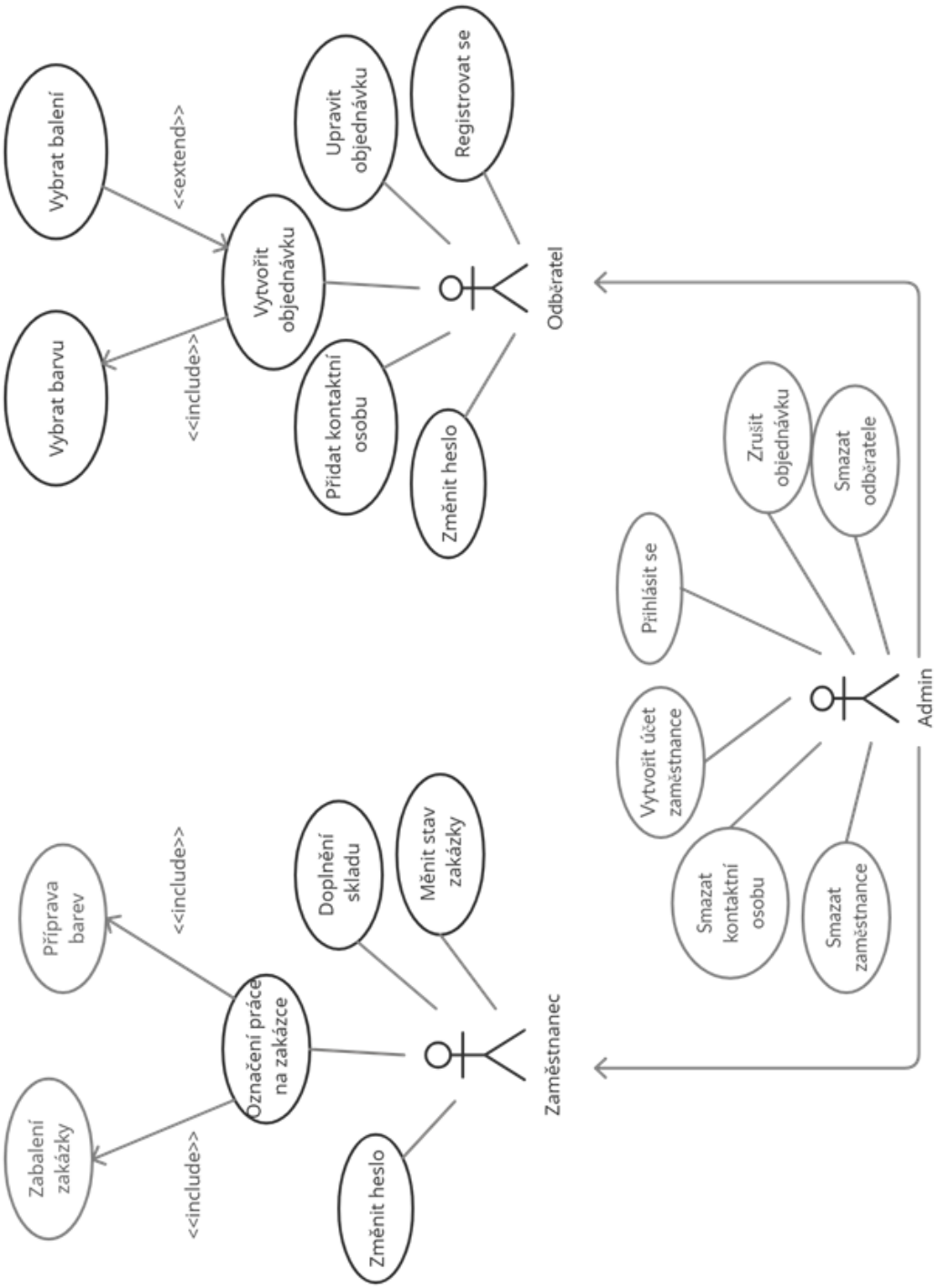
V systému jsou dva typy nádob, těmi jsou: kanystry a plechovky. Každá nádoba má určený objem v litrech. Tyto nádoby do systému zadává, mění a odstraňuje pouze admin. Stejně jako u produktů má každá tato nádoba omezený počet kusů na skladě a je tedy nutné, aby ji zaměstnanec nebo admin naskladňovali.

4.6 Admin

Admin je nejdůležitější rolí systému. V systému může být více adminů, nové adminy může přidávat pouze jiný admin. Po vytvoření přístupu se admin přihlašuje pomocí emailu a hesla. Adminovi je umožněno provádět veškeré činnosti, které mohou provádět zaměstnanci a zákazníci. Navíc mohou přidávat do systému nové produkty, balení, oddělení nebo registrovat nové dodavatele, kteří do firmy dodávají produkty.

4.7 Diagram případů užití

Diagram případů užití (Use Case Diagram) ukazuje chování systému z pohledu samotného uživatele. Jeho hlavním účelem je popis funkcionality systému, tedy to, co od něj klient očekává. Use Case Diagram popisuje funkce, které bude moci uživatel provádět v informačním systému, ale nepopisuje jak přesně budou funkce implementovány. To je také důvod, proč se při návrhu systému nejdříve vytváří tento diagram. Součástí diagramu případů užití jsou užití (use case), aktéři (actors) a vztahy mezi nimi. [43]



Obrázek 4.1: Diagram případů užití.

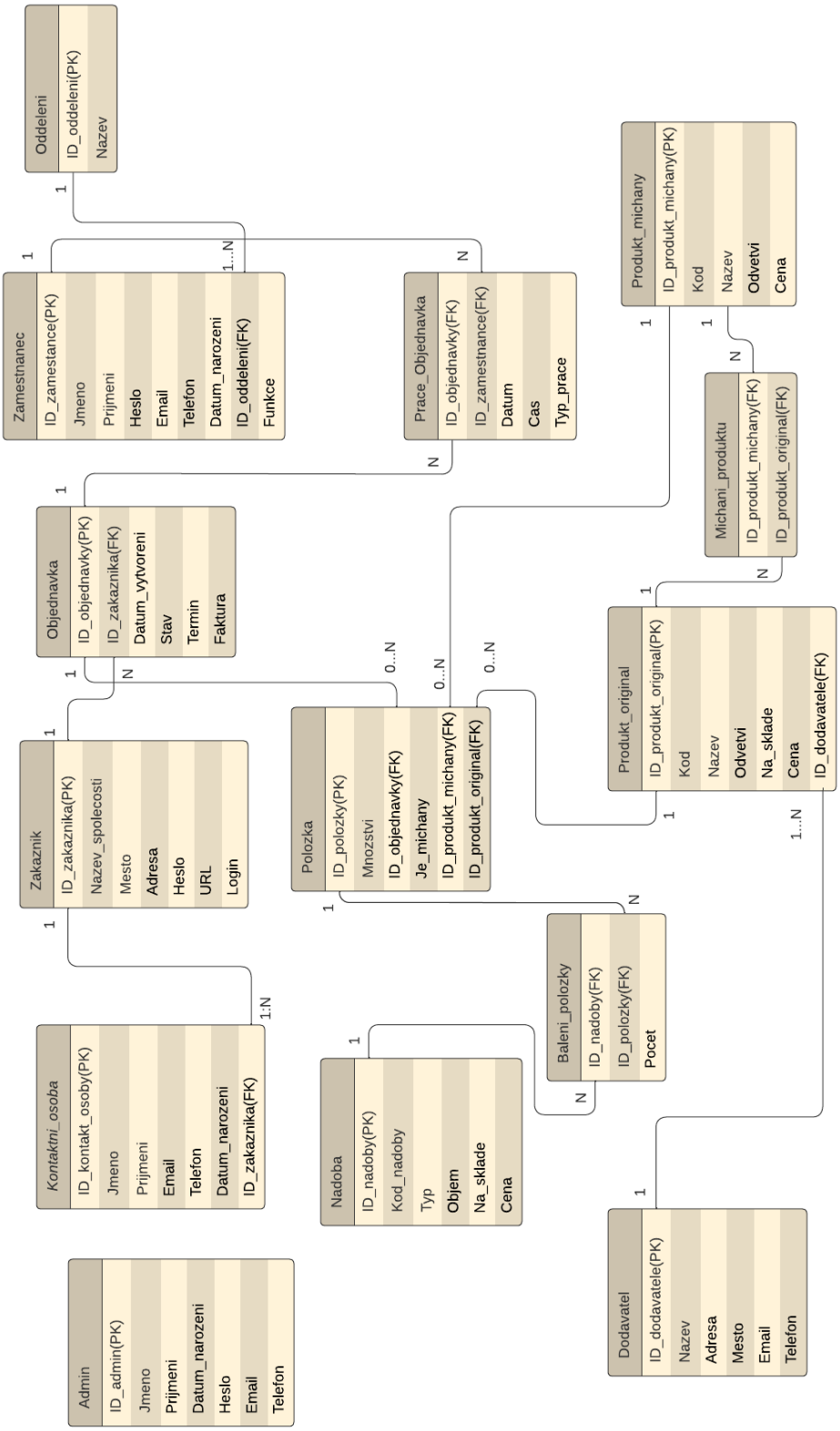
Kapitola 5

Návrh systému

Poté co jsme získali konkrétní požadavky na systém od zákazníka, musíme před implementací nejdříve provést návrh informačního systému. Nejdříve je popsán návrh databáze, který je následně převeden do tabulek relační databáze. Následuje architektonický návrh a návrh uživatelského rozhraní.

