

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky



Tvorba webové aplikace pro organizaci sběru výpočetní techniky při rozsáhlých akcích

BAKALÁŘSKÁ PRÁCE

Studijní program: Aplikovaná Informatika

Studijní obor: Aplikovaná Informatika

Autor: Zeman Petr

Vedoucí bakalářské práce: Ing. Jan Kučera, Ph.D.

Konzultant bakalářské práce: Ing. Jan Kučera, Ph.D.

Praha, březen 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci „Tvorba webové aplikace pro organizaci sběru výpočetní techniky při rozsáhlých akcích“ vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne

.....

[jméno a příjmení autora]

Poděkování

Tuto práci věnuji svým rodičům, kteří mě, i přes všechny problémy během mých studií, vždy podporovali a nepřestali mi věřit.

Tímto také děkuji vedoucímu mé bakalářské práce panu doktoru Ing. Janu Kučerovi, Ph.D., který vždy poradil a byl kdykoliv k dispozici ke konzultaci.

Nakonec bych rád poděkoval mému nadřízenému, který mi věnoval svůj čas a informace potřebné pro vytvoření kvalitní analýzy pro tuto práci.

Abstrakt

Tato práce je zaměřena na analýzu a vývoj webové aplikace asistující při logistice rozsáhlých akcí vyžadujících dodávku a pozdější sběr vybavení na rozdílné lokace. Pro zjednodušení přístupu k této aplikaci v terénu je zapotřebí, aby tato aplikace byla optimalizována pro mobilní telefony. Jak této optimalizace bylo dosaženo si zde přiblížíme z pohledu teorie i praxe.

V první části této práce nejdříve zanalyzujeme existující alternativy, jejich silné a slabé stránky, a odůvodníme vývoj nové aplikace oproti využití již existujících alternativ.

V další části je uvedeno, jaká metodika byla využita pro vývoj a jak bylo postupováno, ale také popisuje optimalizaci webové aplikace pro chytré mobilní telefony a postup při vývoji těchto aplikací

Zbytek práce se věnuje jednotlivým krokům vývoje této aplikace: analýze požadavků, návrhu aplikace, jejího vývoje a konečného testování.

Cílem této práce je seznámit čtenáře s postupem vývoje webové aplikace zaměřené na logistiku a definováním způsobů optimalizace této aplikace pro mobilní telefony.

Klíčová slova

Logistika, Mobile-First, Vývoj, Analýza, Optimalizace, Web, Aplikace

JEL klasifikace

O30

Abstract

This work focuses on analysis and development of a web application that would assist with logistics of wide-spread events requiring supply and later retrieval of necessary requirement to varied locations. To ease access to this application during fieldwork it is required to optimise this application for smart telephones. How this optimisation was reached will be defined from both theoretical and practical standpoint.

In first part of this work we will begin with analysis of existing alternatives. Their strong and weak points will be defined and through analysis we will define why instead of utilising already existing alternatives a new application was developed.

The work will continue with defining which design methodology was utilised for development and which steps were taken during development for smooth progress. Right afterwards a special chapter will be dedicated for defining what it means to have a web application optimised for smartphones and which steps to take during development.

Rest of the work is focused on different parts of application development: Requirements analysis, application design, application development and final testing.

The goal of this work is to show readers how to develop a web application focused on logistics and defining ways to optimise web applications for smartphones.

Keywords

Logistics, Mobile-First, Development, Analysis, Optimalisation, Web, Application

JEL Classification

O30

Obsah

Použité zkratky	9
Úvod.....	11
1 Analýza souvisejících prací	12
1.1 Analýza souvisejících závěrečných prací.....	12
1.2 Analýza související literatury.....	13
1.3 Závěr analýzy	13
2 Analýza existujících aplikací.....	14
2.1 Monday.com	15
2.2 ZohoCreator	16
2.3 OnFleet.....	18
Závěr analýzy:	20
3 Metody a způsob řešení práce	22
3.1 Vodopádový model	22
3.2 Vývojové prostředí	23
3.2.1 Textový editor	23
3.2.2 Verzovací systém	24
3.2.3 HTTP Server	24
3.2.4 SMTP Server.....	24
4 Webové aplikace optimalizované pro chytré mobilní telefony a specifika jejich vývoje	25
4.1 Velikost obrazovky.....	25
4.2 Výkon telefonu	27
4.3 Připojení k datům	27
4.4 Závěr optimalizace	27
5 Analýza požadavků	28
5.1 Funkční požadavky	28
5.2 Nefunkční požadavky	30
5.3 Analýza aplikace.....	32
5.3.1 Technik.....	32
5.3.2 Klient	34
5.3.3 Administrátor	35
6 Návrh aplikace	37
6.1 Technologie využité při vývoji aplikace.....	37

6.1.1 Backend	37
6.1.2 Frontend	38
6.2 Databáze.....	38
6.3 Webová aplikace	39
6.3.1 Model (Manager)	40
6.3.2 View	41
6.3.3 Presenter	43
7 Implementace aplikace	46
7.1 Model	46
7.1.1 UserManager	46
7.1.2 ExcelManager	47
7.1.3 FormManager	50
7.1.4 GridManager	50
7.2 Presenter	51
7.2.1 Presenter	51
7.2.2 Továrny na formuláře	53
7.3 View	54
7.3.1 @Layout.latte	54
7.3.2 Branches	55
7.3.3 Sign	56
7.3.4 Upload	57
7.4 Databáze.....	57
8 Ověření aplikace	58
8.1 Testování požadavků	58
8.1.1 Funkční požadavky	58
8.1.2 Nefunkční požadavky	59
8.2 Testování responzivity.	60
Závěr	62
Seznam obrázků, tabulek a výpisů.....	63
Použitá literatura	65
Přílohy.....	I

Použité zkratky

- Backend – část kódu řešící logiku probíhající na serveru.
- Frontend – část kódu řešící zobrazení a část kódu probíhající na přístroji uživatele.
- HTML – Hypertext Markup Language – základní stavební blok webových stránek. Jedná se o značkovací jazyk, který udává strukturu a obsah stránky.
- CSS – Cascading Style sheets – rozšíření HTML, které dodává možnost definovat vzhled stránek napsaných v HTML, XML a XHTML.
- PHP – Hypertext Preprocessor – skriptovací jazyk určený k vývoji webových stránek a aplikací.
- Javascript / EcmaScript – skriptovací jazyk vycházející z EcmaScriptu. Jeho nejznámější použití je jako jazyk, vnořený do internetových prohlížečů, ale také byl široce adoptován pro serverové a vnořené aplikace.[1]
- Framework – rozšíření jazyka a jeho struktury, které dodává do jazyka možnosti a struktury, které nejsou v originálním jazyce.
- Šablonovací systém – systém umožňující generování webových stránek za pomoci předpřipravených kusů kódu – tzv. Šablon
- Nette – framework pro rozšíření schopností PHP.
- Latte – šablonovací systém pro Nette.
- Bootstrap – knihovna s připravenými komponenty pro HTML, CSS a JS.
- PHPSpreadsheet – knihovna pro PHP zlehčující práci především se soubory Excel.
- MySQL – strukturovaný dotazovací jazyk pro tvorbu a práci s databází vlastněný Oracle.
- MariaDB – strukturovaný dotazovací jazyk, vytvořený původními vývojáři MySQL. Plně kompatibilní s MySQL jako náhrada za něj v případě jeho zpoplatnění po zakoupení Oraclem.
- UML – Unified Modeling Language – standardizovaný způsob zobrazování diagramů pro vývoj softwaru.
- Use Case Diagram – jedná se o jeden z diagramu spadající pod UML. Přesněji řečeno je to standardizovaný způsob zobrazení funkčních požadavků za pomoci případů užití propojených s jednotlivými rolemi.
- Wireframe – návrh definující základní vzhled, funkci a obsah webových stránek.
- MySQL workbench – aplikace pro navrhnutí a propojení s MySQL databází.
- EER – Enhanced Entity-Relationship model – vylepšený entitně-relační model, který rozšiřuje existující princip entitně-relačního modelu a dodává mu nové možnosti zobrazení relací.
- MVP – Model-View-Presenter – způsob vývoje informačních systémů založený na principu oddělení grafiky od logiky. Dělí kód na tři rozdílné části:
 - Model – logická část aplikace obstarávající propojení s databází a veškeré logické prvky.
 - View – grafická část aplikace, která zobrazuje získaná data z ostatních částí systému a kterou uživatel vidí před sebou.

- Presenter – část aplikace obstarávající spojení mezi částmi Model a View. Obstarává generování jemu přiřazeného View a dodává do něj data získaná z Modelu.
- XSS – Crosssitescripting – způsob útoku na webové stránky využitím neopatřenému vstupu. Útočník může do neopatřenému vstupu vložit Javascriptový kód.
- DataTable/DataGrid – tabulky, které kromě schopnosti zobrazit data vlastní i další schopnosti ulehčující práci s zobrazenými daty. Například se jedná o jednoduché vyhledávání, třídění dle sloupců či schopnosti otevřít skryté informace.

Úvod

Tato práce se zabývá problematikou centrálně vedeného sběru výpočetní techniky při rozsáhlých akcích. Práce je určena pro problematiku konkrétní společnosti a počítá se s existencí více týmů techniků obstarávající různé lokace. Lokace jsou rozděleny dle oblastí a každá oblast má své nadřazené oddělení. Kontakt mezi jednotlivými technikami a nadřazenými je zpracováván dispečerem. Na dispečera je tím přesunuto neskutečné množství komunikace a organizace. Tento proces způsobuje velké časové prodlevy při procesu zpracování dat a předávání těchto dat potřebným osobám. Autor se s touto problematikou setkal v jeho aktuálním zaměstnání. Vzhledem k autorově zájmu o vývoj webových aplikací, mu bylo navrženo, aby vyřešení této problematiky zpracoval do své závěrečné práce. Tato práce se bude zabývat procesem vývoje webové aplikace fungující jako centrum pro veškeré techniky, klienty a dispečery. Bez této aplikace probíhá veškerá komunikace o změnách lokalit skrze dispečera, a tím vzniká omezení v toku informací. Přesun dispečera z pozice hlavního datového toku na pozici kontroly tohoto toku odstraní toto omezení, což povede k nárůstu rychlosti přenosu informací mezi týmy a klienty. Jelikož technici pracují v terénu, je nutné optimalizovat aplikaci pro mobilní telefony.

Cílem této práce je vytvořit webovou aplikaci pro podporu a optimalizaci organizování dodávek a sběru výpočetní techniky při rozsáhlých akcích. Aby se vývoj této aplikace dal považovat za splněný musí zvládat – jednak evidovat jednotlivé lokace, na které je technika dovážena společně s informacemi, které týmy a zákazníci jsou spojeni s danými lokacemi, jednak evidovat stav dodávky techniky na jednotlivé lokace, a umožnit technikům a zákazníkům tento stav měnit v lokalitách, které obstarávají. Daná webová aplikace musí být optimalizovaná pro mobilní zařízení.

V dané práci bude nejprve provedena analýza souvisejících prací. V další kapitole proběhne analýza a ohodnocení existujících alternativních aplikací, které by tuto problematiku mohly vyřešit. Následně bude definováno, jaké metodiky a nástroje budou využity při vývoji aplikace. V poslední teoretické kapitole bude definováno, co to jsou webové aplikace optimalizované pro mobilní telefony, a jak je možné této optimalizace dosáhnout. Dále proběhne vývoj podle vybrané metody –Vodopádového modelu. konkrétně v kapitolách Analýzy požadavků, Návrhu, Implementace a Testování. Dle metodiky by se zde měla nacházet i kapitola týkající se údržby aplikace, ale ta již není v kompetenci této práce. Více o tomto modelu je možné nalézt v kapitole 3. Jak byl tento model použit při vývoji aplikace je rozepsáno v kapitolách 5 až 7.

Při vývoji se autor inspiroval závěrečnými pracemi, např. „Optimalizace webu pro mobilní zařízení a analýza jeho vlivu“[2] od Jana Pospíšila, nebo „Responzivní webdesign“[3] od Jakuba Hnízдила. Také se inspiroval literaturou, například knihou „Mobile-first Bootstrap“[4] od autora Alexandre Magno.

1 Analýza souvisejících prací

1.1 Analýza souvisejících závěrečných prací

Nejprve je důležité zjistit, zda tato problematika nebyla řešena již dříve. Při průzkumu vypracovaných prací byly prohledány vnitřní e-zdroje dodávané knihovnou VŠE a systém Theses.cz. Při vyhledávání byly použity následující klíčová sousloví: „Vývoj aplikace, optimalizace webu pro mobilní telefony, aplikace pro organizaci rozvozu a aplikace pro logistiku“ a jejich anglické překlady. Při průzkumu nebyla nalezena žádná práce, která by souvisela přímo s řešenou problematikou dodávek a rozvozu materiálu, či vývoje aplikace pro logistické operace. Nejbližší práce zabývající se podobnou problematikou, byly práce řešící vývoj webových aplikací a optimalizaci webových aplikací pro mobilní telefony. Jedná se především o práce:

- „Optimalizace webu pro mobilní zařízení a analýza jejího vlivu“ [2] od Pospíšila Jana
 - Tato práce pojednává o problematice návrhu a vývoje webu pro mobilních zařízeních a jejich přínosech.
 - Oproti této bakalářské práci, Pospíšil Jan pracuje již s existující stránkou a ukazuje, jak ji optimalizovat pro mobilní telefony. Užívá zde metodiku „Graceful degradation“ oproti metodice „Progressive enhancement“, která je vlastní vývoji Mobile-first.
- „Responzivní webdesign“ [3] od Hnízдила Jakuba
 - Práce „Responzivní webdesing“ pojednává o principech, které je potřeba dodržovat pro dosažení responzivnosti webové aplikace. V této práci je také porovnávána úroveň responzivnosti různých frameworků.
 - Problematika, kterou se Hnízdil Jakub zabývá, je důležitou součástí této bakalářské práce. Ve vyvíjené aplikaci je využíván framework, který zde vyšel jako nejvíce responzivní, společně s technologiemi, které jsou zde zmíněny. Tato práce se zabývá pouze teoretickou částí dané problematiky, nikoli praktickým příkladem na aplikaci.
- „Elektronický obchod v Nette“ [5] od Okosy Michala
 - Okosa Michal vytvořil webovou aplikaci sloužící jako e-shop a popsal, jak postupoval při vývoji této aplikace.
 - Tato práce je typově nejbližší k cíli, o který je zde usilováno. Jedná se pouze o rozdílnou problematiku. V implementaci je využíváno stejného frameworku, jako v práci „Elektronický obchod v Nette“ a různé techniky popsané v této práci byly nápomocné při vývoji aplikace.

Kromě výše uvedených prací, existují i další závěrečné práce řešící problematiku vývoje webových aplikací, ovšem tyto patří mezi nejpodobnější a nejvíce související s problematikou této práce.

1.2 Analýza související literatury

Dále je důležité nalézt odbornou literaturu, řešící tuto problematiku. Zde byly opět využity zdroje dodávané knihovnou Vysoké Školy Ekonomické. Při vyhledávání byla použita stejná hesla, jako při hledání závěrečných prací. Nebyly nalezeny žádné práce, řešící stejnou oblast problematiky. Byly ovšem nalezeny odborné publikace řešící vývoj webových aplikací. Jedná se o následující díla:

- „Building a Web Application with PHP and MariaDB: A Reference Guide“[6] od Sriparasa Sai Srinivas
 - V této knize je popsáno, co jsou technologie PHP a MariaDB, a jak postupovat při vývoji webové aplikace, která využívá tyto technologie.
- „HTML5 and CSS3 Responsive Web Design“[7] od La Grone Benjamin
 - Tato práce definuje, jak je možné dosáhnout responzivnosti webových aplikací za použití technologií HTML5 a CSS3.
- „Mobile-first Bootstrap“[4] od Magno Alexandre
 - Mobile-first Bootstrap představuje rozhraní Bootstrap. Definuje schopnosti tohoto rozhraní a uvádí, jak toto rozhraní umožňuje a zjednodušuje vývoj pro mobilní telefony.

Tyto odborné publikace velmi pomohly při vývoji této webové aplikace. Mnoho zde získaných znalostí bylo následně využito při návrhu a implementaci webové aplikace.

1.3 Závěr analýzy

V oblasti vývoje webových aplikací existuje mnoho prací definujících, jak postupovat při vývoji aplikací a jak dosáhnout optimalizace stránek pro mobilní telefony. Žádná z těchto prací se však nevztahuje k problematice, kterou se tato práce zabývá. V této práci využijeme veškeré znalosti z výše uvedených zdrojů k vytvoření aplikace, která umožní zvýšit efektivnost dovozu a svozu výpočetní techniky při rozsáhlých akcích.

