Univerzita Pardubice
Fakulta elektrotechniky a informatiky

# Analýza a návrh lékařského systému pro správu pacientů

David Krupička

Bakalářská práce

2018

**Univerzita Pardubice**
**Fakulta elektrotechniky a informatiky**
Akademický rok: **2017/2018**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **David Krupička** |
| Osobní číslo: | **I15094** |
| Studijní program: | **B2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Analýza a návrh lékařského systému pro správu pacientů** |
| Zadávající katedra: | **Katedra informačních technologií** |

Zásady pro vypracování:

Cílem bakalářské práce je:
- analýza existujících systémů pro lékaře, které slouží primárně pro správu pacientů
- vytvoření vlastního návrhu a řešení online SAAS systému.

Pro vytvoření praktické části bakalářské práce bude využit programovací jazyk PHP, data systému budou ukládána do databáze MySQL.

Rozsah grafických prací:

Rozsah pracovní zprávy: **30-40 normostran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

**1. Learn PHP 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. New York, NY: Springer Science+Business Media, 2015. ISBN 9781484217290.**
**2. MySQL profesionálně: optimalizace pro vysoký výkon. Brno: Zoner Press, 2009. ISBN 978-80-7413-035-9.**
**3. BÖHMER, Marian. Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj]. Brno: Computer Press, 2012. ISBN 978-80-251-3338-5.**
**4. KOFLER, Michael. Mistrovství v MySQL 5: [kompletní průvodce webového vývojáře]. Brno: Computer Press, 2007. ISBN 978-80-251-1502-2.**

Vedoucí bakalářské práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2017**

Termín odevzdání bakalářské práce: **12. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

Univerzita Pardubice
L.S.
Česká republika

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 20. března 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 11. 5. 2018

David Krupička

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce Ing. Michaelovi Bažantovi, Ph.D. za čas, který mi věnoval a za cenné rady. Také bych rád poděkoval rodině a přátelům, kteří mě podporovali během studia a při zpracování bakalářské práce.

**ANOTACE**

Teoretická část obsahuje analýzu existujících systémů pro lékaře, které slouží primárně pro správu pacientů. Praktická část obsahuje návrh a vytvoření vlastního návrhu a řešení online SaaS systému. Pro vytvoření praktické části bakalářské práce bude využit programovací jazyk PHP, data systému budou ukládána do databáze MySQL.

**KLÍČOVÁ SLOVA**

PHP, webová aplikace, MySQL, SaaS, Composer, HTML, JavaSript, lékařský software.

**TITLE**

Analysis and design of a medical system for patient management

**ANNOTATION**

The theoretical part contains an analysis of existing systems for doctors that are primarily used for patient management. The practical part includes the design and creation of a custom design and solution of the online SaaS system. The PHP programming language will be used to create a practical part of the bachelor thesis, the system data will be stored in the MySQL database.

**KEYWORDS**

PHP, web application, MySQL, Saas, Composer, HTML, JavaScript, medical software.

# OBSAH

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# SEZNAM ZKRATEK

| | |
|---|---|
| API | Application Programming Interface |
| CRUD | Creat, Read, Update, Delete |
| CSS | Cascading Style Sheets |
| DBAL | Database Abstraction Layer |
| DOM | Document Object Model |
| DQL | Doctrine Query Language |
| HTML | HyperText Markup Language |
| MVC | Model View Controller |
| ORM | Object relational mapping |
| PDO | PHP Data object |
| PHP | HyperText Preprocessor |
| REST | Representational state transfer |
| RDBMS | Relational database management system |
| SaaS | Software as a Service |
| SQL | Structured Query Language |
| URL | Uniform Resource Locator |
| WYSIWYG | What you see is what you get |

# ÚVOD

Cílem této bakalářské práce je analýza a návrh lékařského systému pro správu pacientů. Na trhu existuje velké množství systémů, které mají lékaři k dispozici. Pokud chtějí přistupovat ke svým datům z více zařízení, mají možnost využívat existující software s daty v cloudu. Neexistuje ale software, který by byl přístupný odkudkoli bez nutnosti instalace klienta.

Tento problém je řešen v této bakalářské práci. V rámci této bakalářské práce bude vytvořena vlastní aplikace, která bude sloužit jako SaaS (Software as a Service) aplikace lékařům. Lékaři tak získají přístup ke svým datům odkudkoli bez nutnosti instalace klienta na svém zařízení. Lékařům a jejich personálu bude pro přístup k datům stačit pouze internetový prohlížeč.

V první kapitole autor popisuje existující software, který již mohou lékaři používat. Jedná se o úvod do problematiky, kde je popsáno, co existující software lékařům a jejich personálu poskytuje.

Ve druhé kapitole jsou popsány technologie, které se využívají pro tvorbu webových aplikací. Webová aplikace pro správu pacientů potřebuje mít uložená data, proto je v této kapitole nejen popis technologií pro vývoj, ale také popis databází. Jsou zde také popsány frameworky a různá rozšíření, které mohou vývojáři usnadnit vývoj a vést jej k lepšímu návrhu aplikace.

Třetí kapitola se zabývá návrhem a implementací konkrétní webové aplikace. Jelikož je pro úspěšný a funkční systém nutné postupovat při jeho tvorbě systematicky, je tato kapitola zaměřena na sběr požadavků, analýzu, návrh a implementaci. V implementaci je popsáno, jaký typ vývojových technologií byl vybrán pro vývoj a jak vypadá E-R diagram databáze. Závěr této kapitoly obsahuje popis uvedení systému do provozu.

Čtvrtá kapitola obsahuje popis vytvořeného systému. Jsou zde uvedeny funkce, které systém poskytuje. Funkce jsou rozdělené podle rolí, které mohou být uživateli systému přiděleny.

# 1 EXISTUJÍCÍ LÉKAŘSKÝ SOFTWARE

V této kapitole bude podrobně popsán lékařský software, který je dostupný na trhu. Cílem lékařského softwaru je usnadnění práce lékaři, případně celé ordinaci.

## 1.1 FONS Galen

V případě programu nazývaného FONS Galen se jedná o komplexní ambulantní software vyvíjený společností STRAPO s. r. o. FONS Galen využívá cloudová úložiště Microsoft Azure, takže se jedná o cloudový nástroj. [51]

Tento software je možné nainstalovat na počítače s operačním systémem Windows Vista, Windows 7, 8 a 10. Na uživatelské stanice je v případě tohoto programu instalována klientská aplikace, která přistupuje k datům uloženým v Microsoft Azure cloudu. FONS Galen poskytuje řešení pro ambulance, polikliniky, zařízení sociálních služeb i nemocnice. [19]

Následuje tabulka, která udává ceny za pořízení a využívání služeb programu FONS Galen. Ceny zahrnují proškolení pracovníků, moduly pro vyúčtování, e-recept, e-neschopenku a nastavení programu. Uvedený ceník je platný pro maximální počet 5 uživatelů. Pro zařízení s vyšším počtem je nutné kontaktovat obchodní oddělení, jak udává společnost STRAPO s. r. o. na své webové prezentaci. [10]

Tabulka 1: Ceník FONS Galen

| Název přístupu | Cena za pořízení | Měsíční poplatek |
|---|---|---|
| Hlavní Uživatel (lékař) | 19 965,- Kč | 725,- Kč |
| Vedlejší uživatel (sestra, lékař) | 7 260,- Kč | 242,- Kč |

*Zdroj:[10]*

## 1.2 DUNA MEDIK

Dalším informačním systémem určeným pro lékaře je DUNA MEDIK. Tento software je vyvíjený a spravovaný organizací TILL CONSULT a. s. DUNA MEDIK je rozdělen na dvě části podle jeho určení. Části se nazývají DUNA PRIVAT a DUNA DENTA. První část je určena primárně ambulantním specialistům a praktickým lékařům. Oproti tomu je DUNA DENTA určena hlavně stomatologům a ortodontickým ordinacím. Celá tato podkapitola čerpá z webu www.duna.cz.

**DUNA PRIVAT** je rozdělen na 3 základní moduly a poskytuje možnost rozšíření o další 4 moduly. Mezi základní moduly patří **modul ordinace**, který poskytuje správu kartotéky pacientů, výkazů nepravidelné péče a vyúčtování cest lékaře. Druhým základním modulem je **objednávací diář**, který je určen pro evidenci úkolů a objednávání pacientů. Třetím základním modulem je část obsahující **správu výkonů a vyúčtování pro zdravotní pojišťovny**. Moduly, kterými se dá DUNA PRIVAT dále rozšířit, zahrnují platby pacientů, číselníky zdravotních pojišťoven, správu záznamů o vyšetřeních a tiskové sestavy.

**DUNA DENTA** je stejně jako předchozí část rozdělena na základní a rozšiřující moduly. Základní moduly poskytují organizaci dat stomatologické ordinace, objednávací diář, dávky pro zdravotní pojišťovny. Rozšiřující moduly umožňují správu plateb pacientů, stomatologických číselníků, evidenci ošetření a organizaci tiskových sestav. [22]

V následující tabulce je shrnut ceník produktu DUNA PRIVAT. Nákup základní licence obsahuje bezplatnou podporu do konce kalendářního roku a základní moduly popsané výše.

Tabulka 2: Ceník DUNA MEDIK

| Název přístupu | Pořízení | Podpora | Pronájem na 3 měsíce | Pronájem na 6 měsíců |
|---|---|---|---|---|
| Hlavní uživatel | 10 500 Kč | 4 200 Kč | 1 930 Kč | 3780 Kč |
| Další uživatelé | 2 840 Kč | 1 100 Kč | 520 Kč | 1020 Kč |

*Zdroj: [9]*

## 1.3 Medicus

Program Medicus patří k nejrozšířenějším lékařským programům pro správu pacientů. Tento program patří společnosti CompuGroup Medical Česká republika s. r. o. Medicus nabízí řešení pro ambulance, stomatologie, polikliniky, malé nemocnice, lázně a wellness hotely. [46]

Medicus je nabízen pro ambulanci ve třech variantách. Varianty jsou nazvány start, profesional a komfort. Start obsahuje nejméně funkcí pro uživatele systému, komfort naopak nejvíce. Základem pro tyto tři varianty je vyúčtování pojišťovnám, kartotéka pacientů, ambulantní karta, medikace a recepty. Tento základ je dostupný ve všech třech variantách. Na vybranou variantu mohou být navázány další rozšiřující moduly.

Pomocí rozšiřujících modulů je možné přidat správu eReceptu, eNeschopenky, skladu, eNálezů, obrazové dokumentace a mnoho dalších. [45]

## 1.4 WinMed

Dalším komplexním počítačovým programem sloužícím ke správě pacientů je WinMed. Tento software je určený primárně pro soukromé ordinace a ambulantní kliniky. Vzhled pracovní plochy programu WinMed vychází ze stylu Microsoft Office, aby byl uživateli povědomý. WinMed poskytuje evidenci kartotéky pacientů, nálezů, lékařských zpráv, vystavování receptů, vyplňování tiskopisů a mnoho dalších funkcí užitečných pro lékaře. [13], [42]

WinMed nabízí možnost řešení v cloudu. Pomocí dat uložených v cloudu je možné k datům přistupovat odkudkoli. Lékaři tuto možnost využívají, jestliže mají více pracovišť, ze kterých chtějí přistupovat k datům.

Tabulka 3: Ceník WinMed

| | Lékař | Sestra |
|---|---|---|
| Prvotní pořízení | 16 900 Kč | 6 900 Kč |
| Další licence | 6 900 Kč | 5 900 Kč |
| Instalace programu | Zdarma | Zdarma |
| Údržba programu na rok | 6 900 Kč | 2 900 Kč |

*Zdroj: [11]*

## 1.5 Shrnutí

Na trhu existují různé programy pro správu pacientů, které usnadňují každodenní práci lékařům a dalšímu zdravotnickému personálu. Programy, které lékaři používají, jsou z větší části vytvořeny pro Diskový operační systém (DOS) nebo se jedná o jiné aplikace, které vyžadují instalaci. Jestliže chce lékař přistupovat ke svým datům odkudkoli, musí mít na zařízení nainstalovaný program přistupující ke cloudovému úložišti. Tato možnost je značně limitující. Řešením může být webová aplikace přístupná ze všech zařízení, na kterých je nainstalovaný webový prohlížeč.

# 2 TECHNOLOGIE

Tato kapitola bude zaměřena na jednotlivé technologie, které je možné využít při tvorbě webových aplikací. Také zde budou popsány jednotlivé alternativy použitých technologií. Kapitola bude rozdělena na dvě podkapitoly, zabývající se databází a programovacími jazyky.

## 2.1 Databáze

Pro dynamické webové aplikace může být potřebné využívat úložiště dat. Jako úložiště nám může sloužit právě databáze. Databáze je systém pro organizované ukládání strukturovaných údajů. Pro řízení přístupu k datům existuje systém pro řízení báze dat (tzv. SŘBD). Aby mohl být systém označený za SŘBD, musí poskytovat autentizaci a autorizaci. [39]

**Autentizace** má za cíl ověření identity uživatele, který přistupuje k systému. Při tomto procesu ověřujeme, zda je uživatel opravdu tím, za koho se vydává. U webových aplikací je většinou identita uživatele ověřována pomocí přihlašovacího jména a hesla. [6]

**Autorizace** je proces ověření přístupových práv uživatele k dané operaci. Obvykle následuje po úspěšné autentizaci. Nejčastěji jsou autentizovaným subjektům přiděleny role, které mají různá práva. Mezi nejpoužívanější rozdělení rolí patří administrátor, uživatel a bezpečnostní správce. [6]

### 2.1.1 MySQL

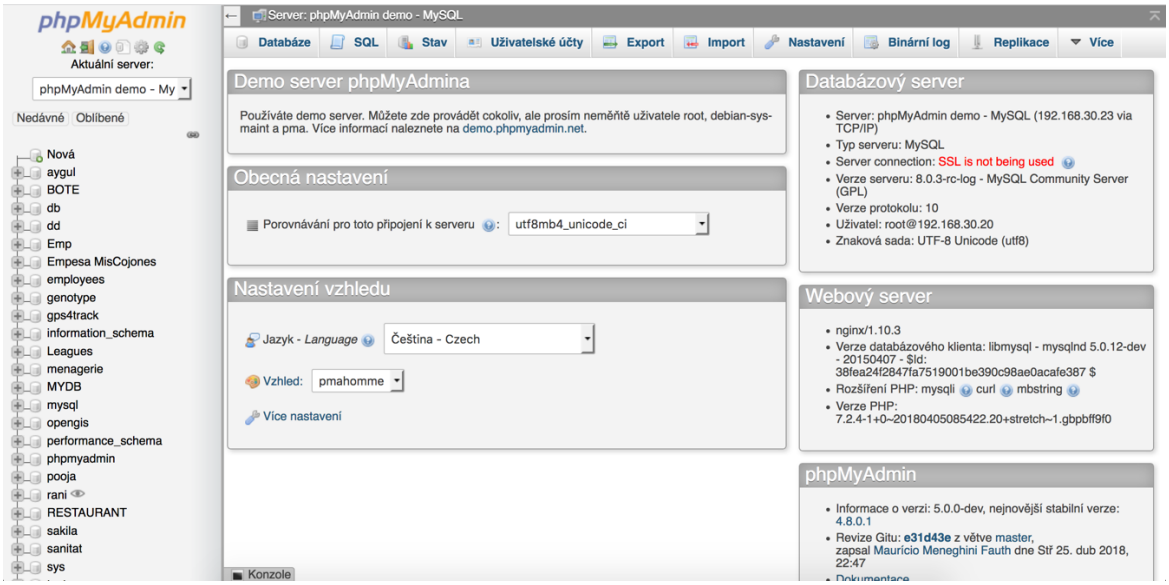
MySQL patří mezi světové nejoblíbenější open source relační databáze. Vděčí za to svému výkonu a spolehlivosti. Mezi nejznámější webové aplikace používající MySQL databázi patří Facebook, Twitter, YouTube a mnoho dalších. [2]

Pojem relační databáze označuje databáze, jejichž základním kamenem jsou relační tabulky, mezi kterými existuje určitá vazba. Tabulky jsou tvořeny sloupci a řádky. Sloupce mají jednoznačné názvy a mohou být chápány jako vlastnosti dané entity. Řádky jsou nazývány záznamy. Pro určení konkrétního záznamu by měl existovat jednoznačný identifikátor daného řádku. [59]

MySQL je vlastněna společností Oracle. Původně MySQL vlastnila společnost MySQL AB, jejímž spoluzakladatelem byl i Michael Widenius, který databázi vymyslel. V roce 2008 byla práva na databázi odkoupena společností Sun Microsystems. Později byl Sun Microsystems odkoupen Oraclem, proto je dnes její dceřinou společností a práva na MySQL patří společnosti Oracle Corporation. [50], [40], [32]

Pro správu databáze MySQL existuje mnoho nástrojů. Mezi nejpoužívanější patří phpMyAdmin, MySQL Workbench a Adminer.