5.1 ER diagram

ER diagram (Entity Relationship diagram) je grafický nástroj, který používáme pro návrh databáze. O entitě hovoříme jako o určitém aspektu reálného světa, může to být fyzicky existující objekt jako je například zaměstnanec, výrobek, nebo také událost jako je návštěva lékaře nebo servis automobilu. Součástí každé entitní množiny jsou určité atributy, sloužící k popisu vlastností entitní množiny, například u entity zaměstnanec mohou být atributy: rodné číslo, jméno, příjmení, datum narození a další. Dalším velice důležitým prvkem v ER diagramu jsou vztahy mezi etitními množinami. Vztah se dá chápat jako spojení mezi několika entitami. Spojujeme entity, které spolu logicky souvisí. [7] Příklad vztahu může být například žák a třída, protože žák navštěvuje třídu a do třídy chodí několik žáků, proto je potřeba mezi těmito entitními množinami v návrhu vztah znázornit.



Obrázek 5.1: ER diagram.

Admin

Admin je entitní množina, která ukládá informace o správcích systému. V tabulce admin jsou atributy, které určují kontaktní údaje správce, těmi jsou email nebo telefon, oba tyto atributy musí být unikátní. V tabulce je také atribut `Heslo`, který se zadává při registraci a pomocí něj a emailu se poté správce do systému přihlašuje.

Zákazník

Entitní množina zákazník nese údaje o všech zákaznících, kteří jsou v systému registrováni a zároveň mají automaticky přidělenou roli zákazníka v systému. V tabulce jsou atributy `Login` a `Heslo`, pomocí kterých se zákazník do systému přihlašuje. Dále jsou v tabulce atributy určující polohu firmy (`Adresa` a `Mesto`) a také `URL`, který ukládá url adresu firmy, tento atribut však není povinný.

Kontaktní osoba

Tato entitní množina slouží pro uchovávání kontaktních osob zákazníků. O kontaktních osobách uchováváme informace nutné pro kontaktování osoby, jako jsou email a telefon, dále jméno, příjmení a datum narození. V této entitní množině je cizí klíč odkazující na tabulku `Zakaznik`, který určuje jakou firmu osoba zastupuje. Každá firma může mít více kontaktních osob. Vztah mezi zákazníkem a kontaktními osobami je 1:N, protože zákazník může mít více kontaktních osob, ale kontaktní osoba zastupuje pouze jednu firmu.

Zaměstnanec

Entitní množina zaměstnanec ukládá údaje o zaměstnancích firmy. Každý zaměstnanec se může přihlásit pomocí emailu a hesla. Dále jsou o zaměstnanci uchovávány v tabulce atributy, sloužící pro jeho kontaktování (`email` a `telefon`), datum narození a funkce, který určuje jakou má pracovník ve firmě funkci. Každý zaměstnanec má v tabulce cizí klíč `ID_Oddeleni`, který určuje na kterém oddělení pracuje.

Oddělení

Entitní množina, která ukládá údaje o odděleních ve firmě. Každý zaměstnanec patří do jednoho oddělení a oddělení má více zaměstnanců, jedná se tedy o vtah 1:N. Tato tabulka má kromě primárního klíče pouze jeden atribut, který určuje název oddělení (`nazev`).

Objednávka

Entitní množina uchovávající informace o konkrétní objednávce, kterou vytvoří odběratel. Tabulka uchovává atributy data vytvoření objednávky (`Datum_vytvoreni`), stav objednávky, ve kterém se právě objednávka nachází(`stav`) a termín, do kdy je potřeba objednávku připravit (`termin`).

Entitní množina `Prace_objednavka` ukládá údaje o provedené práci na objednávce. Uchovává cizí klíče, které označují zaměstnance (`ID_zamestnance`), který práci provedl a na jaké zakázce pracoval (`ID_objednavky`). Dále obsahuje atributy určující čas, kdy práci provedl, o jaký typ práce se jedná a také kolik hodin práce trvala.

Položka

Entitní množina `Polozka` ukládá informace o jednotlivých položkách objednávky. Obsahuje cizí klíč `ID_objednavky`, který určuje k jaké objednávce položka patří, dále obsahuje atribut `mnozstvi`, který určuje množství v litrech. V tabulce je aribut `je_michany`, který určuje, zda se jedná o míchaný produkt, který se míchá ve firmě, nebo o originální produkt. Podle toho je konkrétní položka propojena s tabulkou `Produkt_original` nebo `Produkt_michany`.

Nádoba

Tato entitní množina ukládá údaje o nádobách, ve kterých je možno produkty dodávat. Nese údaje o typu nádoby (`typ`), tedy jestli se jedná o kanystr nebo plechovku. Dále obsahuje atributy, které určují jeho cenu, objem a počet kusů na skladě.

Balení položky

Tabulka `Baleni_polozky` je propojovací tabulka, která nese informace o tom, jak bude konkrétní položka zabalena. V tabulce jsou cizí klíče `ID_nadoby` určující konkrétní nádobu a `ID_polozky`, který určuje o jakou položku se jedná. Také je zde atribut `pocet`, který určuje, kolik těchto nádob bude pro položku použito.

Originální produkt

Entitní množina `Produkt_original` obsahuje informace o produktech, které jsou dodány do firmy jiným dodavatelem a rovnou jsou přeprodávány zákazníkům bez jakýchkoliv úprav. Každý produkt má svůj unikátní kód, dále jsou ukládány informace o tom, do jakého odvětví produkt patří, cena produktu a počet litrů na skladě. Dále uchovává cizí klíč určující výrobce (`ID_dodavatele`), který produkt dodal.

Dodavatel

Entitní množina uchovávající údaje o dodavatelích originálních produktů. Obsahuje adresu, kde dodavatel sídlí a také kontakty jako je `email` a `telefon`. Mezi tabulkou `Dodavatel` a `Produkt_original` je vztah 1:N, protože dodavatel může dodávat více originálních produktů, ale produkt může být dodán pouze jedním dodavatelem.

Míchaný produkt

Tato entitní množina reprezentuje produkty, které jsou míchány přímo ve firmě. Nejčastěji se jedná o barvy, které vznikly smícháním originálních produktů. Každý originální i míchaný produkt má svůj unikátní kód, dále jsou zde uchovány atributy určující cenu za litr nebo odvětví.

Míchání produktů

Entitní množina, která uchovává informace o míchání barev. V tabulce jsou cizí klíče odkazující na originální a míchané barvy.

5.2 MVC architektura

MVC je v dnešní době velice oblíbený architektonický vzor, jehož hlavní myšlenkou je oddělení logiky od výstupu, čímž se aplikace stává znatelně přehlednější. Původně vznikl pro desktop aplikace, ale v dnešní době je využíván zejména pro webové aplikace. Tento model je využíván u mnoha současně oblíbených frameworků jako je již zmíněný Laravel, dále Nette nebo ASP .NET. Architektura MVC rozděluje aplikace do tří komponent, jimiž jsou: [31]

Model

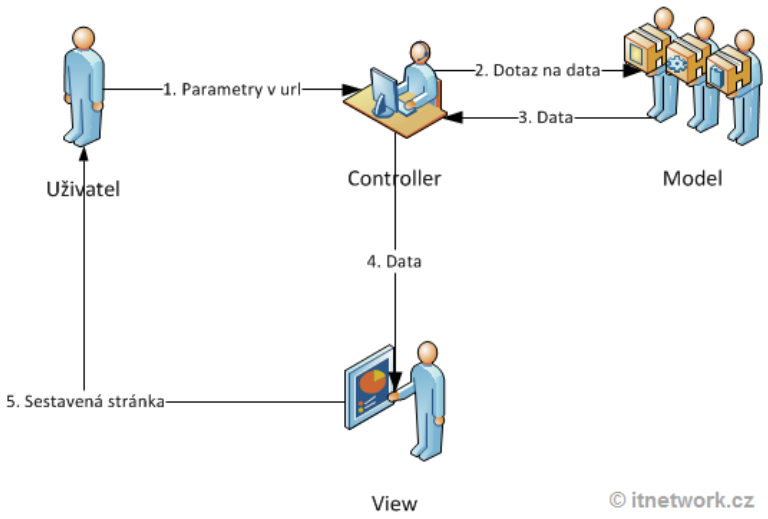
Součástí modelu je logika a vše, co se jí týká. Nejčastěji to bývají databázové dotazy nebo výpočty. Funkce modelu spočívá v přijetí vstupních parametrů zvenku a následnému dodání dat ven. Model se nestará o to, odkud data přišla a ani neví v jakém formátu budou výstupní data, tedy vůbec neví o Controlleru a View. [31]

