
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# VIRTUÁLNÍ ČEKÁRNA U LÉKAŘE

**VIRTUAL WAITING ROOM FOR MEDICAL PRACTICE**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **RICHARD MAJOLA**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **MARTIN KOLÁŘ, M.Sc.**
**SUPERVISOR**



**BRNO 2018**

*Zadání bakalářské práce/20295/2017/xmajol00*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

# Zadání bakalářské práce

Řešitel: **Majola Richard**
Obor: Informační technologie
Téma: **Virtuální Čekárna u Lékaře**
**Virtual Waiting Room for Medical Practice**
Kategorie: Web

Pokyny:

1. Vytvořte koncepci klienta "virtuální čekárny" jako webovou aplikaci s rozhraním pro doktora, mobilního pacienta, a pro čekárnu
2. Seznamte se s metodikou návrhu a implementace s uživatelskými testy
3. Seznamte se s technologiemi webového aplikací, a vyberte vhodný framework
4. Implementujte databázi, back-end, a front-end virtuální čekárny
5. Demonstrujte funkčnost na vhodném příkladě
6. Diskutujte dosažené výsledky a možnosti pokračování práce

Literatura:

- Podle domluvy s vedoucím

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kolář Martin, M.Sc.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
*vedoucí ústavu*

# Abstrakt

Cílem této bakalářské práce je vytvoření virtuální čekárny u lékaře jako webovou aplikaci, tvořenou skrz vývojový framework Laravel za použití jazyka PHP a databázového systému MySQL.Výsledná aplikace umožní lidem se registrodo systét přihlásit se do čekací fronty bez nutnosti být fyzicky přítomen v čekárně.



# Abstract

The aim of this Bachelor's thesis is to develop virtual waiting room at a doctor as web aplication, made up through development environment Laravel with using PHP language and MySQL database system. The final aplication allows people to register into the system and log in to waiting queue without having to be physically present in the waiting room.



# Klíčová slova

PHP, MySQL, databáze, webová aplikace, framework, Laravel, GUI, MVC



# Keywords

PHP, MySQL, database, web aplication, framework, Laravel, GUI, MVC



# Citace

MAJOLA, Richard. *Virtuální čekárna u lékaře.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Kolář, M.Sc.

# Virtuální čekárna u lékaře

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Martina Koláře, M.Sc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Richard Majola
8. května 2018


## Poděkování

Tímto bych chtěl poděkovat vedoucímu mojí bakalářské práce panu Martinovi Kolářovi, M.Sc. za skvělé vedení a odbornou pomoc při vytváření této práce v rámci pravidelných konzultací. Dále bych rád poděloval všem, kteří mi pomáhali při testování návrhů uživatelského rozhraní a za jejich zpětné vazby.

# Obsah

# Kapitola 1

# Úvod

V době, kdy nám vládnou informační technologie a všechno chceme mít co nejdřív, se pochopitelně ruku v ruce s tím projevuje i to, že nechceme na nic čekat, třeba ve frontě u lékaře.

Právě kvůli tomuto problému, vznikl nápad na tuto aplikaci, která nám ušetří spoustu času tím, že u lékaře nebudeme muset zbytečně čekat v čekárně, ale pohodlně se můžeme online přihlásit do čekárny a čas čekání strávit čímkoliv jiným.

Výsledný produkt bude nabývat podoby webové aplikace, která bude částečně sloužit jak informační, tak vyvolávací systém. Ten bude využíván oběma stranami. První stranou je lékařské rozhraní, které bude mít přehled o pacientech v čekárně a bude je vyvolávat do ordinace. Druhou stranou jsou pacienti, kteří využijí mobilní aplikace a po zaregistrování budou mít možnost si vzít číslo online, a ti kteří přijdou rovnou do čekárny, pro ně bude připravena platforma s vlastním rozhraním, přes které získají číslo.

Práce bude představena v několika jednoduchých krocích. V úvodní kapitole se bude zaobírat studiem podobně existujících systému, které jsou pro návrh a vývoj webových aplikací vhodné. V další kapitole bude představen návrh grafického rozhraní aplikace. Poté se vývoj práce přesune k hlavní kapitole a tou je implementace aplikace, v ní bude rozebrána implementace databáze, jazyk programu a popis jednotlivých tříd a jejich vlastnosti. V poslední části bude obsaženo uveřejnění aplikace skupině uživatelů, kteří jí podrobí testování a nakonec zhodnocením výslední webové aplikace.