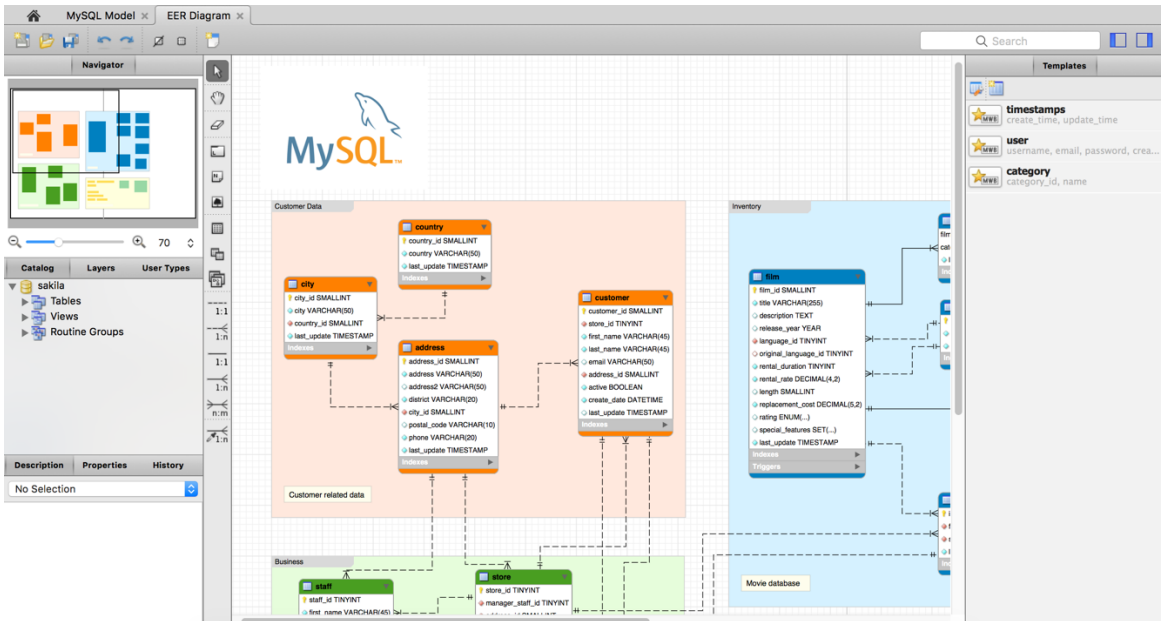
**PhpMyAdmin** je napsaný v programovacím jazyce PHP. Je to bezplatný nástroj a poskytuje široké možnosti práce s databází MySQL a MariaDB. Operace pro vkládání záznamů do tabulek, vytváření tabulek, úpravu záznamů a mnoho dalších lze provádět přes uživatelské rozhraní, které phpMyAdmin poskytuje. Dále je možné spouštět v prostředí tohoto nástroje SQL skripty. [8]




Obrázek 1: phpMyAdmin

*Zdroj: Vlastní zpracování*

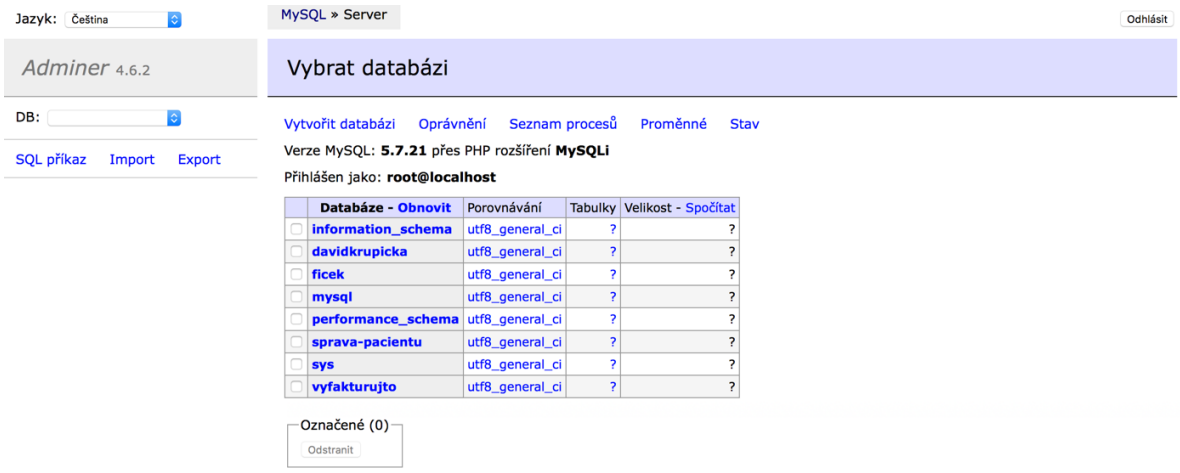
**MySQL Workbench** je nástroj pro vizuální modelování databází MySQL. Pomocí nástroje MySQL Workbench je možné také editovat, vytvářet nebo mazat databáze. Jedná se o multiplatformní aplikaci, která je spustitelná na operačních systémech Linux, Windows a macOS. [41]




Obrázek 2: MySQL Workbench

*Zdroj: Vlastní zpracování*

**Adminer** slouží ke správě MySQL databáze. Stejně jako phpMyAdmin je napsaný v jazyce PHP. Oproti aplikaci phpMyAdmin je Adminer napsaný v jediném PHP souboru. Adminer slouží ke správě databází MySQL, MariaDB, PostgreSQL a dalších. [56]




Obrázek 3: Adminer

*Zdroj: Vlastní zpracování*

### 2.1.2 Alternativní databáze

K databázi MySQL existují různé alternativy. Mezi nejpoužívanější patří MariaDB, PostgreSQL či Oracle.

Databázový systém **MariaDB** vznikl jako odnož MySQL. Nyní je vyvíjen společností MariaDB foundation, ve které působí zakladatel MySQL Michael Widenius. Do verze 5.5 jsou MariaDB s MySQL kompatibilní. [50]

**PostgreSQL** je na rozdíl od MySQL pomalejší ale bezpečnější a spolehlivější. Je vyvíjen jako svobodný software a umožňuje spouštět procedury psané v jazycích jako je Python nebo Pearl. [51]

**Oracle** je multiplatformní databázový server, který poskytuje správu strukturovaných dat. Jedná se o velmi používaný databázový server, protože podporuje přístup více uživatelů ke stejným datům. [49]

## 2.2 Vývojové technologie

Pro vytvoření webové aplikace je zapotřebí využívat určité technologie, které jsou k dispozici. Mezi tyto technologie patří skriptovací jazyky, programovací jazyky, verzovací systémy vyvíjeného softwaru a mnohé další technologie. V této podkapitole budou popsány technologie využitelné pro vytvoření webové aplikace a v některých případech i jejich alternativy.

### 2.2.1 HTML

Zkratka HTML vyjadřuje HyperText Markup Language. Jedná se o značkovací jazyk, který slouží k tvorbě statických webových stránek. První definice jazyka HTML byla vytvořena v roce 1991 a vytvořil ji Tim Barners-Lee. [33]

HTML dokument má danou strukturu, která je tvořena tagy. První řádek obsahuje doctype, který určuje typ dokumentu a použitou verzi HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

Následují párové tagy html, head a body. Tagy se dělí na **párové** a **nepárové**. Párové tagy obsahují otevírací a uzavírací tag. Například pro odstavec je to otevírací tag `<p>` a uzavírací tag `</p>`. Tagy je možné do sebe vnořovat, není ale možné, aby se křížily. [48]

Aktuální verze HTML je verze 5.1. Oproti verzi 4 přibyly nové tagy, jako například tag pro hlavičku, patičku nebo video. Verze 5.0. se v prohlížečích podporuje od roku 2010. [29]

### 2.2.2 CSS

CSS je zkratka pro kaskádové styly a znamená Cascading Style Sheets. Styly jsou nazývané kaskádové, protože na sebe mohou vrstvit další definice stylu a platí vždy poslední definice. CSS slouží k vzhledovému formátování HTML dokumentu. Existují tři způsoby použití CSS stylu v dokumentu.

**Přímý styl** je vložení CSS přímo do tagu HTML. K tomu slouží využití atributu style. Druhým způsobem je využití tzv. **stylopisu**. CSS styly se v tomto případě píší do hlavičky HTML dokumentu mezi párové tagy <style></style>. Poslední způsob je zápis stylů do **souboru *.css**. Tento způsob má výhodu při načítání stránky, webové prohlížeče tyto dokumenty nemusí stahovat znovu, to má za následek rychlejší načtení stránky.

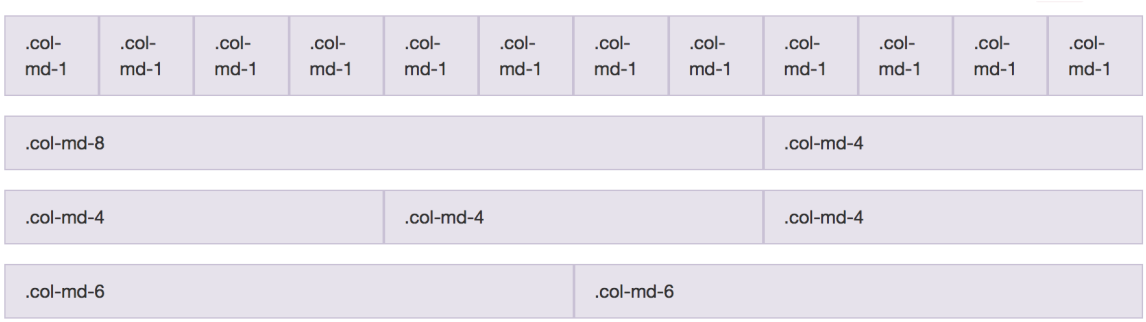
Pomocí CSS je možné formátovat například následující vlastnosti:

- nastavení velikosti písma,
- odsazení řádku odstavce a jeho řádkování,
- formátování nadpisů,
- nastavení barev textu,
- změna pozadí stránky nebo jednotlivých element na stránce,
- zobrazení rolovacích lišt nebo oříznutí element stránky,
- zvýrazňování odkazů na stránce, změna vlastností po najetí kurzorem na odkaz. [13]

Pro usnadnění práce s CSS existují frameworky. Jedním z takových frameworků je **Twitter Bootstrap**. Tento Framework obsahuje sadu připravených vzhledů tlačítek, formátování nadpisů, styly modálních oken a mnoho dalších funkcí. Mezi nejvyužívanější část Twitter Bootstrapu patří grid system. [38]

**Grid system** pomáhá programátorovi rozdělit webovou stránku na určité části a mezi tyto části dělit obsah stránky. V případě Bootstrapu je stránka rozdělena na 12 částí.

Podle přidělovaných tříd v atributu class u elementů nastavíme, na kolik z těchto 12 částí se bude prvek rozpínat.




Obrázek 4: Grid systém

*Zdroj: [25]*

Například pro využití 7 částí z těchto 12, použijeme class=”col-7”. Bootstrap poskytuje možnost využití různých velikostí podle zařízení. Názvy tříd odlišuje pomocí infixů sm, md, lg a xl. Názvy jsou seřazeny podle velikosti zařízení, kde sm jsou nejmenší zařízení (obvykle mobilní telefony) a xl největší zařízení (monitory stolních počítačů). [18], [26]

### 2.2.3 PHP

PHP je akronym pro Hypertextový Preprocesor. Programovací jazyk PHP vyvinul Remus Lerdorf v roce 1994 pro svoji soukromou potřebu. Roku 1995 byla veřejnosti poskytnuta první verze PHP, tehdy se nazývala Personal Home Page Tools. Ještě téhož roku vyšla PHP verze 2, která měla přepracovaný parser. Do druhé verze byla také přidána podpora SQL. V roce 2004 byla vydána verze 4, která byla vyvinuta Zeevem Suraskim a Andim Gutmansem. Tato verze měla až 200 násobně vyšší výkon než PHP 3.0. Aktuálně nejnovější stabilní verze je PHP 7.2. [35], [7], [34]

Programovací jazyk PHP má podobnou syntaxi jako jazyk C. Pracuje na straně serveru a je multiplatformní. Jeho použití je možné na operačních systémech typu UNIX i Windows. Zdrojové kódy jazyka PHP jsou uloženy na serveru v souborech s příponou .php. Ve výjimečných případech se setkáme s příponami .php3, .php4, .php5. Jazyk PHP je velmi oblíbený, protože má jisté výhody, mezi které patří: [44]

- je jednoduchý na pochopení,
- snadná komunikace s databází MySQL, MariaDB nebo PostgreSQL,

- je podporován velkým množstvím poskytovatelů webhostingu,
- má velkou podporu komunity a jedná se o otevřený projekt.

Mezi nevýhody jazyka PHP patří, že je interpretovaný, nejedná se o kompilovaný jazyk. Pokud někdo získá přístup k serveru, může nahlédnout do PHP skriptů. I přes tyto nevýhody je jazyk PHP velmi používaný, protože se pomocí něj dají vytvořit aplikace jako internetové obchody, sociální sítě, diskusní fóra, osobní webové prezentace, vyhledávače a mnoho dalších. [58]

Pro usnadnění práce programátorů existují knihovny. Pro PHP je mnoho knihoven veřejně dostupných a je možné je stáhnout na internetu. Jednotlivé knihovny mezi sebou mohou mít závislosti. Pokud existuje taková závislost, bylo by složité se v nich orientovat a dohledávat další knihovny které musí programátor stáhnout. Proto existuje nástroj Composer.

### 2.2.4 Composer

Composer je nástroj, který se stará o správu závislostí v PHP. Tento nástroj je distribuován jako spustitelný soubor .phar. Jestliže je potřeba nainstalovat nějaké knihovny do projektu, stačí vytvořit soubor composer.json, do kterého se zapíší informace o potřebných knihovnách a jejich závislostech. Pokud je v composer.json zapsaný název potřebné knihovny, tak se po spuštění příkazu composer install tyto knihovny stáhnou do adresáře vendor. Nástroj Composer stahuje knihovny primárně z centrálního repozitáře Packagist.

V případě používání verzovacích systémů (například git) je potřeba složku vendor ignorovat, protože obsahuje externí zdrojové kódy, stažené pomocí Composeru. Pokud se jedná o verzovací systém git, je nutné přidat název adresáře vendor do souboru .gitignore. [14]

### 2.2.5 Git

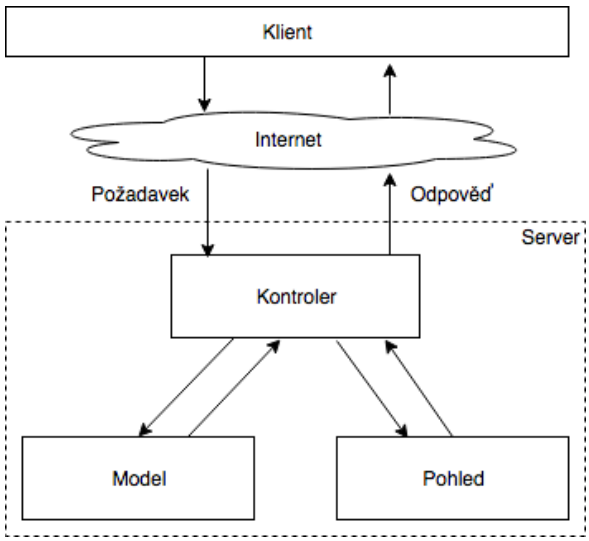
Git je systém využívaný pro správu verzí vyvíjeného softwaru. Git vyvinul jeden z hlavních vývojářů Linux kernelu Linus Torvalds. Za pomocí Gitu má programátor přístup k historii projektu. V této historii se může vývojář přesouvat a vracet se ke starším verzím. Git také poskytuje větvení, díky kterému můžeme vytvářet jednotlivé nové funkcionality systému bez jakéhokoli dopadu na již funkční část systému. Pokud se v nově vyvíjené větvi cokoli pokazí, nemusí se kód větve zahrnout do funkčního celku. Pro týmovou práci je možné využít hostingy pro ukládání repozitářů, například GitHub, Bitbucket nebo GitLab. Alternativou verzovacího systému Git je Apache Subverison, CVS, Mercurial a další. [55]

### 2.2.6 Symfony

Symfony je PHP framework, který obsahuje sadu komponent usnadňující tvorbu webových aplikací. Symfony patří mezi světově proslulé frameworky. Některé komponenty Symfony jsou využívány známými redakčními systémy jako například PrestaShop, Drupal a Joomla.

Symfony využívá architekturu MVC. Tato architektura je pojmenována podle tří částí, anglicky model, view a controller.

- **Kontroler** se stará o řízení. S touto částí komunikuje uživatel. Kontroler získává od uživatele parametry, které pošle modelu. Následně od modelu získá data, která potřebuje k zaslání HTML stránky uživateli. Stránku HTML získá od pohledu, kterému předá data získaná z modelu.
- **Model** obsahuje výpočetní logiku a stará se o práci s databází. Z databáze získává data, která může modifikovat a předat kontroleru.
- **Pohled** obsahuje HTML kód a využívá šablonovací nástroj. V Symfony je ve výchozím nastavení použit šablonovací nástroj Twig. Tento nástroj umožní vkládat data předaná kontrolerem přímo do HTML. [37]




Obrázek 5: MVC architektura

*Zdroj: Vlastní zpracování*

Společně s frameworkem Symfony je možné využívat šablonovací nástroj **Twig**, který obsahuje 3 základní syntaktické konstrukce. První konstrukce se značí pomocí dvou složených

závorek {{ … }} a využívá se například pro vypisování hodnoty určité proměnné. Další konstrukce se využívá pro určitou logiku, například k větvení programu a je zapisována pomocí složené závorky a znaku procenta {% … %}. Konstrukce sloužící pro komentování se zapíše jako složená závorka a znak křížek {# … #}. [54]

Další podstatnou částí frameworku Symfony je směrovač neboli **router**. Router zpracovává zadanou URL adresu, podle které se vybírá kontroler zpracovávající daný požadavek. Router umožňuje využití hezkých URL. Hezké URL je uživatelsky přívětivější, například místo adresy `index.php?id_uzivatele=5` lze využít adresu `/uzivatel/5`. Router v Symfony poskytuje možnost zadávat adresu přímo do anotačních komentářů funkce v kontroleru, která bude daný požadavek zpracovávat.

```
class UserController extends Controller
{
    /**
     * @Route("/user/{id}", name="user_detail")
     */
    public function userDetailAction($id)
    {
        // kód funkce zpracovávající požadavek
    }
}
```

Na předchozím příkladu je ukázka kontroleru zpracovávající požadavek s URL `/user/id` s využitím routeru nastaveného přes anotační komentář funkce. [47], [17]

### 2.2.7 Alternativní PHP frameworky

Místo Symfony lze využít alternativní frameworky, mezi které patří například Nette, Laravel, CakePHP, Zend Framework a další.

**Nette** framework patří k nejoblíbenějším v České republice, kde byl také založen. Je distribuován jako open source a je vyvíjen českou komunitou, kde je hlavním vývojářem David Grudl. Jako hlavní šablonovací systém je zde využit Latte a je možné použití knihovny Tracy pro pohodlné ladění. Na rozdíl od frameworku Symfony využívá Nette MVP architekturu. MVP značí Model, View a Presenter. [36]

**Laravel** byl vydán v roce 2011. Podle průzkumu na webu Sitepoint je Laravel nejoblíbenějším frameworkem. Tento framework má vlastní šablonovací systém nazývaný Blade. Laravel využívá architekturu MVC, stejně jako framework Symfony. [12]

**CakePHP** je starší než Laravel. Byl vydán již v roce 2005 a patří mezi nejoblíbenější PHP frameworky. Stejně jako Laravel nebo Symfony využívá CakePHP architekturu MVC.

CakePHP byl využit pro vývoj webových aplikací pro slavné značky jako například BMW nebo Hyundai. [36]

**Zend Framework** je dalším open source PHP frameworkem. Doporučuje se pro větší projekty, protože obsahuje spoustu konfiguračních voleb.

### 2.2.8 Doctrine 2

Doctrine 2 je ORM pro jazyk PHP. ORM je zkratka pro objektově relační mapování, anglicky object-relation mapping. Pomocí ORM lze mapovat relační databázi na objekty. Pomocí Doctrine 2 lze mapovat i opačně, objekty na relační databázi. Doctrine 2 mapuje strukturu databáze na takzvané entity. Zároveň podporuje databáze MySQL, Oracle, PostgreSQL a SQLite. Minimálním požadavkem pro fungování je PHP verze 5.3.

**Entita** je základ ORM. Jedná se o objekt, který slouží k mapování. Entity se vytváří za účelem uchovávání dat, představuje konkrétní řádek v databázové tabulce. Entita může obsahovat kromě metod typu get a set i další metody sloužící k práci s daty dané entity. Vlastnosti atributů entity jako například délku, povolení hodnoty NULL, název sloupce v tabulce, relace a další můžeme u entit popisovat pomocí anotačních komentářů. S entitami pracuje další důležitá část Doctrine 2, takzvaný Entity manager.