View

View (pohled) zajišťuje zobrazení formátovaných dat uživateli. K zobrazení jsou většinou využívány šablonovací systémy, ve kterých je možno provádět iterace a podmínky. View neví o ostatních komponentech. Neví odkud mu data přišla, pouze je zobrazí uživateli ve správném formátu. [31]

Conroller

Jelikož Model neví o View a naopak, je tedy potřeba zajistit jejich komunikaci. O to se stará právě Controller. Ten komunikuje nejen s View a Modelem, ale také s uživatelem a drží tak celý systém pohromadě. [31]



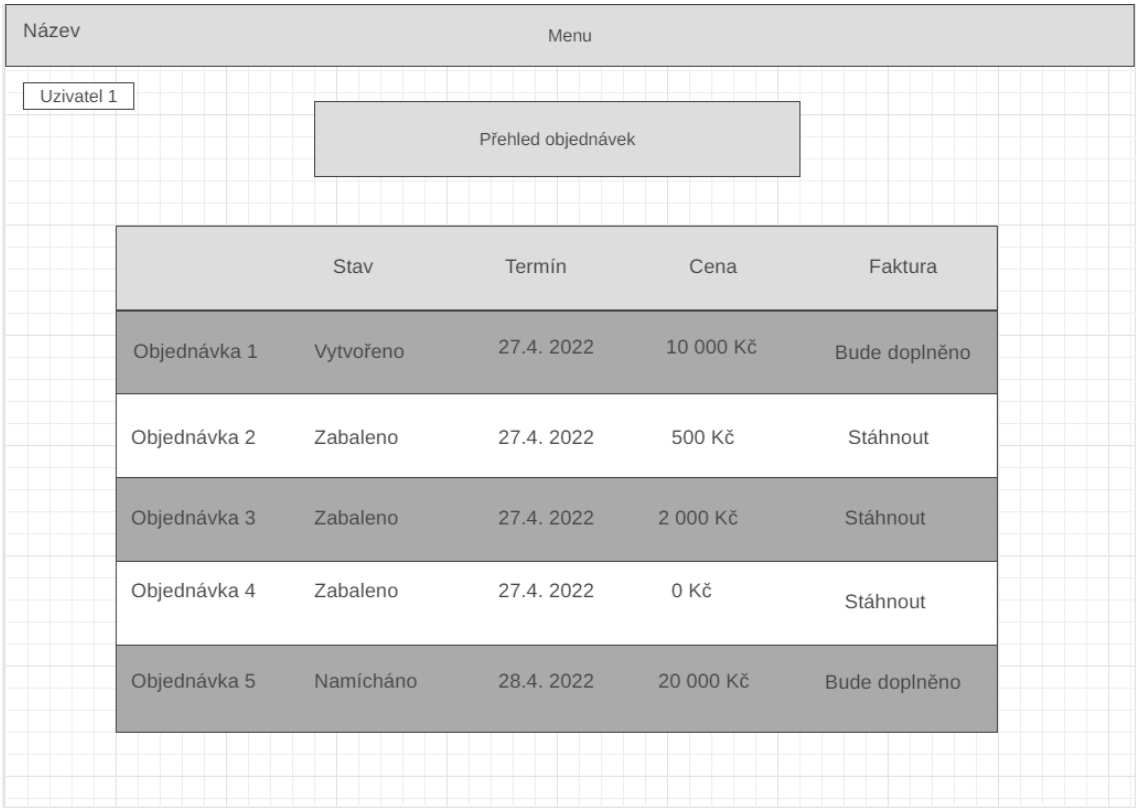
Obrázek 5.2: MVC architektura. Převzato z: [32].

5.3 Návrh uživatelského rozhraní

Důležitou součástí informačního systému je uživatelské rozhraní. Při návrhu uživatelského rozhraní je velice důležité, aby bylo přehledné a uživatelsky přívětivé. Je vhodné udělat si hrubý návrh uživatelského rozhraní ještě před implementací, aby jsme se při samotné implementaci nemuseli soustředit na to, co vše musíme do uživatelského rozhraní zahrnout.

Návrh stránky s objednávkami

Pro ukázku jsem si vybral stránku, která bude obsahovat přehled objednávek. Tato stránka je pro uživatele velice důležitá. Součástí informačního systému bude menu, které bude k dispozici ze všech sekcí systému po přihlášené uživatele. Veškeré objednávky budou zobrazeny v tabulce. U každé objednávky budou zobrazeny její vlastnosti, například cena, termín, stav a další. Po rozkliknutí čísla objednávky se do ní budou přidávat položky.



Obrázek 5.3: Návrh šablony pro přehled objednávek.

Kapitola 6

Použité technologie

V této kapitole jsou rozepsány technologie, které byly použity při tvorbě této práci. Nejdříve je popsán PHP framework Laravel a je porovnán s dalšími frameworky. Poté je popsáno Google Calendar API a na závěr kapitoly jsou popsány front-end technologie.

6.1 Technologie použité při tvorbě Back-endu

V této podkapitole se věnuji back-end technologiím, konkrétně PHP frameworkům a jejich vzájemnému porovnání.

Laravel

Laravel[1] je PHP framework, který je poskytován zdarma jako open source. V současné době patří k nejpoužívanějším frameworkům. Poprvé byl vydán v roce 2011, vytvořil jej Taylor Otwell a je založen na frameworku Symfony, avšak výhodou je, že není nutná znalost Symfony, k tomu aby mohl vývojář pracovat s Laravelem. Webové aplikace vytvořené pomocí Laravelu pracují na architektuře model-view-controller (MVC). [29] U Laravelu je možnost psát PHP konstrukce přímo do HTML kódu, což nám usnadní mnoho práce a výsledný kód je přehlednější. To umožňuje šablonovací systém blade, který použijeme tak, že za soubor přidáme příponu `.blade.php`. K instalaci a založení projektu v Laravelu se používá nástroj Composer. Součástí Laravelu je také Artisan.

Composer

Composer[2] je nástroj nutný pro založení projektu v Laravelu, protože přes něj probíhá instalace. Composer při instalaci stáhne vše, co je potřeba pro správné fungování Laravelu. Za pomoci composeru můžeme později snadno stahovat a instalovat další knihovny. [12]

Artisan

Součástí Laravelu je konzolová aplikace Artisan[3], která je velmi užitečná a usnadní mnoho práce s vývojem. Artisan se používá pomocí příkazu `php artisan`, poté se nám zobrazí všechny dostupné příkazy. Pomocí artisanu můžeme vytvářet soubory, jako jsou například

[1] Dostupné z: https://laravel.com/.
[2] Dostupné z: https://getcomposer.org/.
[3] Dostupné z: https://laravel.com/docs/9.x/artisan.

Modely nebo Controllery, k vytvoření souboru slouží příkaz: `php artisan make`. Dále pomocí Artisanu probíhají migrace databáze a umožňuje mnoho dalších funkcí, jako například generování registračního a přihlašovacího formuláře. [2]

Další PHP frameworky

V této sekci jsou popsány další známé PHP frameworky a jsou porovnány s Laravelem.

Symfony

Dalším frameworkem pro tvorbu webových aplikací v PHP je Symfony[4]. Jedná se o open-source framework, který je inspirován například frameworkem Spring nebo Django a sám se stal inspirací pro framework Laravel, který je mu velice podobný. První verze byla vydána v roce 2005 a stále přicházejí nové inovace, dnes patří k nejpoužívanějším frameworkům na světě. Symfony je stejně jako Laravel postaveno na MVC architektuře. Symfony využívá šablonovací systém Twig, jedná se o obdobu blade u frameworku Laravel. [40]

Rozdíly mezi Symfony a Laravelem: [30]

- Laravel je oproti Symfony využíván spíše pro menší projekty, u kterých je důležitá rychlá a jednoduchá implementace a nejsou vyžadovány nestandardní funkce.
- Symfony je vhodnější pro projekty, u kterých je důležitá dlouhodobá údržba.
- Laravel má v porovnání se Symfony poměrně snadnou migraci databází a jednoduchou konfiguraci.

Nette

Nette[5] je, stejně jako Laravel, framework pro tvorbu webových aplikací v PHP. Nette má své kořeny v Česku, původním autorem je David Grudl, z tohoto důvodu je u nás Nette velice rozšířené, ale po světě příliš rozšířené není. Stejně jako Laravel také Nette používá MVC architekturu.