2 Analýza existujících aplikací

Než začneme vývoj nové aplikace, je důležité zjistit, zda již neexistuje alternativa, která by vyřešila veškeré požadavky uživatele. Není zapotřebí „znovu vynalézt kolo“, pokud by alternativa existovala, ale byla pouze zadavatelem přehlédnuta. Při hledání podobných existujících aplikací byly vytvořeny tři kategorie nejvíce odpovídající zadání. Jedná se o kategorie: Aplikace pro organizaci týmů, Logistická aplikace pro logistické firmy, a Nástroje pro jednoduché vytvoření vlastní logistické aplikace. Existující žebříčky těchto kategorií byly analyzovány a z každé kategorie byly vybrány aplikace nejvíce odpovídající zadaným požadavkům. Nalezené aplikace budou následně oznámkovány dle naplnění určených kritérií. Známky jsou následující:

- 1 – Naprosto nedostačující/Nemožné použít
- 2 – Nedostačující/Narušuje to chod procesu
- 3 – Stěží dostačující/Možné použít bez kritických narušení procesu
- 4 – Dostačující/Při implementaci jsou lehké problémy
- 5 – Zcela dostačující/Není problém při implementaci.

Aby bylo možné aplikaci použít, nesmí mít v hodnocení známku 1, jelikož je z popisu známky možné vidět, že v tom případě stav této funkcionality definuje aplikaci jako nepoužitelnou. Pokud je ohodnocena známkou 2, musí mít aplikace dostatek ostatních výhod, aby mohla být použita i přes narušení chodu procesu.

Kromě těchto hodnocení, bude mít každá aplikace definováno, zda má veškeré požadované funkcionality, či zda některé chybí.

Kritéria pro porovnávání:

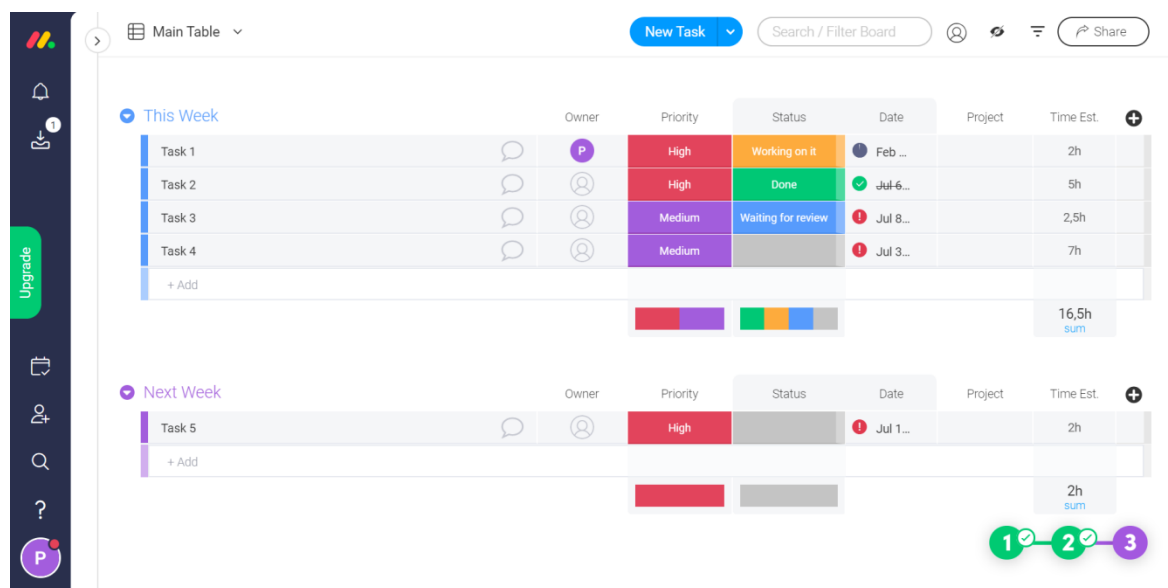
- **Instalace** – udává míru složitosti potřebnou k úvodnímu nastavení aplikace pro využití.
- **Naplnění požadavků** – do jaké míry je aplikace schopna naplnit požadavky firmy definované v kapitole 5). Nejdůležitější kritérium pro aplikaci.
- **Mobilní aplikace** – zde je hodnocena existence a přehlednost mobilní verze této aplikace.
- **Uživatelské rozhraní** – toto kritérium definuje přehlednost a rychlost navigace v uživatelském rozhraní aplikace.
- **Cena** – účelem této aplikace je neztrácet peníze časovými prodlevami na straně dispečera, proto je nutné, aby cena aplikace nepřesáhla možný užitek vzniklý ušetřením času techniků.

Autor zde podotýká, že vzhledem k absenci hlubších znalostí těchto aplikací mohou existovat funkce, které vyřeší zde kritizované problémy. Tyto funkce však není možné nalézt základní analýzou s využitím veřejně přístupných informací, návodů a recenzí dodávaných společnostmi provozujícími tyto aplikace. Aplikace byly hodnoceny samostatně bez asistence.

2.1 Monday.com

Monday.com je univerzální webová aplikace pro organizaci projektů. Umožňuje vytváření tabulí s jednotlivými úkoly. Je navržena tak, že pouze změnou nastavení a vzorů lze vyřešit mnoho různých problematik. Docházka zaměstnanců, kanban, přehled stavu práce týmů, čistá změna vzoru umožní vaši aplikaci úplně rozdílné využití[8](Volně přeloženo).

Tato aplikace je představena jako platforma pro organizaci práce postavená pro společnosti. Aplikace je definována jako perfektně škálovatelná, plně upravovatelná a naprosto intuitivní[8].



Task	Owner	Priority	Status	Date	Project	Time Est.
This Week						
Task 1	P	High	Working on it	Feb ...		2h
Task 2		High	Done	Jul 6...		5h
Task 3		Medium	Waiting for review	Jul 8...		2,5h
Task 4		Medium		Jul 3...		7h
						16,5h sum
Next Week						
Task 5		High		Jul 1...		2h
						2h sum

Obrázek 1 - Tabule Monday.com – Monday.com [8]


Při analýze této aplikace byla nejprve analyzována obtížnost instalace. Samotné využití aplikace je jednoduché, stačí se přihlásit na stránku Monday.com a uživatel má hned přístup ke svým tabulkám. Ovšem nastávají problémy, jakmile se uživatel dostane k úpravě tabulí pro využití na potřeby práce. Neexistují žádné připravené vzory k tomu, co zadavatel potřebuje, proto je zapotřebí vytvořit vlastní vzor. Vytvoření vzoru pro naše potřeby je jednoduché na pochopení, ale časově velmi náročné. Po vytvoření tohoto vzoru, má uživatel možnost si tento vzor uložit a při budoucím použití ho pouze načíst.

Dále bylo analyzováno, zda aplikace naplňuje požadavky firmy. Aplikace podporuje načítání dat z excelu, což je jeden z požadavků. Bohužel nelze dosáhnout zobrazování záznamů dle skupin uživatelů. Další problém je, že není možné různým uživatelům přiřadit rozdílné pravomoci ohledně jednotlivých řádků a sloupců. Také je velkou nevýhodou, že pro sdílení tabulek s návštěvníky je zapotřebí pokročilého platebního plánu. V této aplikaci je pro potřeby firmy příliš problémů a nedostatků.

Třetím kritériem je mobilní aplikace. Toto je prostor, kde se Monday.com dostává do popředí. Mobilní aplikace Monday.com je velmi přehledná a responzivní bez jakéhokoliv problému.

Uživatelské rozhraní Monday.com je kvalitně propracované, vysoce intuitivní a na pohled příjemné. Jak je možné vidět na obrázku 1, rozhraní je přehledné a funkční.

Jako poslední z kritérií byla analyzována cena. Zde bohužel skončila jakákoliv možnost využití této aplikace pro potřeby firmy. Jak je možné vidět na ceníku v obrázku 2, platební systém Monday.com počítá s měsíční, nebo roční platbou za každého uživatele. Pro potřeby firmy by byl zapotřebí minimálně jeden účet úrovně Pro, následně za každý tým minimálně jeden účet Basic. Vzhledem k počtu týmů a častosti potřeby této aplikace, je velmi jednoduché zjistit, že jakékoliv možné peníze ušetřené z používání této aplikace budou pohřbeny hluboko pod poplatky potřebnými k provozu této aplikace. Je možné, že součástí platebního systému „Enterprise“ by vznikla nabídka, která by byla více akceptovatelná, ale stále to cenově přesahuje přijatelné hodnoty.

Pick your plan	Basic	Standard	Pro	Enterprise
	\$49 USD / month billed monthly	Most Popular \$59 USD / month billed monthly	\$99 USD / month billed monthly	 Contact our sales team

Obrázek 2 - Cenový plán Monday.com za pět uživatelů – Monday.com [8]

Nyní nastává čas ohodnotit tuto aplikaci známkami.

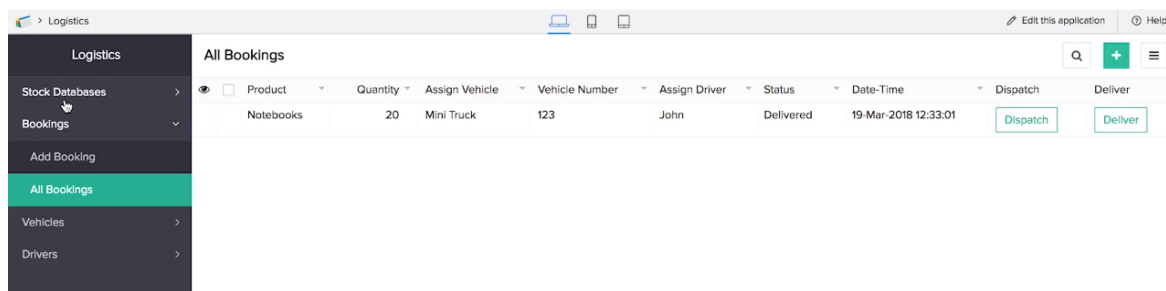
- Instalace - 3
- Naplnění požadavků - 1
- Mobilní aplikace - 5
- Uživatelské rozhraní - 5
- Cena - 1

Závěr: Aplikace je pro použití firmy naprosto neakceptovatelná z důvodu nenaplnění požadavků a příliš vysoké ceny využití.

2.2 ZohoCreator

Počínaje sběrem dat, konče přeměnou informací na znalosti, ZohoCreator zvládne všechno. Automatizace vašich procesů, dodání přístupů, co vaši kolegové potřebují, vizualizace informací skrz zprávy, toto všechno hlavně odkudkoliv. Řekněte aplikaci a ZohoCreator ji obsahuje: připravenou na použití nebo připravenou k postavení.[9] (Volně přeloženo).

Takto zní úvod přehledu aplikace ZohoCreator. Ve skutečnosti se jedná o rozhraní, které vám dodá stavební bloky a zjednoduší jakékoliv kódování potřebné pro automatizaci vašich procesů.



Obrázek 3-Logistický nástroj ZohoCreator – ZohoCreator [9]

Pojďme si tuto aplikaci prohnat skrze kritéria. Podobně jako u Monday.com, ZohoCreator je možné spustit hned z webové aplikace bez jakékoliv instalace na váš počítač. Oproti Monday.com je, ale příprava tabulí pro potřeby zadavatele, kde tato aplikace uvízne. O ZohoCreator by se dalo říci, že spíše než o aplikaci, se jedná o rozhraní, které může uživatel využít pro vytvoření vlastní aplikace. To znamená, že je zapotřebí, aby při každé instalaci byla vytvořena aplikace, která zmizí, jakmile vám vyprchá měsíční platba. Pokud by firma využívala tuto aplikaci častěji, toto by byla skvělá volba. Bohužel vzhledem k tomu, že tato aplikace bude použita jednou až třikrát za rok, potřeba vytvoření aplikace po každém vytvoření účtu se stává neúnosnou.

Vzhledem k tomu, že tuto aplikaci si uživatel vytváří sám s použitím dodaných nástrojů, je možné dosáhnout veškerých požadavků, které zadavatel potřebuje, tím že si aplikaci sám navrhnete.

ZohoCreator dodává nástroje i pro vytvoření vlastní mobilní aplikace, kde uživateli stačí pouze sestavit části, které potřebuje a může aplikaci hned využít dle potřeb, které si nastaví.

Uživatelské rozhraní aplikace je přehledné a vysoce konfigurovatelné. Jedna z možností zobrazení se nachází na obrázku 3.

Samotná cena je přátelská. Uživatel platí pouze za přístup k nástrojům a službám aplikace ZohoCreator. Za samotné uživatele a zaměstnance již uživatel neplatí. Problémem je, že se opět jedná o měsíční platbu. Oproti ostatním příkladům je cena této aplikace již mnohem akceptovatelnější. Firma opět nabízí flexibilní ceny pro společnosti, což by mohlo vést k ceně, kterou by firma byla ochotná dát. Ceník je možné si prohlédnout v obrázku 4.

<p>BASIC</p> <p>€15</p> <p>/user</p> <p>BEGIN TRIAL</p>	<p>PREMIUM</p> <p>€30</p> <p>/user</p> <p>BEGIN TRIAL</p>	<p>ULTIMATE</p> <p>Available only in yearly plan</p>	<p>ENTERPRISE</p> <p>Custom Pricing</p> <p>CONTACT US</p>
--	--	--	--

Obrázek 4 – Měsíční cenový plán ZohoCreator – ZohoCreator [9]

Finální skóre tedy je následující

- Instalace - 1
- Naplnění požadavků – 5
- Mobilní aplikace – 5
- Uživatelské rozhraní – 5
- Cena – 4

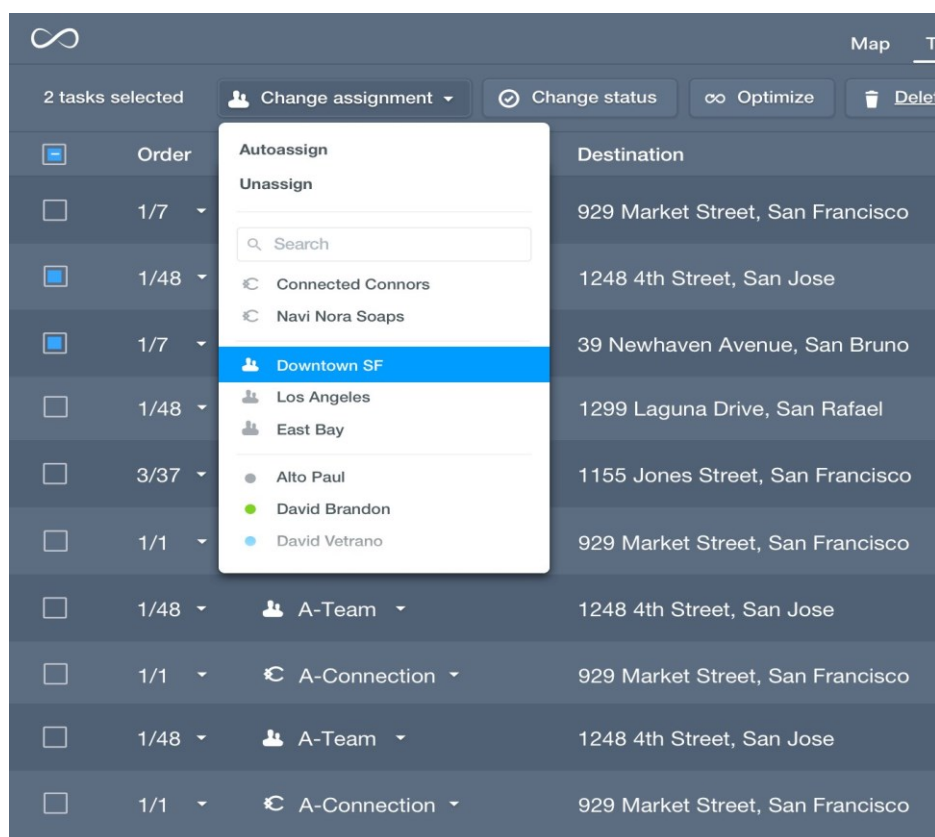
Tato aplikace nechává autora vysoce rozpolceným. Na jednu stranu se jedná o vysoce konfigurovatelné rozhraní usnadňující mnohé problémy, které by mohly vzniknout při vlastní výrobě. Na druhou stranu by samotné sestavování aplikace a vyznání se v nástrojích vedlo k neskutečnému množství práce. Většímu, než by osoba zběhlá ve vývoji webových stránek věnovala vytvoření vlastní aplikace.

Tuto aplikaci by autor doporučil pro uživatele, mající základní, povrchní přehled IT.

2.3 OnFleet

Světově nejpokročilejší platforma pro dodávky na poslední míli. Heslem této firmy je: Pozdravte jednodušší operace, chytřejší rozhodnutí a příjemnější zákaznické zkušenosti [10]. (Volně přeloženo).

OnFleet je populární řešení pro logistické operace pocházející ze San Franciska. Umožňuje uživateli plánovat trasy, sledovat dodávky, držet statistiky dodavatelů a mnohem více.