Na následujícím kódu lze vidět třídu, která bude sloužit pro vytváření instance entity Osoba. Osoba odpovídá tabulce s názvem osoba a má jeden atribut. U atributu jméno je v anotaci určen datový typ string a maximální délka je 30 znaků. Z důvodu stručnosti byly vynechány metody get a set, které lze vygenerovat nebo dopsat.

```
/**
 * Osoba
 * @ORM\Table(name="osoba")
 */
class Osoba
{
    /**
     * @var string
     * @ORM\Column(type="string", length=90)
     */
    private $jmeno;

}
```

**Entity manager** slouží ke správě entit. Umí entity vytvářet, editovat, mazat, provádět nad nimi vyhledávání a poskytuje mnoho dalších funkcí pro práci s entitami. Entity manager uchovává frontu změn. Entity do fronty zařadíme pomocí zavolání metody `persist` a po úspěšném volání metody `flush` se tyto změny projeví v databázi.

Další příklad ukazuje vytvoření instance entity osoba, pomocí metod typu set nastavení hodnot atributů jméno a příjmení. Následně je volána metoda persist, po zařazení entity do fronty je zavolána metoda flush, která data osoby uloží do databáze.

```
$osoba = new Osoba;
$osoba->setJmeno('Michal');
$osoba->setPrijmeni('Novák')
$entityManager->persist($osoba);
$entityManager->flush();
```

Entity manager využívá při dotazování tzv. **DQL**. DQL je zkratka pro Doctrine Query Language, jedná se o dotazovací jazyk který kombinuje snadnost použití SQL a objektový přístup. Při dotazování se využívají názvy entit místo názvů tabulek. Výstupem využití DQL a Entity manageru jsou instance, namísto řádků tabulky. Ukázku práce s DQL lze vidět na dalším stručném přikladu.

```
$query = $entityManager->createQuery(„SELECT o FROM Osoba o
WHERE o.jmeno = 'Michal'“);
$result=$query->getResult();
```

**Architektura Doctrine** je rozdělena mezi tři balíčky:

- **Common** balíček je nezávislý na zbylých dvou balíčcích. Obsahuje knihovny třídy a rozhraní a rozšiřuje základní funkce PHP.
- Balíček **DBAL** pojmenovaný jako Database Abstraction Layer je závislý na balíčku Common a obsahuje rozšířenou abstraktní vrstvu nad PDO. Cílem tohoto balíčku je definování DQL a poskytování aplikačního rozhraní API, které zmírní rozdíly mezi různými RDBMS.
- Třetím balíčkem je **ORM**, již zmiňovaný Object Relation Mapping se stará o mapování objektů do relační databáze a zpět. Tento balíček je závislý na balíčcích DBAL a Common. [53]

### 2.2.9 JavaScript

Klientský skript nazývaný jako JavaScript se používá při vývoji webových stránek. JavaScript se zpracovává v prohlížeči na straně klienta a je ze serveru posílán společně s HTML stránkami. Stejně jako u PHP se jedná o interpretovaný jazyk, který se nemusí kompilovat, je objektový a case sensitivní. JavaScript umožňuje [28]:

- hodnocení dat ve formulářích,
- počítání výpočtů na klientské straně v prohlížeči,
- tvorbu různých prvků k oživení webu,
- vytváření otevíracích menu a dalších dynamických prvků. [15], [30]

Důvodem pro využití JavaScriptu k validaci formulářů je ušetření času. Odeslání požadavku na server a zobrazení odpovědi serveru zabere více času, než přímá validace JavaScriptem. Nevýhoda JavaScriptu vyplývá z toho, že běží u klienta v prohlížeči a tím pádem může být JavaScript klientem vypnut. Proto je důležité provést validaci i na serveru.

Syntaxe JavaScriptu je velmi podobná syntaxi jazyka C. JavaScript umožňuje předávat funkce v parametru jiných funkcí tzv. callback, díky využití funkcionálního paradigma. To vývojářům umožňuje do běžné proměnné uložit funkci.

### 2.2.10 JavaScriptové knihovny a frameworky

Vývojáři mají k dispozici velké množství JavaScriptových knihoven a frameworků, které jsou vyvíjené jako open source. Mají za úkol usnadnění práce programátorovi, zmenšení obsahu kódu, který musí vývojář psát a rozšířit vývojáři možnosti. Mezi nejznámější patří jQuery, Dojo, Prototype a MooTools. [43]

**jQuery** je JavaScriptová knihovna. Má jednoduchou syntaxi a velmi usnadňuje práci s DOM objekty. jQuery používá alias, pomocí kterého se místo názvu jQuery využívá znak $. Tuto knihovnu je potřeba do webové stránky vložit. Je možné ji stáhnout jako jeden soubor, obsahující veškerý kód umožňující manipulaci s DOM objekty, Ajaxem a událostmi. [16], [20]

### 2.2.11 WYSIWYG Editory

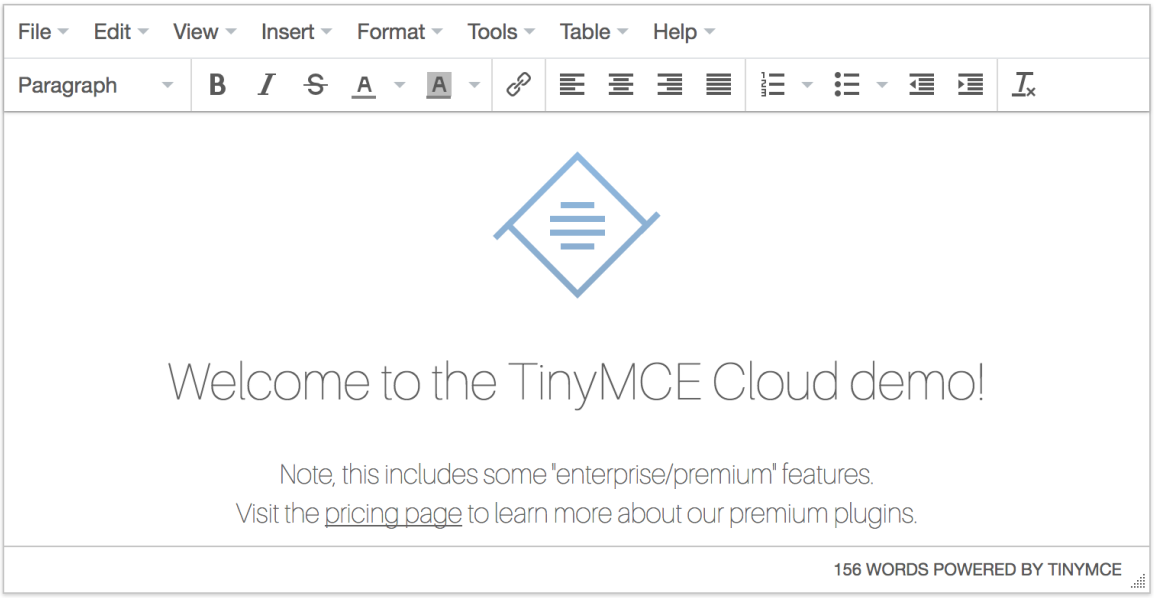
HTML jazyk poskytuje základní pole pro vložení dlouhého textu zvané textarea. Toto pole může být často velmi nedostačující a limitující, jelikož neobsahuje grafické formátování textu. Tento nedostatek odstraňují právě WYSIWYG editory. Zkratka WYSIWYG je odvozená

od anglického názvu „What you see is what you get“, přeloženo do češtiny „Dostaneš co vidíš“. WYSIWYG editory představují nadstavbu nad základním polem textarea, obsahem tohoto pole je XHTML, které je uživateli zobrazeno graficky. Uživatel může v textu editovat styl písma, tloušťku, kurzívu, velikost písma a spoustu dalších vlastností. Mezi nejoblíbenější WYSIWYG editory patří TinyMCE, CKEditor, FreeTextBox a další.

**TinyMCE** editor má velmi jednoduchou implementaci. Do HTML stránky stačí implementovat JavaScriptový soubor tohoto editoru.

```
<script src='https://cloud.tinymce.com/stable/tinymce.min.js'>
</script>
```

Následně je nutné zavolat metodu init a pomocí selektoru vybrat identifikátor prvku textarea, který se má uživateli zobrazovat jako editor. Grafické uživatelské prostředí editoru TinyMCE je zobrazeno na následujícím obrázku.




Obrázek 6: TinyMCE editor

*Zdroj: Vlastní zpracování*

TinyMCE disponuje funkcemi poskytujícími vkládání tabulek, obrázků, editaci textu, vkládání hypertextových odkazů a spoustu dalšími funkcemi. Výhodou tohoto editoru je možná rozšiřitelnost, vývojář může do editoru přidávat vlastní tlačítka a funkce. [43]

**CKEditor** vznikl z původního FCKEditoru. CKEditor se snaží odstraňovat nedostatky, které měl FCKEditor. CKEditor také rozšiřuje základní vstupní pole textarea. Tento editor je

graficky velmi podobný produktům Microsoft Word nebo textovému editoru z řady Open Office.

**FreeTextBox** je využívaný primárně pro ASP.NET Vzhledově je stejně jako CKEditor velmi podobný nástroji Word od společnosti Microsoft. [52]

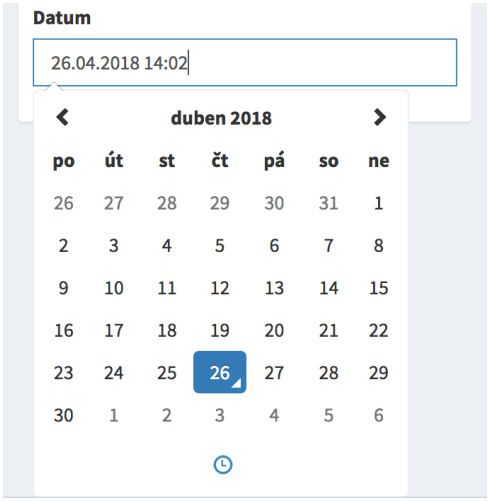
### 2.2.12 Další rozšíření pro Frontend

Část webové aplikace nazývaná Frontend je ta část, která je viditelná běžným uživatelem aplikace. Technologie, které jsou potřeba k vytvoření této Frontendové části lze rozšířit různými komponentami, šablonami a nadstavbami. Většina těchto komponent je vytvářena z důvodu šetření času programátora. Komponenty jsou znovupoužitelné a velká část z nich je vyvíjena jako open source. Mezi takové komponenty patří například FullCalendar.

**FullCalendar** je open source jQuery JavaScriptový plugin, který slouží k zobrazení kalendáře na webové stránce. Hodí se k zobrazování událostí, je velmi intuitivní a vzhledově podobný kalendáři, který poskytuje Google. Tento kalendář je velmi jednoduché vložit do stránky, pro jeho správnou funkci stačí přidat JavaScriptový soubor a soubor s CSS kaskádovými styly. [1]

Dalším možným rozšířením může být **šablona**, využívající Bootstrap a jQuery popsaná níže. Příkladem takové open source šablony je **Admin LTE**. Tato šablona byla navržena primárně pro Bootstrap verze 3. Na internetu je velké množství šablon, které se využívají k návrhu webových aplikací. [4]

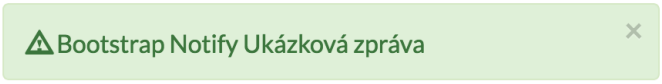
Jestliže má uživatel na webové stránce vyplňovat pole s datem, je velmi nepohodlné napsat vedle pole formát, ve kterém je po uživateli datum žádáno. Takovým formátem může být například dd-mm-yyy. Většina uživatelů by na první pohled nepochopila, co je po nich žádáno. K tomu slouží rozšíření textového pole zvané datepicker. Jestliže je na Frontendu použit datepicker, je uživateli po kliknutí do pole zobrazen kalendář. Po jednoduchém kliknutí na datum v kalendáři se pole vyplní s přesným formátem, který je po uživateli požadován. Jedním z takových rozšíření je Bootstrap Datepicker. Uživatelská část tohoto rozšíření je vidět na následujícím obrázku. [21]




Obrázek 7: Bootstrap Datepicker

*Zdroj: Vlastní zpracování*

Uživateli je občas potřeba zobrazovat hlášení, zda daná akce dopadla podle očekávání nebo ne. K tomu slouží rozšíření Bootstrap Notify. Tyto notifikace lze barevně rozlišit, využívají pro svůj správný běh Bootstrap a jsou responzivní. [31]




Obrázek 8: Bootstrap Notify

*Zdroj: Vlastní zpracování*

Pro zobrazení fotografií na webu mohou sloužit různé přehledné fotogalerie. Jedním pluginem určeným pro vytvoření fotogalerie je Bootstrap Lightbox. [57]

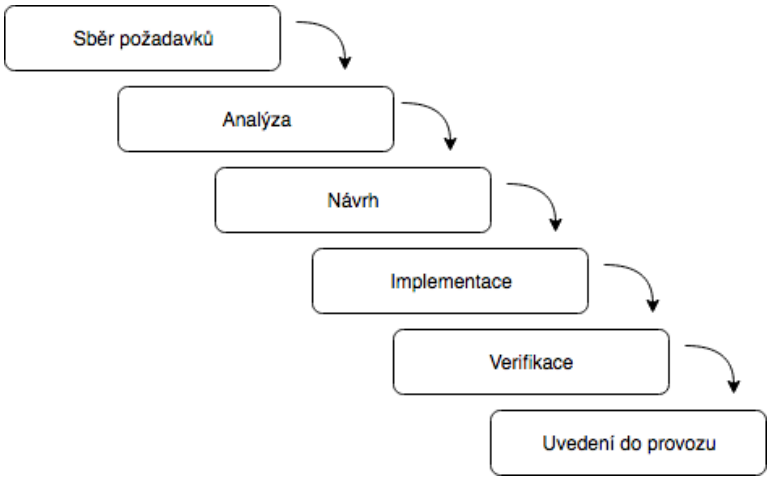
# 3 NÁVRH A IMPLEMENTACE

Součástí této bakalářské práce je implementace vlastního řešení. Jelikož existující řešení dostupné na trhu není přístupné z každého webového prohlížeče, tak se autor rozhodl vytvořit řešení jako webovou aplikaci. Celá tato kapitola čerpá ze zdrojů [5], [24].

Aby vyvíjený software nevznikal chaoticky a vývoj měl určitý řád, existují určité metodiky vývoje software, které jsou vyzkoušené v praxi. Tyto metodiky zajišťují efektivní vytváření nového softwaru. Metodiky bychom mohli dělit na:

- **klasické**, někdy označované jako tradiční. Mezi tyto metodiky řadíme vodopádový model, spirálovitý model nebo unifikovaný proces vývoje software.
- **agilní**, kde zákazník rychleji dostává dodávanou práci, která se v cyklech stále upravuje a vylepšuje.

Pro vývoj softwaru pro správu pacientů byla vybrána metodika **vodopádového modelu**. Zde je proces vývoje softwaru rozdělen na činnosti, které na sebe postupně navazují. Těmito částmi jsou sběr požadavků, analýza, návrh, implementace, verifikace a zavedení aplikace do provozu.




Obrázek 9: Vodopádový model

*Zdroj: Vlastní zpracování*

V ideálním světě by se z jedné fáze vývoje nevracelo k fázi předchozí. V praxi se ale často stává, že se vrátit o jednu fázi musíme. Stává se tak z důvodu, že například u implementace zjistíme, že je potřeba doplnit nějaké detaily z analýzy. Pokud se v životním cyklu vývoje

softwaru tyto zpětné kroky vyskytují, měly by být velmi výjimečné a při vývoji by měla být maximální snaha o jejich minimalizaci.

## 3.1 Požadavky

Podkapitola zaměřená na požadavky bude obsahovat specifikaci požadavků na nově vyvíjený systém pro správu pacientů. Požadavky popisují, co by měl systém dělat, nepopisují jak. Existuje více druhů kategorizace, v této práci jsou rozděleny požadavky na funkční a nefunkční. Každému požadavku je také nastavena priorita dle následující tabulky.

Tabulka 4: Priority požadavků

| Zkratka | Anglický název | Český název | Popis |
|---|---|---|---|
| M | Must have | Nezbytný | Povinný požadavek |
| S | Should have | Možný | Důležitý požadavek |
| C | Could have | Eventuální | Nepovinný požadavek |
| W | Want to have | Chceme mít | Může být zahrnut do další verze |

*Zdroj: Vlastní zpracování*

### 3.1.1 Funkční požadavky

Funkční požadavky popisují, jaké chování bude systém nabízet. Požadavky jsou rozděleny podle čtyř částí, na které bude systém rozdělen a části společné pro přihlášené uživatele a administrátora.

Následují požadavky na část systému, která je přístupná veřejnosti bez nutnosti registrace.

Tabulka 5: Funkční požadavky na veřejnou část systému

| ID | Název | Priorita |
|---|---|---|
| FR1 | Systém bude umožňovat vytvoření účtu pro doktora | M |
| FR2 | Systém bude umožňovat vytvoření účtu pro zdravotní sestru | M |
| FR3 | Systém bude umožňovat získání základních informací o systému | C |
| FR4 | Systém bude umožňovat zobrazení náhledových fotografií ze systému | C |
| FR5 | Systém bude umožňovat přihlášení pro všechny role | M |

*Zdroj: Vlastní zpracování*

Na administrátorskou část systému byly analyzovány další požadavky, které jsou sepsané v následující tabulce.

Tabulka 6: Funkční požadavky na administrátorskou část systému

| ID | Název | Priorita |
|---|---|---|
| FR6 | Systém bude umožňovat zablokování účtu uživatele | S |
| FR7 | Systém bude umožňovat odblokování účtu uživatele | S |
| FR8 | Systém bude umožňovat evidování zdravotních pojišťoven | M |

*Zdroj: Vlastní zpracování*

Pro část systému určené k přístupu doktorům existují požadavky, které budou popsány v tabulce, která následuje. Tyto požadavky jsou sepsány pro roli doktor.

Tabulka 7: Funkční požadavky na část systému pro doktory

| ID | Název | Priorita |
|---|---|---|
| FR9 | Systém bude umožňovat evidenci pacientů v kartotéce | M |
| FR10 | Systém bude umožňovat evidenci vyšetření u pacientů | S |
| FR11 | Systém bude umožňovat evidenci plánovaných návštěv pacientů | S |
| FR12 | Systém bude umožňovat evidenci událostí v kalendáři | C |
| FR13 | Systém bude umožňovat evidenci souborů u vyšetření pacienta | S |
| FR14 | Systém bude umožňovat evidenci zdravotních sester pro roli doktor | M |

*Zdroj: Vlastní zpracování*

Stejně jako u doktorů, existují požadavky pro část systému určené zdravotním sestrám. Pro přístup do této části bude potřeba, aby měl uživatel přidělenou roli zdravotní sestra.