Rozdíly mezi Nette a Laravelem

- Nette klade větší důraz na bezpečnost výsledné aplikace, obsahuje totiž nástroj, který automaticky generovaný kód ošetřuje od nejčastějších bezpečnostních rizik.
- Při práci s Nette trvá vývoj webových aplikací delší dobu než u Laravelu.
- Nette má oproti Laravelu velmi jednoduchou dokumentaci, což je velká nevýhoda.

CodeIgniter

Codelgniter[6] je další PHP framework, jehož původní vydání bylo v roce 2006. Codelgniter má výhodu v podobě rychlé přímočaré instalace, u které je vyžadováno jenom minimum konfigurace. Při jeho používání nedochází ke konfliktům kvůli verzím PHP, protože funguje

[4] Dostupné z: https://symfony.com/.
[5] Dostupné z: https://nette.org/cs/.
[6] Dostupné z: https://www.codeigniter.com/.

dobře téměř na všech hostovacích platformách. Oproti Laravelu a mnoho dalších frameworků není striktně založen na MVC architektuře. Je sice nutné používat Controllery, ale modely a pohledy povinné nejsou, můžeme tak používat své vlastní jmenné a kódovací konvence. CodeIgniter tedy vývojářům při vývoji webových aplikací umožňuje značnou volnost, což může být považováno za velkou výhodu. [42]

Rozdíly mezi Laravelem a CodeIgniterem [11]

- Laravel je relačně objektově orientovaný, zatímco Codeigniter je pouze objektově orientovaný
- Laravel přináší funkce ověřování, zatímco CodeIgniter je nepřináší.
- CodeIgniter oproti Laravelu nepracuje striktně s MVC architekturou.
- Codeigniter je vhodnější pro začátečníky, protože Laravel nabízí mnoho funkcí, které je složitější pochopit.

6.2 Google Calendar

Google Calendar je internetový kalendář vyvíjen firmou Google od roku 2006. Google Calendar je možné používat na mobilních telefonech s operačním systémem Android i iOS, nebo na desktopových zařízeních pomocí webové aplikace. Tento kalendář podporuje spoustu různých funkcí. U událostí se dá nastavit počáteční i konečný čas nebo je možné nastavit konání na celý den. Je možno natavit místo konání události a také si můžeme nastavit připomenutí pomocí emailu nebo vyskakovacího okna. Pro vývojářské potřeby je k dispozici vývojové aplikační rozhraní Google Calendar API.

Google Calendar API

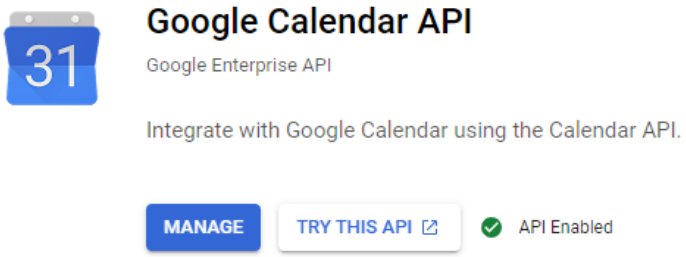
Díky Google Calendar API je možné vytvářet aplikace, které mohou pracovat přímo s událostmi v Google kalendáři u zvoleného Google účtu. Google Calendar API je RESTful API, ke kterému můžeme přistupovat prostřednictvím klientských knihoven Google. Toto rozhraní nám zpřístupní většinu funkcí, které jsou dostupné v aplikaci Google Calendar. [6]

Při práci s Google Calendar API můžeme pracovat s několika komponentami kalendáře, přičemž každá má své vlastní metody, které nám dovolují s nimi pracovat. [6]

- Event – event (událost) v kalendáři, jejíž součástí je čas začátku a konce události. Může to být jednorázová událost nebo opakující se událost.
- Calendar – kalendář je sbírka akcí. Každý kalendář má svá metadata, která nesou určité vlastnosti, například výchozí časové pásmo kalendáře.
- Calendar list – seznam všech kalendářů uživatele v uživatelském rozhraní kalendáře. Metadata obsahují vlastnosti kalendáře specifické pro jednoho uživatele jako je například barva kalendáře nebo způsob upozornění na události.
- Setting – předvolby uživatele, například časové pásmo uživatele.
- ACL – access control rule (pravidlo řízení přístupu) přiděluje uživateli nebo skupině uživatelů úroveň přístupu ke kalendáři.

Příprava pro práci s Google Calendar API

Ze všeho nejdříve se založí nový projekt na domovské stránce Google API. Zadá se název projektu a stiskne se tlačítko create. Po vytvoření projektu je vybrána položka Library, kde jsou API, které si lze aktivovat. V případě této práce bylo vyhledáno a aktivováno Google Calendar API. [20]



Obrázek 6.1: Aktivace Google Calendar API. Převzato z: [20].

Po vytvoření projektu a aktivace Google Calendar API je potřeba vytvořit servisní účet. Účet se vytváří na domovské stránce. Musíme vybrat záložku `Credentials` a poté `CREATE CREDENTIALS`, otevře se nabídka, ze které vybereme možnost Service Account (servisní účet). Jakmile vytvoříme účet, vygeneruje se nám soubor `credentials.json`. Díky tomuto souboru můžeme přidávat události do google kalendáře bez autorizace. [5]

6.3 Technologie použité při tvorbě front-endu

Při práci na front-endu bylo využito klasické HTML a CSS v kombinaci s frameworkem Bootstrap. Na doplňky ve front-endu byl využit jazyk javascript.

Bootstrap

Bootstrap[2] vytvořil Jacob Thornton a Mark Otto. Jedná se o open source framework pro tvorbu webu a webových aplikací, které využívají HTML, CSS a Javascript. Používání bootstrapu je velmi jednoduché, zejména pro vytváření responzivních webů. To je také důvod, proč patří mezi nejoblíbenější frameworky současnosti. Při práci lze používat předem definované styly, které lze aplikovat na HTML elementy a není potřeba definovat vlastní styly. Součástí bootstrapu je také používání různých komponent, jako jsou například různá tlačítka nebo rozbalovací nabídky a další. [4]

Chart.js

Javascriptová knihovna Chart.js[3] je open source knihovna, která slouží pro vytváření různých druhů grafů ve webových aplikacích. Chart.js nám umožňuje vymodelovat 8 druhů grafů, pro příklad to jsou grafy: sloupcové, plošné nebo spojnicové. V této práci je tato knihovna využita pro zobrazení statistik prodejních dat. [8]

[2]Dostupné z: https://getbootstrap.com/.
[3]Dostupné z: https://www.chartjs.org/.

Kapitola 7

Implementace

Když máme specifikovány požadavky od zákazníka a máme vytvořen návrh informačního systému, můžeme konečně přistoupit k samotné implementaci. Tato kapitola se věnuje implementaci informačního systému, budou zde popsány různé části informačního systému a postup jejich implementace. Pro implementaci byly využity technologie, které byly popsány v předešlé kapitole.

7.1 Důležité soubory

Jakmile vytvoříme nový projekt v Laravelu, vygeneruje se adresářová struktura s mnoha různými adresáři a soubory. Zde popíšu, se kterými soubory jsem nejvíce pracoval a k čemu slouží.

Nejdůležitějším adresářem je určitě adresář **app**. Obsahuje kódy, které tvoří samotné jádro informačního systému. V něm se nachází podadresář **Http**, který obsahuje Middlewary a hlavně Controllery. Právě Controllery tvoří logiku systému. Dalším podadresářem je adresář **Models**, který obsahuje modely. Každá tabulka v databázi je propojena s jedním modelem, který provádí změny nad daty v tabulce, například přidává nebo odstraňuje záznamy a také provádí dotazy nad tabulkami.

Dalším velmi důležitým adresářem je adresář **resources**. Jehož součástí jsou veškeré pohledy neboli views. To znamená soubory, které definují vzhled stránek, zobrazující se uživateli.

Adresář **database** obsahuje migrace databáze. Pomocí migrací se vytváří tabulky v databázi.

V kořenovém adresáři je také velice důležitý soubor **.env**, který obsahuje proměnné prostředí. Můžeme pomocí něj nastavit například přístup k databázi včetně přihlašovacích údajů do databáze.

7.2 Připojení a následná práce s databází

K databázi se připojíme pomocí souboru **.env**, jak již bylo zmíněno dříve. Každá tabulka má svůj vlastní model, těch je v systému okolo patnácti. Abychom mohli později manipulovat s atributy tabulek, musíme je nejdříve do každého modelu zadat.