Obrázek 5-OnFleet logistika – OnFleet [10]

Podobně jako v předchozích aplikacích jsou zde postupně analyzována jednotlivá kritéria.

První část je instalace. OnFleet je také webová aplikace a umožňuje využívat aplikaci bez jakékoliv dlouhé instalace, postačí pouze přihlásit se na webovou stránku. Nastavení tabulek a informací je také relativně jednoduché, jakékoliv propojení s vnějším světem však potřebuje aspoň základní znalosti kódu. OnFleet má vlastní vysoce kvalitně dokumentované API, ke kterému je možné se připojit, což za cenu zvýšení obtížnosti velmi rozšiřuje možnosti této aplikace.

Dále máme dosáhnout požadavků. Tato aplikace dokáže téměř vše, co je požadováno. Hlavním problémem zde je, že aplikace počítá pouze s organizací vaší firmy a absolutně neřeší stranu klienta, kromě informování o doručení položky. Klient nemá přehled o svých lokalitách, což trochu ubírá na efektivnosti aplikace, ale není to neakceptovatelné. Především vzhledem k existenci kvalitní API, s jejíž asistencí a trochou programování je možné tento problém vyřešit.

Mobilní aplikace OnFleet je velmi intuitivní s kvalitním designem. Zde dávám aplikaci nejlepší známku.

Jak je možné vidět na obrázku 5 uživatelské rozhraní aplikace je vysoce efektivní a přehledné. Chybí zde však rozhraní pro klienty. Proto musím této aplikaci v této funkcionalitě snížit známku.

Poslední a velmi důležitým kritériem je cena. Bohužel zde OnFleet podobně jako Monday.com končí. Jak je možné vidět v ceníku nacházejícím se v obrázku 6, aplikace opět

počítá s integrováním aplikace do vaší firmy a každodenním používáním, a podle toho také vypadá cenový plán. Jedná se o měsíční platby, jejichž výše se kromě úrovně plánu pohybuje také podle počtu aktivit provedených za měsíc. Přesněji řečeno máte v základním plánu 1000 aktivit a za každou další platíte 0.15 \$ za aktivitu.

	Starter	Basic	Premium	Professional
Monthly price	\$199	\$449	\$999	\$2,499
Included tasks	1,000	2,500	5,000	12,500
Additional tasks	\$0.15	\$0.18	\$0.20	\$0.20

Obrázek 6- Měsíční cenový plán OnFleet – OnFleet [10]

Finální skóre aplikace je následující:

- Instalace - 4
- Naplnění požadavků – 4
- Mobilní aplikace – 5
- Uživatelské rozhraní – 4
- Cena – 1

Tato aplikace je vysoce kvalitní a naplnila by hlavní část potřeb pro firmu, ovšem z důvodu neakceptovatelného cenového plánu se nejedná o možné řešení.

Závěr analýzy:

Tabulka 1 - Analýza aplikací (vlastní tvorba)

Kritéria	Monday.com	ZohoCreator	OnFleet
Instalace	3	1	4
Naplnění požadavků	1	5	4
Mobilní aplikace	5	5	5
Uživatelské rozhraní	5	5	4
Cena	1	4	1

Většina aplikací na trhu pro udržování týmů, nebo logistické operace na trhu počítá s tím, že firma se zabývá pouze touto aktivitou každý den a také podle toho plánuje schopnosti a ceny své služby. Jediné další možnosti jsou rozhraní, které vám umožní vytvořit si vlastní aplikaci. Bohužel toto řešení by stále požadovalo využití programátora pro sestavení potřebné aplikace a mělo by to navíc tu nevýhodu, že veškerý kód by byl závislý na službách firmy, které byste museli měsíčně platit poplatek za možnost využití jejich nástrojů.

Z tohoto důvodu se firma rozhodla vytvořit jednoduchou „Budget“ aplikaci kterou si v případě potřeby zprovozní na vlastních serverech.

3 Metody a způsob řešení práce

Na začátku jakéhokoliv projektu je důležité rozhodnout, jak se bude postupovat k jeho vyřešení. Existuje mnoho různých metodik pro vedení práce. Pro účely této práce, byl z důvodu menšího projektu a jednočlenného týmu vybrán základní vodopádový model.

3.1 Vodopádový model

„Vodopádový model je nejstarší a nejznámější model životního cyklu vývoje programu.“
[11]

Jedná se o model, který je založen na stupňovitosti vývoje, které na sebe přesně navazují a v původní verzi modelu, nemohou, jakkoliv přecházet. Model se stává z 5 jednotlivých a rozdílných stupňů. Tyto stupně se nazývají: Požadavky, Návrh, Implementace, Verifikace a Zprovoznění. V této kapitole budou tyto jednotlivé stupně popsány.

Požadavky

V této části proběhne analýza problematiky, a od zadavatelů se zjistí, jaké jsou přesné požadavky a specifikace požadované na výstupní aplikaci. Po zjištění se tyto informace nejprve zanalyzují a výsledky těchto analýz budou uloženy do potřebných dokumentů pro další využití ve vývoji.

Při analýze požadavků probíhala komunikace se zákazníkem pro podrobnější definici problematiky. Během této komunikace byly vytvořeny tabulky požadavků pro aplikaci. Také byl vytvořen UML diagram způsobů užití ukazující rozdělení funkčních požadavků dle rolí. Výsledky jsou k dispozici v kapitole 5.

Návrh

Návrhem se zde myslí vytvoření postupu, který bude využit při implementaci kódu. Přesněji řečeno se v této části vytvoří dokumenty definující architekturu aplikace. Z těchto dokumentů se vychází během implementace kódu. Jedná se například o dokumenty, definující strukturu programu nebo databáze.

Pro návrh aplikace byly využity následující metody. Pro návrh vzhledu webové aplikace byly vytvořeny Wireframe pomocí aplikace diagrams.net[12]. Dále byly vytvořeny EER modely databáze za pomoci aplikace MySQL Workbench[13]. Pro definici architektury a jednotlivých tříd, byla využita aplikace EnterpriseArchitect[14]. Výsledky této části jsou představeny v kapitole 6.

Implementace

V této části se přechází do produkčního cyklu. V této části se postupuje podle dokumentů vytvořených v předchozích částech. Nastává zde realizování navrhnutých struktur do vybraných technologií. Výsledkem této realizace by měla být hotová aplikace připravená pro použití.

Při implementaci kódu bylo nejprve připraveno vývojové prostředí, které bude definované později. Následně byl návrh převeden do kódu. Výsledek implementace je možné nalézt v kapitole 7.

Ověření

V této části probíhá testování hotové aplikace pro ověření, zda splňuje veškeré požadavky zadané v prvním stupni vodopádového modelu. Také se zde podchytí veškeré chyby, které unikly v implementačním stupni. Výsledkem této části by měla být aplikace splňující veškeré požadavky s žádným, či minimálním počtem chyb.

Pro danou práci bude realizováno testování responzivnosti na straně vývojáře a základní kontrola splnění požadavků. Pokročilé testování funkcionality již se uskuteční na straně klienta. Více o této kapitole se nachází v kapitole 8.

Udržování

Zde je specifikováno, jak bude aplikace zavedena do provozu u klienta, a jak bude provozována. Jakých postupů bude využito v případě úprav, a také jak budou opravovány jakékoliv další chyby.

Udržování aplikace již není součástí této práce

3.2 Vývojové prostředí

Další důležitý bod pro zvýšení efektivnosti práce je kvalitní nastavení vývojového prostředí. V této kapitole si využité prostředí popíšeme.

3.2.1 Textový editor

Pro kvalitní práci s kódem je důležité využívat editor, který je schopný inteligentně rozpoznávat kód a asistovat při vývoji. Pro vývoj této aplikace byl použit produkt PhpStorm. Aplikace PhpStorm je ucelené vývojové prostředí dodávající mnohé funkcionality včetně pokročilého editoru kódu. Jedná se například o funkcionality automatického importu tříd, propojení s verzovacím systémem, automatické indentace kódu, či přehledu nad celým projektem. Jedná se o vývojové prostředí, vydané společností JetBrains, specializované na vývoj aplikací v jazyce PHP. Více o této aplikaci lze nalézt na oficiálních stránkách JetBrains. [15]

3.2.2 Verzovací systém

Při vývoji se často stává, že změnou kódu se aplikace stane nepoužitelnou. Z tohoto důvodu je nezbytné vytvářet zálohy aplikace v různých stavech vývoje. Pro tento účel existuje open-source aplikace Git. Tato aplikace dodává mnohé možnosti pro vývoj aplikace, tj. udržovat záznamy o jednotlivých verzích aplikací, vytvářet rozdílné větve aplikací pro řešení rozdílných problémů a mnohem více. Další informace lze získat na stránkách dokumentace Git.[16]

Další důležitou částí ukládání informací je ochrana proti chybě vašeho úložného zařízení. Z tohoto důvodu je vhodné využívat verzovací systém na vzdáleném serveru, určeném čistě pro ukládání dat. Existují mnohé Cloud service či aplikace, které umožňují tuto službu. Pro vývoj aplikace pro danou práci byla vybrána populární služba GitHub. Informace o této službě je možné nalézt na stránkách dokumentace Github[17]. Při vývoji byl používán privátní repozitář. Po dokončení bude odevzdaná verze práce uložena do veřejného repozitáře určeného čistě pro danou verzi. Repozitář bude pod open-source licencí MIT pro povolení dalších rozšíření tohoto kódu. Informace o tomto repozitáři jsou v příloze.

3.2.3 HTTP Server

Pro testování a vývoj aplikace je nutné mít připravené prostředí, na kterém aplikace bude běžet. Bez existence tohoto prostředí je vývoj aplikace nemožný. Pro potřeby vývoje dané aplikace postačil lokální server. Pro dosažení plné funkcionality vyvíjené aplikace na webové stránce bylo zapotřebí, aby kromě http serveru běžely i aplikace MariaDB a PHP. Vzhledem k operačnímu systému Windows, byl zvolen předpřipravený balíček WAMP. Tento balíček automaticky připraví prostředí s aplikacemi odpovídající svému názvu. Windows, Apache, MySQL/MariaDB a PHP.[18]

3.2.4 SMTP Server

Součástí této aplikace je i posílání upozornění skrz email. Pro naplnění požadavků bezpečnosti dnešních poštovních aplikací je zapotřebí propojit aplikace s SMTP serverem. Pro testování byla využita služba Mailtrap.io poskytující službu, tvářící se jako SMTP server. Tato služba však místo odeslání zprávy, tuto zprávu zachytí a uloží do poštovní schránky. Jedná se o naprosto ideální službu pro testování odesílání emailových zpráv. Více o této službě můžete najít na oficiálních stránkách. [19]

4 Webové aplikace optimalizované pro chytré mobilní telefony a specifika jejich vývoje

Při vývoji aplikací pro mobilní telefony je zapotřebí si uvědomit, jak se telefony odlišují od stolních či přenosných počítačů. Jedná se o hlavně o následující vlastnosti:

- Menší velikost obrazovky.
 - I největší mobilní telefony, mají stále mnohem menší plochu pro zobrazení obsahu než jakékoliv počítače.
- Menší výkonnost.
 - Byť v dnešní době jsou již nejvýkonnější telefony obdobně silné jako slabší počítače, mnoho uživatelů stále používá telefony, které nejsou příliš výkonné.
- Pohyblivost/Připojení k internetu.
 - Mobilní telefony jsou často používány v místech, kde neexistuje pevné (lokální) připojení k internetu, což vede k závislosti na mobilních datech. V aktuálních časech zavádění sítí 5G jsou tyto problémy již z větší části překonány, bohužel vzhledem k vysokým cenám mobilních operátorů v Česku, mnoho uživatelů stále využívá pomalá data.

4.1 Velikost obrazovky.

Jak zde již bylo uvedeno, menší velikost obrazovky vede k menší ploše pro zobrazení obsahu stránky. Z tohoto důvodu je potřeba stránku navrhnout takovým způsobem, aby byly stále plně zobrazitelné a čitelné. Pro dosažení tohoto stavu jsou důležité následující dva principy.

Metodika Mobile First

Metodika Mobile First říká, že pro vývoj stránky je lepší nejprve začít na prostředí, které dodává více omezení a teprve ve chvíli, kdy je aplikace schopna plně fungovat na těchto platformách přidávat další rozšíření. Tomuto principu říkají „Progressive advancement“ – Postupné vylepšování, pro přesnější definici zde využijí citát:

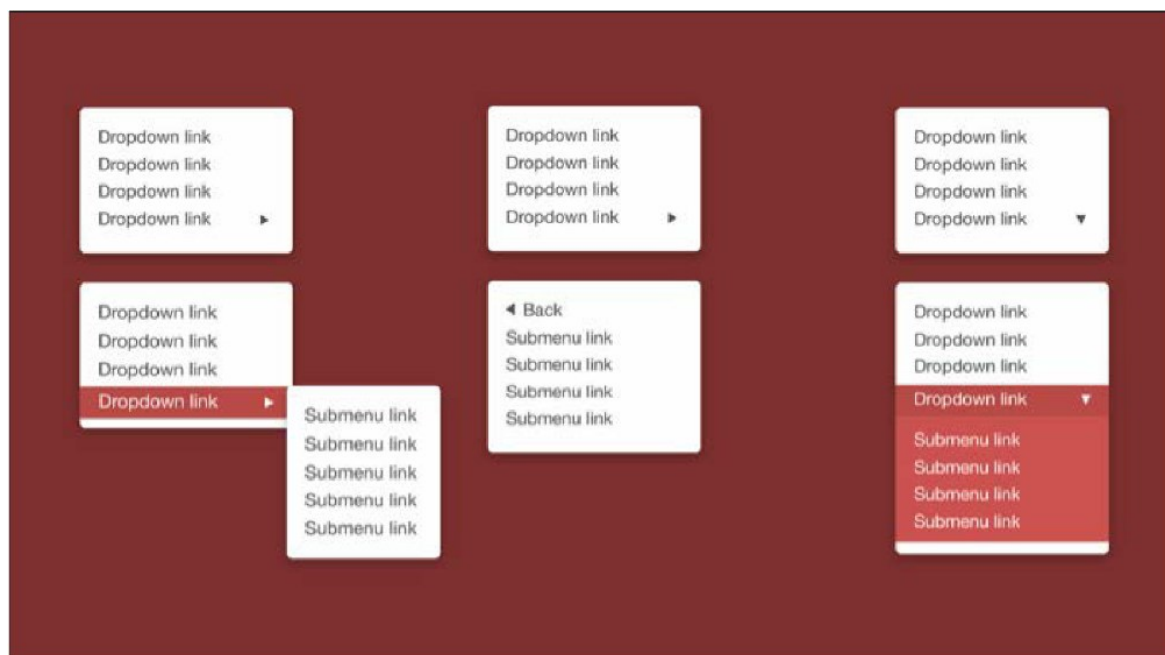
Mobile First zapadá právě do filosofie postupného vylepšování. Návrh webu začíná pro zařízení s největším množstvím omezení a postupně se web obohacuje o další funkčnost, podle toho, co dovolují zařízení a prohlížeče s pokročilejšími vlastnostmi[2]

Opakem tohoto principu je tzv. „Gracefuldegradation“ – Elegantní degradace. Zde se nejprve vytvoří stránka pro platformu, která přivede uživateli nejlepší zážitek, a pro ostatní platformy je postupně zbavovaná méně důležitějších součástí, než dosáhne potřebné jednoduchosti při stále funkčnosti aplikace. Více o této metodice je možné nalézt v práci Jana Pospíšila – Optimalizace webu pro mobilní zařízení [2].

Responzivita

Responzivita je jedna z nejdůležitějších vlastností pro zprovoznění webových stránek a aplikací pro mobilní telefony. Jedná se o navrhnutí stránky takovým způsobem, aby byla schopna reagovat na různé velikosti obrazovek, a následně dle toho změnit svoji strukturu. Velmi kvalitní ukázka této problematiky je v následujícím obrázku a citátu:

„Jak ukázáno v následujícím obrázku, je vidět menu nalevo, které se rozevírá do dalších podmenu napravo. Je zde i druhá možnost přeměnit toto na „převrácenou“ verzi, kde můžete navigovat skrz předmět do jiného, jako kdybyste přeskakovali mezi stránkami. Třetí řešení funguje jak pro stolní počítače tak pro telefony, kdy otevřená podmenu jsou vždy viditelná.“[4] (volně přeloženo z odkazu)



Obrázek 7- Responsive Dropdown – Mobile-first Bootstrap [4]

Jak je možné vidět, existují různé způsoby jak vyřešit stejný problém. Každý ze způsobů má své výhody a nevýhody. Responzivnost umožní nahradit využívanou metodu ve chvíli, kdy přestane být funkčním řešením pro dané zařízení. Jedná se také například o základní navigační okno dodávané rozšířením Bootstrap, které dokud může, zobrazuje tlačítka zvlášť. Ve chvíli, kdy tyto tlačítka není schopné zobrazit, tak je všechny přesune do rozevíracího menu.