Tabulka 8: Funkční požadavky na část systému pro zdravotní sestry

| ID | Název | Priorita |
|---|---|---|
| FR15 | Systém bude umožňovat výběr doktora pro evidenci jeho dat | M |
| FR16 | Systém bude umožňovat evidenci pacientů vybraného doktora | M |
| FR17 | Systém bude umožňovat evidenci vyšetření pacientů vybraného doktora | S |
| FR18 | Systém bude umožňovat evidenci naplánovaných návštěv pacientů vybraného doktora | S |
| FR19 | Systém bude umožňovat evidenci událostí v kalendáři vybraného doktora | C |
| FR20 | Systém bude umožňovat evidenci souborů u vyšetření pacienta pro vybraného doktora | S |
| FR21 | Systém bude umožňovat evidenci čekárny | C |
| FR22 | Systém bude umožňovat přijetí požadavku od doktora, pro evidenci jeho dat | M |

*Zdroj: Vlastní zpracování*

Požadavky vypsané v následující tabulce jsou společné pro všechny uživatele přihlášené do systému. Jedná se tak o uživatele přihlášené s rolí zdravotní sestra nebo doktor. Neplatí ale pro nepřihlášeného uživatele s přístupem do veřejné části systému.

Tabulka 9: Funkční požadavky společné pro zdravotní sestry a doktory

| ID | Název | Priorita |
|---|---|---|
| FR23 | Systém bude umožňovat změnu hesla | M |
| FR24 | Systém bude umožňovat změnu osobních informací | S |
| FR25 | Systém bude barevně rozlišovat události v kalendáři | W |

*Zdroj: Vlastní zpracování*

### 3.1.2 Nefunkční požadavky

Nefunkční požadavky se zabývají vlastnostmi nebo omezujícími podmínkami na systém.

Tabulka 10: Nefunkční požadavky na systém

| ID | Název | Priorita |
|---|---|---|
| NFR1 | Systém bude dostupný na internetu | M |
| NFR2 | Systém bude možné provozovat na operačním systému typu UNIX | S |
| NFR3 | Systém bude napsán v jazyce PHP | M |
| NFR4 | Data systému budou uložena v databázi MySQL | S |
| NFR5 | Zabezpečený přístup pro uživatele na serveru do složky s daty | |

*Zdroj: Vlastní zpracování*

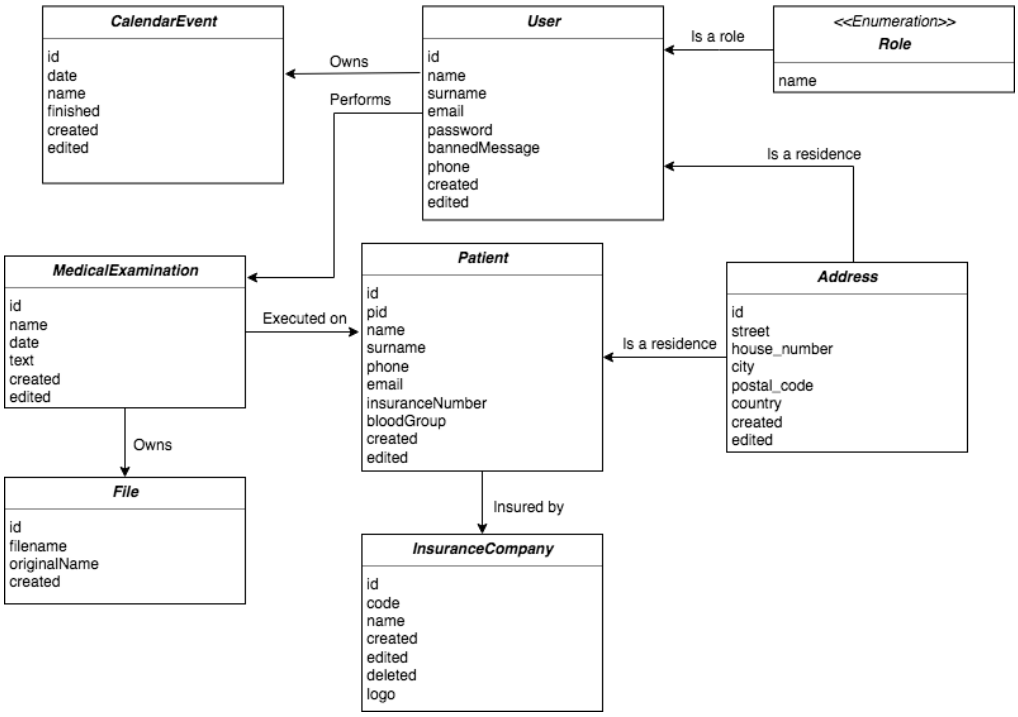
## 3.2 Analýza

Další částí při vytváření softwaru je analýza. Tato analýza navazuje na předchozí část specifikaci požadavků. Analýza by měla přinést jisté artefakty, kterými jsou **analytické třídy** a **realizace případů užití**. Cílem analýzy je tedy vytvoření analytického modelu a diagram realizace případu užití. Při práci na části analýzy a specifikace požadavků se může stát, že se pracovní postupy budou často překrývat. Při vytváření analýzy je nejdůležitější snaha o určení základního chování systému.

### 3.2.1 Analytický model

Tvorba analytického modelu je první částí analýzy. Výstupem by měl být analytický model, který se bude zaměřovat primárně na to, co systém dělá. Analytický model by neměl řešit, jak bude systém určitou činnost dělat. Na tvorbě analytického modelu by se měl podílet primárně analytik, ale měl by být srozumitelný pro všechny osoby zainteresované do vývoje software a klienta, pro kterého je software vyvíjen. Je to důležité z důvodu možnosti se vyjádřit ke směru vývoje softwaru a jeho funkcí.

Dále je uvedený diagram, který zobrazuje analytický model sestavený z analytických tříd s klíčovými pojmy. Z tohoto analytického modelu se bude vycházet v pozdější fázi tvorby softwaru, tj. v návrhu.



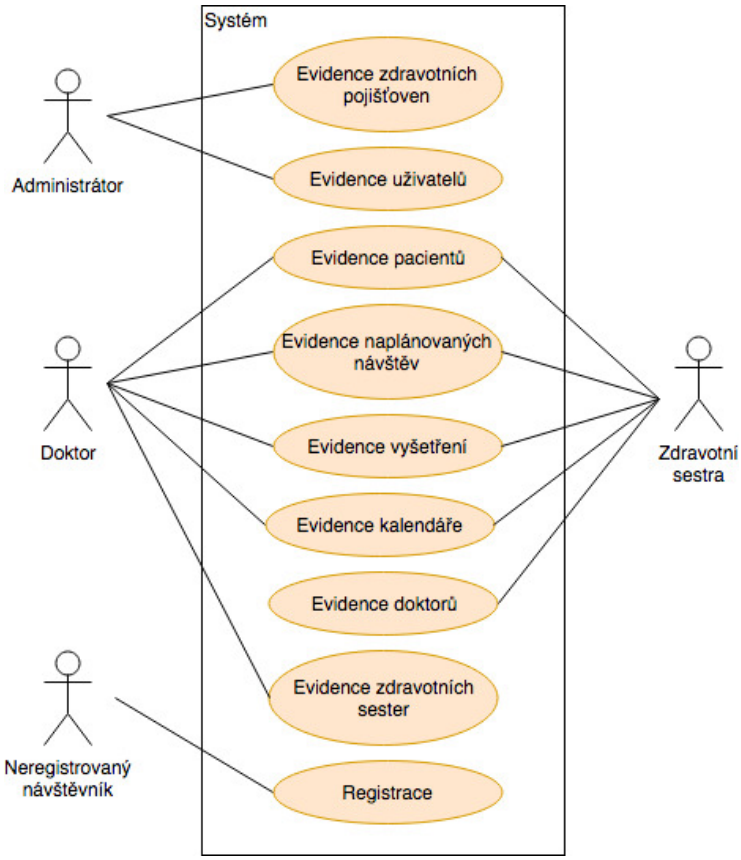

Obrázek 10: Analytický model

*Zdroj: Vlastní zpracování*

### 3.2.2 Realizace případů užití

Realizace případů užití je další fází, navazující na předchozí vytvoření analytického modelu. Výsledkem realizace případů užití by měl být **diagram případů užití** a **specifikace případů užití**.

**Diagram případů užití** zobrazuje jednotlivé aktéry, případy užití, ohraničení systému a vztahy. Aktérem je označen někdo nebo něco, co interaguje se systémem, který je v případu užití ohraničený. Toto ohraničení pomáhá rozpoznat, co je součástí systému a co nebo kdo je mimo systém. Případem užití se rozumí jistá činnost, kterou je nutné provést pro dosažení určitého cíle. Mezi činnostmi a aktéry jsou vztahy, které popisují jaký typ vtahu mezi nimi je.




Obrázek 11: Diagram případů užití

*Zdroj: Vlastní zpracování*

Na výše uvedeném obrázku je zobrazen diagram případů užití pro nově navrhovaný systém. Jsou zde 4 aktéři, kteří mohou přijít se systémem do styku a určitým způsobem s ním pracovat. Mezi tyto aktéry patří 3 registrované role a jedna neregistrovaná. Činnosti jsou znázorněny jako elipsy.

**Specifikace případů užití** je detailní popis jednotlivých případů užití z Use Case diagramu. Jelikož UC diagram zobrazuje pouze názvy případů užití, tak by nemusel být v některých případech moc vypovídající. Od toho existuje právě detailnější specifikace případů užití. Která obsahuje unikátní identifikátor ID, název, krátký popis, aktéry, základní tok a podmínky pro dokončení. Specifikace je možné rozšířit o alternativní toky, podmínky pro spuštění a mnohé další parametry.

Následují dvě specifikace, které jsou určené roli registrovaného administrátora. Bez role administrátora jsou funkce těchto Use Case specifikací zcela nepřístupné. První se zabývá evidencí zdravotních pojišťoven přístupných doktorům a zdravotním sestrám.

Tabulka 11: Případ užití evidence zdravotních pojišťoven

| | |
|---|---|
| ID | UC1 |
| Název | Evidence zdravotních pojišťoven |
| Krátký popis | Evidence zdravotních pojišťoven obsahuje CRUD operace nad zdravotními pojišťovnami. |
| Aktéři | Administrátor |
| Základní tok | 1. Systém zobrazí seznam pojišťoven<br>2. Aktér vybere operaci, kterou chce provést<br>3. Uloží upravené údaje nebo zvolí smazání<br>4. Systém provedenou operaci provede nad daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli administrátor |

*Zdroj: Vlastní zpracování*

Specifikace UC2 je druhá specifikace zaměřená za aktéra s rolí Administrátor. Detailní popis specifikace je v následující tabulce.

Tabulka 12: Případ užití evidence uživatelů

| | |
|---|---|
| ID | UC2 |
| Název | Evidence uživatelů |
| Krátký popis | Aktér s rolí administrátor má možnost spravovat registrované uživatele, daným uživatelům odebrat přístup nebo jim přístup vrátit |
| Aktéři | Administrátor |
| Základní tok | 1. Aktér obdrží od systému seznam registrovaných uživatelů<br>2. U konkrétního uživatele vybere akci, kterou chce provést (odebrání přístupu, navrácení přístupu)<br>3. V případě odebírání přístupu aktér vyplní důvod<br>4. Systém uloží změnu do databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli administrátor |

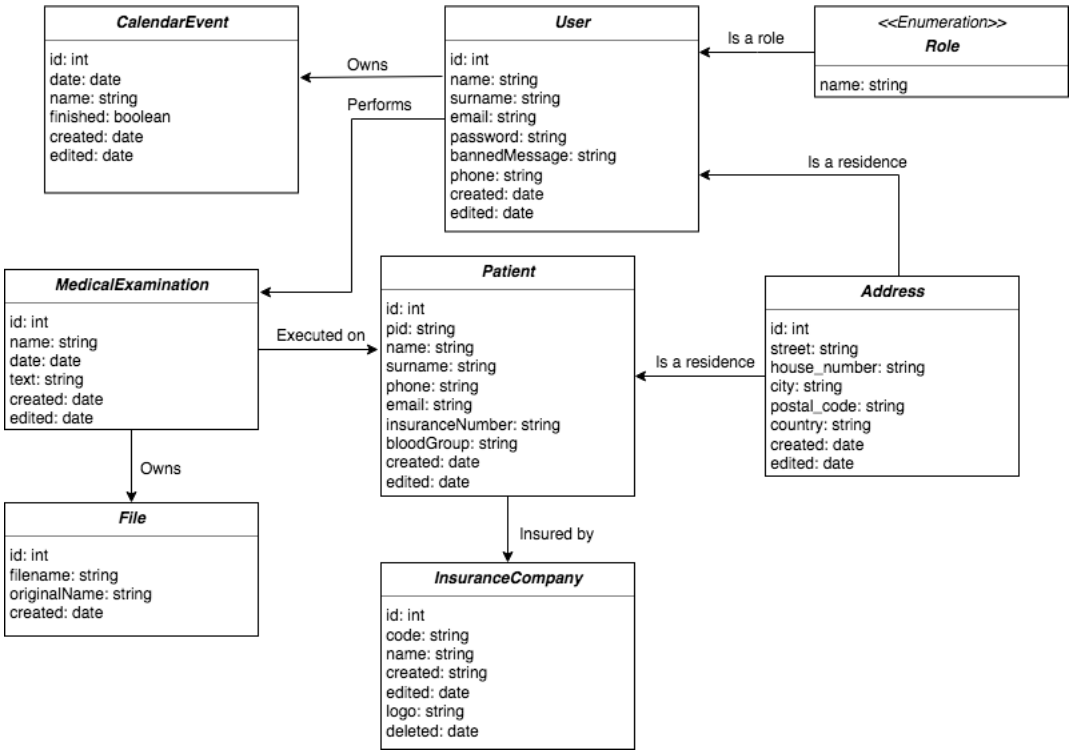
*Zdroj: Vlastní zpracování*

Další specifikace případů užití jsou přidány do přílohy.

## 3.3 Návrh a implementace

Fáze návrhu a implementace navazuje na fázi analýzy. Výstupem fáze návrhu by měl být návrhový model, obsahující detailnější informace o třídách než model analytický. Jelikož třídy

v návrhovém modelu budou odpovídat třídám využitým k vytvoření entit Doctrine 2, bude využit k následnému vytvoření databázového modelu.

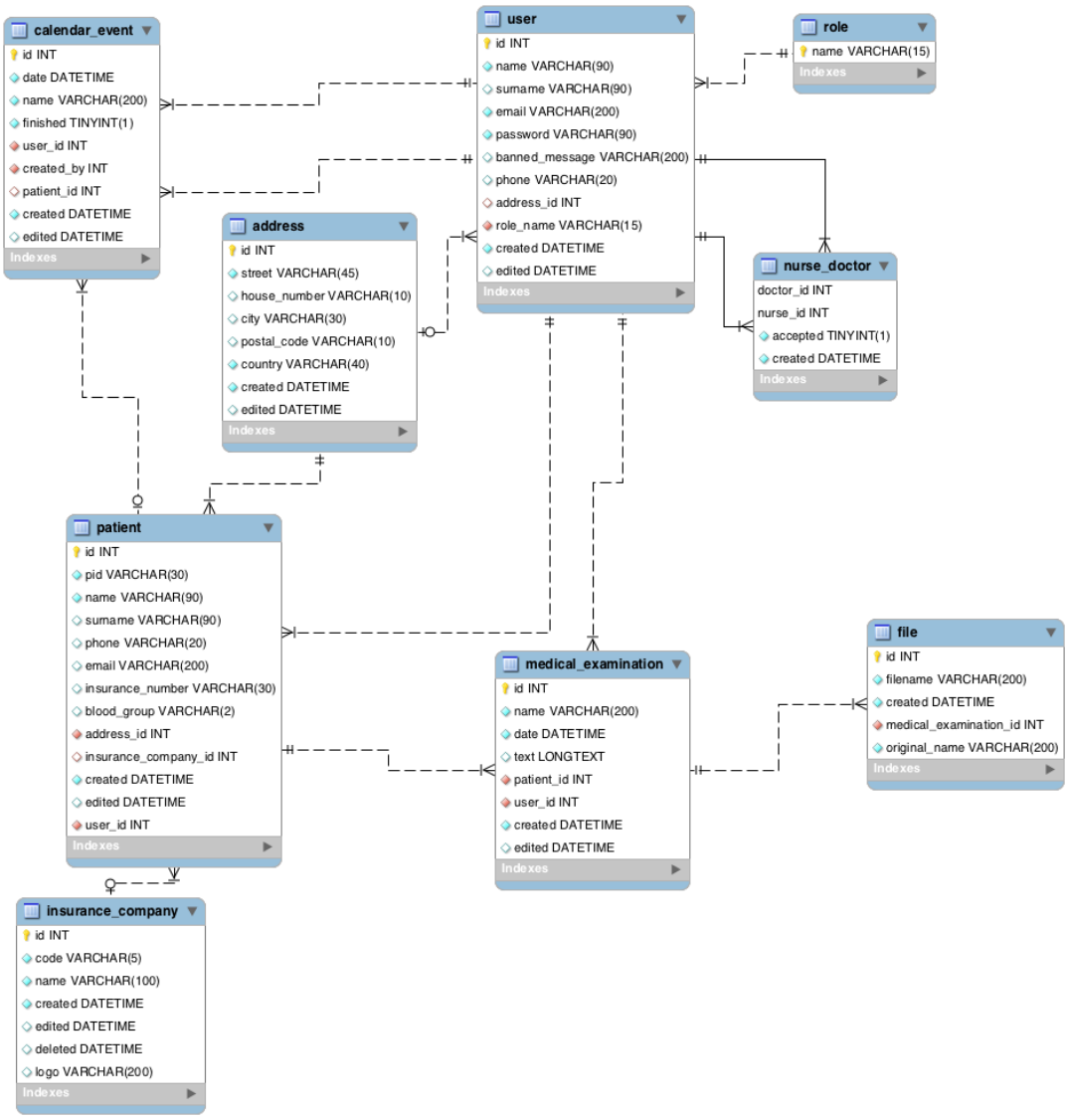



Obrázek 12: Návrhový model

*Zdroj: Vlastní zpracování*

Jak lze vidět na předchozím diagramu, návrhový model je velmi podobný modelu analytickému. Je doplněn o detailní informace, které budou potřeba při vytváření databázového modelu.

V aplikaci bude využita třívrstvá architektura MVC popsána již výše v podkapitole zabývající se PHP frameworkem Symfony. Tato třívrstvá architektura bude v části nazývané Model pracovat s entitami Doctrine 2. Jednotlivé entity odpovídají tabulkám, zobrazeným v následujícím ER diagramu vytvořeném v programu MySQL Workbench.




Obrázek 13: ER diagram

*Zdroj: Vlastní zpracování*

### 3.3.1 Využité technologie

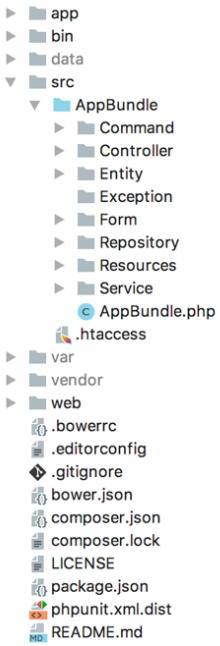
V teoretické části byly popsány technologie a jejich alternativy, které je možné využít při tvorbě webových aplikací. V této podkapitole autor uvádí, které konkrétní technologie se rozhodl zvolit při tvorbě vlastní aplikace pro správu pacientů.