# Kapitola 2

# Získávání informací

V této kapitole bude rozebráno studium informací o tom, jak nahlížet na projekt, co všechno se bude k vývoji aplikace potřebovat a jak jsou jednotlivé části pro vývoj aplikace důležité.

## 2.1 Studium existujících systémů

Koncept virtuálních front je vysoce žádaným tématem, díky kterému mnoho firem vytváří spousty virtuálních systémů. S těmito systémy se můžeme běžně setkat na poštách či úřadech a jsou také označovány jako "Vyvolávací systémy" [14].

Vyvolávací systém, jak napovídá název, pracuje na principu vyvolávání, to znamená, že klienti nejsou nuceni stát v řadě fronty, ale jsou vyzváni systémem ve chvíli, kdy na ně přijde řada k odbavení. Zpravidla bývá systém centralizovaný a rozdělen do více komponent:

- Tiskárna lístků s pořadovým číslem,
- obrazovka pro zobrazení čísla,
- aplikace či jiný systém pro zaměstnance k vyvolání dalšího klienta,
- server pro správu dat celého systému.

Důležitým faktem zůstává, že tyto systémy se vyskytují pouze v místě, kde jsou provozovány a není k nim vnější přístup, a tak vzniká rozšíření o online přistup do fronty. POZOR! důležité je nezaměňovat to s rezervací lístků ve frontě.

## 2.2 Vývojové technologie

Předtím, než se začne vytvářet určitý program, je potřeba zapřemýšlet o tom, které vývojové prostředí, a které technologie k tomu budou neideálnější. Protože se práce zaměří na vývoj webové aplikace, je tudíž z hlediska vývojových technologií nepodstatnější to, který framework[1] a programovací jazyk bude použit pro tuto aplikaci. Další částí je GUI[2], jenž uživatelům umožní ovládat aplikaci pomocí interaktivních grafických prvků.

### 2.2.1 Laravel 5

Když se bude mluvit o Laravelu, bude se tím mít na mysli verze 4.2 a vyšší. V tomto případě se vývoj aplikace zaměří na verzi Laravel 5 [11]. Laravel je volně dostupným frameworkem pro vývoj webových aplikací, který využívá architekturu MVC[3] [7]. Základní kamenem Laravelu je Symfony (viz kapitola 2.2.2), který poskytuje moduly jako například Browserkit a FileSystem. Další důležitou komponentou je Composer, což je nástroj pro správu závislostí v PHP a umožňuje deklarovat knihovny, na kterých aplikace závisí a spravuje je [18]. V neposlední řadě to je Eloquent ORM[4], který v Laravelu poskytuje, krásnou a jednoduchou implementaci ActiveRecord pro práci s databází. A nakonec důležitou součástí Laravelu je jeho vlastní šablonový engine "Blade", který spravuje veškeré pohledy s příponou .blade.php.

### 2.2.2 Symfony 2

Stejně tak jako Laravel i Symfony je volně dostupný framework, založený na MVC architektuře, určen pro vytváření webových aplikací. Symfony usiluje o urychlení tvorby a údržby webových aplikací, má za cíl dát vývojářům plnou kontrolu nad konfigurací od struktury adresářů až po knihovny. Zkáladní komponenty obsažené ve frameworku symfony jsou výše zmíněný Browserkit, který simuluje chování webového prohlížeče a umožňují nám klikat na odkazy, či odesílat formuláře ovšem neumožní zadávat požadavky na externí weby. Další již zmíněná komponenta je FileSystem, ta poskytuje základní nástroje pro abstrakní souborový systém [12].

Mezi obrovské výhody frameworku Symfony patří to, že se jedná o nejvýkonnější a nejflexibilnější PHP framework s inovativními možnostmi. Na

[1] specializovaný soubor knihoven pro usnadnění práce
[2] Graphical User Interface
[3] model-view-controler
[4] object relation mapper

druhou stranu je to složitý framework, který vyžaduje spoustu času na učení a osvojení si i jiných technologií, se kterými spolupracuje.