Ve chvíli, kdy je aplikace plně responzivní, neboli plně funkční pro veškeré plánované velikosti zařízení a prohlížečů, máme hlavní část vývoje pro mobilní telefony za sebou. Více o responzivnosti lze nalézt například v knize *Mobile-first Bootstrap* [4], nebo v pracích Jakuba Hnízдила, *Responzivní webdesign* [3], či v práci *Webové mobilní aplikace pro správu osobních financí* [20], kde je této problematice věnována jedna kapitola.

4.2 Výkon telefonu

Mnoho uživatelů stále využívá starší telefony, které mají mnohem slabší schopnost zpracovávat data než ostatní zařízení. Navíc jazyk Javascript, který řeší zpracovávání dat na straně klienta, není optimalizován pro použití na mobilních telefonech. Mnoho rozšíření tyto problematiky řeší, například o tomto píše kniha *Mobile-First Bootstrap*, která říká, že: *Javascriptové komponenty Bootstrap řeší chyby a problémy, které vznikají při použití Javascriptu na mobilních telefonech.*[4] (volně přeloženo ze zdroje)

Pro omezení využití nízkého výkonu telefonu je dobré zpracovávání dat řešit na serveru a u klienta tyto data pouze zobrazovat. Pokud je server dostatečně silný a optimalizovaný, toto povede k zrychlení zobrazení stránek na mobilních dat.

4.3 Připojení k datům

Tento problém se s neustálým rozvojem technologie zmenšuje, bohužel stav mobilních operátorů v Česku vede uživatele, stále užívat pomalé připojení k datům. Z tohoto důvodu je důležité co nejvíce zjednodušit obsah předávaných dat. Znamená to nemít ve stránkách zbytečné obrázky, videa a animace, a ty existující důležité kompresovat pro zmenšení objemu dat.

4.4 Závěr optimalizace

V této kapitole bylo představeno, které části vývoje je nejvíce zapotřebí brát v potaz pro optimalizaci stránky pro mobilní telefony. V dané aplikaci byly tyto problematiky vyřešeny následovně:

- Velikost obrazovky
 - Pro optimalizaci stránky na mobilní telefony byl vybrán framework Bootstrap. Tento framework je navrhnut pro maximální responzivnost.
- Výkon telefonu
 - Pro omezení množství probíhajících operací na straně uživatele, byla většina logiky přemístěna na stranu serveru. Jediné operace probíhající na straně klienta jsou generace tabulek z dat a interakce s touto tabulkou.
- Připojení k datům
 - Stránky jsou navrhnuty jednoduše. Na stránkách nejsou žádná zbytečná multimédia, která by navyšovala množství dat na stránce.

5 Analýza požadavků

V každém projektu je nutné přesně definovat, co a jak aplikace dokáže. Je potřebné vypsát požadavky klienta na aplikaci. Tyto požadavky budou v této práci rozděleny na dvě kapitoly. Nejprve zde budou nefunkční požadavky, které budou vypsány do tabulky. Následně za pomoci diagramu užití vytvořeném dle principů UML budou zobrazeny funkční požadavky.

5.1 Funkční požadavky

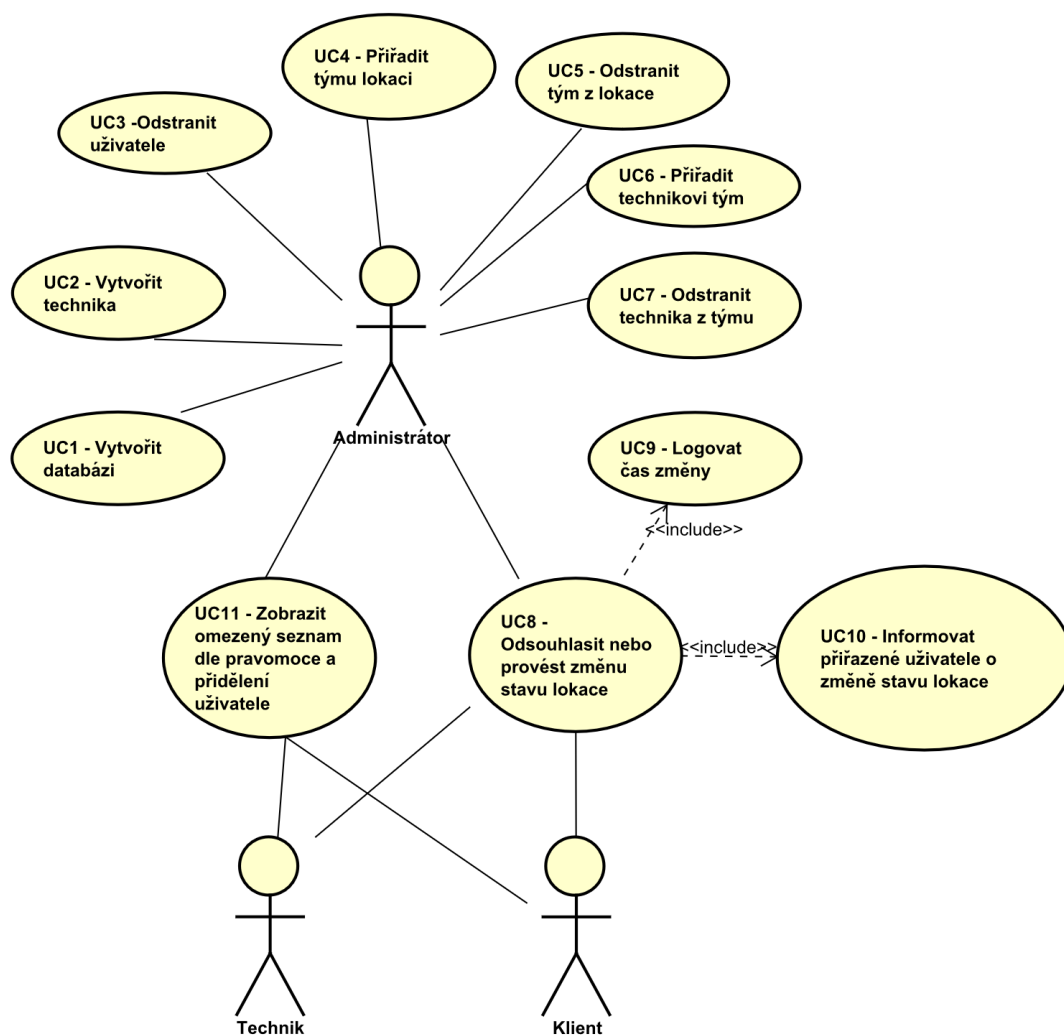
Prvními požadavky jsou funkční požadavky, tj. ty, které definují funkcionality, které bude schopna aplikace poskytovat.

Pro zobrazení funkčních požadavků byl využit diagramový jazyk UML pro znázornění přiřazení daných funkčních požadavků k potřebným rolím v aplikaci.

Tabulka 2 - Funkční požadavky (vlastní tvorba)

ID	Název	Popis	Use Case
FOO1	Naplnit databázi	Administrátor musí být schopen naplnit databázi předpřipravenými daty. Ideálně z Excel souboru	UC1
Foo 2	Vytvořit technika	Administrátor má v aplikaci schopnost vytvořit nového technika.	UC2
Foo 3	Odstranit technika	Administrátor má v aplikaci schopnost odstranit technika.	UC3
Foo 4	Přiřadit týmu lokaci	Administrátor má možnost přiřadit týmu lokaci.	UC4
FOO5	Odstranit tým z lokace	Administrátor má možnost odstranit tým z lokace.	UC5
Foo 6	Přiřadit technikovi tým	Administrátor má možnost přiřadit technika do týmu.	UC6
FOO7	Odstranit technika z týmu	Administrátor má možnost odstranit technika z týmu.	UC7
Foo 8	Odsouhlasit nebo provést změnu	Všechny role mají možnost provést změnu role na k sobě přiřazených lokacích.	UC8

	lokace	Role technik a klient mají tuto možnost omezenou pouze pod změny stavu, které obstarávají.	
F009	Logovat čas změny	Veškeré změny stavu budou zaznamenány s informacemi o času, uživateli provádějícím změnu, a změněné lokaci.	UC9
F010	Informovat přiřazené uživatele o změně stavu a lokace	Ve chvíli, kdy proběhne změna lokace, proběhne upozornění všech uživatelů k této lokaci přiřazených.	UC10
F011	Zobrazit omezený seznam	Každý uživatel má možnost zobrazit seznam lokací. Tento seznam je omezený pouze na lokace, které uživatel obstarává.	UC11



Obrázek 8 - Use Case Diagram - Autor

5.2 Nefunkční požadavky

Dále hovoříme o nefunkčních požadavcích. V těchto požadavcích je definováno, jak by měla aplikace fungovat. Přesněji řečeno sem patří definice hovořící o technických kritériích typu efektivita, obtížnost údržby, vlastnosti, ale i rychlost.

V následující tabulce jsou vypsány požadavky, které byly zadány jako důležité pro vývoj a provoz této aplikace.

Tabulka 3 - Nefunkční požadavky (vlastní tvorba)

ID	Název	Popis
N001	Levný provoz	Aplikace nesmí při provozu přesáhnout cenou za údržbu výšku úspor získaných jejím provozem.
N002	Rychlá odezva	Aplikace nesmí při základních krocích vytvářet vlastním provozem zpoždění vyšší jak 10 sekund za činnost.
N003	Jednoduchá instalace na vlastní server	Aplikaci včetně databáze musí být jednoduché zapnout a nastavit i pro zaměstnance se základní schopností programování.
N004	Optimalizace rozhraní pro technika na chytrý telefon.	Aplikace musí pro rozhraní technika mít kvalitní mobilní optimalizaci vzhledem k primárnímu přístupu těchto uživatelů z terénu.
N005	Přehlednost	Uživatelské rozhraní musí být přehledné a jednoduché pro zvýšení schopnosti orientace a tím i rychlosti v situacích kde se každá sekunda počítá.

5.3 Analýza aplikace

Je potřebné, aby tato aplikace měla tři rozdílné rozhraní. Aktuální rozhraní bude definováno dle role přihlášeného uživatele. Role uživatelů budou existovat tři: Technik, Klient a Administrátor.

5.3.1 Technik

Toto bude rozhraní pro technika společnosti. Zaměstnance, kteří obstarávají rozvoz a svoz potřebného vybavení pro jednotlivé klienty.

Tito technici budou mít k dispozici pouze přehled kanceláří přidělených týmu, ke kterému byl daný technik přidělen. V daném přehledu kanceláří bude mít zobrazené informace o kanceláři, jejím aktuálním stavu a také informace o kontaktních osobách těchto kanceláří.

Jak by asi tato stránka měla vypadat je znázorněno v následujícím wireframe.

Page 1

https://www.draw.io

LOGO

Profile Log Off

Filter by supervisor

Search

Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State

Janitor Phone	Branch Description	Branch Note	Change State		
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State

Footer ~ copyright

Obrázek 9 - Wireframe – Pohled technika ve větším zařízení – Autor

Případná alternativní podoba stránky pro technika využívající mobilní zařízení.

Obrázek 10 - Wireframe – Pohled technika v mobilním zařízení – Autor

V tomto zobrazení je vysoce důležité, aby technik byl schopen se rychle orientovat dle adresy a kódu dané kanceláře a zároveň viděl přehled všech stavů těchto kanceláří. Proto jsou tyto informace uloženy v primárních řádkách. V případě potřeby dalších informací, bude možné tyto řádky rozevřít a získat další skryté informace, včetně tlačítka umožňujícího technikovi informovat aplikaci a dispečera o dokončení činnosti v dané kanceláři a následném změnění stavu v databázi. Toto tlačítko se nachází v sekundárních (skrytých) řádcích pro snížení pravděpodobnosti náhodného zakliknutí tohoto tlačítka. Kromě toho při každém pokusu o změnu stavu uživateli vyskočí konfirmační okno žádající uživatele o ujištění se o rozhodnutí.

Vzhledem k menší velikosti mobilních telefonů rozevření řádku otevře více řádků s méně sloupci oproti větším aplikacím pro udržení potřebné čitelnosti. I zde se tlačítko pro změnu stavu nalézá v sekundárních řádcích, a i zde vyskočí konfirmační okno pro omezení možností nechtěných změn.

5.3.2 Klient

Zde se pod slovem klient hovoří o nadřízené kontaktní osobě dohlížející nad funkcí jednotlivých kanceláří pod jeho správou. Klient, podobně jako technik, bude mít pouze přístup k informacím a kancelářím, které sám spravuje.

Možnou podobu rozhraní pro roli klienta je možné nalézt v následujícím Wireframe.

The wireframe shows a web application interface for a client. At the top, there is a header bar with a 'LOGO' on the left, and 'Profile' and 'Log Off' links on the right. Below the header, there is a search bar labeled 'Search' and a filter bar labeled 'Filter by Team'. The main content area contains a table with 6 columns: 'Branch Code', 'Branch Address', 'Branch Leader', 'Branch Leader Phone', 'State', and 'Change State'. The table has 10 rows of data. Each row has a 'Change State' button in the last column. At the bottom of the page, there is a footer bar with the text 'Footer ~ copyright'.

Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State
Branch Code	Branch Address	Branch Leader	Branch Leader Phone	State	Change State

Obrázek 11 - Wireframe – Pohled klienta ve větším zařízení – Autor

Oproti pohledu technika, při pohledu klienta není potřebné znázornění tolika informací, proto se zde nalézají pouze primární řádek s veškerými potřebnými informacemi a tlačítkem pro změnu stavu. Tím se sice oproti rozhraní technika, kde se tato tlačítka nalézají v sekundárních řádcích, přijde o částečnou ochranu proti nechtěné aktivaci, zůstává zde však ochrana za pomoci konfirmačního tlačítka.

5.3.3 Administrátor

V rozhraní této role se bude nalézat většina funkcionalit této aplikace. Jak je možné vidět z diagramu užití, je potřebné, aby administrátor byl schopný nejenom vidět veškeré informace o všech kancelářích, ale také informace o uživatelích, týmech, klientech a některé z těchto informací i upravovat, ale také být schopen připravit samotnou databázi před každou akcí.

Jak z definice této role lze vidět, zde nebude stačit jedna stránka, která stačí ostatním rolím. Bude zde potřeba implementovat navigaci mezi jednotlivými stránkami řešící rozdílné problematiky. Návrh této stránky můžete vidět v následujícím Wireframe.

The wireframe shows a web application interface for an administrator. At the top, there is a browser-like header with 'Page 1' and a URL 'https://www.draw.io'. Below this is a navigation bar with a 'LOGO' and several menu items: 'Branches' (highlighted), 'Teams', 'Users', 'Database', 'Statistics', 'Profile', and 'Log Off'. The main content area features a 'Filter' input field and two radio buttons: 'By Supervisor' and 'By Team' (selected). Below the filter are two tables. The first table has 7 columns: 'Branch Code', 'Supervisor', 'Branch Address', 'Branch Leader', 'Branch Leader Phone', 'State', and a 'State' button. It contains 6 rows of data. The second table has 5 columns: 'Team Name', 'Team Leader Phone', 'Branch Description', 'Branch Note', and a 'StateName' dropdown menu. It contains 4 rows of data. At the bottom, there is a footer with the text 'Footer ~ copyright'.

Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State	State
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone	State	State

Team Name	Team Leader Phone	Branch Description	Branch Note	StateName
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone
Branch Code	Supervisor	Branch Address	Branch Leader	Branch Leader Phone

Obrázek 12 - Wireframe – Pohled administrátora ve větším zařízení – Autor

Jak je možné na tomto Wireframe vidět, oproti rozhraní technika a klienta jsou zde navíc navigační tlačítka, které administrátora přeměrují na dané stránky. Wireframe těchto dalších stránek bude možné vidět v příloze. V této hlavní stránce je potřebné, aby administrátor měl možnost nejenom tento stav změnit na další v pořadí, ale i jej vrátit na jakýkoliv z jiných existujících stavů, což bude umožněno za pomoci dropdown menu s možností výběru jiného stavu. Základní hodnota tohoto dropdown menu, bude vždy hodnota stavu o stupeň vyšší pro urychlení ovládání.

Zbylé Wireframe se již budou nalézat v příloze a zde budou pouze popsány. Na stránce Teams budou zobrazeny jednotlivé týmy techniků, které po rozbalení vypíší všechny techniky spadající pod daný tým. Také zde bude možné přejít do vygenerované stránky daného týmu, kde bude možno tento dále upravovat členy a kanceláře přiřazené tomuto týmu. Další stránka v navigaci je stránka users, kde bude mít administrátor možnost upravovat vlastnosti jednotlivých uživatelů, nebo také vytvořit zcela nového uživatele. Předposlední stránkou je zde stránka Database, kde má administrátor možnost naimportovat data do databáze z excel souboru vytvořeného přesně podle určené šablony, také má možnost vyexportovat existující data z databáze to takového excel souboru.