V modelech se dá snadno definovat vztah mezi tabulkami. Předvedu na příkladu mezi tabulkami `Zakaznik` a `Objednavka`, kde je vztah 1:N. Zákazník může mít více objedná-

vek, ale objednávka přísluší pouze jednomu zákazníkovi. V tomto případě vypadá funkce v modelu pro objednávku takto:

```
public function customer() {
    return $this->belongsTo(Customer::class);
}
```

Metoda v modelu pro zákazníka vypadá pro změnu takto:

```
public function order() {
    return $this->hasMany(Order::class);
}
```

V každém modelu je možné mít metody, které nějakým způsobem pracují s databází a získávají z nich potřebná data. Tyto metody poté v rámci modelu volají Controllery.

7.3 Registrace uživatelů

Uživatelé se do systému registrují sami, nebo je registruje admin. Registrace je implementována v Controllerech pro registraci. Registrace každé role probíhá ve zvláštním Controlleru.

Zákazník

Pro registraci zákazníka je potřeba vyplnit registrační formulář. Ve formuláři je potřeba vyplnit pole s údaji, některá z nich nejsou povinná. Po vyplnění formuláře a stisknutí tlačítka se zavolá metoda `store()` v souboru `CustomerController`. Nejdříve jsou v rámci této metody data z formuláře zkontrolovány pomocí validátoru. Validátor ověří, zda jsou povinná pole opravdu vyplněna, zda jsou ve správném formátu a unikátní v rámci tabulky, pokud je to vyžadováno. Pokud jsou zadané údaje úspěšně validovány, do tabulky zákazníků se vloží záznam s nově vytvořeným zákazníkem. Heslo zákazníka, které se do databáze ukládá, je z důvodu bezpečnosti hashováno pomocí metody `Hash::make()`. Po vytvoření účtu je zákazník okamžitě přihlášen do systému.

Zaměstnanec

Účet zaměstnance je vytvořen adminem ihned poté, co je do firmy přijat. Admin zaregistruje zaměstnance pomocí registračního formuláře. Další postup je obdobný jako u registrace zákazníka. Pouze se zavolá metoda `store()` ze souboru `EmployeeController`, data jsou opět nejdříve validována. Po úspěšné validaci se vytvoří nový záznam v tabulce zaměstnanců.

Admin

Admin může přidávat další adminy. Princip vytváření účtu je stejný jako u zaměstnance a zákazníka, pouze se zavolá metoda `store()` ze souboru `AdminController`.

7.4 Role v systému a jejich autorizace

Informační systém budou využívat uživatelé tří rolí: admin, zákazník a zaměstnanec. Každému uživateli je přiřazena role podle toho, jakým způsobem se do systému zaregistruje

(nebo ho zaregistruje admin). Po vytvoření projektu je v Laravelu pouze jedna role `user`. Při registraci a přihlašování se automaticky pracuje s tabulkou v databázi `users`. Přidání rolí jsem provedl v souboru `auth.php`, kde jsem nejdříve musel upravit tzv. **providery**.

```
'providers' => [
    'employees' => [
        'driver' => 'eloquent',
        'model' => App\Models\Employee::class,
    ],
]
```

U každého **providera** bylo nutné vybrat model, pomocí kterého se po registraci uživatele pozná, jakou roli bude mít. Poté bylo ještě potřeba vytvořit přímo role (tzv. guards):

```
'guards' => [
    'employee' => [
        'driver' => 'session',
        'provider' => 'employees',
    ],
]
```

U každé z nich byl vybrán provider, který byl již dříve vytvořen. Po této úpravě jsou uživatelům při vytvoření přiřazeny příslušné role. Po vytvoření rolí je nutné nastavit, k jakému obsahu webové aplikace budou mít přístup příslušníci každé role. Toto je v informačním systému vyřešeno pomocí tzv. **Middlewarů**. Zde je příklad využití Middlewaru, v tomto případě se jedná o routu, která provede přesměrování na stránku zobrazení všech zákazníků:

```
Route::group(['middleware' => 'auth:admin'], function () {
    Route::get('/customers/index',
    [CustomerController::class, 'index'])
    ->name('customers.index');
}
```

Pokud by cestu této routy zadal příslušník jiné role (v tomto případě zákazník nebo zaměstnanec), byl by ze sytému odhlášen a přesměrován na úvodní přihlašovací obrazovku, kde se může přihlásit jako příslušník jiné role.

7.5 Přihlašování do systému

Přihlašování do systému je provedeno pomocí přihlašovacího formuláře, do kterého zadá uživatel email a heslo, pokud se jedná o admina nebo zaměstnance. V případě, že se však jedná o zákazníka, je nutné, aby zadal login a heslo. Pro přihlášení zákazníků každé role slouží jeden Controller. Pro Zákazníka je to `LoginCustomerController`, pro admina `LoginAdminControler` a pro zaměstnance `LoginEmployeeController`. Každý z těchto Controllerů obsahuje metodu `index()`, která po zavolání zobrazí uživateli příslušný přihlašovací formulář. V každém tomto Controlleru je také metoda `login()`, tato metoda slouží k ověření přihlašovacích údajů a následnému přihlášení do systému uživatele s příslušnou rolí. O odhlášení se stará metoda `logout()`, která je také v každém Controlleru, po odhlášení je uživatel přesměrován zpět na úvodní přihlaovací obrazovku.

7.6 Správa uživatelských účtů

Správu uživatelských účtů si může provést každý uživatel sám různými způsoby. Po vytvoření účtu si zákazník, zaměstnanec i admin mohou upravit svoje údaje, které zadali při registraci. K těmto úpravám slouží stejné Controllery, které sloužily pro registraci uživatelů. Pokud chce uživatel změnit údaje svého účtu, je mu zobrazen formulář pro úpravu profilu pomocí metody `edit()` v rámci patřičného Controlleru. Ve formuláři jsou již vyplněné jeho původní údaje pro přehlednost, pouze stačí uživateli je přepsat údaji novými. Po vyplnění a kliknutí na tlačítko se zavolá metoda `update()`, která údaje v databázi změní.

Dále je umožněno uživateli změnit svoje heslo. Uživateli je zobrazen formulář pro změnu hesla, do kterého je nutné zadat staré heslo, nové heslo a zopakovat nové heslo. Heslo je stejně jako při registraci hashováno z důvodu bezpečnosti.

Úpravu profilu i změnu hesla mohou kromě samotných uživatelů provádět logicky také správci informačního systému. Ti mají možnost zobrazit si seznam všech zaměstnanců a zákazníků. K tomu slouží `EmployeeController` a `CustomerController`, ve kterých je metoda `index()`. Tato metoda vrátí příslušný pohled i s proměnnou, která obsahuje všechny zákazníky nebo zaměstnance. Tato proměnná se v pohledu procyklí a do tabulky se vypíšou údaje o patřičných uživatelích. Admin má u každého z těchto uživatelů tlačítko pro změnu hesla i úpravu ostatních údajů.

Správce systému má navíc možnost oproti uživatelům odstranit jejich účty pomocí tlačítka. Po stisknutí tohoto tlačítka je zavolána metoda `destroy()` s parametrem, který určuje ID uživatele. Tento uživatel je v rámci této metody vyhledán v databázi a jeho účet je odstraněn. U zaměstnanců má admin navíc tlačítko pro označení práce na zakázce. U zákazníka má možnost vytvořit novou objednávku a také přidat novou kontaktní osobu.

7.7 Správa produktů v systému

Produkty se v informačním systému dělí do dvou hlavních skupin. Těmi jsou produkty originální a míchané. Originální produkty mají kód produktu začínající písmenem „O“ a míchané mají kód začínající písmenem „M“. Pro správu produktů slouží soubory `ProductOriginalController` a `ProductMixedController`. Produkty může do systému přidávat pouze admin, a to pomocí formuláře. Po zadání potřebných údajů o produktu a kliknutí na tlačítko se zavolá metoda `store()`, která zařídí uložení produktu do databáze. Produkty může naskladnit zaměstnanec nebo také admin. Při zobrazení produktů je u každého z nich textové pole, do kterého se zadá množství, které se naskladní a potvrdí tlačítkem. Při naskladnění je volána metoda `addOnStore()`, která změní množství na skladě v databázi. Při naskladnění míchaných produktů se cyklem projdou všechny originální produkty, které jsou v receptu tohoto míchaného produktu a odečtou se ze skladu, protože byly při míchání spotřebovány.