### 2.2.3 Nette

V neposlední řadě se představí Nette, který je svými vlastnostmi je označován jako nejpopulárnější framework v České republice a třetí na celém světe. Všechny zmíněné frameworky byl s MVC arichtekturou a ani Nette není vyjímkou. Čím se liší od jiných frameworků je to, že je stavěný především tak, aby byl co nejsrozumitelnější a nejvstřícnější uživatelům. Dále je přívětivé, že eliminuje bezpečnostní rizika a mohou v něm být vytvářené i e-shopy. Co se dá ovšem považovat za nevýhodů je, že je nutné ručně promazávat cache a databázová vrstva se chová výrazně jinak u tabulek bez primárního klíče [13].

### 2.2.4 Bootstrap

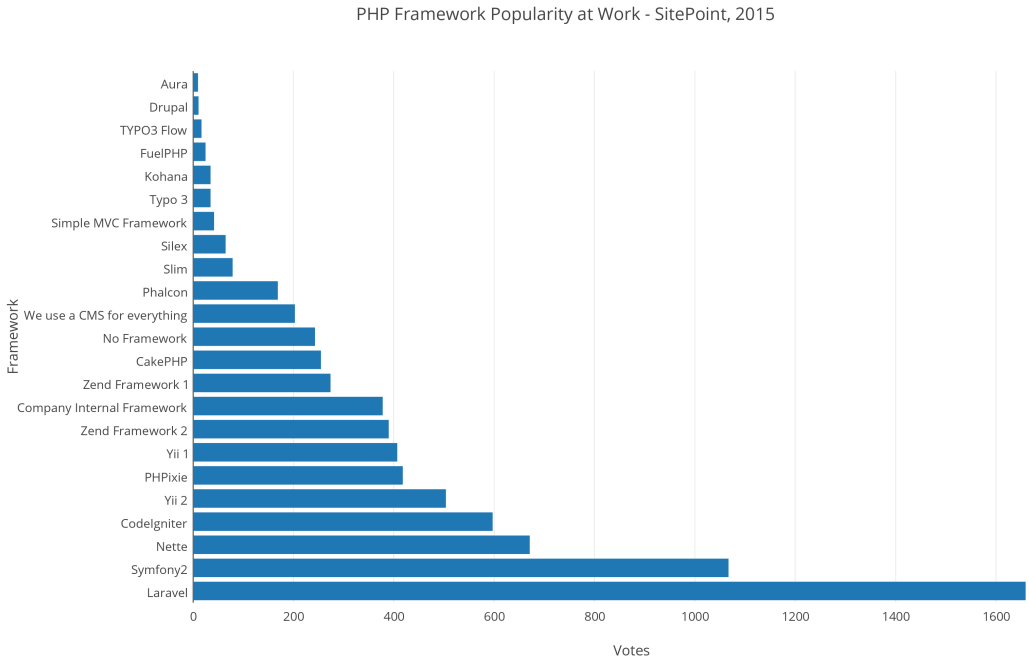
Bootstrap je volně dostupný front-end[5] framework, který je navržený pro elegantní, výkonnou a rychlou tvorbu grafické části webu a webových aplikací. V podstatě se jedná o sadu nástrojů, která obsahuje připravené HTML a CSS upravitelné šablony a další komponenty uživatelského rozhraní na webové stránce. Stejně tak obsahuje rozšíření pro využití JavaScriptu, přičemž jeho největší předností je responzivita[6] [17].

Tabulka 2.1: Popularita používaných frameworků v jednotlivých zemí (převzato z [19]

| **Stát** | **Oblíbený framework** | **Hlasy** |
|---|---|---|
| USA | Laravel | 819 |
| Česká republika | Nette | 770 |
| Velká Británie | Laravel | 496 |
| Německo | Symfony2 | 428 |
| Francie | Symfony2 | 343 |
| Rusko | Yii 2 | 235 |

[5]část webové aplikace, kterou vidí návštěvník stránek

[6]přizpůsobivost webové stránky zařízením s různým rozlišením




Obrázek 2.1: Popularita frameworků za rok 2015 (převzato z [19])

## 2.3 Databáze

Jelikož se jedná o webovou aplikaci, která obsahuje registraci uživatelů, musí se z toho důvodu vytvořit pochopitelně databáze, kde bude vše uloženo. Pro tento případ webových aplikací existuje databáze MySQL [1]. MySQL je relační databázový systém, který umožňuje technologiím PHP a Apache spolupracovat na zpřístupnění a zobrazení dat ve formátu čitelném pro internetové prohlížeče. Jako relační systém umožňuje spojování mnoha různých tabulek, což nabízí maximální efektivitu a rychlost.