6 Návrh aplikace

Po kapitole o analýze potřebných požadavků a vytvoření definic aplikace přichází ve vodopádovém modelu na řadu samotné navrhnutí architektury vytvářené aplikace. Nejprve si představíme, které technologie budou použity pro vývoj aplikace. Následně proběhne návrh Databáze obstarávající uložení dat. Poslední část navrhne webovou aplikaci, která tyto data zobrazí a umožní uživatelům s těmito daty manipulovat.

6.1 Technologie využité při vývoji aplikace

Při výběru technologií byly vybrány začáteční technologie dle znalosti autora, a dané technologie byly rozšířeny dle potřeb. Vzhledem k tomu, že během studia autor dokončil předměty webové technologie a webové aplikace, získal základní znalosti technologií HTML5 CSS3 Javascript a PHP. Pro rozšíření těchto znalostí tyto technologie využil i při vývoji této aplikace.

6.1.1 Backend

Pro tuto část kódu byly vybrány technologie MariaDB pro správu databáze a framework Nette založeném na PHP. Při výběru databáze nastávala otázka, zda vybrat mezi technologiemi MySQL a MariaDB. Vzhledem k tomu, že MariaDB podporuje veškeré funkcionality MySQL, je stabilnější a nehrozí u ní zpoplatnění od Oracle. Z tohoto důvodu byla technologie MariaDB zvolena pro tuto část práce.

Pro samotnou logiku serveru již byl vybrán jazyk PHP. Pro usnadnění práce se autor rozhodl využít framework. Vzhledem k autorově neznalosti práce s frameworkem, se původně rozhodl nalézt jednoduchý framework, který by obsahoval veškeré potřebné funkcionality. Během výzkumu byl nejprve vybrán framework CodeIgniter4, který měl tyto požadavky splnit. Bohužel později během počátků vývoje se přišlo na to, že čtvrtá iterace frameworku CodeIgniter je nová verze, která je naprostým přepisem předchozích verzí. To znamenalo, že postupy a kvalitní dokumentace, které existovaly u předchozích verzí, by zde již nefungovaly. Vzhledem k tomu, že tato verze byla i relativně nová, znamenalo to, že neexistovala kvalitní dokumentace a podpora byla také velmi nekvalitní. Z těchto důvodů se přešlo na populárnější framework. Vybrán byl český framework Nette, který je v České republice velmi oblíbený. Velkou výhodou tohoto frameworku oproti CodeIgniter4 je kvalitní a rychlá odezva podpory i v českém jazyce.

Pro vývoj aplikace byly zapotřebí určité funkcionality, které není lehké vytvořit v základním kódu, proto bylo zapotřebí v kódu použít dodatečné knihovny:

- PhpSpreadsheet.

- Při práci s databází je potřebné pracovat s tabulkovými soubory vytvářenými v aplikaci Excel. Tato knihovna umožní při vývoji jednoduše s těmito soubory pracovat.
- Bootstrap-Tables.
 - Velké množství dat v této aplikaci je uloženo v tzv. „DataTable“, tj. tabulkách, které kromě zobrazení dat zvládají i další schopnosti, jako jednoduché vyhledávání, třídění dle sloupců aj. Pro vyřešení této problematiky byla využita knihovna Bootstrap-Table, která rozšiřuje schopnosti frameworku Bootstrap pro tabulky. Další výhodou využití této knihovny je využití responzivnosti přenesené z původního frameworku Bootstrap.
 - Alternativní knihovny pro problematiku datagrid/datatables byly knihovny Mesour Data sheets, či Ublaboo/ContributteDatagrids. První knihovna bohužel nebyla kompatibilní s verzí Nette, která je v aplikaci používána a druhá knihovna neposkytovala potřebované funkcionality. Toto vedlo k přesunutí tvorby tabulek na stranu klienta s knihovnou Bootstrap-Table.
- Nette.Ajax.js.
 - Samotný framework Nette sice má existující propojení Ajax, bohužel toto propojení je hlavně pro propojení s jazykem HTML a pro propojení se skripty Javascriptu nemá schopnosti, které jsou zapotřebí pro korektní funkci aplikace. Tato knihovna přináší mnohé zlepšení pro toto propojení.

6.1.2 Frontend

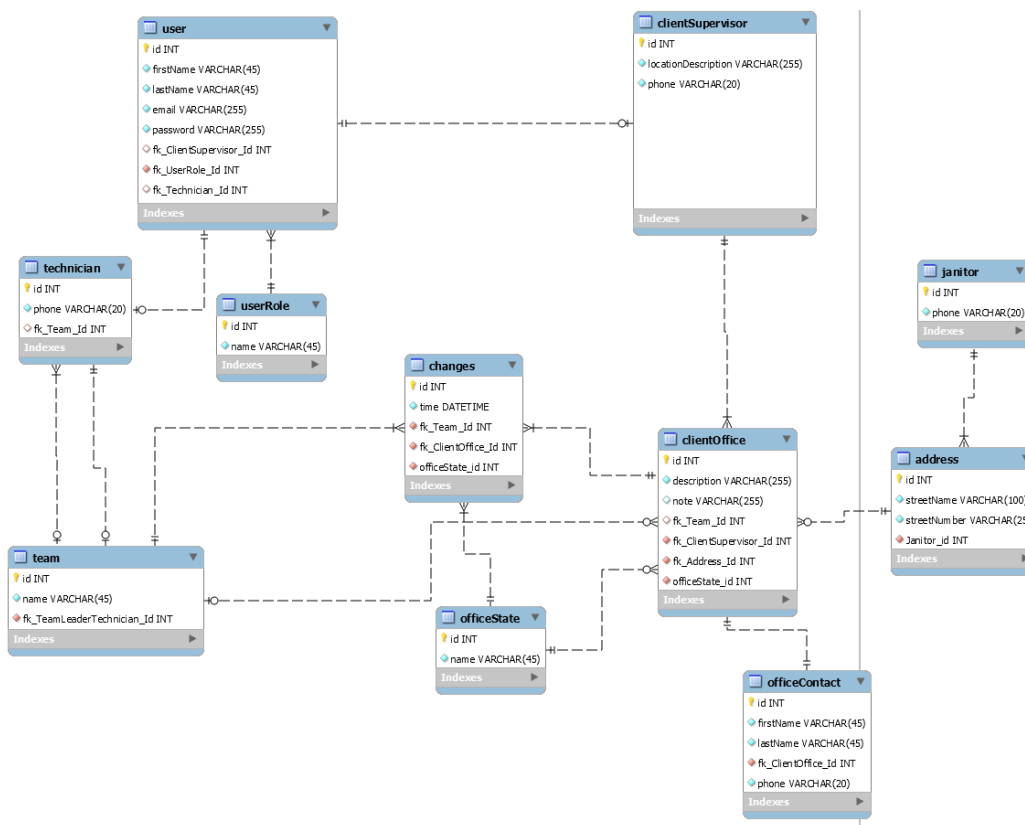
Pro část kódu, která generuje, co uživatel vidí, je v aplikaci využit šablonovací systém Latte, který je součástí frameworku Nette. Tento systém je zde využit pro rozšíření a zjednodušení základních jazyků HTML a CSS. Dále je zde využíván framework Bootstrap, který přináší zjednodušení dosažení responzivnosti stránek.

Pro omezení požadovaného výkonu od přístroje uživatele, a tím zpřístupnění aplikace pro více uživatelů, je omezeno množství zpracovávání dat na straně klienta.

6.2 Databáze

Databáze je strukturované uložení dat umožňující uživatelům efektivně vyhledávat uložená data. Nejčastěji využívané databáze jsou tzv. relační databáze založené na propojených tabulkách. Vzhledem k autorovým pozitivním zkušenostem s tímto typem databáze byl tento typ databáze využit i při vývoji této aplikace.

Databáze byla navržena za pomoci aplikace MySQL Workbench. Tato aplikace umožňuje navrhnout databázi za pomoci jejích nástrojů pro modelování EER. S pomocí této aplikace byl vytvořen následující model.

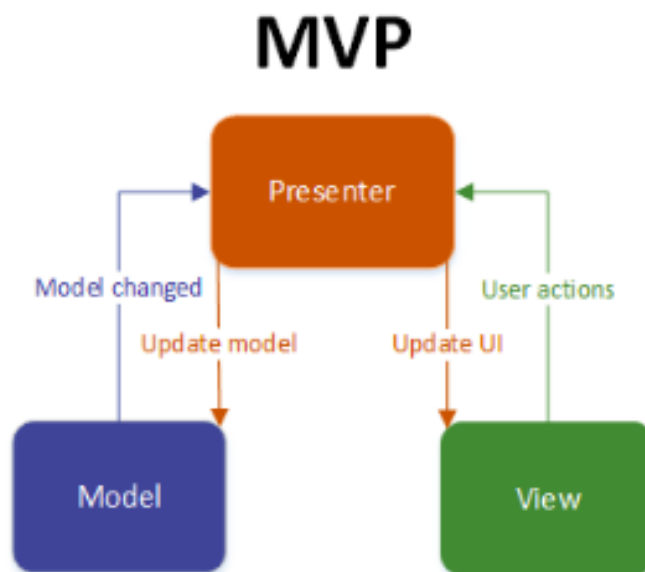


Obrázek 13 - EER model databáze – Autor

Tento model naplňuje všechny potřeby aplikace a MySQLWorkbench již dokáže z tohoto modelu vygenerovat potřebné skripty. Tyto skripty je zapotřebí lehce upravit vzhledem k lehkým rozdílům mezi originálním MySQL vlastněným Oracle, pro který tato aplikace byla vytvořena, a alternativní verzí MySQL – MariaDB vytvořenou původními vývojáři po zakoupení MySQL Oracle.

6.3 Webová aplikace

Dále je důležité navrhnout architekturu samotné webové aplikace. Pro vývoj této aplikace byla využita architektura typu MVP. Architektura MVP je založená na myšlence oddělení grafické (View) a logické (Model) části kódu. Pro propojení těchto částí kódu vznikla navíc kontrolní (Controller) část umožňující efektivnější propojení. Oproti Model-View-Controller, Model-View-Presenter nerozhoduje které View bude zobrazeno, jelikož má vždy přiřazeno právě jedno určité View. Také MVP neumožňuje View, jakkoliv přistoupit přímo model.[21]



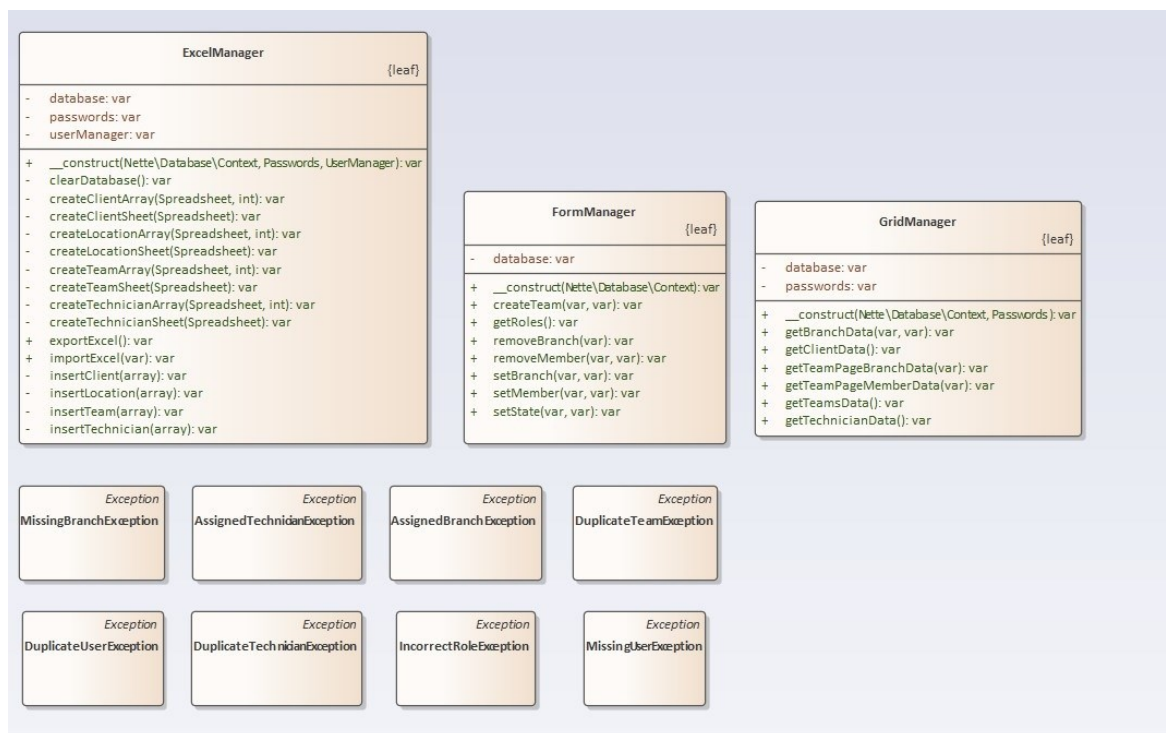
Obrázek 14 - grafické zobrazení MVP – Ben Elghali Beyram [21]

6.3.1 Model (Manager)

Část model obstarává hlavní logické operace aplikace a také veškeré propojení aplikace z databáze. Ve frameworku Nette se třídy, které obstarávají model obsahují ve jménu slovo Manager.

Pro tuto aplikaci bude potřeba vytvořit čtyři třídy, vzhledem k jmenným pravidlům Nette a jejich funkcionalitě byly tyto třídy pojmenovány následovně:

- ExcelManager
 - Třída řešící import a export excel souborů a jejich interakci s databází.
- GridManager
 - Třída řešící získávání informací z databáze pro tabulky a následné předávání těchto informací kontrolní části.
- UserManager
 - Třída, co obstarává přihlašování a odhlašování uživatelů. Také ukládá uživatelské informace do objektu Identity. Tento objekt následně předává kontrolním částem.
- FormManager
 - Třída zpracující veškerou komunikaci s databází vzniklou skrze interakci s formuláři.



Obrázek 15 - ClassDiagram – Model – Autor

6.3.2 View

Část View obstarává veškeré grafické části, které uživatel vidí na svém monitoru. Grafika webových aplikací je založena na značkovacím jazyce HTML a CSS. Vzhledem k potřebám vývoje, je tento jazyk často dynamicky generován použitím jiných jazyků. Framework Nette pro zobrazování využívá šablonovací systém Latte. Tento systém přináší jednoduchý způsob, jak implementovat předané informace do značkovacích jazyků, znovu využívat již existující části kódu, a také přináší větší bezpečnost díky automatickému zneplatňování speciálních znaků a kontrole odkazů. Toto zjednoduší práci při ochraňování vašeho kódu proti některým bezpečnostním útokům. Jeden z těchto útoků je například Cross-SiteScripting.

Framework Nette generuje Latte následujícím způsobem. Vyhledá, zda existuje hlavní šablona, neboli soubor typu layout.latte. Z tohoto souboru vezme základní podobu stránky, kterou využije při generaci všech stránek. Následně zkontroluje, s jakou kontrolní třídou pracuje. Vezme jméno této třídy a zkusí nalézt soubor s tímto názvem. Zde, pokud nebyly vloženy další argumenty, nalezne soubor default.latte a z něj vybere informace, které vloží do hlavní šablony.

Hlavní šablona

Nejprve je třeba zpracovat hlavní šablonu layout.latte . Rozdělme si tuto šablonu na části.

- Head
 - Meta-data

- V této části se nalézají informace o stránce. Zde se jedná třeba o název stránky, nebo typ písma.
- Importy
 - Zde jsou informace o potřebných balíčcích pro importaci. Jedná se například o balíčky Bootstrap nebo FontAwesome.
- Body
 - Navbar
 - Navigační menu stránky, měněné dle role uživatele. Možnost nahradit separátním okrajem stránky v případě některých stránek.
 - Content
 - Hlavní obsah stránky, zde je veškerý kód importovaný ze separátní šablony každé stránky.
 - Footer
 - Banner
 - Koncová část stránky se jménem autora zobrazovaná na každé stránce.
 - Skripty
 - Zde se nalézají odkazy na skripty v jazyce Javascript využívané v této aplikaci.
 - Možnost importovat dodatečné skripty z šablony určité stránky.

Jednotlivé šablony stránek

Další část View, na kterou je potřeba se zaměřit jsou jednotlivé šablony pro každý Controller. V naší aplikaci se jedná především o následující šablony:

- Branches
 - default.latte
 - Stránka zobrazující stav kanceláří dle přístupu každého uživatele.
 - Zobrazuje se pro každou roli jinak.
- Teams
 - default.latte
 - Stránka zobrazující seznam týmů a jejich informace.
 - Přístup možný pouze pro administrátora.
- Users
 - default.latte
 - Stránka zobrazující seznam uživatelů a jejich informace.
 - Přístup možný pouze pro administrátora.
- Database
 - default.latte
 - Stránka umožňující následující akce.
 - Importování dat do databáze ze vloženého souboru Excel.
 - Stáhnutí souboru Excel s exportovanými daty z databáze.
 - Přístup možný pouze pro administrátora.