Recepty produktů

Každý míchaný produkt má recept, který obsahuje originální proukty. Vytváření, úprava i odstranění produktu z receptu je prováděno v rámci souboru `MixingProductController`. Upravit recepty je možno na stránce seznamu produktů, kde je u každého míchaného produktu tlačítko pro změnu receptu. Po stisknutí tlačítka se nám zobrazí pohled `mixingProduct.show`, kde jsou zobrazeny položky, které již jsou v receptu obsaženy. Tuto přísadu je možné smazat. Dále je zde formulář pro přidání produktu do receptu. Formu-

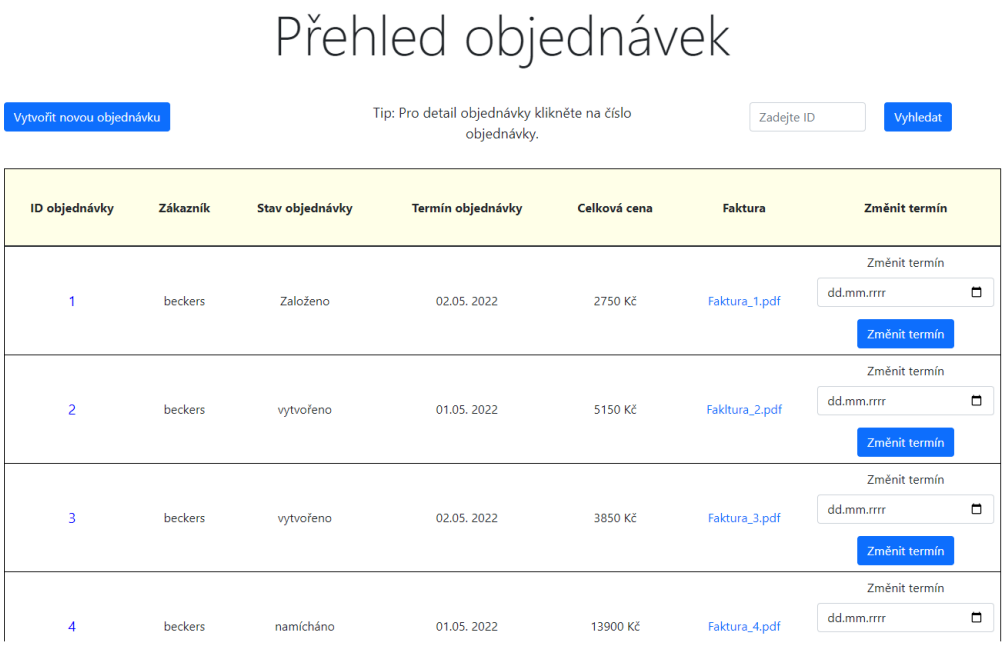
lář obsahuje pole, do kterého se zadá kód originálního produktu, který chceme do receptu přidat a potvrdí se tlačítkem. Po stisknutí tlačítka se zavolá metoda `store()`. Nejdříve se ověří zda, bylo vyplněno pole originálním produktem, který existuje. Také se ověří, zda tento produkt již není součástí receptu.

7.8 Správa objednávek

Veškerá správa objednávek je prováděna v rámci souboru `OrderController`. Objednávku může vytvořit zákazník nebo také admin. Zákazník si může vytvořit objednávku při zobrazení svých objednávek, kde stiskne tlačítko pro vytvoření, poté je zavolána metoda `store()`, která objednávku vytvoří. Admin objednávku vytvoří, když si zobrazí seznam zákazníků a u konkrétního zákazníka stiskne tlačítko pro vytvoření. Poté je zavolána metoda `storeAdmin()`.

Po vytvoření objednávky může k objednávce zaměstnanec nebo admin nahrát fakturu, kterou si později stáhne zákazník. Po nahrání fakturu vidí také admin a zaměstnanec, ti navíc mají možnost fakturu změnit. Při nahrání se zavolá metoda `uploadFile()`, která soubor uloží do vybraného adresáře a do databáze uloží název tohoto souboru. Pokud je již soubor nahrán, zákazník vidí jeho název a má možnost ho stáhnout.

U každé objednávky je možné po vytvoření měnit její stav (metoda `changeState()`) a termín (metoda `changeTerm()`). Termín se zároveň uloží do google kalendáře pomocí Google Calendar API, poté mohou vidět zaměstnanci v google kalendáři termíny objednávek.



Obrázek 7.1: Stránka pro zobrazení objednávek.

Správa položek v objednávce

S položkami v objednávkách pracuje soubor ItemController. Po rozkliknutí detailu objednávky a kliknutí na tlačítko pro přidání položky se zavolá metoda `create()`, která zařídí zobrazení seznamu produktů, u kterých bude také textové pole pro zadání množství a

tlačítko. Po stisknutí tlačítka se zavolá metoda `store()`, která nejdříve zjistí podle kódu produktu, zda se jedná o míchaný nebo originální produkt a podle toho vyplní atribut `is_mixed` v tabulce s položkami. Také vyplní atribut `ID_produktu` a zkontroluje se, zda je na skladě dostatek tohoto produktu. Pokud ne, vypíše se varovné hlášení. Pokud je na skladě dostatek kusů, po přidání k objednávce se ze skladu odečtou. Pro odstranění položky z objednávky slouží metoda `destroy()`. Ta naopak po smazání položky doplní množství automaticky zpět na sklad.

Správa balení

Úprava balení položek se provádí v souboru `PackageItemController`. U každé položky objednávky je tlačítko pro úpravu balení. Po stisknutí vidíme všechny již vybrané nádoby a také tlačítko pro přidání nové nádoby. Po stisknutí, stejně jako u produktů, vidíme seznam nádob a textové pole, do kterého zadáme množství kusů. Po přidání je volána metoda `store()`, která nejdříve zkontroluje počet položek na skladě. V případě, že je na skladě dostatek těchto nádob, přidá se do databáze záznam s ID položky, ID nádoby a počtem kusů. Později je možné upravit množství kusů této nádoby, což zařídí metoda(`changeCount()`) a také je možné nádobu úplně odstranit (metoda `destroy()`).

7.9 Google Calendar

V informačním systému je využito Google Calendar API, které umožňuje zobrazit přehled objednávek a jejich termíny dokončení v Google kalendáři zaměstnanců. Objednávka je do kalendáře přidána jako celodenní událost na den, kdy je termín této objednávky. Událost je do Google kalendáře přidána při vytvoření objednávky v rámci souboru `OrderController`. Pro přidávání událostí do kalendáře byla využita tato knihovna: Manage events on a Google Calendar[1]. Ukázka přidání události do google kalendáře je uvedena zde [19]:

```
Event::create([
    'id' => 'eventid'.$newOrder->id,
    'name' => 'Číslo objednávky: '.$newOrder->id,
    'startDate' => Carbon::now()->add(1, 'week'),
    'endDate' => Carbon::now()->add(1, 'week'),
]);
```

Událost (Event) má několik atributů. Jako identifikátor události jsem zvolil řetězec „eventidxx“, kde „xx“ je ID objednávky, které bude stejné jako v databázi. Počáteční i konečné datum události je nastaveno na stejný den. Tím se docílí toho, aby byla událost celodenní. V tomto případě je to týden po založení objednávky, protože každá objednávka má po založení tento termín. Při úpravě termínu objednávky se vyhledá událost podle ID objednávky, kterou chceme upravit.

Pokud by událost nebyla nalezena, znamená to, že se z nějakého důvodu při založení objednávky nevytvořila. V tom případě je v rámci tohoto kódu zachycena výjimka `Google_Service_Exception`, poté je tato událost vytvořena znovu. Pokud je událost nalezena, pouze se upraví počáteční a konečné datum události.

[1] Dostupné z: https://github.com/spatie/laravel-google-calendar

7.10 Asociační pravidla

V informačním systému je implementováno získávání asociačních pravidel. Konkrétně se vyhledají v systému položky, které jsou často objednávány společně v rámci jedné objednávky a pak je zákazníkům na základě tohoto zjištění doporučováno zboží, které se často vyskytuje s položkami, které již mají v objednávce.

V této práci je využita metoda Apriori. Nalezení frekventovaných množin je řešeno v souboru `OrderController`, kde je metoda `getProductsByOrder()`. Tato metoda projde celou databázi a vrátí pole, ve kterém jsou na každém indexu uloženy položky, které byly zakoupeny v rámci jedné objednávky. Toto pole je později využito pro získání frekventovaných množin.

Pro implementaci Apriori je využita knihovna PHP-ML[2]. Nejdříve je třeba vytvořit instanci třídy `Apriori`, do které se zadají parametry podpory a spolehlivosti. Na základě provedeného testování byly zvoleny následující hodnoty, protože při nižší podpoře bylo doporučováno příliš mnoho produktů. Při vyšší podpoře byly produkty doporučovány naopak jen zřídka.

```
$apriori = new Apriori($support = 0.2, $confidence = 0.2);
```

Po vytvoření instance nad ní zavoláme metodu `train()`. Jako vstup použijeme pole, které jsme dostali jako výstup metody `getProductsByOrder()`. Po vykonání této metody máme „vytrénovaný“ asociátor. Můžeme nad ním tedy volat další metody, pomocí kterých získáme potřebné výsledky.

```
$apriori->train($items1, $items2);
```

Dále bylo vytvořeno pole `recommendedItems`, do kterého budou uloženy doporučené produkty. Do tohoto pole je uložen výsledek metody `predict()`, která na vstup dostala pole s produkty nacházející se v objednávce, pro kterou hledáme doporučené produkty.