Komunikace virtuální čekárny využívá architekturu klient-server[7] a pro tuto aplikaci bude databáze zastupovat funkci serveru. Databáze bude obsahovat data s primárně dvěma datovými typy:

- Statické,
- dynamické.

[7]klient se dotazuje na server, server odpovídá na požadavky klienta

Mezi statická data patří neměnné informace, bez možnosti úpravy, jako například odhadovaný čas příchodu na řadu a počet čekajících pacientů. Dynamické data jsou učena především pro lékařské rozhraní, kde lékař může určit průměrný čas vyšetření jednoho pacienta nebo čas ordinačních hodin. Pacient bude mít možnost upravit si vlastní profil, pro případ aktualizace osobních údajů, např. změnu pojišťovny.

### 2.3.1 E-R Diagram

Před tvorbou vlastní databáze je v prvé řadě nejdůležitější, aby byl vytvořen E-R Diagram (Entity-relationship diagram) [2]. ERD je nejznámější a nejčastěji používanou modelovací technikou pro návrh relačních databází. Jelikož se databáze skládá z tabulek, tak každá tabulka reprezentuje samostatnou entitu, kterou může v případě této aplikace být uživatel nebo návštěva v ordinaci.

## 2.4 Návrhy testování uživatelského rozhraní

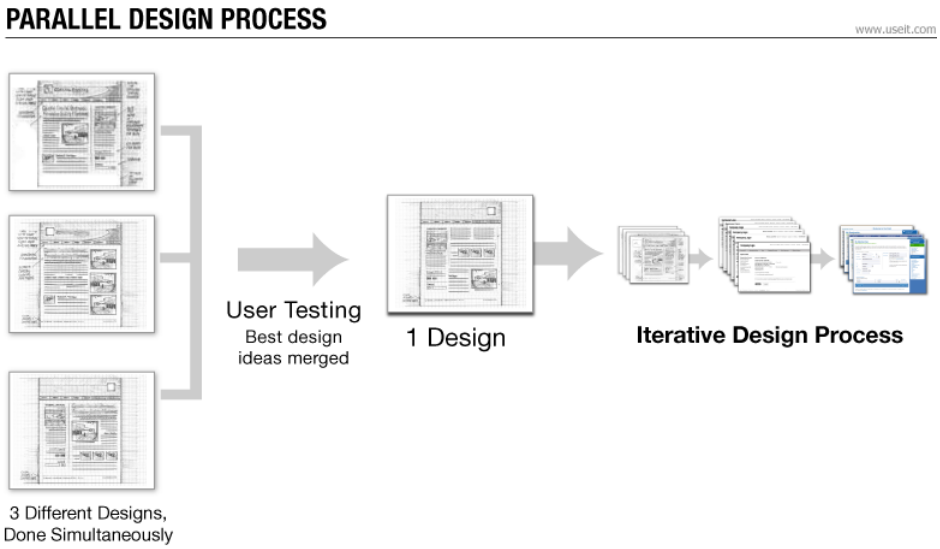
Uživatelské rozhraní, dále jen UI (User Interface), je podstatnou částí při návrhu jakékoliv aplikace. Nyní budou ukázány různé techniky, které se používají pro vytváření UI. Nedílnou součástí testování nejsou jen způsoby, jak testování provádět, ale také uživatelé, kteří s aplikací budou pracovat. Ti poskytnou nadhled, jakým směrem by se UI mělo vydat tak, aby obsahovalo vše potřebné bez zbytečností a bylo přehledné.

Jednou skupinou testovacích osob budou lékaři, protože právě u nich se tento výsledný produkt bude používat. Důležité je také rozsah skupiny testovacích uživatelů, která ve studii Jakoba Nielsena [10] poukazuje na to, že optimální počet uživatelů pro testování by neměl být víc než 5 [9]. Další skupinou testovacích subjektů budou uživatelé, kteří by využívali online možnosti virtuální čekárny. Musíme brát v potaz věkový rozsah, jelikož starší uživatelé zpravidla nevyužívají tyto možnosti. Bude pro ně připraven také nemobilní klient umístěný v čekárně přímo v ordinaci.

Musí být ovšem zmíněno, také to jaký způsob testovaní se bude v průběhu vývoje aplikovat. Pokud nebude zvolen správný postup, celý vývojový proces aplikace by se mohl při pozdějších úpravách zadrhnout, nebo dokonce zkrachovat. Aby se vyvarovalo tomuto problému, je vhodné pro aplikaci začít s technikou paralelní návrhu designu a poté se zaměřit na techniku iterativního návrhu.