Pro tvorbu systému byl využit jazyk PHP 7.1 společně s PHP frameworkem Symfony 3.4. Dále byla využita knihovna jQuery 3.3 a šablona Admin LTE 2.4.3. Například šablona Admin LTE obsahuje další závislosti, které jsou instalovány díky Boweru 1.8.2. Pro zobrazování

chybových nebo potvrzovacích hlášení byl využit Bootstrap Notify 3.1.3. Pro ukládání dat byla využita databáze MySQL verze 5.7.

### 3.3.2 Struktura projektu

V této podkapitole autor popisuje strukturu vytvořeného projektu. Struktura je velmi ovlivněna frameworkem Symfony a jeho architekturou MVC.




Obrázek 14: Struktura projektu

*Zdroj: Vlastní zpracování*

**App** obsahuje konfigurační soubory .yml a AppKernel, jádro aplikace.

V adresáři **bin** jsou spustitelné PHP soubory. Adresář obsahuje Symfony console a Symfony requirements. **Symfony requirements** slouží ke kontrole, zda je pro spuštění projektu nainstalované vše potřebné. **Symfony console** slouží pro spouštění PHP skriptů z konzole, jsou zde skripty pro mazání cache, vytvoření účtu administrátora a další.

Adresář **data** slouží pro ukládání uživatelských dat, například se sem ukládají soubory z vyšetření pacientů.

**Src** obsahuje kód samotné aplikace. Je zde AppBundle, který obsahuje kontrolery, entity, výjimky, repozitáře pro práci s entitami a mnohé další. V této složce je kód, který autor vytvořil.

Adresář **var** slouží pro ukládání cache na serveru, sessions a logování chyb.

**Vendor** obsahuje externí knihovny, které se instalují pomocí composeru. Tato složka se vytvoří po spuštění příkazu composer install.

Poslední adresář **web** obsahuje externí knihovny stažené pomocí boweru, vlastní CSS, také obsahuje hlavní přístupový bod a tím je app.php. Symfony 3.4 ještě nebere v potaz soubory .env takže pro vývojáře je zde app_dev.php. Tento adresář také obsahuje favicon, robots.txt a další soubory.

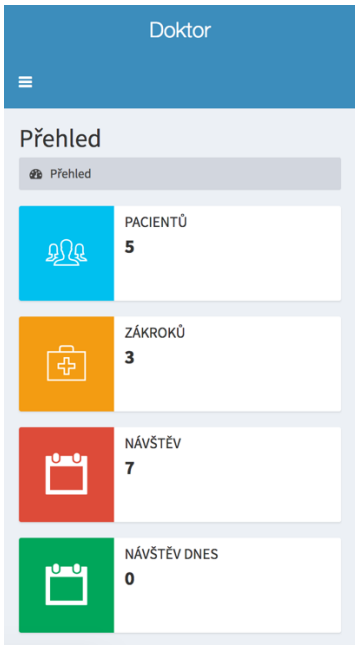
## 3.4 Uvedení do provozu

Postup a nastavení nutné pro uvedení systému do provozu je popsané v souboru README.md který je součástí aplikace. Soubor README.md je zohledněný na GitHubu, proto používá pro formátování textu syntaxi Markdown. V souboru README.md jsou také vypsány požadavky pro spuštění. [3]

# 4 POPIS FUNKCÍ SYSTÉMU

Tato kapitola bude popisovat jednotlivé části systému a funkce, které daná část poskytuje. Bude rozdělena do čtyř podkapitol podle jednotlivých částí, které systém obsahuje. V této kapitole bude také popsáno, které role jsou potřebné pro přístup do určitých částí systému.

Celý systém má sjednocený vzhled, který je pro přehlednost barevně odlišený podle toho, ve které části se uživatel nachází. Všechny části systému jsou responzivní, tudíž se přizpůsobí velikosti displeje zobrazovacího zařízení bez ztráty přehlednosti.

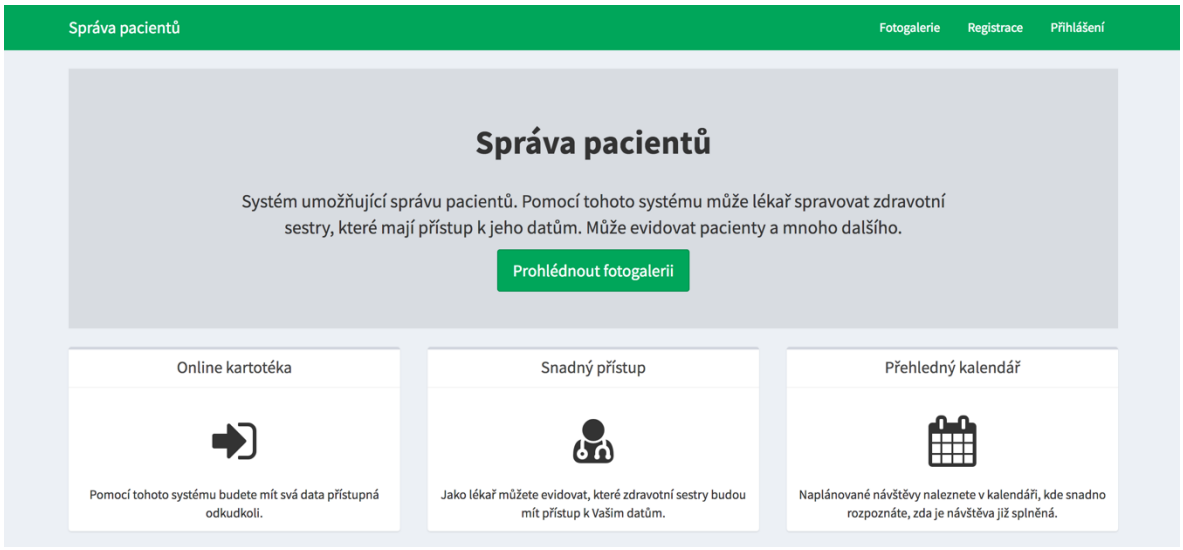



Obrázek 15: Responzivní přehled

*Zdroj: Vlastní zpracování*

## 4.1 Veřejná část systému

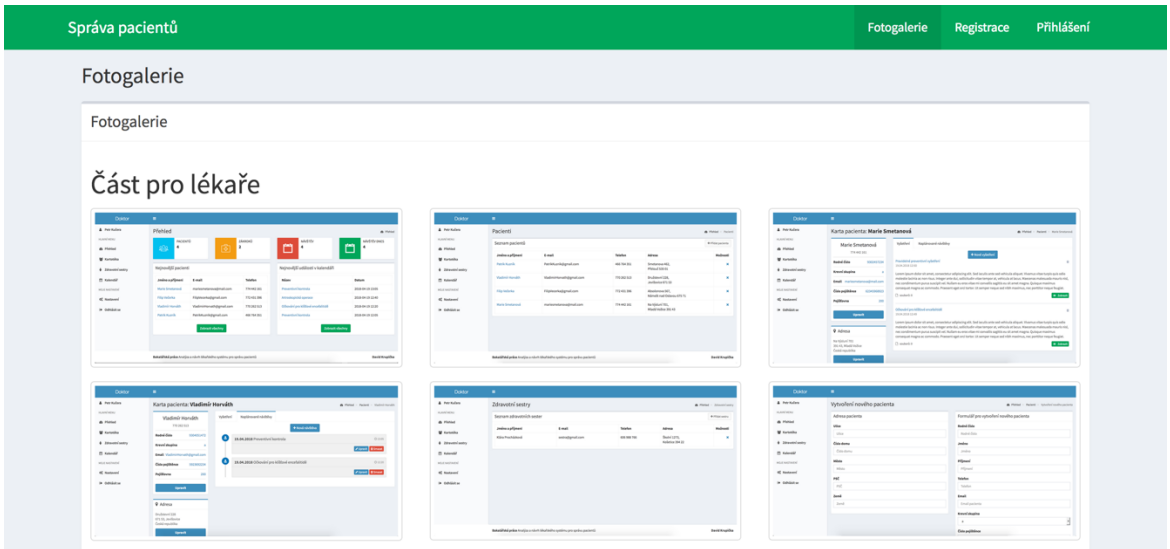
Tato část webové aplikace je jako jediná přístupná neregistrovaným návštěvníkům. Zde může neregistrovaný uživatel získat základní informace o systému. Tato část obsahuje horní menu, pomocí kterého může návštěvník procházet stránky veřejné části.




Obrázek 16: Veřejná část systému

*Zdroj: Vlastní zpracování*

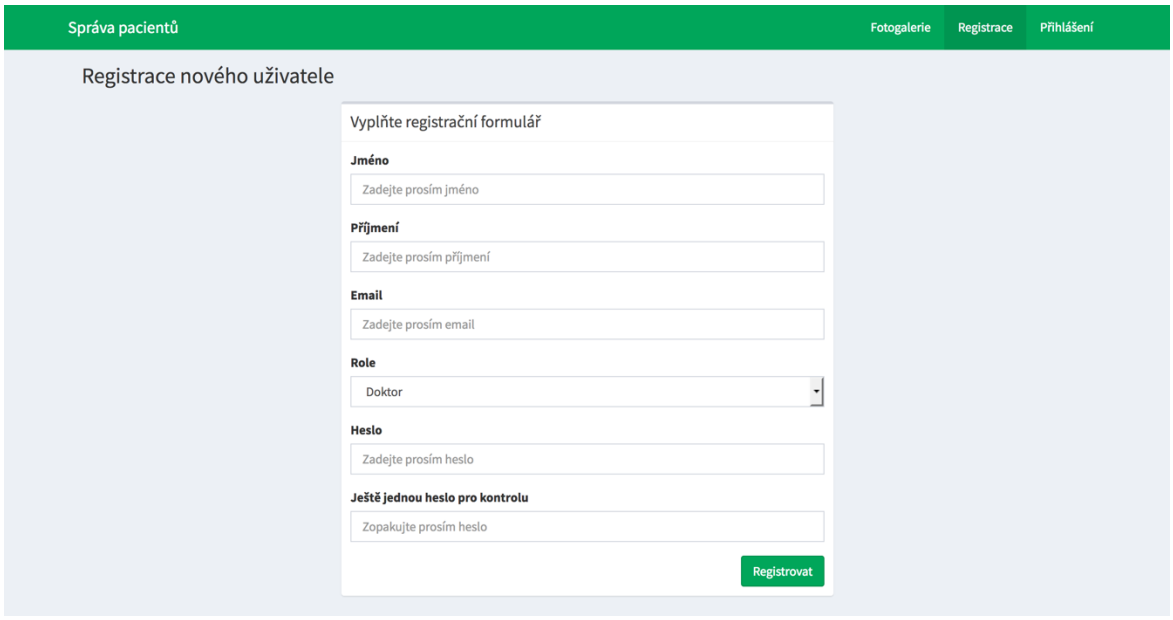
Než se uživatel registruje, má možnost si ve fotogalerii prohlédnout fotografie systému. Fotogalerie obsahuje několik snímků obrazovky ze systému, rozdělených podle částí pro doktory a zdravotní sestry. Uživatel tak získává představu o obsahu systému, ještě před zadáním osobních údajů při registraci.




Obrázek 17: Fotogalerie ve veřejné části

*Zdroj: Vlastní zpracování*

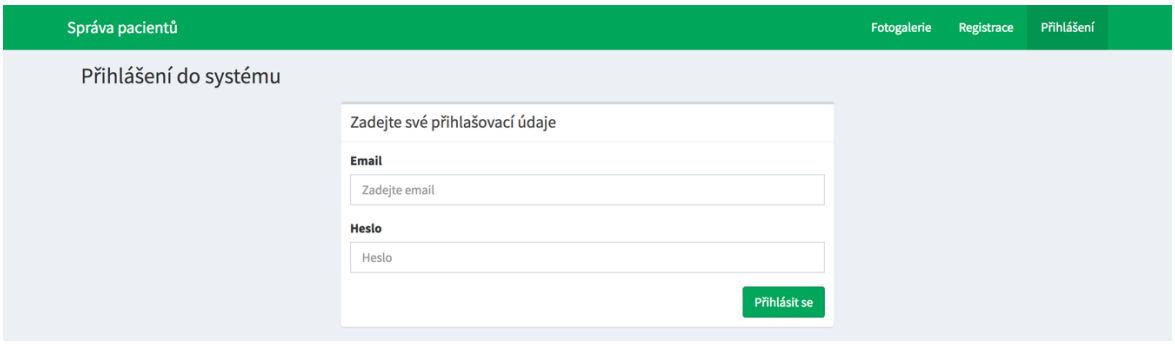
Další možností, kterou tato část systému poskytuje je **registrace**. Jestliže chce uživatel systém využívat, je nutné, aby se registroval. Při registraci má uživatel na výběr, zda chce mít roli sestra nebo doktor. Kromě výběru role je po uživateli vyžadováno vyplnění formuláře se základními kontaktními údaji a hesla. Heslo je vyžadováno po uživateli dvakrát.




Obrázek 18: Registrace uživatele

*Zdroj: Vlastní zpracování*

Po úspěšné registraci má uživatel možnost se do systému přihlásit. Veřejná část systému obsahuje formulář pro přihlášení, na který bude návštěvník přesměrován po kliknutí na tlačítko přihlášení v horním menu. K tomu, aby se mohl uživatel úspěšně přihlásit je potřeba znát email a heslo. Po úspěšném přihlášení je uživatel přesměrován do části systému, která odpovídá jeho přidělené roli.



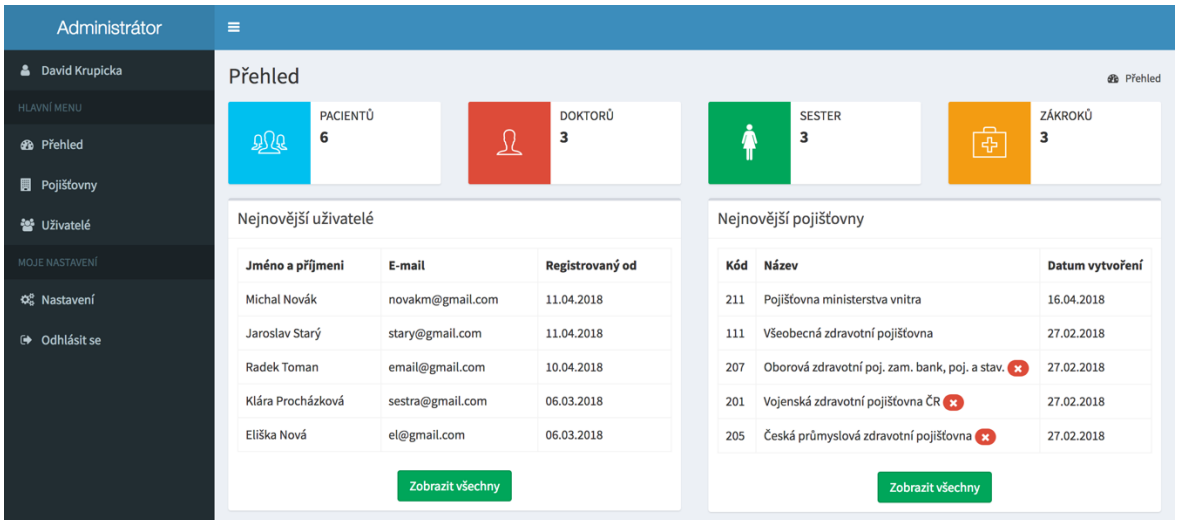

Obrázek 19: Přihlášení

*Zdroj: Vlastní zpracování*

## 4.2 Část systému určená administrátorům

Pro přístup do administrátorské části je potřeba být přihlášený a mít přidělenou roli administrátor. Po přihlášení administrátora se zobrazí přehled, ve kterém je zobrazen počet registrovaných doktorů, zdravotních sester, kolik systém obsahuje pacientů a kolik je v databázi uloženo zákroků provedených na pacientech.

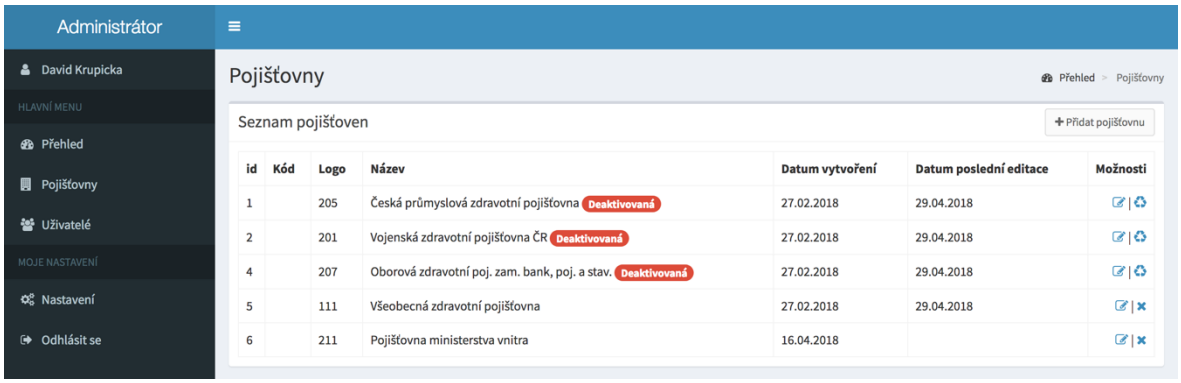
Na levé straně nalezne administrátor menu, které umožňuje procházet administrátorskou část systému. Je zde k nalezení odkaz pro správu pojišťoven, uživatelů, nastavení, přehled a možnost se odhlásit ze systému.




Obrázek 20: Přehled administrátora

*Zdroj: Vlastní zpracování*

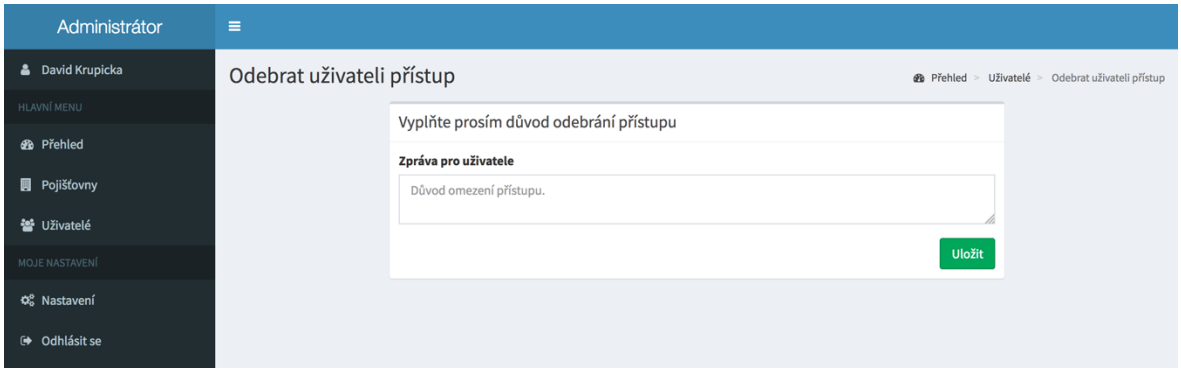
První z možností administrátora je **správa pojišťoven**. Každá pojišťovna má evidovaný kód, název a logo, které pomáhá pojišťovnu lépe vizuálně identifikovat. Administrátor má možnost výše uvedená data měnit, nebo přidat novou pojišťovnu do systému. Tyto pojišťovny jsou pak nabízeny doktorům a sestrám při registraci nových pacientů nebo jejich editaci.