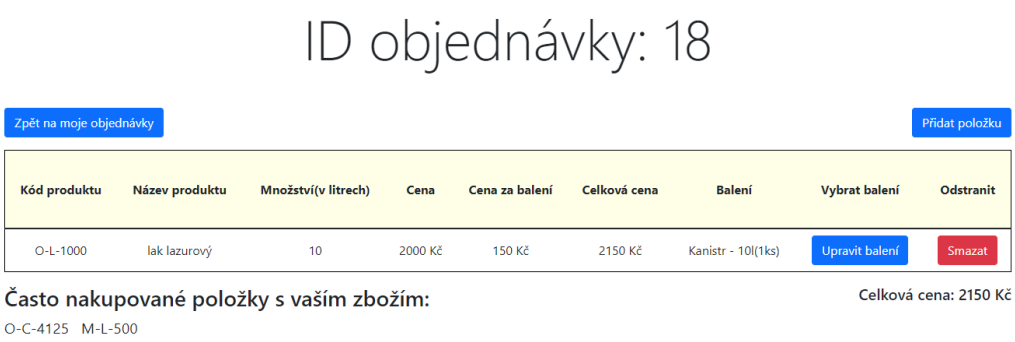
```
$recommendedItems = $apriori->predict($itemsInOrder);
```

Metoda `predict()` vrátí pole, které obsahuje další pole, ve kterých jsou produkty, které jsou frekventované s produkty v objednávce. Příklad výstupu metody je uveden zde:

```
[[['O-C-4125']], [['M-L-500']], [['O-C-4125', 'M-L-500']]]
```

Jelikož se položky v poli vyskytují vícekrát, musely být data upravena tak, aby se každá položka zobrazila uživateli jako doporučené zboží pouze jednou. Tyto data jsou následně odeslána pohledu, který zobrazí detail objednávky. V tomto pohledu jsou poté zobrazeny doporučené produkty pro zákazníka. [1]

[2] https://php-ml.readthedocs.io/en/latest/



Obrázek 7.2: Zobrazení doporučených produktů u objednávky

7.11 Statistiky

V informačním systému je zpracována analýza dat, která je zobrazena uživatelům systému formou grafů. Uživatelé mají možnost zobrazení statistiky týkajících se zejména prodeje produktů, ale také odvedené práce zaměstnanců nebo největších odběratelů. Ke statistikám mají přístup všichni uživatelé systému, avšak každé roli uživatelů jsou zobrazeny odlišné statistiky. Všechny statistiky vidí pouze admin. Zaměstnanec například nevidí statistiky týkající se nejproduktivnějších zaměstnanců a zákazníci nevidí statistiky největších odběratelů.

Pro tvorbu grafů byla využita javascriptová knihovna chart.js[3]. Datová analýza je zpracována v modelu `Stats.php`, kde jsou volány SQL dotazy nad daty z databáze. Součástí tohoto souboru je také metoda `getMaxFrom()`, které je jako parametr zadán výsledek dotazu. Tato metoda poté vybere záznamy s nejvyšší četností. Metody z tohoto modelu jsou volány v souboru StatsController.php, kde se data z těchto metod uloží do proměnných a jsou odeslány pohledu **`stats.index.blade.php`**, kde jsou zpracovány formou grafů.

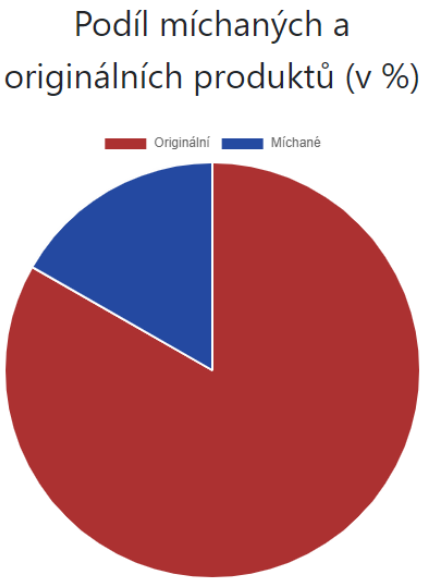
Každý graf je zpracován v jednom canvasu. Každý tento canvas má svůj identifikátor, pomocí kterého se určí, který graf do něj bude vymodelován.

Zde vidíme příklad dotazu, jehož výsledkem je seznam zaměstnanců a součet hodin, které odpracovaly.

Na obrázku 7.3 vidíme nejprodávanější produkty vymodelované do sloupcového grafu. Na obrázku 7.4 vidíme podíl míchaných a originálních produktů, které si zákazníci objednali. Data jsou vymodelována do koláčového grafu, ve kterém jsou uvedeny procenta, které nám říkají v kolika případech lidé objednávají originální produkty a v kolika případech míchané produkty.

[3] `https://www.chartjs.org/`

Obrázek 7.3: Ukázka sloupcového grafu.



Obrázek 7.4: Ukázka koláčového grafu.

Kapitola 8

Testování

Posledním krokem, který následoval po implementaci, bylo testování výsledné aplikace. Jedná se o velice důležitou fázi, při které ověřujeme, zda výsledný produkt funguje tak, jak se očekává.

8.1 Testování při implementaci

Testování bylo provedeno po každé větší části informačního systému. Při tomto testování byly odhaleny drobné chyby, které se týkaly například chybného předávání parametrů mezi třídami a šablonami. Bylo také často zapotřebí opravit chybové hlášky. Toto testování probíhalo na lokálním serveru a aplikace pracovala s lokální MySQL databází.

8.2 Testování uživateli

Jakmile byla implementace informačního systému dokončena, začalo uživatelské testování. Hlavním účelem při tomto testování bylo zjistit výslednou kvalitu systému v praxi. Také bylo potřeba ověřit, zda jsou správně splněny všechny požadavky na výsledný systém. Tohoto testování se účastnila firma, která je potencionálním zákazníkem. Dále systém testovaly příslušníci rodiny, přátelé a blízcí.

Kroky, které provedli zákazníci

Nejdříve ze všeho zákazník dostal za úkol vytvořit si svůj účet v systému a následně se přihlásit. Po přihlášení si vyzkoušel změnit svoje heslo a také některé údaje, které zadal při registraci. Poté si v systému prohlédl nabízené produkty a vytvořil si novou objednávku, do které si přidal několik produktů a vybral jejich zabalení. Zákazník si přidal do několika objednávek stejné produkty a vyzkoušel, jestli v další objednávce dostane doporučení na produkty, které se opakují v již vytvořených objednávkách. Poté si u objednávky zkusil stáhnout fakturu a změnit termín, do kterého chce mít objednávku připravenou. Po otestování objednávek si zkusil přidat novou kontaktní osobu, změnit její údaje a následně ji zkusil také smazat. Jako poslední krok bylo odhlášení ze systému.

Kroky, které provedli zaměstnanci

Zaměstnanec dostal přihlašovací údaje, pomocí kterých se do systému přihlásil. Po přihlášení změnil svoje údaje a heslo. Poté změnil stav objednávek a označil na objednávkách

vykonanou práci. Svoji odvedenou práci si zobrazil a zkontroloval. K objednávce nahrál fakturu. V rámci svého Google účtu si zkusil do svého Google kalendáře přidat firemní kalendář, ve kterém se mu zobrazují termíny všech objednávek. Po splnění těchto úkolů se odhlásil ze systému.

Kapitola 9

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat informační systém pro prodejce nátěrových hmot. Při návrhu i implementaci byl kladen důraz zejména na přehlednost a jednoduchost systému. Při vývoji informačního systému byly pořádány konzultace s potencionálním zákazníkem, tak aby výsledný systém co nejvíce odpovídal jeho požadavkům. V rámci systému je zákazníkům umožněno prohlížet si dostupné produkty, vytvářet nové objednávky, do kterých přidávají produkty a vybírají si u nich konkrétní zabalení. U každé objednávky si zákazník může nastavit termín, do kterého chce mít objednávku vyřízenou. Zaměstnanec má přehled o všech objednávkách, může měnit jejich stav a označovat na nich vykonanou práci. Pro lepší přehlednost o termínech objednávek je zaměstnancům umožněno propojení s Google kalendářem, ve kterém se všechny termíny zobrazí. Kromě těchto bodů, je poskytnuto zobrazení statistik týkajících se prodeje produktů a dalších dat v systému. Také je implementována analýza asociačních pravidel, pomocí kterých jsou zákazníkům doporučovány produkty v závislosti na produktech, které jsou součástí objednávky. Informační systém je dostupný online na webové adrese: http://bakal-colors.online/.