### 2.4.1 Paralelní návrh designu

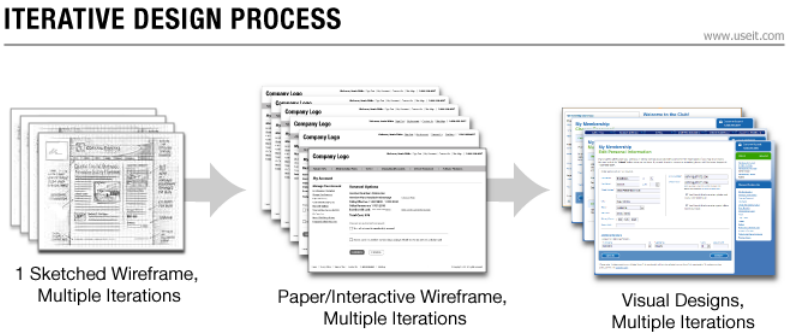
Pro zajištění nejvhodnějšího vývoje aplikace závislé na testování je vhodné začít tak, že se vytvoří několik návrhů aplikace ve stejným moment [8]. Nejvhodnější by bylo mít vytvořené nejméně 3 odlišné verze designu a tyto alternativy předložit testovacím subjektům aplikace, od nich se vrátí zpětná vazba, která bude využita jako kreativní impuls pro další návrhy. Jakmile bude provedeno testování, zhotoví se jeden kombinovaný návrh vytvořený z informací získaných během testování, které obsahují nejlepší poznatky z různých verzí. Jakmile se vytvořen ucelený návrh, přesune se návrh k iterativnímu testování.




Obrázek 2.2: Paralelní návrh (převzato z [8])

### 2.4.2 Iterativní návrh designu

Iterativní testování využije výsledný navržený design paralelního návrhu. Nad tímto návrhem budou prováděny iterace, jednoduše řečeno to znamená, že bude následovat krok za krokem úprava z jednoho návrhu k dalšímu. O tom kolik bude potřeba iterací celkem se přesně neví, přičemž by měly být minimálně dvě až tři, vhodné je ovšem používat rozmezí 5-10 iterací, kde každá iterace by měla zahrnovat alespoň jeden až dva nové návrhy designu [8].




Obrázek 2.3: Iterativní návrh (převzato z [8])

# Kapitola 3

# Návrhy uživatelského rozhraní

Základním pravidlem, o který se bude tento projekt opírat bude "méně je více" nebo-li také "dostat se co nejdál za nejkratší dobu", tím se bude rozumět to, že čím méně kliknutí provedeme, tím více času ušetříme hledáním a prováděním toho, co chceme uskutečnit. Budou představeny techniky, které projektu pomůžou při tvorbě uživatelského rozhraní způsobem, který bude vyhovovat většině uživatelů a navíc bylo přehledné.

Celá aplikace bude mít tři základní odvětví. V první části se zaměří na rozhraní pro lékaře, které bude předně obsahovat vyvolávání pacientů. Pacientské UI bude mít oproti tomu lékařskému za úkol přihlašování pacienta do front a UI pro nemobilního klienta bude vytvořené pro pacienty, kteří nepoužili aplikaci virtuální čekárny, ale zařadili se do fronty přímo ve fyzické čekárně u lékaře.

Všichni uživatelé, kteří se rozhodnou aplikaci využívat, budou vlastnit svůj samostatný profil, který bude uchovávat jejich osobní údaje a informace ohledně návštěv. Lékařský profil bude nabízet stejné možnosti a navíc bude obsahovat přehled a informace o čekárně a o tom, kolik pacientů se v ní nachází v daný moment.

## 3.1 GUI pro lékaře

Lékařské rozhraní, které bude vyvolávat pacienty z čekací fronty, by nemělo postrádat přehled o pacientech, kteří se v čekárně vyskytují. Pro přehlednost se fronta s pacienty bude znázorňovat ve formě tabulky, kde ve sloupcích se budou udávat jména pacientů, jejich pořadová čísla a stav zdali jsou již v čekárně. Další funkcí, kterou bude virtuální čekárna v tomto rozhraní disponovat, se bude týkat úpravy času ordinačních hodin. Tato možnost by neměla překážet v hlavním navrženém vyvolávacím okně. Možnost úpravy

ordinačních hodin bude mít své vlastní prostředí, k němuž bude jednoduchý přístup pomocí odkazovacího tlačítka.