Obrázek 21: Správa pojišťoven

*Zdroj: Vlastní zpracování*

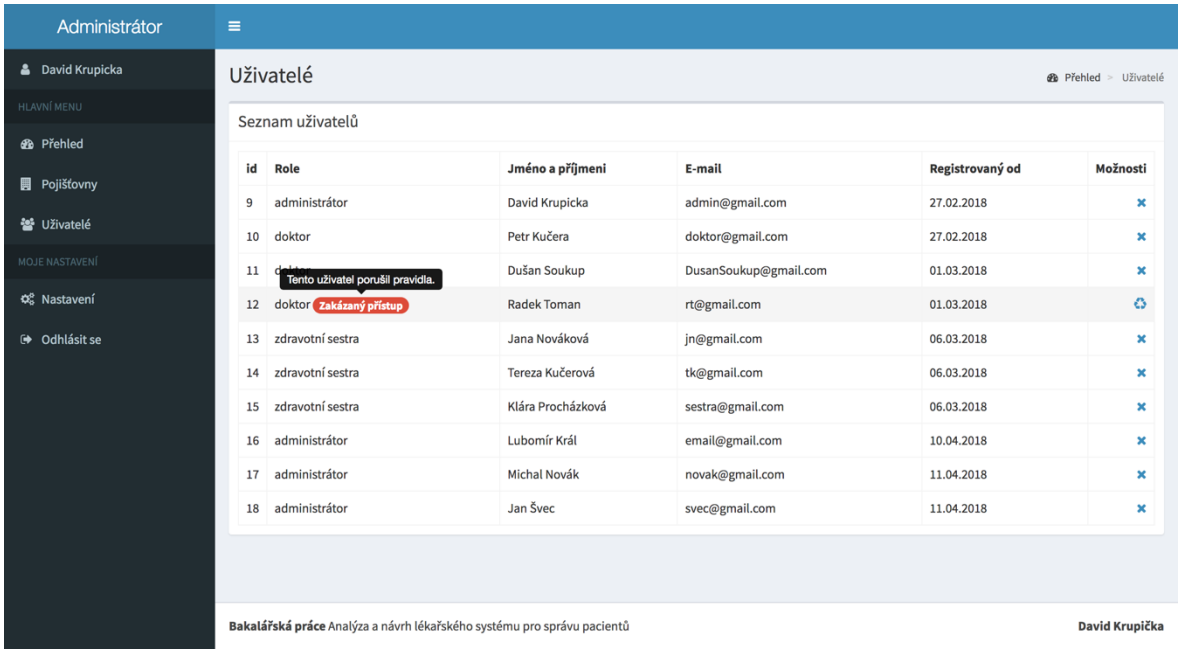
Administrátorská část dále poskytuje možnost **správy uživatelů**. Přihlášený administrátor může doktorům a sestrám zakázat přístup do systému s uvedenými přihlašovacími údaji. Jestliže se administrátor rozhodne zakázat uživateli přístup do systému, udělí mu zákaz a napíše krátkou zprávu vysvětlující důvod udělení zákazu přístupu do systému. Tato zpráva se pak zobrazí uživateli, když se bude snažit přihlásit do systému.




Obrázek 22: Odebrání přístupu uživateli

*Zdroj: Vlastní zpracování*

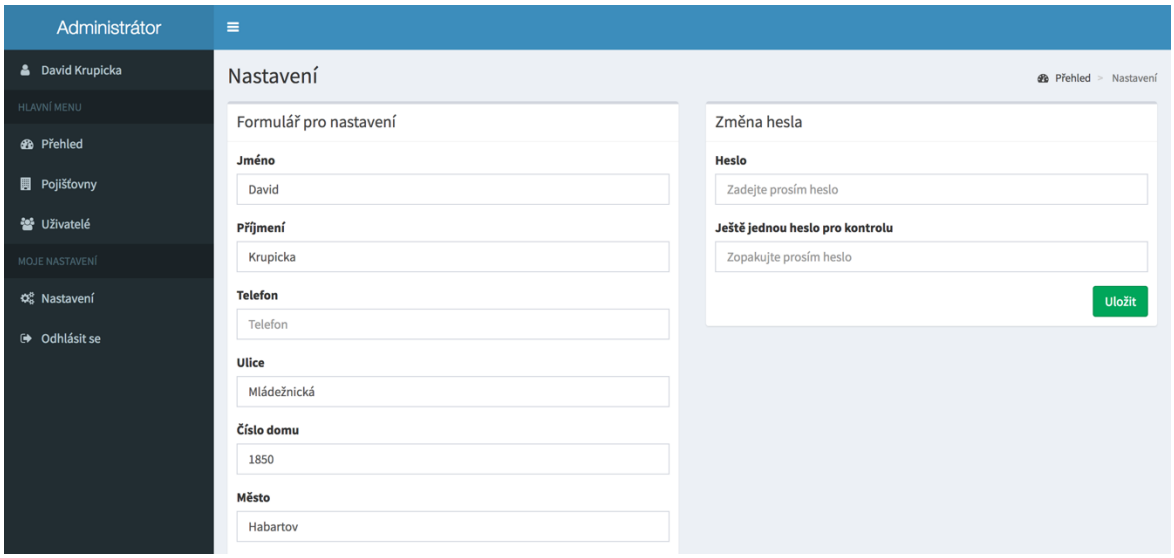
V přehledném seznamu uživatelů je možné zjistit, kdy byl uživatel registrovaný, jakou má roli, jméno, příjmení a email. Email je pro administrátora viditelný z důvodu možné potřeby kontaktování uživatele. Uživatelé s odebraným přístupem jsou označeni štítkem a po přejetí kurzorem nad tímto štítkem se zobrazí zpráva, která byla zadána administrátorem jako důvod pro odebrání přístupu do systému.




Obrázek 23: Správa uživatelů

*Zdroj: Vlastní zpracování*

Další možností v menu administrátora je **nastavení**. Nastavení slouží ke změně hesla a osobních údajů administrátora. Tato stránka obsahuje dva formuláře, jeden pro osobní údaje a druhý pro heslo. Změna hesla vyžaduje zadání nového hesla dvakrát. Druhé vyplnění hesla je pro kontrolu, že se administrátor nepřepsal. Stránka nastavení vypadá stejně pro všechny role využívající tento systém.




Obrázek 24: Nastavení

*Zdroj: Vlastní zpracování*

### 4.2.1 Vytvoření účtu administrátora

Účet administrátora může vytvořit pouze správce serveru, na kterém aplikace běží. Proto se nemůže administrátor registrovat na veřejné části, ale byl pro registraci administrátorského účtu vytvořen speciální PHP příkaz. Tento příkaz je nutné spouštět z terminálu. Po otevření adresáře nainstalované aplikace stačí zadat následující příkaz.

```
bin/console app:create-admin-account
```

Příkaz pro vytvoření administrátorského účtu se spouští se třemi argumenty. Těmito argumenty jsou jméno, příjmení a email. Po správném zadání proběhne výzva o zadání hesla pro nově vytvářený účet.

```
david:sprava-pacientu$ bin/console app:create-admin-account Michal Novak novakm@gmail.com
Please enter your new password

A new administrator account has been created. Name: Michal, Surname: Novak, email: novakm@gmail.com
```

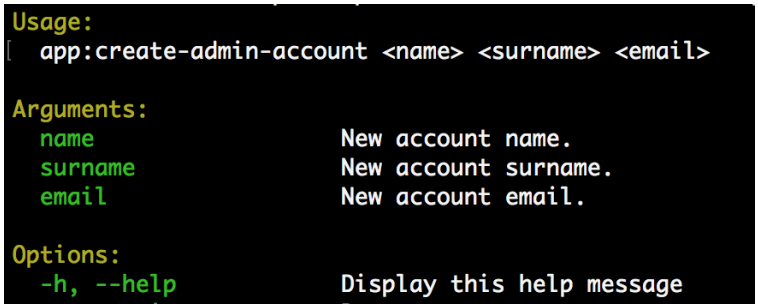
Obrázek 25: Vytvoření administrátorského účtu

*Zdroj: Vlastní zpracování*

Správce má možnost si pro daný příkaz zobrazit nápovědu, pomocí přepínače --help nebo -h.

```
bin/console app:create-admin-account --help
```

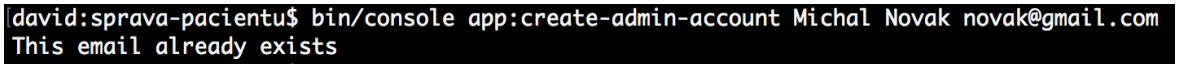
Po zadání příkazu s přepínačem --help se správci zobrazí následující nápověda.




Obrázek 26: Příkaz pro vytvoření nového účtu administrátora

*Zdroj: Vlastní zpracování*

Stejně jako u registrace ve veřejné části se i zde kontroluje, zda nedochází k registraci již registrovaného emailu. Jestliže se správce pokusí vytvořit administrátorský účet s emailem, který je již v systému, zobrazí se mu chybová hláška.



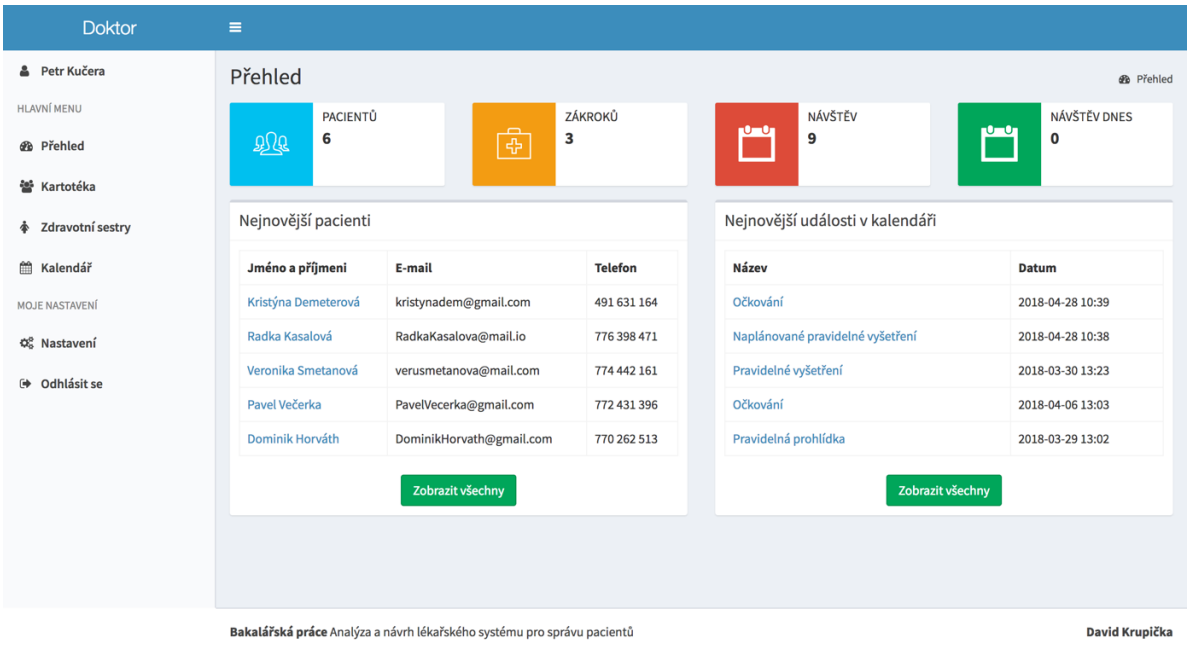

Obrázek 27: Neúspěšné vytvoření administrátorského účtu

*Zdroj: Vlastní zpracování*

## 4.3 Část systému přístupná doktorům

Tato část je vzhledově velmi podobná administrátorské části. Pro přehlednost je odlišená barevně. Administrátor má tmavé menu, doktoři a zdravotní sestry mají menu světlé. Stejně jako administrátor má doktor v menu odkazy na nastavení, přehled a tlačítko pro odhlášení.

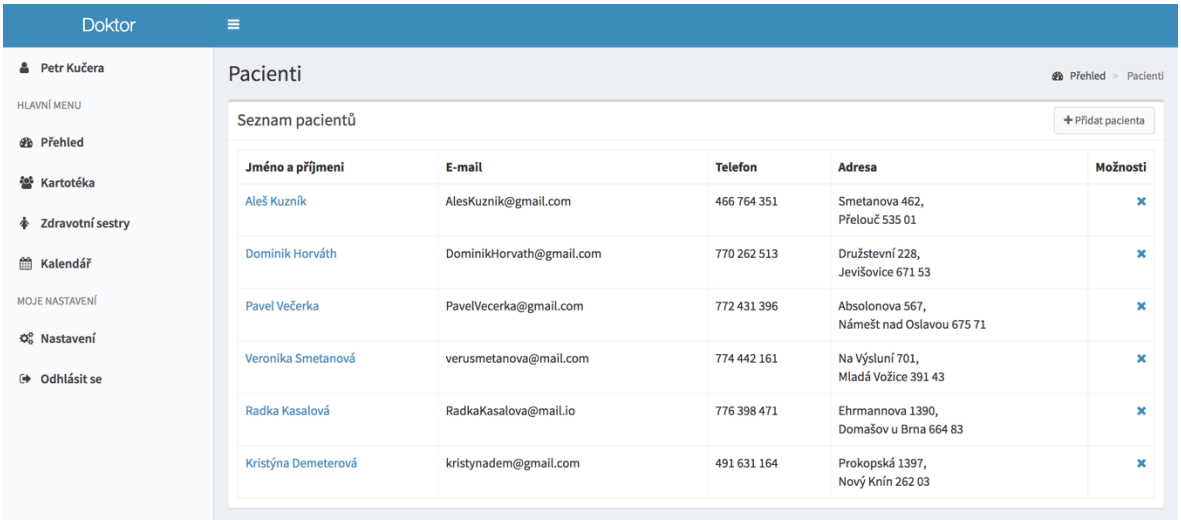
Dále doktorská část obsahuje správu pacientů, kterou nalezne v kartotéce, správu zdravotních sester, vlastní kalendář, evidenci vyšetření a možnost plánování návštěv u pacientů.




Obrázek 28: Přehled u role doktor

*Zdroj: Vlastní zpracování*

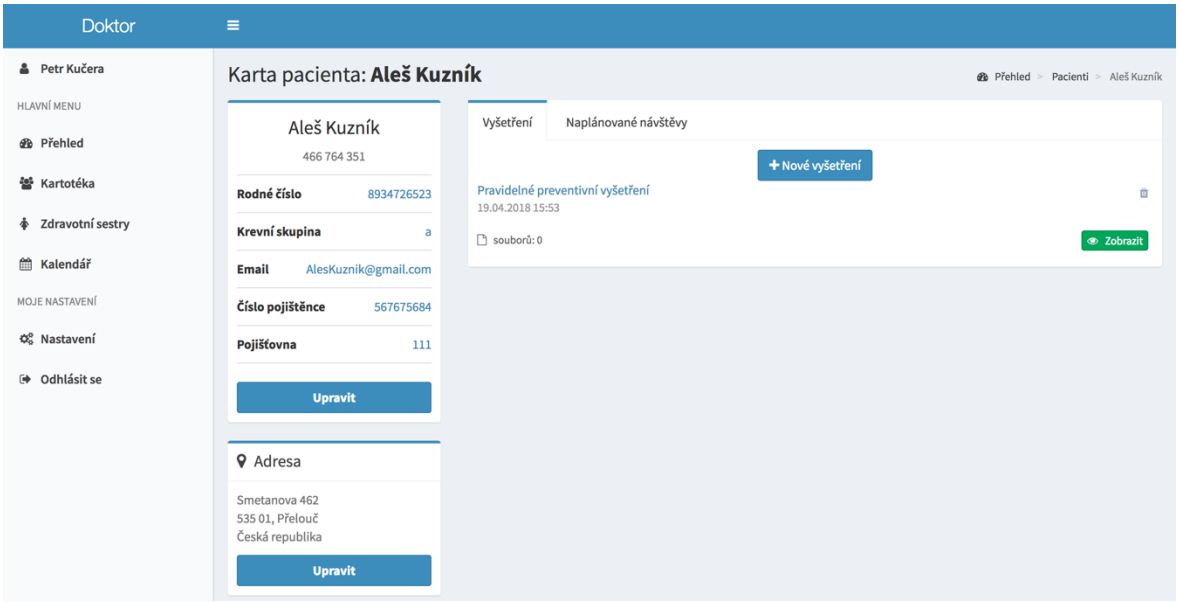
Stránka **kartotéka** obsahuje seznam všech pacientů konkrétního přihlášeného doktora. Z této stránky umožňuje systém přejít na formulář sloužící pro přidání nového pacienta do systému.




Obrázek 29: Správa pacientů

*Zdroj: Vlastní zpracování*

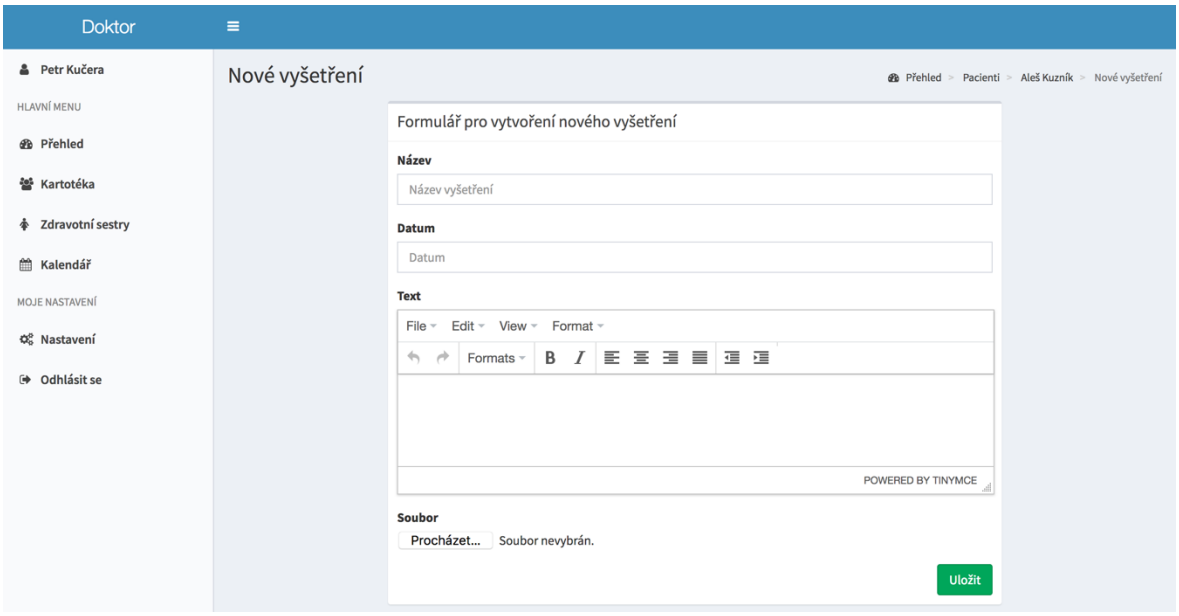
Ve formuláři je nutné vyplnit údaje o pacientovi a případně i jeho adresu. Po úspěšném uložení formuláře je uživatel přesměrován na **detail karty pacienta**. Zde nalezne doktor přehledně uspořádané informace o pacientovi.