9.1 Možná vylepšení

Informační systém vyhovuje všem požadavkům, které zákazník požadoval. Avšak jsou určitá vylepšení, která by mohla být do budoucna implementována. Jedno z těchto vylepšení by mohla být pokročilejší analýza pomocí OLAP technologií. Například by mohly být dostupné statistiky, které by ukazovaly nejprodávanější produkty v závislosti na ročních obdobích. Také by mohlo být implementováno využití nějaké shlukovací metody. Například pro shlukování zákazníků, kteří často nakupují stejné produkty. Systém by do budoucna mohl navíc obsahovat volbu dopravy. Kromě webové aplikace by systém mohl být také součástí mobilní aplikace, aby zákazník mohl snáze vytvářet a spravovat své objednávky.

Literatura

[1] *Apriori Associator* [online]. [cit. 2022-05-02]. Dostupné z: https://php-ml.readthedocs.io/en/latest/machine-learning/association/apriori/.

[2] *Artisan Console* [online]. [cit. 2022-04-24]. Dostupné z: https://laravel.com/docs/9.x/artisan.

[3] *Backend* [online]. [cit. 2022-04-24]. Dostupné z: https://www.shoptet.cz/slovnik-pojmu/backend/.

[4] Ouellette, A. *What is Bootstrap: A Beginner's Guide* [online]. [cit. 2022-04-24]. Dostupné z: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/.

[5] Kutáč, P. *Google Calendar API - události z vlastního kalendář* [online]. [cit. 2022-04-25]. Dostupné z: https://www.kutac.cz/weby-a-vse-okolo/google-calendar-api-udalosti-z-vlastniho-kalendare.

[6] *Google Calendar API Overview* [online]. [cit. 2022-04-24]. Dostupné z: https://developers.google.com/calendar/api/guides/overview.

[7] Zendulka, J. a Bartík, V. *Konceptuální modelování* [online]. [cit. 2022-04-25]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIDS-IT%2Flectures%2Fcz%2F2_kmod.pdf&cid=13296.

[8] *Simple yet flexible JavaScript charting for designers & developers* [online]. [cit. 2022-04-24]. Dostupné z: https://www.chartjs.org/.

[9] Pohořelský, M. *Klient-server* [online]. 2015 [cit. 2022-04-25]. Dostupné z: http://www.multimediaexpo.cz/mmecz/index.php/Klient-server.

[10] *Architektura klient-server (Client–server model)* [online]. [cit. 2022-04-24]. Dostupné z: https://managementmania.com/cs/architektura-klient-server.

[11] Jackson, P. *Laravel vs CodeIgniter: Which is Better?* [online]. [cit. 2022-04-25]. Dostupné z: https://www.guru99.com/laravel-vs-codeigniter.html.

[12] *Getting Started* [online]. [cit. 2022-04-24]. Dostupné z: https://getcomposer.org/doc/00-intro.md.

[13] *What is CSS?* [online]. [cit. 2022-04-24]. Dostupné z: https://www.thatcompany.com/what-is-css.

[14] *The steps for data mining process* [online]. 2020 [cit. 2022-04-26]. Dostupné z: https://www.researchgate.net/figure/The-steps-for-data-mining-process_fig3_344166043.

[15] Štráfelda, J. *Databáze* [online]. [cit. 2022-04-25]. Dostupné z: https://www.strafelda.cz/databaze.

[16] *DBMS (Database Management System) - systém řízení báze dat* [online]. [cit. 2022-04-25]. Dostupné z: https://managementmania.com/cs/dbms-database-management-system-system-rizeni-baze-dat.

[17] Chand, M. *Most Popular Databases In The World* [online]. [cit. 2022-04-25]. Dostupné z: https://www.c-sharpcorner.com/article/what-is-the-most-popular-database-in-the-world/.

[18] *Frontend* [online]. [cit. 2022-04-26]. Dostupné z: https://www.seobility.net/en/wiki/Frontend.

[19] Herten, F. V. der. *Manage events on a Google Calendar* [online]. [cit. 2022-05-02]. Dostupné z: https://github.com/spatie/laravel-google-calendar.

[20] *Google API Console: APIs and Services* [online]. [cit. 2022-04-25]. Dostupné z: https://console.cloud.google.com/apis/dashboard.

[21] Han, J., Pei, J. a Kamber, M. *Data Mining: Concepts and Technuiques*. 3. vyd. Morgan Kaufmann, 2012. ISBN 978-0-12-381479-1.

[22] Čápka, D. *Lekce 1 - Úvod do HTML a váš první web* [online]. [cit. 2022-04-24]. Dostupné z: https://www.itnetwork.cz/html-css/webove-stranky/jak-psat-moderni-web-html-tutorial-uvod-do-html.

[23] Vladimír Šmíd. *Pojem informačního systému* [online]. [cit. 2022-04-24]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsys.htm.

[24] Burget, R. *Pojem informačního systému, data, informace* [online]. 2020 [cit. 2022-04-24]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp01_Informacni_systemy.pdf&cid=13985.

[25] Malhotra, A. *Java Tutorial - Tutorialspoint* [online]. 2019 [cit. 2022-04-24]. Dostupné z: https://www.xicom.biz/blog/9-best-reasons-to-choose-java-for-web-development/.

[26] Čápka, D. *Lekce 1 - Úvod do JavaScriptu* [online]. [cit. 2022-04-24]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk.

[27] Koďousková, B. *Javascript pro začátečníky: co to je a jak funguje* [online]. [cit. 2022-04-24]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky.

[28] Lacko, L. *Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle*. Brno: Computer Press, 2003. ISBN 80-7226-969-0.

[29] Szecsei, S. *Advantages and Disadvantages of The Laravel Framework* [online]. 2020 [cit. 2022-04-25]. Dostupné z: https://www.popwebdesign.net/popart_blog/en/2020/03/advantages-and-disadvantages-of-the-laravel-framework/.

[30] Chojrin, M. *Laravel vs. Symfony: A Side by Side Comparison* [online]. [cit. 2022-04-25]. Dostupné z: https://adevait.com/laravel/laravel-vs-symfony-comparison.

[31] Čápka, D. *MVC architektura* [online]. [cit. 2022-04-23]. Dostupné z: https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury.

[32] Čápka, D. *MVC architektura* [online]. [cit. 2022-04-26]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

[33] *What is MySQL? Everything You Need to Know* [online]. [cit. 2022-04-25]. Dostupné z: https://www.talend.com/resources/what-is-mysql/.

[34] *CO JE TO DATABÁZE MYSQL?* [online]. [cit. 2022-04-25]. Dostupné z: https://best-hosting.cz/cs/napoveda/co-je-to-databaze-mysql.

[35] *IBM Cloud Education: OLTP* [online]. 2020 [cit. 2022-04-24]. Dostupné z: https://www.ibm.com/cloud/learn/oltp.

[36] Shweta, D. *What is PHP & Reasons You Should Learn PHP* [online]. 2019 [cit. 2022-04-24]. Dostupné z: https://medium.com/javarevisited/what-is-php-reasons-you-should-learn-php-38f898a8e287.

[37] *What is Python? Executive Summary* [online]. [cit. 2022-04-25]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[38] Můčka, J. *Relační databáze vs. nerelační databáze: jaké jsou mezi nimi rozdíly?* [online]. 2021 [cit. 2022-04-25]. Dostupné z: https://www.master.cz/blog/relacni-databaze-nerelacni-databaze-jake-jsou-rozdily/.

[39] Daniel, R. *Datový sklad* [online]. 2010 [cit. 2022-04-24]. Dostupné z: https://homel.vsb.cz/~dan11/is_skripta/IS%202010%20-%20Danel%20-%20Datovy%20sklad.pdf.

[40] Máca, J. *Lekce 1 - Úvod do Symfony frameworku pro PHP* [online]. [cit. 2022-04-25]. Dostupné z: https://www.itnetwork.cz/php/symfony/zaklady/uvod-do-symfony-frameworku-pro-php.

[41] *Třívrstvá architektuta* [online]. 2015 [cit. 2022-04-24]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.

[42] Monus, A. *10 PHP Frameworks For Developers – Best of* [online]. 2018 [cit. 2022-05-02]. Dostupné z: https://www.hongkiat.com/blog/best-php-frameworks/.

[43] Čápka, D. *Lekce 2 - UML - Use Case Diagram* [online]. [cit. 2022-04-24]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram.

[44] *Web application* [online]. [cit. 2022-04-23]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app.

Příloha A

Obsah přiloženého paměťového média

- **app** – adresář se zdrojovými soubory aplikace.
- **zprava-src** – adresář obsahující zdrojové soubory pro tvorbu technické zprávy.
- **xkrene15.pdf** – technická zpráva ve formátu PDF.
- **instalace** – návod na instalaci.