## 3.2 GUI pro mobilního klienta

Pro mobilního klienta je vytvoření samostatného rozhraní nepostradatelnou součástí. V tom rozhraní bude pro klienta v prvé řadě připraveno tlačítko k přihlášení se do fronty. Jednou z možných informací, které nám nabízí fyzická čekárna u lékaře, je hlavně přehled o tom, kolik pacientů se v ní nachází, a proto i tato informace je podstatná. Tudíž uživatel, který používá rozhraní mobilního klienta, bude znát průběžný počet lidí v čekárně. Další informací, která se nabídne uživateli mobilního klienta, bude pravděpodobný čas, který v čekárně stráví, než se dostane na řadu do ordinace. Ta se zjistí průběžným počtem pacientů před námi a průměrným časem, který pacient stráví na vyšetření.

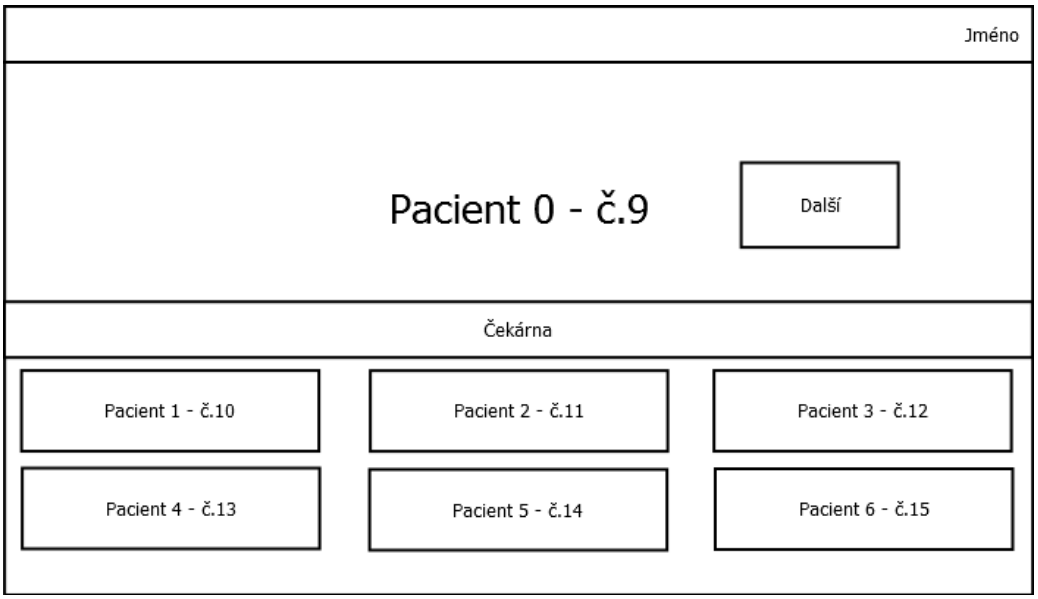
## 3.3 GUI pro fyzickou čekárnu

Grafické uživatelské rozhraní, bez kterého se aplikace neobejde, je i takové, které bude využíváno v čekárně lékaře. To bude muset být především přehledné a navíc jednoduché na pochopení. Mělo by obsahovat hlavně možnost přihlášení se do fronty čekajících, stejně tak jako potvrzení pacienta, který se do fronty přihlásil online. Bez potvrzení tohoto místa přímo v čekárně by nebylo možné systémově zachytit, zda se pacient vůbec dostavil. V případě nepotvrzení bude pacient vyřazen z fronty a na řadu se dostane následující pacient přítomný v čekárně.