Obrázek 30: Detail pacienta

*Zdroj: Vlastní zpracování*

V detailu pacienta má doktor možnost přejít na formulář editace informací o pacientovi. Další možností, která se nachází v detailu pacienta, je **evidence vyšetření** a **plánování návštěv**. Doktor zde vidí seznam všech již proběhlých vyšetření. Pomocí formuláře, na který je přesměrován po kliknutí na tlačítko nové vyšetření, může doktor vytvořit nové vyšetření pro konkrétního pacienta.

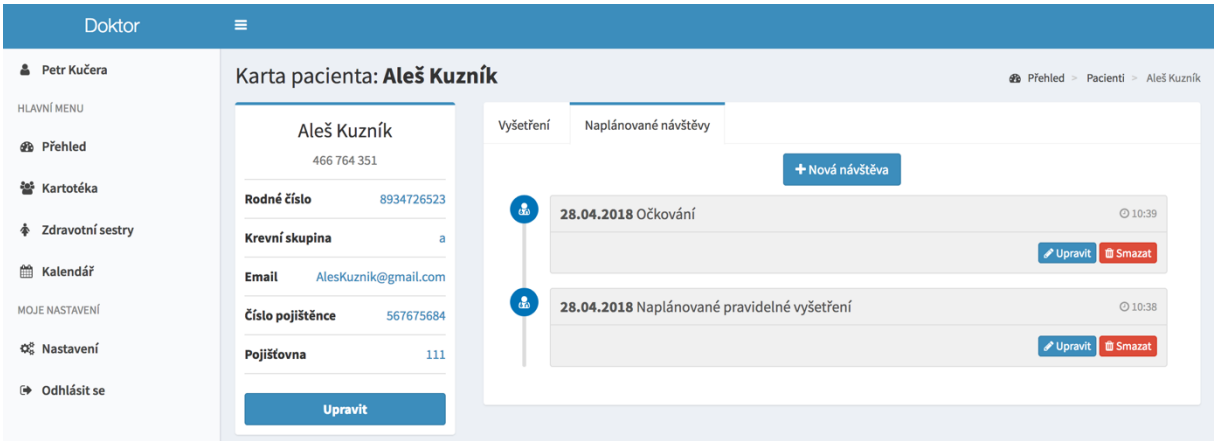



Obrázek 31: Nové vyšetření

*Zdroj: Vlastní zpracování*

K vyšetření lze případně nahrát fotografie. Již vytvořené vyšetření je možné editovat. Formulář editace vypadá velmi podobně jako formulář pro vytvoření nového vyšetření.

Na kartě **plánování návštěv** nalezne doktor seznam všech naplánovaných návštěv pro daného pacienta. Opět tu existuje možnost přidání nové návštěvy pomocí jednoduchého formuláře. Již naplánované návštěvy lze editovat nebo odstranit.

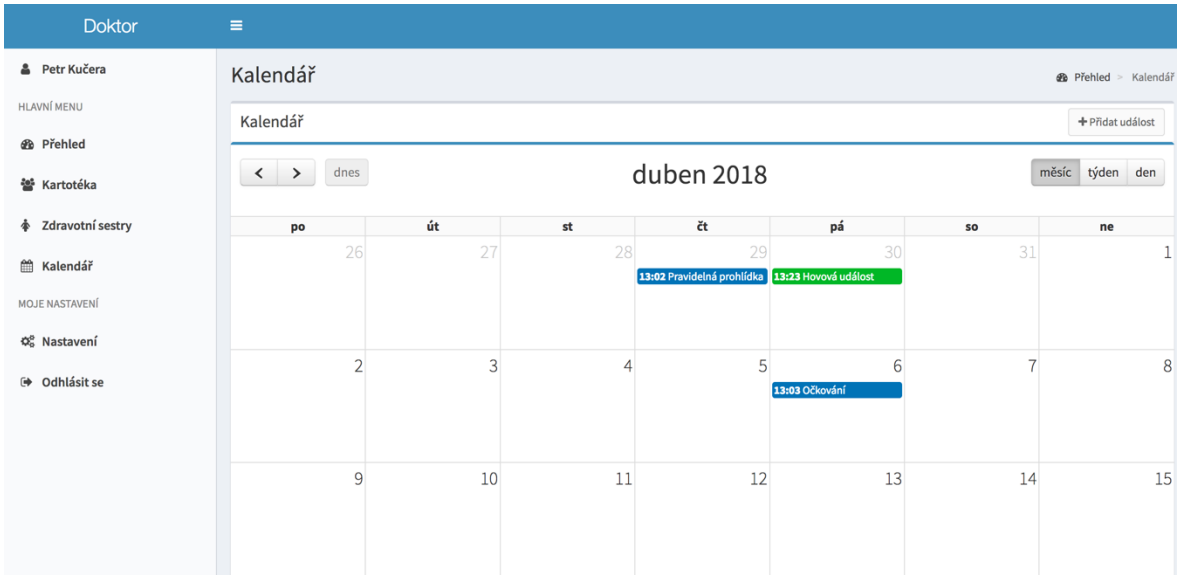



Obrázek 32: Plánování návštěv

*Zdroj: Vlastní zpracování*

Naplánované návštěvy se zároveň přidají do **kalendáře** doktora daného pacienta. Detail kalendáře lze zobrazit po kliknutí na tlačítko kalendář v levém menu. Kalendář také obsahuje události, které přidá doktor přes tlačítko pro přidání události. Dále obsahuje události, které doktorovi může přidat do kalendáře zdravotní sestra.

Pro přehlednost jsou položky v kalendáři barevně oddělené. Zelené jsou události, které byly označeny za vyřízené. Modré jsou události přidané doktorem a žlutě jsou vyznačené události přidané zdravotní sestrou.

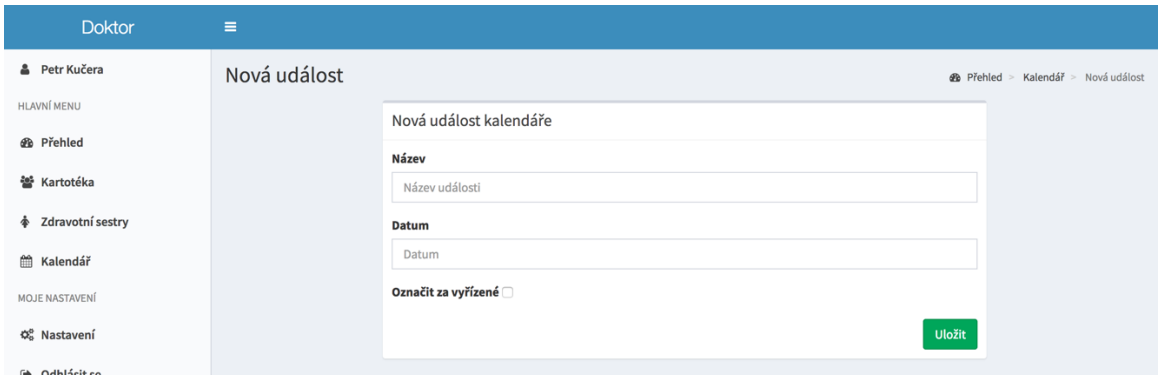



Obrázek 33: Kalendář

*Zdroj: Vlastní zpracování*

Pro snadnou práci s formulářem jsou zde tlačítka, které ovlivňují zobrazení kalendáře. Doktor si tak může zobrazit události z vybraného měsíce, dne nebo týdne. V základním nastavení se doktorovi zobrazuje náhled měsíce.

Po kliknutí na tlačítko přidat událost je doktor přesměrován na formulář pro přidání nové události. Formulář obsahuje pole pro zadání názvu události, data a možnost označit událost za vyřízenou. Vyřízené události se nezobrazují zdravotní sestře v čekárně. Stránka čekárny bude popsána v další kapitole, věnující se části systému pro roli zdravotní sestra.



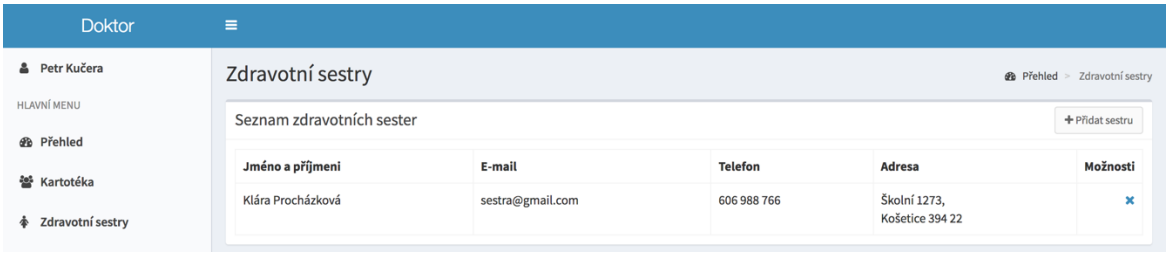

Obrázek 34: Formulář události

*Zdroj: Vlastní zpracování*

Editace události pak vypadá velmi podobně, zobrazí se po kliknutí na položku v kalendáři, kterou chce doktor editovat.

Další možností, kterou nabízí levé menu, je přechod na stránku **správy zdravotních sester**. Zde je doktorovi zobrazen seznam zdravotních sester, které mají přístup ke správě jeho pacientů v systému a jeho kalendáři a dalších již zmíněných dat.

Doktor nemá možnost zdravotní sestře v systému vytvořit účet. Zdravotní sestra se musí zaregistrovat v části přístupné veřejnosti. Doktor jí následně může zaslat požadavek, aby získala přístup k doktorovým datům. Tento požadavek musí být zdravotní sestrou potvrzen. Po úspěšném potvrzení získá zdravotní sestra přístup ke správě výše zmíněných dat doktora.

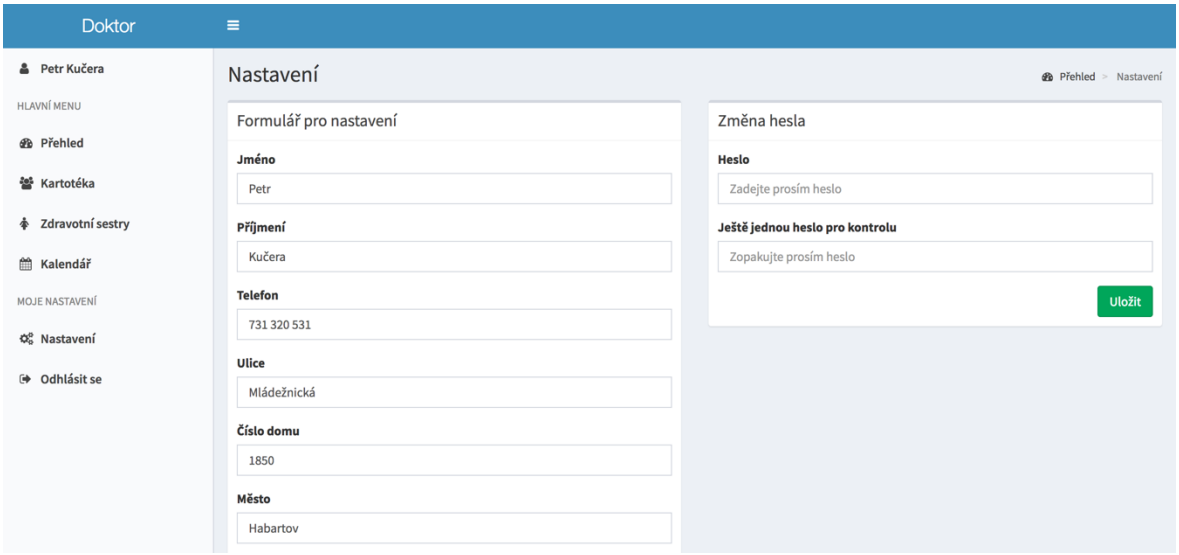



Obrázek 35: Evidence zdravotních sester

*Zdroj: Vlastní zpracování*

Stejně jako administrátor má doktor možnost nastavit své osobní údaje a heslo v nastavení. Stránka pro nastavení osobních údajů a hesla vypadá stejně jako u administrátora, pouze je barevně odlišená jako zbytek systému. Osobní údaje vyplňuje doktor pro zdravotní sestru nebo

administrátora, těmto dvěma rolím budou osobní údaje zobrazeny kvůli možnosti potřeby doktora kontaktovat.

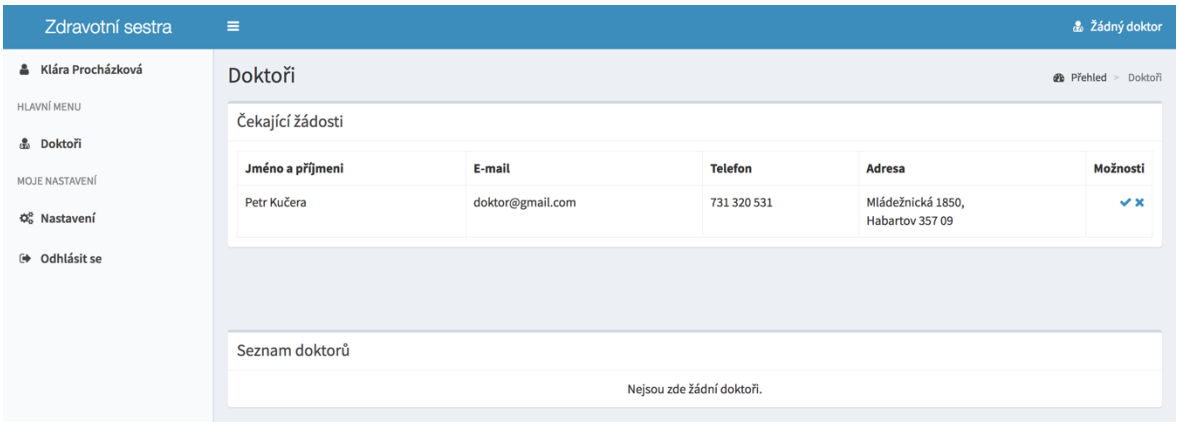



Obrázek 36: Nastavení doktora

*Zdroj: Vlastní zpracování*

## 4.4 Část systému pro zdravotní sestry

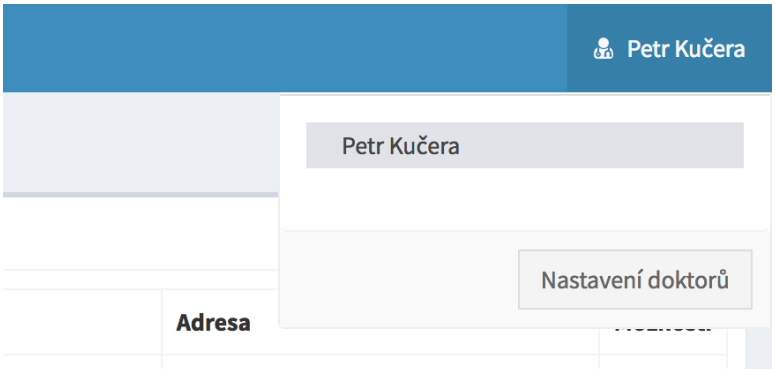
Tato část je vzhledově velmi podobná uživatelské části určené doktorům. Zdravotní sestra nemá v systému uložené žádné vlastní pacienty ani údaje v kalendáři. Veškerá data, která spravuje patří doktorovi, pro kterého pracuje. Aby mohla zdravotní sestra spravovat informace uložené doktorem, musí od doktora obdržet žádost, kterou příjme. Evidenci těchto žádostí nalezne v pravém menu po kliknutí na položku doktoři.




Obrázek 37: Evidence doktorů

*Zdroj: Vlastní zpracování*

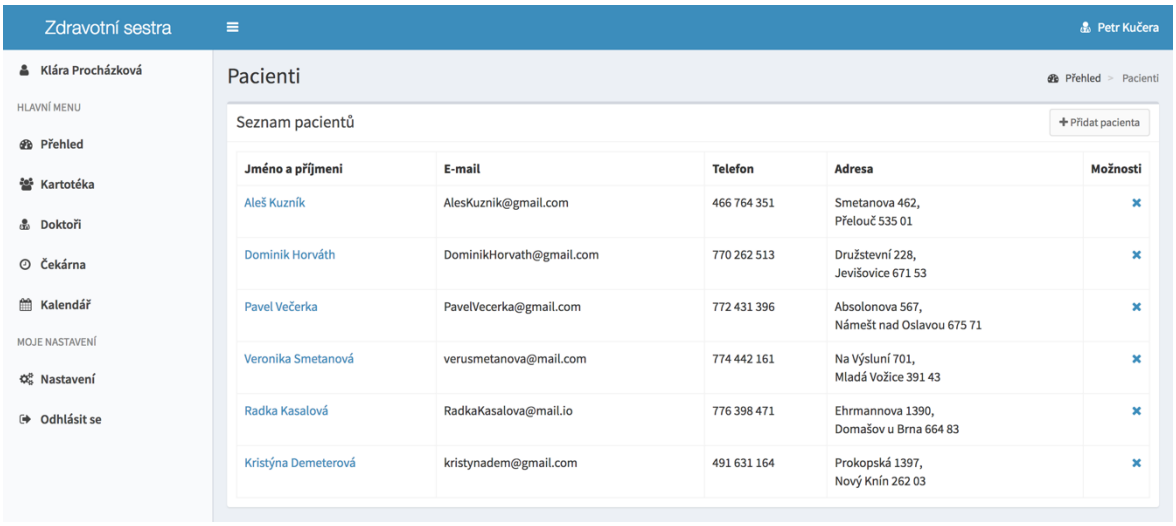
V pravém horním rohu se zdravotní sestře zobrazuje doktor, jehož data právě spravuje. Zdravotní sestra může doktora přepnout na libovolného, kterému schválila žádost. Po výběru doktora se zdravotní sestře zobrazí informační hláška, že byl doktor změněn a zobrazí se jí relevantní data.