- TeamPage
 - default.latte
 - Stránka generující stránku jednotlivých týmů.
 - Přístup možný pouze pro administrátora
- Sign
 - In.latte
 - Stránka umožňující přihlášení uživatele.
 - Out.latte
 - Stránka umožňující odhlášení uživatele – Ihned přesměruje na přihlašovací stránku.

Formuláře

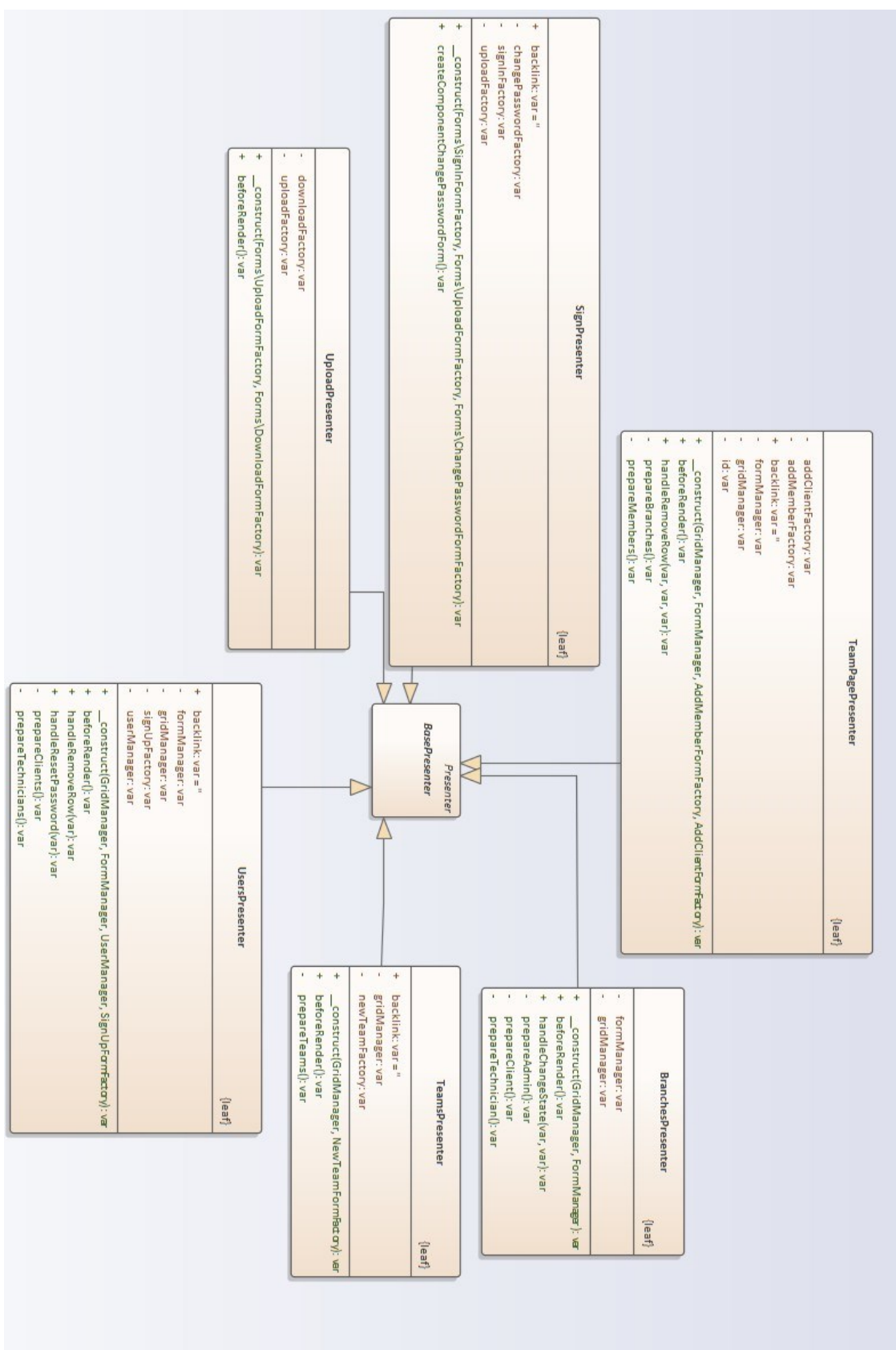
Kromě těchto šablon je taky Nette schopné dynamicky generovat formuláře a efektivně je propojit s kontrolní částí. Tyto formuláře jsou generovány pomocí návrhového vzoru „tovární metoda“ a každý formulář má vlastní třídu.

6.3.3 Presenter

Část Presenter propojuje logickou část Model (Manager) a grafickou část View (šablony latte). Obstarává veškeré informace, které View potřebuje pro svoji funkci. Jedná se například o uživatelské údaje, názvy stránek, nebo data formulářů. V Nette tyto třídy obsahují ve svém názvu slovo Presenter. Pro naši aplikaci potřebujeme Presentery pro každou stránku. Jedná se o třídy:

- SignPresenter
 - Kontrolní třída obstarávající přihlašování a odhlašování uživatele.
 - Komunikuje s formulářem SignInFormFactory pro vytvoření grafické podoby formuláře, kterou předá své šabloně In.latte.
 - Pokud z tohoto formuláře získá data, předá je Modelu do třídy UserManager, který tyto data zkontroluje a v případě úspěšného přihlášení vytvoří identitu uživatele.
 - Obstarává odhlašování uživatele za pomoci šablony Out.latte
 - Obstarává změnu hesla.
- BranchesPresenter
 - Kontrolní třída obstarávající zobrazení informací o kancelářích. Jedná se o dynamicky generovanou stránku vysoce rozdílnou podobou podle typu role uživatele.
 - Komunikuje s třídou GridManager pro získání dat odpovídající danému uživateli. Tyto data následně předá do své šablony, kde s využitím těchto dat vygeneruje tabulku Bootstrap-table, kterou zobrazí uživateli.
 - Od uživatele může dostat informace o změně stavu kanceláře. Tuto informaci následně předá zpět do třídy GridManager, která tuto informaci změni na databázi.

- TeamsPresenter
 - Kontrolní třída obstarávající zobrazení informací o týmech.
 - Kontroluje, zda uživatel je administrátor, pokud není, je přesměrován na stránku Branches.
 - Komunikuje s třídou GridManager pro získání seznamu týmu a jejich informací. Tyto data následně předá do své šablony, kde s využitím těchto dat vygeneruje tabulku Bootstrap-table, kterou zobrazí uživateli.
 - Umožňuje uživateli vygenerovat stránku týmu pro provedení změn ve vybraném týmu.
 - Umožní uživateli vytvořit nový tým dle zadaných informací.
- TeamPagePresenter
 - Kontrolní třída obstarávající zobrazení informací o týmech.
 - Kontroluje, zda uživatel je administrátor, pokud není, je přesměrován na stránku Branches.
 - Komunikuje s třídou GridManager pro získání seznamu přidělených techniků a lokací a jejich informací. Tyto data následně předá do své šablony, kde s využitím těchto dat vygeneruje tabulku Bootstrap-table, kterou zobrazí uživateli.
 - Třída je vygenerována dle zaslání Id týmu.
 - Umožňuje přidávat a odstraňovat techniky a lokace pro daný tým
- UserPresenter
 - Kontrolní třída obstarávající zobrazení informací o uživateli.
 - Kontroluje, zda uživatel je administrátor, pokud není, je přesměrován na stránku Branches.
 - Komunikuje s třídou GridManager pro získání seznamu uživatelů a jejich informací. Tyto data následně předá do své šablony, kde s využitím těchto dat vygeneruje tabulku Bootstrap-table, kterou zobrazí uživateli.
 - Umožňuje administrátorovi vytvořit nového technika, kterého následně předá třídě UserManager, kde zkontroluje autentičnost této informace a v případě, že není žádný problém je tento uživatel vytvořen v databázi.
- DatabasePresenter
 - Kontrolní třída obstarávající stránku pro upravování dat v databázi.
 - Kontroluje, zda uživatel je administrátor, pokud není, je přesměrován na stránku Branches.
 - Komunikuje s třídou ExcelManager pro zpracování dat z vloženého Excel souboru a jejich následné vložení do databáze v případě importování dat.
 - Komunikuje s třídou ExcelManager pro zpracování dat z databáze souboru a jejich následné vložení do nově vytvořeného souboru Excel dle definované šablony v případě exportování dat.



Obrázek 16- ClassDiagram – Presenter – Autor

7 Implementace aplikace

Ve chvíli, kdy je aplikace navrhnutá, přichází na řadu tento návrh přenést v realitu. V této kapitole bude popsáno, jak autor postupoval při tvorbě kódu. Tato kapitola bude rozdělena do jednotlivých podkapitol dle jednotlivých částí aplikace.

7.1 Model

V této části bude popsána tvorba tříd v části model. Tato část obstarává hlavní část logiky aplikace a veškeré propojení aplikace s databází. Tyto informace dále předává třídám v části Presenter, které tyto informace zpracují a předají je pro zobrazení v části View.

Dle návrhu existují 4 třídy rozděleny dle typu informací které zpracovávají.

7.1.1 UserManager

Tato třída je propojena s Nette Framework jako autentikátor obstarávající propojení s funkcemi „Identity“ a „User“, které v tomto frameworku zpracovávají veškerou autorizaci a autentizaci při komunikaci s Presentery. Toto propojení je realizováno díky napojení na rozhraní IAuthenticator. Tato třída byla vyvinuta upravením základní třídy dodávané v základní struktuře tohoto frameworku a rozšířením o dodatečné funkce.

Nejdůležitější funkce v této třídě je funkce authenticate, kterou právě využívá framework Nette pro upravenou autentizaci uživatele podle potřeb jednotlivých vývojářů.

Výpis 1 - UserManager – funkce authenticate (vlastní tvorba).

```
/**
 * Performs authentication.
 * @throws Nette\Security\AuthenticationException
 */
public function authenticate(array $credentials): Nette\Security\IIdentity
{
    [$email, $password] = $credentials;

    $row = $this->database->table(self::TABLE_NAME)
->where(self::COLUMN_EMAIL, $email)
->fetch();

    if(!$row) {
        throw new Nette\Security\AuthenticationException('The email is incorrect.',
self::IDENTITY_NOT_FOUND);
    }
}
```

```

        } elseif(!$this->passwords->verify($password, $row[self::COLUMN_PASSWORD_HASH]))
    {
        thrownew Nette\Security\AuthenticationException('The password is incorrect.',
self::INVALID_CREDENTIAL);

        } elseif ($this->passwords->needsRehash($row[self::COLUMN_PASSWORD_HASH])) {
            $row->update([
self::COLUMN_PASSWORD_HASH => $this->passwords->hash($password),
            ]);
        }

        $name = $row->ref('userRole', 'fk_UserRole_Id')->name;

        $arr = $row->toArray();
        unset($arr[self::COLUMN_PASSWORD_HASH]);
        return new Nette\Security\Identity($row[self::COLUMN_ID], $name, $arr);
    }

```

Této třídě je předán při přihlašování email a heslo uživatele, co se pokouší přihlásit. Podle emailu je následně nalezeno, zda tento uživatel opravdu existuje. Pokud uživatel existuje, zadané heslo je porovnáno s uloženým heslem v databázi. Pokud si všechna data navzájem odpovídají, jsou informace uživatele uloženy do objektu Identity a předány zpět pro použití.

Kromě této funkce, se v zde třídě nachází funkce pro vygenerování nového hesla při generaci nového uživatele a znovuvytvoření hesla v případě, že nějaký uživatel heslo zapomněl. Při generování nového hesla je dočasné heslo vždy zasláno danému uživateli, který se přes toto heslo přihlásí a je hned donucen si toto heslo změnit. Poslední funkce v této třídě jsou funkce pro přidání a odebrání technika. Pro tyto třídy byly vytvořeny speciální třídy výjimek pro případ, že zde vznikla při tvorbě chyba. Tyto výjimky jsou odchytávány u formulářů propojených s těmito funkcemi a ukládány jako chybový text informující o chybě při pokusu odeslání formuláře.

7.1.2 ExcelManager

Třída ExcelManager obstarává veškerou potřebnou logiku pro práci s tabulkovými soubory známými z aplikací excel. Jsou navrženy pro přesně danou strukturu tohoto typu souboru, z daného souboru vyberou data, z kterých vygenerují záznamy v databázi. Alternativně do této struktury, (uložené v šabloně) vygeneruje data z databáze.

V této třídě jsou dvě hlavní funkce, které každá má velké množství podfunkcí řešící jednotlivé karty souboru. Jedná se o funkce importExcel a exportExcel. Ostatní funkce zde existují pouze pro upravení struktury a přehlednost kódu.

Výpis 2 - ExcelManager – funkce exportExcel (vlastní tvorba).

```
/** Exports data from database into excel.
 * @throws \PhpOffice\PhpSpreadsheet\Exception
 * @throws \PhpOffice\PhpSpreadsheet\Reader\Exception
 * @throws \PhpOffice\PhpSpreadsheet\Writer\Exception
 */
public function exportExcel()
{
    $spreadsheet = IOFactory::load('assets/Template.xlsx');
    $spreadsheet = $this->createTechnicianSheet($spreadsheet);
    $spreadsheet = $this->createClientSheet($spreadsheet);
    $spreadsheet = $this->createTeamSheet($spreadsheet);
    $spreadsheet = $this->createLocationSheet($spreadsheet);
    $spreadsheet = $this->createChangeSheet($spreadsheet);
    $writer = \PhpOffice\PhpSpreadsheet\IOFactory::createWriter($spreadsheet, 'Xls');
    header('Content-Type: application/vnd.ms-excel');
    header('Content-Disposition: attachment; filename="output.xls"');
    $writer->save("php://output");
}
```

Výpis 3 - ExcelManager – funkce importExcel (vlastní tvorba).

```
/**
 * Function that receives spreadsheet as file from form and inserts data into database.
 * @param $file
 * @throws \PhpOffice\PhpSpreadsheet\Reader\Exception
 */
public function importExcel($file)
{
    $spreadsheet = IOFactory::load($file);

    $this->clearDatabase();
    $this->database->table('user')->insert([
        'firstName' => 'Admin',
        'lastName' => 'Admin',
        'email' => 'Adm@adm.cz',
        'password' => $this->passwords->hash('Admin'),
        'tempPassword' => false,
        'fk_UserRole_Id' => 1,
    ]);

    $startCoord = 2;
```



```

        $rowValue = $this->createTechnicianArray($spreadsheet, $startCoord);

while(!is_null($rowValue['email'])) {
    $startCoord++;
    $this->insertTechnician($rowValue);
    $rowValue = $this->createTechnicianArray($spreadsheet, $startCoord);

}

$startCoord = 2;
$rowValue = $this->createTeamArray($spreadsheet, $startCoord);
while(!is_null($rowValue['name'])) {
    $startCoord++;
    $this->insertTeam($rowValue);
    $rowValue = $this->createTeamArray($spreadsheet, $startCoord);
}

$startCoord = 2;
$rowValue = $this->createClientArray($spreadsheet, $startCoord);
while(!is_null($rowValue['email'])) {
    $startCoord++;
    $this->insertClient($rowValue);
    $rowValue = $this->createClientArray($spreadsheet, $startCoord);
}

$startCoord = 2;
$rowValue = $this->createLocationArray($spreadsheet, $startCoord);
while(!is_null($rowValue['id'])) {
    $startCoord++;
    $this->insertLocation($rowValue);
    $rowValue = $this->createLocationArray($spreadsheet, $startCoord);
}

return $rowValue;
}

```

Jak je možné v těchto funkcích vidět, je zde pracováno s knihovnou PhpSpreadsheet. Ve funkci export Excel je nejdřív načtena šablona, do které jsou následně vygenerovány za pomoci podtříd data. Tyto data jsou následně poslány uživateli pro stažení. Ve funkci importExcel je nejdříve vyčištěna databáze od starých dat, a následně jsou do databáze uloženy informace získané procházením jednotlivých řádků dat v zaslaném souboru. Při tvorbě kódu bylo počítáno s tím, že tato funkce bude použita pouze na začátku každé akce, proto se na začátku databáze čistí.

7.1.3 FormManager

Třída FormManager obstarává veškerou logiku formulářů, která se netýká základní správy uživatele. Obstarává formuláře pro vygenerování nového týmu, přerazování jednotlivých členů a lokací týmům a nejdůležitější funkci celé aplikace, změnu stavu lokací.