Obrázek 38: Výběr doktora

*Zdroj: Vlastní zpracování*

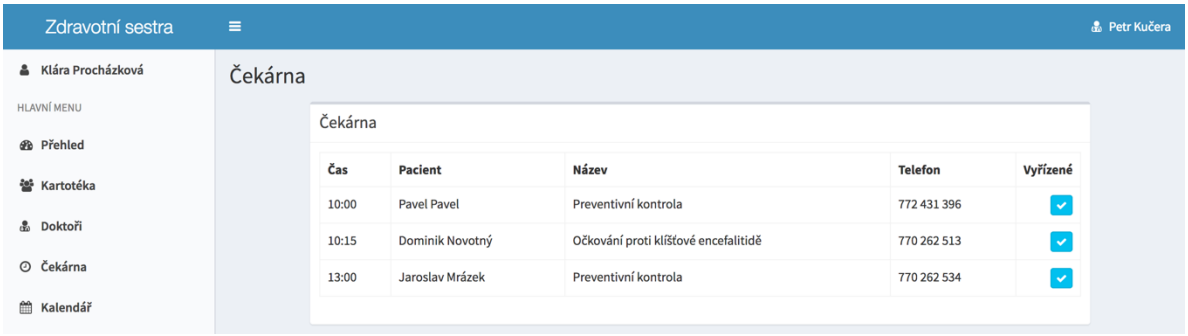
Stránka s pacienty pak vypadá stejně jako v části systému určené doktorům. Sestra má také stejné možnosti jako doktor. Pro vybraného doktora může spravovat evidenci kartotéky, kalendáře, vyšetření a plánování návštěv pacientů.




Obrázek 39: Kartotéka

*Zdroj: Vlastní zpracování*

Po kliknutí na odkaz **čekárna**, se zdravotní sestře zobrazí informace o naplánovaných návštěvách na dnešek. Zde má možnost některou z dnešních návštěv označit za vyřízenou. Po označení návštěvy za vyřízenou tato návštěva ze seznamu zmizí. Návštěvy jsou v seznamu zobrazeny podle data.




Obrázek 40: Čekárna

*Zdroj: Vlastní zpracování*

Možnost nastavení je pro zdravotní sestru stejná jako pro uživatele s rolí doktor. Osobní údaje zdravotní sestra vyplňuje pro případnou nutnost kontaktování zdravotní sestry ze strany doktora nebo administrátora.

# ZÁVĚR

Cílem této bakalářské práce bylo analyzovat existující software pro správu pacientů, navrhnout vlastní řešení, které bude fungovat jako SaaS aplikace a toto řešení také vytvořit. Nejprve byly popsány existující systémy, které jsou k dispozici na trhu a které mohou lékaři používat. Poté následoval popis technologií, které mohou být využity k tvorbě webové aplikace.

Na základě analyzovaných existujících systémů a za pomocí konzultací s odborným personálem z lékařských oborů byly vytvořeny požadavky na systém, jak funkční tak nefunkční. Po sepsání požadavků následovala analýza, která měla za cíl vytvořit analytický model systému. Na analýzu navazoval návrh a implementace, za použití technologií, popsaných ve druhé kapitole. Výsledkem je systém, který je možné spustit jako SaaS. V rámci této bakalářské práce byly také popsány požadavky pro úspěšné spuštění systému a postup, který je pro spuštění nutné dodržet.

Stanovené cíle byly dodrženy a implementovány podle zadání. Systém poskytuje správu pacientů, přihlašování pod různými rolemi, podle kterých jsou uživateli poskytovány různé funkcionality.

Lékaři mohou spravovat seznam zdravotních sester, které mají přístup k jejich datům. Lékař může v systému evidovat vlastní pacienty, plánovat pacientům návštěvy, evidovat vyšetření u jednotlivých pacientů, dále může k vyšetřením ukládat vlastní soubory. Zdravotním sestrám je umožněno spravovat data lékaře, jestliže se přes systém spárují účty lékaře a zdravotní sestry. Dále mohou zdravotní sestry spravovat evidenci čekárny, kde se zobrazí seznam objednaných pacientů na konkrétní den.

Tato práce byla vytvořena pro účel vypracování bakalářské práce. Jestliže by se systém spouštěl na produkci, je možné, že by se objevily problémy, které by bylo potřeba vyřešit, protože se s nimi autor nemusel setkat při tvorbě bakalářské práce. Dále by bylo možné pro další rozšíření funkčnosti implementovat další funkce, například napojení na elektronickou evidenci tržeb, eRecept, eNeschopenku a další. Díky REST API jsou možnosti dalšího rozšíření velmi široké, takže může vytvořený systém poskytovat i kvalitní základ pro další nadstavby.

# POUŽITÁ LITERATURA


[1] A JavaScript event calendar. *FullCalendar* [online]. [cit. 2018-04-30]. Dostupné z: https://fullcalendar.io/

[2] About MySQL. *MySQL* [online]. [cit. 2018-04-30]. Dostupné z: https://www.mysql.com/about

[3] About writing and formatting on GitHub. *GitHub* [online]. [cit. 2018-04-30]. Dostupné z: https://help.github.com/articles/about-writing-and-formatting-on-github/

[4] ALMSAEED, Abdullah. AdminLTE Control Panel Template. *AdminLTE* [online]. [cit. 2018-04-30]. Dostupné z: https://adminlte.io/

[5] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[6] Autentizace a autorizace. *Tirsul.cz* [online]. [cit. 2018-04-30]. Dostupné z: http://www.trisul.cz/bezpecnost-autentizace-autorizace

[7] BÖHMER, Marian. Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj]. Brno: Computer Press, 2012. ISBN 978-80-251-3338-5.

[8] Bringing MySQL to the web. *PhpMyAdmin* [online]. [cit. 2018-04-30]. Dostupné z: https://www.phpmyadmin.net

[9] Ceník. *Duna Medik* [online]. [cit. 2018-04-30]. Dostupné z: https://www.duna.cz/duna-medik/cenik

[10] Ceník. *Fons Galen* [online]. 2016 [cit. 2018-04-30]. Dostupné z: http://www.fonsgalen.cz/cenik

[11] Ceník. *WinMed* [online]. [cit. 2018-04-30]. Dostupné z: http://winmed.cz/o_programu/cenik

[12] Co je Laravel? *Laravel blog* [online]. [cit. 2018-04-30]. Dostupné z: http://laravelblog.cz/co-je-laravel

[13] Co všechno umí. *WinMed* [online]. [cit. 2018-04-30]. Dostupné z: ttp://winmed.cz/o_programu/co-vsechno-umi/

[14] Composer. *Nette Framework* [online]. [cit. 2018-04-30]. Dostupné z: https://doc.nette.org/cs/2.4/composer

[15] ČÁPKA, David. 1. díl - Úvod do JavaScriptu. *ITnetwork* [online]. [cit. 2018-04-30]. Dostupné z: https://www.tvorba-webu.cz/javascript/

[16] ČÁPKA, David. 1. díl - Úvod do jQuery. *ITnetwork* [online]. [cit. 2018-04-30]. Dostupné z: https://www.itnetwork.cz/javascript/jquery-zaklady/javascript-tutorial-funkcionalni-programovani-a-jquery-webova-kalkulacka

[17] ČÁPKA, David. 3. díl - Směrovač (router). *ITnetwork* [online]. [cit. 2018-04-30]. Dostupné z: https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-router-smerovac

[18] ČÁPKA, David. 27. díl - Bootstrap - Grid systém Bootstrapu. *ITnetwork.cz* [online]. [cit. 2018-04-30]. Dostupné z: https://www.itnetwork.cz/html-css/bootstrap/bootstrap-grid-system-bootstrapu

[19] Demoverze FONS Galen. *Fons Galen* [online]. 2016 [cit. 2018-04-30]. Dostupné z: http://www.fonsgalen.cz/demoverze/

[20] Downloading jQuery. *JQuery* [online]. [cit. 2018-04-30]. Dostupné z: https://jquery.com/download/

[21] EONASDAN. Bootstrap 3 Datepicker v4 Docs. *GitHub* [online]. 2017 [cit. 2018-04-30]. Dostupné z: https://eonasdan.github.io/bootstrap-datetimepicker/

[22] Evidence. *Duna Medik* [online]. [cit. 2018-04-30]. Dostupné z: https://www.duna.cz/duna-medik/

[23] Fons Galen. *Fons Galen* [online]. 2016 [cit. 2018-04-30]. Dostupné z: http://www.fonsgalen.cz/

[24] FOWLER, Martin. *Destilované UML*. Praha: Grada, 2009. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

[25] Grid system. *Bootstrap* [online]. [cit. 2018-04-30]. Dostupné z: https://getbootstrap.com/docs/3.3/css/#grid

[26] Grid system. *Bootstrap* [online]. [cit. 2018-04-30]. Dostupné z: https://getbootstrap.com/docs/4.0/layout/grid/

[27] JANOVSKÝ, Dušan. CSS styly - úvod. *Jak psát web* [online]. [cit. 2018-04-30]. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html

[28] JANOVSKÝ, Dušan. Úvod do JavaScriptu. *Jak psát web* [online]. 2016-03-08 [cit. 2018-04-30]. Dostupné z: https://www.jakpsatweb.cz/javascript/javascript-uvod.html

[29] JANOVSKÝ, Dušan. Verze HTML. *Jak psát web* [online]. [cit. 2018-04-30]. Dostupné z: https://www.jakpsatweb.cz/html/verze-html.html

[30] JavaScript. *Tvorba webu* [online]. [cit. 2018-04-30]. Dostupné z: https://www.tvorba-webu.cz/javascript/

[31] KEANE, Danny. Bootstrap Notify. *Bootstrap Notify* [online]. [cit. 2018-04-30]. Dostupné z: http://bootstrap-notify.remabledesigns.com/

[32] KOFLER, Michael. Mistrovství v MySQL 5: [kompletní průvodce webového vývojáře]. Brno: Computer Press, 2007. ISBN 978-80-251-1502-2.

[33] KOSEK, Jiří. Historie a vývoj HTML. *HTML 5* [online]. 2013 [cit. 2018-04-30]. Dostupné z: http://htmlguru.cz/uvod-historie.html

[34] Learn PHP 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. New York, NY: Springer Science+Business Media, 2015. ISBN 9781484217290.

[35] LEISS, Oliver a Jasmin SCHMIDT. *PHP v praxi: pro začátečníky a mírně pokročilé*. Praha: Grada, 2010. Průvodce (Grada). ISBN 978-80-247-3060-8.

[36] MACHAČ, Marek. 10 nejlepších PHP frameworků pro vývojáře. *Interval.cz* [online]. 2015-10-29 [cit. 2018-04-30]. Dostupné z: https://www.interval.cz/clanky/10-nejlepsich-php-frameworku-pro-vyvojare/

[37] MÁCA, Jindřich. 1. díl - Úvod do Symfony frameworku pro PHP. *ITnetwork* [online]. [cit. 2018-04-30]. Dostupné z: https://www.itnetwork.cz/php/symfony/zaklady/uvod-do-symfony-frameworku-pro-php

[38] MICHÁLEK, Martin. K čemu je dobrý Bootstrap a frontend frameworky?. *Zdroják* [online]. 2013-06-12 [cit. 2018-04-30]. Dostupné z: https://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/

[39] MISHA. *Databáze* [online]. [cit. 2018-04-30]. Dostupné z: http://www.databaze.chytrak.cz

[40] MySQL profesionálně: optimalizace pro vysoký výkon. Brno: Zoner Press, 2009. ISBN 978-80-7413-035-9.

[41] MySQL Workbench. *MySQL* [online]. [cit. 2018-04-30]. Dostupné z: https://www.mysql.com/products/workbench

[42] O programu. *WinMed* [online]. [cit. 2018-04-30]. Dostupné z: http://winmed.cz/o_programu/

[43] ODELL, Den. *JavaScript: průvodce programováním ajaxových aplikací*. Brno: Computer Press, 2010. ISBN 978-80-251-2733-9.

[44] PHP /základy/. *Tvorba webu* [online]. [cit. 2018-04-30]. Dostupné z: https://www.tvorba-webu.cz/php/

[45] Porovnání programů. *Medicus* [online]. [cit. 2018-04-30]. Dostupné z: http://www.medicus.cz/ambulance/programy/porovnani-programu/

[46] Programy. *Medicus* [online]. [cit. 2018-04-30]. Dostupné z: http://www.medicus.cz/ambulance/programy/

[47] Routing. *Symfony* [online]. [cit. 2018-04-30]. Dostupné z: https://symfony.com/doc/3.4/routing.html

[48] STOHWASSER, Petr. Co je HTML. *Pěstujeme web* [online]. 2013 [cit. 2018-04-30]. Dostupné z: http://www.pestujemeweb.cz/obsah/html/html-tagy-struktura.php

[49] ŠIKA, Michal. Oracle – základní pojmy. *IT Blook* [online]. 2010-01-12 [cit. 2018-04-30]. Dostupné z: http://itblok.bastler.cz/subdom/itblok/oracle-zakladni-pojmy/

[50] ŠTRAUCH, Adam. Databáze MariaDB válcuje MySQL. *Root.cz* [online]. 2012-12-21 [cit. 2018-04-30]. Dostupné z: https://www.root.cz/clanky/databaze-mariadb-valcuje-mysql

[51] ŠTRÁFELDA, Jan. PostgreSQL. *Adaptic* [online]. [cit. 2018-04-30]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/postgresql

[52] The 10 Best WYSIWYG HTML Editors. *1stWebDesigner* [online]. [cit. 2018-04-30]. Dostupné z: https://1stwebdesigner.com/best-wysiwyg-html-editor/

[53] TICHÝ, Jan. Doctrine 2: úvod do systému. *Zdroják* [online]. 2010-07-21 [cit. 2018-04-30]. Dostupné z: https://www.zdrojak.cz/clanky/doctrine-2-uvod-do-systemu/

[54] VÍTEK, Rostislav. Symfony po krůčkách – Twig. *Zdroják* [online]. 2016-03-08 [cit. 2018-04-30]. Dostupné z: https://www.zdrojak.cz/clanky/symfony-po-kruckach-twig/

[55] VOLAKOVIČ, Patrik. 1. díl - Git - Historie a principy. *ITnetwork* [online]. [cit. 2018-04-30]. Dostupné z: https://www.itnetwork.cz/software/git/git-tutorial-historie-a-principy/

[56] VRÁNA, Jakub. Adminer - správa databáze v jednom souboru. *Adminer* [online]. [cit. 2018-04-30]. Dostupné z: https://www.adminer.org

[57] WHITE, Ashley. Bootstrap Lightbox. *GitHub* [online]. 2017-10-19 [cit. 2018-04-30]. Dostupné z: https://github.com/ashleydw/lightbox

[58] ZAJÍC, Petr. PHP (1) - Historie a budoucnost. *Tvorba webu* [online]. 27.5.2004 [cit. 2018-04-30]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=171

[59] Základy relačních databází, jejich využití v programování webu. *Vysoká škola ekonomická v Praze* [online]. [cit. 2018-04-30]. Dostupné z: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

# PŘÍLOHY

# PŘÍLOHA A – SPECIFIKACE PŘÍPADŮ UŽITÍ

| ID | UC3 |
|---|---|
| Název | Evidence pacientů |
| Krátký popis | Evidence pacientů obsahuje CRUD operace nad pacienty daného doktora. |
| Aktéři | Doktor, zdravotní sestra |
| Základní tok | 1. Systém zobrazí seznam pacientů<br>2. Aktér vybere operaci, kterou chce provést<br>3. Uloží upravené údaje nebo zvolí smazání<br>4. Systém provedenou operaci provede s daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli doktor, nebo zdravotní sestra. V případě zdravotní sestry je nutné aby měla přístup k datům daného doktora. |

| ID | UC4 |
|---|---|
| Název | Evidence naplánovaných návštěv |
| Krátký popis | Evidence naplánovaných návštěv obsahuje CRUD operace nad návštěvami. |
| Aktéři | Doktor, zdravotní sestra |
| Základní tok | 1. Systém zobrazí seznam naplánovaných návštěv pacienta<br>2. Aktér vybere operaci, kterou chce provést<br>3. Uloží upravené údaje nebo zvolí smazání<br>4. Systém provedenou operaci provede s daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli zdravotní sestra nebo doktor. Je nutné aby měl aktér přístup k danému pacientovi. |

| ID | UC5 |
|---|---|
| Název | Evidence vyšetření |
| Krátký popis | Evidence vyšetření obsahuje CRUD operace nad vyšetřeními daného pacienta. |
| Aktéři | Doktor, zdravotní sestra |
| Základní tok | 1. Systém zobrazí seznam vyšetření pacienta<br>2. Aktér vybere operaci, kterou chce provést<br>3. Uloží upravené údaje nebo zvolí smazání<br>4. Systém provedenou operaci provede s daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli doktor nebo zdravotní sestra. V případě zdravotní sestry musí mít přístup k danému pacientovi vybraného doktora. |

| ID | UC6 |
|---|---|
| Název | Evidence kalendáře |
| Krátký popis | Evidence kalendáře obsahuje CRUD operace nad událostmi kalendáře. |
| Aktéři | Doktor, zdravotní sestra |
| Základní tok | 1. Systém zobrazí kalendář se všemi událostmi<br>2. Aktér vybere operaci, kterou chce provést<br>3. Uloží upravené údaje nebo zvolí smazání<br>4. Systém provedenou operaci provede s daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli doktor nebo zdravotní sestra. V případě zdravotní sestry je nutné aby měla přístup k datům daného doktora. |

| ID | UC7 |
|---|---|
| Název | Evidence doktorů |
| Krátký popis | Pomocí evidence doktorů sestra potvrzuje nebo zamítá požadavky od doktorů. Při potvrzení požadavku získaného od doktora získává přístup k jeho datům. |
| Aktéři | Zdravotní sestra |
| Základní tok | 1. Systém zobrazí seznam doktorů a požadavků od doktorů<br>2. Aktér vybere schválení, zamítnutí nebo smazání<br>3. Systém vybranou operaci provede s daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli zdravotní sestra. |

| ID | UC8 |
|---|---|
| Název | Evidence zdravotních sester |
| Krátký popis | Doktor může určit, kterou zdravotní sestru požádá o správu jeho dat. Zdravotní sestře pošle požadavek, po jeho schválení může zdravotní sestra pracovat s daty doktora. |
| Aktéři | Doktor |
| Základní tok | 1. Systém zobrazí seznam zdravotních sester<br>2. Aktér vybere přidání požadavku, smazání požadavku<br>3. Systém provedenou operaci provede s daty databáze |
| Podmínky pro dokončení | Aktér má přidělenou roli doktor |

| | |
|---|---|
| ID | UC9 |
| Název | Registrace |
| Krátký popis | Na veřejné části se může zaregistrovat uživatel s výběrem role zdravotní sestra nebo doktor. |
| Aktéři | Neregistrovaný návštěvník |
| Základní tok | 1. Systém zobrazí formulář pro vyplnění registrace<br>2. Aktér vyplní formulář a ten odešle<br>3. Systém uloží data do databáze |
| Podmínky pro dokončení | Aktér není přihlášený |

# PŘÍLOHA B – PŘILOŽENÉ CD

Přiložené CD obsahuje:

- kód aplikace,
- PDF verzi bakalářské práce,
- skripty pro vytvoření databáze.