Výpis 4– FormManager – setState (vlastní tvorba).

```
/** Sets a newstate to a branch.
 * @param $code - Code of a branch to have state changed.
 * @param $state - State to set to a branch.
 * @param $userId - Id of user changing state.
 * @return mixed|Nette\Database\Table\ActiveRow
 */
public function setState($code, $state, $userId)
{
    $stateId = $this->database->table('officeState')
->where('name = ?', $state)
->fetch();
    $this->database->table('clientOffice')
->where('id = ?', $code)
->update(['fk_officeState_id' => $stateId->id]);
    $this->noticeRelated($code, $state);
    $this->archiveChange($code, $state, $userId);

    $newerState = $this->database->table('officeState')
->where('id=?', $stateId->id + 1)
->fetch();
    if ($newerState == null) {
        $newerState = $stateId;
    }
    return $newerState->name;
}
```

Jedná se o základní funkci, která pouze změní stav lokace s daným kódem na zasláný stav a následně zavolá funkci pro informování zúčastněných uživatelů a funkci archivující informace o proběhnuté změně. I přes svoji zdánlivou jednoduchost je to funkce, která obstarává nejdůležitější funkcionalitu celé aplikace.

7.1.4 GridManager

Tato třída je hlavní forma komunikace mezi databází a knihovnou Bootstrap-table obstarávající zobrazení data tabulek. Tato třída pro každý Presenter získá data z databáze. Vytvoří strukturované pole zakódované v JSON pro potřeby tabulky, a toto pole je předáno přes Presenter pro zpracování jednotlivým skriptům ve View. Zde nebudu zobrazovat

ukázky kódu, jelikož se jedná pouze o základní získávání dat z databáze a ukládání těchto dat do polí.

7.2 Presenter

Podkapitola Presenter bude popisovat, jak bylo postupováno při tvorbě tříd zpracovávající komunikaci mezi modelem a View. Pod tuto kapitolu také zahrnu množství továren na formuláře.

7.2.1 Presenter

Většina tříd obstarávající funkci Presenter má podobnou strukturu a všechny jsou dědici třídy BasePresenter. Nejprve je načtena funkce beforeRender, kde předtím, než se vytvoří stránka, je zkontrolována role uživatele, a v případě nedostatečné role, je uživatel přesměrován na úvodní přihlašovací obrazovku.

Výpis 5– TeamsPresenter – beforeRender (vlastní tvorba).

```
/** Checksforcorrect role beforerenderingthepage.
 * @throws \Nette\Application\AbortException
 */
functionbeforeRender()
{
    if (! $this->user->isInRole('Admin')) {
        $this->redirect('Sign:In');
    }
}
```

Tato funkce je používána ve většině Presenterů, jedinými rozdíly jsou třídy BranchesPresenter, jelikož k těmto stránkám mají přístup i jiné role než administrátor a třída SignPresenter, která obstarává autentizaci uživatelů. Dále zde ve většině tříd existuje funkce renderDefault která předává základní data pro vygenerování View.

Výpis 6– TeamsPresenter – renderDefault (vlastní tvorba).

```
/**
 * Default rendererofofpage
 */
public functionrenderDefault(): void
{
    $this->template->presenter = $this->presenter->name;
    if (isset($this->user)) {
        $this->template->role = $this->user->getRoles()[0];
    }
    if ($this->user->getIdentity() != null) {
        $this->template->user = $this->user
->getIdentity()
    }
}
```

```

->getData()['lastName'];
    } else {
        $this->template->user = '';
    }
}
$columns = $this->prepareTeams();
$this->template->columns = $columns;
$rows = $this->gridManager->getTeamsData();
$this->template->rows = $rows;
$this->template->rowNumber = sizeof($rows);
$this->template->expand = false;
$this->template->columnNumber = 6;
$this->template->secondaryColumnNumber = 6;
$this->template->tertiaryColumnNumber = 4;

}

```

Tato funkce nejprve předá view informace o přihlášeném uživateli. Následně začne předávat data potřebná pro generaci data tabulky Bootstrap-table. Tato funkce využívá podtřídu PrepareTeams, která vygeneruje strukturu sloupců tabulky. Jedná se o pole, které definuje jméno sloupce, hodnotu hlavičky sloupce pro zobrazení a zda je možné podle sloupce třídit tabulku. Tato data jsou zakódovány do formátu JSON pro přenos. Různé druhy těchto „prepare“ tříd využívají veškeré třídy typu Presenter co pracují s data tabulkami. Zde se opět třída BranchesPresenter odlišuje od ostatních tím, že má různé vstupy pro tabulky dle role uživatele. Vzhledem ke způsobu, jak jsou předávány data tabulkám Bootstrap-table, veškerá interakce s tlačítky uvnitř těchto data tabulek, musí být vedena přes skripty na straně uživatele. Ty komunikují s Presentery za pomoci knihoven Nette.Ajax.js. Pro správu této komunikace zde existují třídy „handle“. Třída SignPresenter se zde také liší. Tato třída je propojena s různými šablonami a nemá defaultní renderování pro předání dat. Obsahuje pouze komponenty pro vygenerování autentizačních formulářů.

Výpis 7– UsersPresenter – handleResetPassword (vlastní tvorba).

```

/** Ajaxhandlerforreseting a passwordof a user.
 * @param $id - idof user whoshouldgetpassword reset.
 * @throws \Nette\Application\AbortException
 */
public functionhandleResetPassword($id)
{
if ($this->isAjax()) {
    $this->userManager->resetPassword($id);
    $this->sendResponse(
new\Nette\Application\Responses\JsonResponse(
        [
            "status" => "ok",
        ]
    )
)
}
}

```

```

        )
    );
}
}

```

Jak je možné vidět, tyto funkce získají informace předávané z View pomocí technologie Ajax a předají je třídám typu Model pro zpracování. Po dokončení informují Ajax o dokončení procesu. Poslední funkce, které se nacházejí v těchto třídách jsou funkce řešící formuláře mimo data tabulky. Ty jsou zpracovány přes komponenty, které volají Továrny na formuláře. Tyto funkce zpravidla vypadají následovně.

Výpis 8– UsersPresenter – createComponentTechnicianForm (vlastní tvorba).

```

/**
 * Componentforcreationofformforaddingnewtechnician.
 * @return Form - Formforcreationoftechnician.
 */
protectedfunctioncreateComponentTechnicianForm(): Form
{
    return $this->signUpFactory->create(function (): void {
        $this->restoreRequest($this->backlink);
    });
}

```

7.2.2 Továrny na formuláře

Pro tvorbu formulářů je využívána funkcionalita frameworku Nette a návrhový vzor tovární metoda. Po tvorbě formuláře skrz tovární metodu, jsou jednotlivé součásti formuláře sestaveny, a je zde definováno, co proběhne po úspěchu metody, a co se stane při zachycení které výjimky.

Výpis 9– TeamsFormFactory (vlastní tvorba).

```

/** Functionthatcreatesformforassigningtechnicians to team.
 * @param $id - Idof team thatishavingtechnicianassigned.
 * @paramcallable $onSuccess
 * @return Form - Formforassigningtechnicians to team.
 */
    publicfunctioncreate($id,callable $onSuccess): Form
    {
        $form = $this->factory->create();
        $form->addText('member', 'Member: ');
        ->setRequired('Please enter email ofmember to add to team.');
```

```

        $form->onSuccess[] = function (Form $form, \stdClass $values)
        use ($onSuccess): void {
            try {
                $this->formManager->setMember($values->id, $values->member);
            } catch (Model\MissingUserException $e) {
                $form['member']
->addError('No such technicianexists.');
```

```

                return;
            } catch (Model\IncorrectRoleException $e) {
                $form['member']
->addError('This user is not a technician.');
```

```

                return;
            } catch (Model\AssignedTechnicianException $e){
                $form['member']
->addError('User withthis email isalready in different team.');
```

```

            }

            $onSuccess();

        };

        return $form;
    }

```

7.3 View

Pro část řešící zobrazení uživatelského rozhraní byl využit šablonovací systém Latte. Tento systém se při generaci každé stránky nejprve pokusí nalézt soubor základního rozložení „@layout.latte„. Pokud tento soubor nalezne, tak do tohoto souboru vkládá jednotlivé bloky definované v dodatečných šablonách každé stránky. V této podkapitole nejprve bude popsána hlavní šablona. Následně za pomoci šablon stránek, lišících od ostatních, ukáží, jaké technologie byly použity. Nebudou zde vypsány veškeré šablony, které existují v kódu, vzhledem k opakujícímu se využití technologií.

7.3.1 @Layout.latte

V souboru „@layout.latte“ této aplikace je hlavička všech stránek definující meta data stránky, a využívané CSS knihovny. V této hlavičce existuje speciální bloky umožňující jednotlivým šablonám stránek přidávat do této informace hlavičky. Jedná se o definování titulů stránky a možnost přidávat speciální informace do hlavičky.

Po definování hlavičky přichází hlavní část stránky: Body. V této hlavní šabloně je pro body vytvořen defaultní navigační lišta. Tuto navigační lištu je však možné nahradit za vlastní. Tato možnost byla využita pro autentizační stránky, kde jsou dosud neznámé informace o uživateli využívané pro vygenerování lišty. V případě, že uživatel není administrátor, jsou zde pouze zobrazeny odkazy na změnu hesla, skrze jméno uživatele a

odhlášení uživatele. Ostatní stránky jsou přístupné pouze pro roli administrátor a pro tuto roli se navigace na tyto stránky v navigační liště zobrazí. Po nastavení navigační lišty je importován hlavní blok „content“, kde se nachází veškerý hlavní obsah stránky, který je dodán již jednotlivými šablonami stránek. Po vložení „content“ se v této hlavní šabloně nachází patka stránky, kde jsou zobrazeny informace o stránce, aktuálně se zde nachází jméno autora a datum tvorby stránky.

Poslední část, která se zde nachází je blok pro skripty. V této části jsou importovány veškeré knihovny využívající jazyk Javascript. Po importu těchto knihoven jsou importovány skripty z jednotlivých stránek, potřebné pro funkcionalitu odpovídajících stránek

7.3.2 Branches

Toto je hlavní stránka, kam mají přístup veškeré role. Na této stránce je pouze vygenerována data tabulka Bootstrap-table podle informací zadaných z odpovídající třídy Presenter, proto hlavní část této šablony se nachází ve skriptu. V samotné části content se nachází pouze základní definice tabulky s jejími vlastnostmi.

Výpis 10– Branches\default.latte – Tabulka pro zobrazení lokací (vlastní tvorba)

```
<div class="container-fluid">

<div class="table-responsive">
<table id="table" class="table" data-unique-id="BranchCode"
      data-detail-view-by-click="true"
      data-search="true"
      data-visible-search="true"
      data-mobile-responsive="true"
      {if $expand}data-detail-view="true"{/if}>
<theadclass="thead-dark">
</thead>
</table>
</div>
</div>
```

Ve zbytku šablony se již nachází pouze blok se skripty. V tomto bloku jsou vytvořeny dvě funkce. Jedná se o funkce pro tvorbu základní tabulky a funkci pro vytvoření tabulky s rozšířenými informacemi, které se zobrazí po kliknutí na řádek záznamu. Lehké úpravy těchto funkcí jsou využity ve všech šablonách využívající Bootstrap-table.

Funkce „buildTable“ si nejprve přeparseruje informace o tabulce předané z Presenteru iterací, následně tyto informace předá funkci Bootstrap table pro vygenerování tabulky. V případě, že se v těchto tabulkách nachází tlačítka, jsou k těmto tlačítkům přiřazeny události, které proběhnou při kliknutí. Součástí těchto událostí je Ajaxrequest, který se propojuje s funkcí typu „handle“ v odpovídajícím Presenteru, který data předá dál modelu pro předání informací z události databázi.

```

/**
 * Creates primary table
 */
$el.bootstrapTable({
columns: columns,
    data: rowData
    {if $expand}
    , detailView: cells > 1,
onExpandRow: function (index, rows, $detail) {
expandTable($detail, index, rowData)
    }
    {/if}
});

```

Funkce `expandTable` je zavolána ve chvíli, kdy uživatel zažádá o další informace z nějakého určitého záznamu. Získá předané informace o řádce a datech, a z těchto dat vygeneruje sekundární tabulku.

7.3.3 Sign

Šablony pro třídu `SignPresenter` jsou rozdílné oproti ostatním hlavně tím, že využívají tři rozdílné šablony místo jedné. Jedná se spíše o pomocné stránky, které jsou tvořeny pouze formuláři Nette. Jedná se o šablony „in.latte“, „out.latte“ a „changePassword.latte“. Šablona „out.latte“ je, kromě krátkého nápisu, který nebude možné zahlédnout, prázdná. Tato šablona existuje pouze pro umožnění odhlášení uživatele po stisknutí tlačítka. Většina logiky zde probíhá v `SignPresenteru`. Oproti této šabloně, šablony „in.latte“ a „changePassword.latte“ sestávají z dvou částí. Speciálně vytvořené navigační lišty, která nahrazuje základní, a zavolané komponenty vytvářející formuláře.

```

{block navbar}
<div class="navbarnavbar-expand-lg navbar-darkbg-darkfixed-top">
<span class="navbar-brand"><a class="navbar-brand p-0" n:href=Branches: >Fleenk</a></span>
</div>
{/block}
<div class="mt-4">
<h1 n:block=title>Sign In</h1>
    {includebootstrap-formsigninForm}
</div>

```


7.3.4 Upload

Toto jsou šablony, které obstarávají stránku Database. Tato stránka sestává z dvou částí, které jsou vytvořeny za pomoci Bootstrap třídy „Card“. Toto jim dá specifický, celistvý vzhled. Do těchto „karet“ je následně importována komponenta vytvářející formuláře.

7.4 Databáze

Samotný kód databáze již byl vygenerován z modelu vytvořeného při návrhu. Byly vytvořeny troje soubory. CreateScript obstarávající původní vytvoření tabulek souboru. DeleteScript, který obstarává vyčištění databáze ode všech existujících tabulek, a nakonec byl vytvořen InsertScript který databázi naplní počátečními informacemi.

Výpis 13– InsertScript (vlastní tvorba).

```
INSERT INTO `userrole`(`id`, `name`) VALUES (1,'Admin');
INSERT INTO `userrole`(`id`, `name`) VALUES (2,'Technik');
INSERT INTO `userrole`(`id`, `name`) VALUES (3,'Klient');

INSERT INTO `officestate` (`id`, `name`) VALUES
(4, 'Collected'),
(3, 'Readyfor Pickup'),
(2, 'In use'),
(1, 'starting');

INSERT INTO `user` (`id`, `firstName`, `lastName`, `email`, `password`, `tempPassword`,
`fk_ClientSupervisor_Id`, `fk_UserRole_Id`, `fk_Technician_Id`) VALUES
(1, 'Admin', 'Admin', 'Adm@adm.cz',
'$2y$10$i5at1myh753dOURT6r2R9OQY9VXnuQ7Xu1w0u3PYC5sZXAs0eFYki', 0, NULL, 1, NULL);
```

Jak je možné vidět z této ukázky kódu, je nutné předem plně vyplnit dvojce tabulky, a zároveň přidat administrátorský účet do aplikace. Nejprve jsou vloženy uživatelské role. Existují troje, Administrátor, Technik a Klient. Následně jsou vloženy možné stavy, kterých mohou nabývat lokace klientů. Pro účely této práce byly vybrány stavy začínající proces, definující, že přístroje zatím nebyly dovezeny. Využívané, informující, že jsou přístroje stále využívány. Připravené k vyzvednutí, stav, který definuje, že technici mají přístroje odvézt, a nakonec je zde stav sebráno, který říká, že přístroje byly vybrány a práce na lokaci je již dokončena.

8 Ověření aplikace

Pro odstranění možných chyb aplikace je vysoce důležité aplikaci kvalitně testovat. Po dokončení vývoje byla u aplikace otestována responzivnost na různé prohlížeče a zařízení. Testování funkcionality je plánované při spolupráci s klientem. Bohužel, vzhledem ke krizové situaci probíhající během vývoje bakalářské práce nebylo možné toto testování stihnout do termínu dokončení bakalářské práce. Z tohoto důvodu, proběhnou pouze akceptační testy požadavků a responzivity.

8.1 Testování požadavků

Jak již zde bylo zmíněno, plánované testování na straně klienta nebylo možné provést před termínem odevzdání bakalářské práce. Z tohoto důvodu proběhlo alespoň manuální testování se třemi anonymními uživateli, pro zjištění, zda byly požadavky naplněny. Jednotlivé zprávy techniků se nalézají v příloze. Úroveň splnění požadavků bude ohodnoceno známkou v rozmezí 1 až 5. Hodnoty známky definují následující informace.

1. Požadavek je zcela implementován.
2. Požadavek je částečně implementován.
3. Požadavek není implementován.

8.1.1 Funkční požadavky

Tabulka 4 - Funkční požadavky – Testování (vlastní tvorba)

ID	Název	1.Uživatel	2.Uživatel	3. Uživatel
FOO1	Naplnit databázi	1	1	1
FOO 2	Vytvořit uživatele	1	2	1
FOO 3	Odstranit uživatele	1	1	1
FOO 4	Přiřadit týmu lokaci	1	2	1
FOO5	Odstranit tým z lokace	1	1	1
FOO 6	Přiřadit technikovi tým	1	2	1

FO07	Odstranit technika z týmu	1	1	1
FO08	Odsouhlasit nebo provést změnu lokace	2	1	1
FO09	Logovat čas změny	1	2	2
FO10	Informovat přiřazené uživatele o změně stavu a lokace	1	1	1
FO11	Zobrazit omezený seznam	1	1	1

8.1.2 Nefunkční požadavky

Ne všechny požadavky, které se zde nacházejí je možné testovat manuálně. U požadavků, kde je nemožné testovat, je popsán důvod netestování a případná závislost požadavku. Požadavky, které bylo možné otestovat budou hodnoceny známkami. Znamky budou mít následující hodnotu:

1. Aplikace naplňuje požadavek.
2. Aplikace má určité problémy pro splnění požadavku.
3. Aplikace není schopna splnit požadavek.

Tabulka 5 - Nefunkční požadavky – Testování (vlastní tvorba)

ID	Název	1.Uživatel	2.Uživatel	3.Uživatel
N001	Levný provoz	Tento požadavek je nemožné ověřit, jelikož výše úspor závisí na velikosti akce (množství techniků a jejich platu).Tyto údaje podléhají internímu utajení a nebylo je možno použít pro v této práci. Náklady na tuto aplikaci jsou kombinace cen hostingu HTTP a SMTP serveru.		
N002	Rychlá odezva	1	1	1
N003	Jednoduchá instalace na vlastní server	1	1	1
N004	Optimalizace	1	2	2

	rozhraní pro technika na chytrý telefon.			
No05	Přehlednost	1	1	1

8.2 Testování responzivity.

Pro překontrolování schopnosti zobrazení na různých nosičích byla aplikace otestována autorem s využitím různými přístroji. Při testování budou hlavní dvě vlastnosti. Jedná se o vlastnost zobrazení definující, do jaké úrovně je uživatelské rozhraní na daném nosiči efektivní a vlastnost funkcionalita, která informuje, zda všechny potřebné funkce na daném nosiči fungují. Tyto vlastnosti budou ohodnoceny známkou v rozmezí 1 až 5. Hodnoty známky definují následující informace.

4. Vlastnost aplikace nemá problém.
5. Vlastnost lehce narušuje chod aplikace.
6. Vlastnost přivádí problémy a způsobuje zpoždění při chodu aplikace.
7. Vlastnost velmi narušuje chod aplikace. Aplikaci je téměř nemožné efektivně použít.
8. Vlastnost způsobuje aplikaci nepoužitelnou. Aplikace je nepoužitelná.

Výsledky testování

Během testování bylo nalezeno, že při menším rozlišení není celá tabulka zobrazena najednou. Pro zobrazení informací musí uživatel tabulku posunout horizontálně. Tuto chybu není možné jednoduše opravit, jelikož původní plán implementace s atributem Bootstrap-table „data-mobile-responsive“ obsahuje chybu, která při přeměně tabulky na karty odstraňuje možnost expandovat řádky, což je vysoce důležitá součást této aplikace.[22]

Tabulka 6 - Testování Responzivnosti (vlastní tvorba)

Přístroj	Prohlížeč	Zobrazení	Funkcionalita	Poznámka
Stolní počítač vlastní sestavy	Google Chrome 81.0.4044.129	1	1	Plná responzivnost
	Microsoft Edge 81.0.416.68	1	1	Plná responzivnost
Lenovo T420	Mozilla Firefox 75.0	1	1	Plná responzivnost
	Opera 68.0.3618.63	1	1	Plná responzivnost
RedmiNote 8 Pro Android 9	Opera Mini Extreme data saving 47.2.2254.147957	2	1	Nutnost posunovat tabulky
	Google Chrome 81.0.4044.117	2	1	Nutnost posunovat tabulky
IPhone 8 IOS 13.4	Safari	2	1	Nutnost posunovat tabulky
Tablet LG-V500	Mozilla Firefox	2	1	Nutnost posunovat tabulky

Závěr

Při centrálně vedené organizaci týmů techniků dochází k časovým prodlevám při komunikaci s dispečerem. Pro omezení těchto časových prodlev byla započata analýza potřebných funkcionalit a požadavků na aplikaci. V dané práci bylo zaznamenáno, jak bylo při vývoji postupováno. Nejprve byla provedena analýza potřebných funkcionalit, které musí aplikace dosahovat. Vzhledem k vybrané metodice byla tato kapitola v práci přesunuta nad část návrhu, pro udržení posloupnosti vybrané metodiky. Následně proběhla rešerše existujících informací o dané problematice a bylo zkontrolováno, zda již neexistují aplikace, které potřebnou problematiku. Při analýze souvisejících aplikací bylo zjištěno, že aplikace řešící tuto problematiku existují, ale jsou naprosto neakceptovatelné vzhledem ke své cenové politice či nedostatku požadovaných funkcionalit. Následně byla zvolena metoda a způsoby, které budou využity při vývoji. Po definování prostředí začala tvorba návrhu aplikace. Návrh byl rozdělen na tři části. V první části byl navrhnut vzhled stránek. V druhé části byla definována struktura a architektura logiky. V třetí a poslední části návrhu byla navržena struktura relační databáze. Po dokončení těchto návrhů proběhla jejich realizace. V kapitole pojednávající o části implementace je popsána funkcionalita programu a kód, který byl použit pro dosažení těchto funkcionalit. Poslední část je ověření funkcionality programu. Do termínu dokončení bakalářské práce bylo možné otestovat pouze ověření responzivnosti napříč různými přístroji a webovými prohlížeči. Samotné ověření funkcionality proběhne u klienta mimo oblast bakalářské práce. Věřím, že i přes své nedostatky tato aplikace splňuje veškeré požadavky po ní požadované, a bude využívána v situacích, pro které byla stvořena.

Bohužel, vzhledem k nedostatečným zkušenostem, byla daná analýza neúplná a určité části musely být během vývoje pozměněny, což společně s neznalostí využívaných technologií vedlo k velkému množství prodlev a zpoždění. Další zlepšování systému již muselo být odloženo, protože z důvodu prodlev ve vývoji a probíhající karantény způsobené pandemií Covid-19, bylo nemožné provést testování funkcionality. Absence tohoto testování vede k existenci nezachycených chyb v odevzdané verzi aplikace. I přes problémy během vývoje však byla aplikace vyvinuta do použitelného stavu splňující veškeré zadané cíle.

Evidence jednotlivých lokací je kvalitně zpracována a jednotlivé role mají zobrazené pouze lokace, ke kterým jsou přiřazeny, a mohou upravovat pouze ty stavy. Evidování stavu dodávky techniky v jednotlivých lokacích funguje a jednotlivé role mají možnost měnit stav přiřazených lokací ve chvíli, kdy na to mají právo. Optimalizace webové aplikace na mobilní telefony má stále určité nedostatky, které jsou způsobeny chybou ve využívané knihovně pro zpracování data tabulek. Tato knihovna má atribut, který umožní přeměnu tabulky na jednotlivé karty s informacemi, ve chvíli, kdy je obrazovka nedostačující pro zobrazení celé tabulky. Bohužel nalezená chyba způsobuje, že ve chvíli, kdy se tabulka přemění na kartu, již není možné otevřít sekundární tabulky skryté v každém řádku. Z tohoto důvodu tento atribut není používán.

Seznam obrázků, tabulek a výpisů

Seznam Obrázků

Obrázek 1 - Tabule Monday.com – Monday.com [8]	15
Obrázek 2 - Cenový plán Monday.com za pět uživatelů – Monday.com [8]	16
Obrázek 3-Logistický nástroj ZohoCreator – ZohoCreator [9]	17
Obrázek 4 – Měsíční cenový plán ZohoCreator – ZohoCreator [9]	18
Obrázek 5-OnFleet logistika – OnFleet [10]	19
Obrázek 6- Měsíční cenový plán OnFleet – OnFleet [10]	20
Obrázek 7- Responsive Dropdown – Mobile-first Bootstrap [4]	26
Obrázek 8 - Use Case Diagram - Autor	30
Obrázek 9 - Wireframe – Pohled technika ve větším zařízení – Autor	32
Obrázek 10 - Wireframe – Pohled technika v mobilním zařízení – Autor	33
Obrázek 11 - Wireframe – Pohled klienta ve větším zařízení – Autor	34
Obrázek 12 - Wireframe – Pohled administrátora ve větším zařízení – Autor	35
Obrázek 13 - EER model databáze – Autor	39
Obrázek 14 - grafické zobrazení MVP – Ben Elghali Beyram [21]	40
Obrázek 15 - ClassDiagram – Model – Autor	41
Obrázek 16- ClassDiagram – Presenter – Autor	45

Seznam Tabulek

Tabulka 1 - Analýza aplikací (vlastní tvorba)	20
Tabulka 2 - Funkční požadavky (vlastní tvorba)	28
Tabulka 3 - Nefunkční požadavky (vlastní tvorba)	31
Tabulka 4 - Funkční požadavky – Testování (vlastní tvorba)	58
Tabulka 5 - Nefunkční požadavky – Testování (vlastní tvorba).....	59
Tabulka 6 - Testování Responzivnosti (vlastní tvorba)	61

Seznam výpisů

Výpis 1 - UserManager – funkce authenticate (vlastní tvorba).....	46
Výpis 2 - ExcelManager – funkce exportExcel (vlastní tvorba).	48
Výpis 3 - ExcelManager – funkce importExcel (vlastní tvorba).....	48
Výpis 4– FormManager – setState (vlastní tvorba).	50
Výpis 5– TeamsPresenter – beforeRender (vlastní tvorba).	51
Výpis 6– TeamsPresenter – renderDefault (vlastní tvorba).	51
Výpis 7– UsersPresenter – handleResetPassword (vlastní tvorba).	52
Výpis 8– UsersPresenter – createComponentTechnicianForm (vlastní tvorba).....	53
Výpis 9– TeamsFormFactory (vlastní tvorba).	53
Výpis 10– Branches\default.latte – Tabulka pro zobrazení lokací (vlastní tvorba)	55
Výpis 11– Branches\default.latte – Generace Bootstrap-table (vlastní tvorba)	56
Výpis 12– Sign\in.latte – Přihlašování uživatele (vlastní tvorba).....	56
Výpis 13– InsertScript (vlastní tvorba).	57

Použitá literatura

- [1] *ECMAScript® 2019 Language Specification* [online]. [vid. 2020-03-04]. Dostupné z: <https://www.ecma-international.org/ecma-262/10.0/index.html#Title>
- [2] POSPÍŠIL JAN. *Optimalizace webu pro mobilní zařízení a analýza jejího vlivu* [online]. Praha, 2013 [vid. 2020-04-15]. Diplomová práce / info:eu-repo/semantics/masterThesis. University of Economics, Prague. Dostupné z: <https://vskp.vse.cz/eid/38467>
- [3] HNÍZDIL JAKUB. *Responzivní webdesign* [online]. Praha, 2015 [vid. 2020-04-15]. Diplomová práce / info:eu-repo/semantics/masterThesis. University of Economics, Prague. Dostupné z: <https://vskp.vse.cz/eid/45301>
- [4] MAGNO, Alexandre. *Mobile-first Bootstrap* [online]. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2013 [vid. 2020-03-03]. ISBN 978-1-78328-580-8. Dostupné z: <http://ebookcentral.proquest.com/lib/vsep/detail.action?docID=1441776>
- [5] OKOSY MICHAL. *Elektronický obchod v Nette* [online]. Praha, 2015 [vid. 2020-04-15]. Bakalářská práce. University of Economics, Prague. Dostupné z: <https://vskp.vse.cz/eid/45271>
- [6] SRIPARASA, Sai Srinivas. *Building a Web Application with PHP and MariaDB: A Reference Guide* [online]. Olton, UNITED KINGDOM: Packt Publishing, Limited, 2014 [vid. 2020-03-03]. ISBN 978-1-78398-163-2. Dostupné z: <http://ebookcentral.proquest.com/lib/vsep/detail.action?docID=1688630>
- [7] LAGRONE, Benjamin. *HTML5 and CSS3 Responsive Web Design Cookbook* [online]. Olton, UNITED KINGDOM: Packt Publishing, Limited, 2013 [vid. 2020-03-03]. ISBN 978-1-84969-545-9. Dostupné z: <http://ebookcentral.proquest.com/lib/vsep/detail.action?docID=1192670>
- [8] *monday - team management software | Enterprise* [online]. [vid. 2020-02-27]. Dostupné z: <https://monday.com/1/enterprise>
- [9] *See the endless list of possible applications in Zoho Creator* [online]. [vid. 2020-02-27]. Dostupné z: <https://www.zoho.com/creator/overview.html>
- [10] Onfleet - Delightful delivery management software. *Onfleet* [online]. [vid. 2020-02-29]. Dostupné z: <https://onfleet.com>
- [11] *A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model - ProQuest* [online]. [vid. 2020-03-03]. Dostupné z: <http://search.proquest.com/docview/1660801422?pq-origsite=summon>
- [12] *Diagram Software and Flowchart Maker* [online]. [vid. 2020-05-07]. Dostupné z: <https://www.diagrams.net/>
- [13] *MySQL :: MySQL Workbench* [online]. [vid. 2020-05-07]. Dostupné z: <https://www.mysql.com/products/workbench/>

- [14] *UML modeling tools for Business, Software, Systems and Architecture* [online]. [vid. 2020-05-07]. Dostupné z: <https://www.sparxsystems.com/>
- [15] PhpStorm: The Lightning-Smart IDE for PHP Programming by JetBrains. *JetBrains* [online]. [vid. 2020-05-06]. Dostupné z: <https://www.jetbrains.com/phpstorm/promo/>
- [16] *Git - Documentation* [online]. [vid. 2020-05-06]. Dostupné z: <https://git-scm.com/doc>
- [17] *GitHub.com Help Documentation - GitHub Help* [online]. [vid. 2020-05-06]. Dostupné z: <https://help.github.com/en/github>
- [18] WampServer. *SourceForge* [online]. [vid. 2020-05-06]. Dostupné z: <https://sourceforge.net/projects/wampserver/>
- [19] *Mailtrap.io — Fake smtp testing server. Dummy smtp email testing* [online]. [vid. 2020-05-08]. Dostupné z: <https://mailtrap.io/>
- [20] HNÍZDIL, JAKUB. *Webové mobilní aplikace pro správu osobních financí*. [online]. Prague, 2013 [vid. 2020-04-15]. University of Economics, Prague. Dostupné z: <https://vskp.vse.cz/eid/66336>
- [21] BEYRAM, Ben Elghali. Introducing the Android Model-View-Presenter (MVP) Design Pattern. *Medium* [online]. 24. srpen 2017 [vid. 2020-05-02]. Dostupné z: <https://medium.com/@beyram.ghali/android-model-view-presenter-mvp-design-pattern-b27c6bf938de>
- [22] detail view doesn't work on mobile view... · Issue #4972 · wenzhixin/bootstrap-table. *GitHub* [online]. [vid. 2020-05-05]. Dostupné z: <https://github.com/wenzhixin/bootstrap-table/issues/4972>

Přílohy

Veškeré přílohy jsou uloženy jak v přiřazeném archívu, tak na repozitáři <https://github.com/zempo2/Fleenk>.

Obě tyto uložistiště využívají následující strukturu adresářů a souborů. V přehledu * značí, že se zde nachází větší množství souborů, které zde nebude rozepisováno

- docs
 - Database
 - Model
 - EER – EER model databáze
 - Scripts
 - CreateScript.sql – Skript pro sestavení databáze
 - DropScript.sql – Skript pro vymazání databáze
 - InsertScript.sql – Skript pro vložení základních hodnot
 - Documents
 - Excel
 - InputExcel.xlsx – Testovací data pro aplikaci
 - Template.xlsx – Šablona pro vkládání dat do aplikace
 - Testing
 - Formulář_testování.xlsx – Šablona formuláře pro akceptační testy
 - Formulář_uživatel_1.xlsx – Vyplněný formulář uživatele 1
 - Formulář_uživatel_2.xlsx – Vyplněný formulář uživatele 2
 - Formulář_uživatel_3.xlsx – Vyplněný formulář uživatele 3
 - Uml
 - ClassDiagram – Adresář s UML diagramy tříd
 - UseCaseDiagram – Adresář s UML diagramy případů užití.
 - Wireframes – Adresář s Wireframy aplikace
 - *
- src – Adresář se zdrojovým kódem aplikace
 - *
- readme.md – Manuál pro aplikaci v jazyce Markdown
- readme.html – Manuál pro aplikaci v jazyce HTML