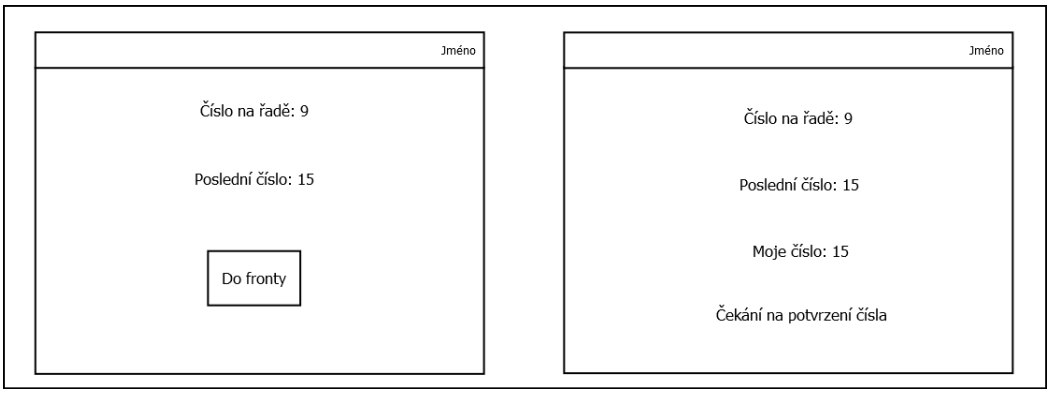
## 3.4 Návrhy uživatelských rozhraní

Dříve než se začne aplikace implementovat, je důležité si vytvořit několik různých návrhů, jak by mohla vypadat její grafická stránka. Tyto jednotlivé návrhy budou předloženy testovacím subjektům, kteří by mohli aplikaci v budoucnu využívat. Vzhledem k tomu, že věkový rozsah není omezen, může být aplikace testována na lidech všech věkových kategorií. Jedny z prvních návrhů byly připravené na papíře, aby se vytvořil ucelený přehled, jaké parametry by mělo uživatelské rozhraní mít. Jednotlivé prvotní návrhy pro všechna rozhraní je možné vidět na Obrázcích 3.1, 3.2 a 3.3. Dalším krokem po otestování návrhů je převést tyto návrhy do elektronické podoby. Každý z návrhů bude jednotlivě upraven a ztvárněn jako webová stránka, která se

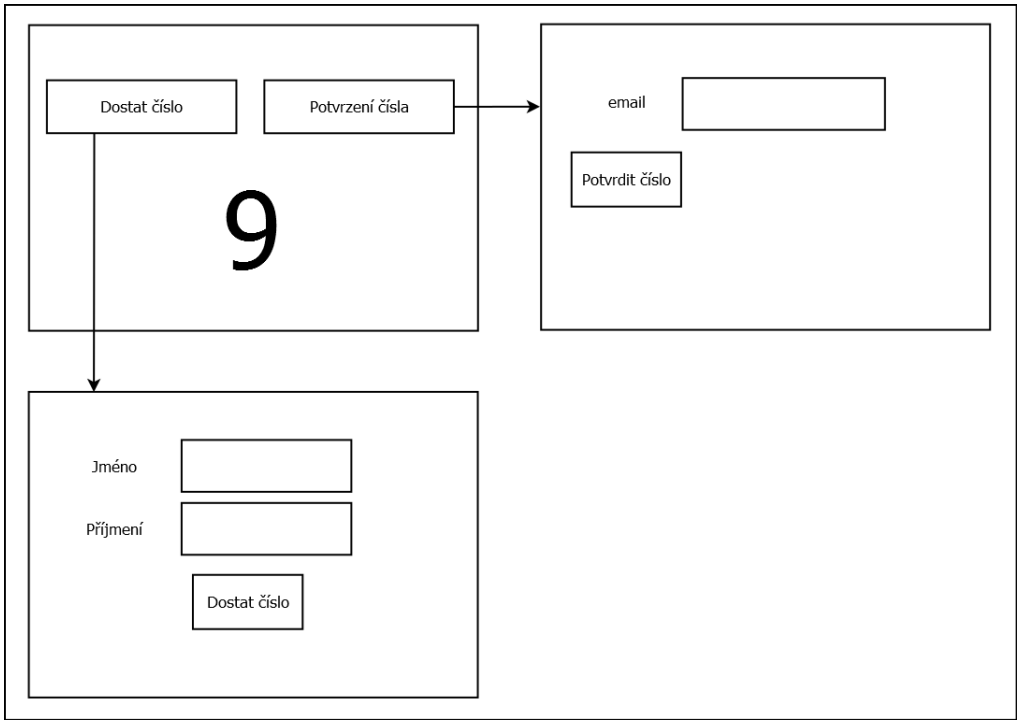
automaticky přizpůsobí různým velikostem zobrazovacích zařízení od mobilu přes iPad až po monitor PC.




Obrázek 3.1: Návrh lékařského rozhraní




Obrázek 3.2: Návrh rozhraní mobilního klienta




Obrázek 3.3: Návrh rozhraní v čekárny

### 3.4.1 Testování návrhů

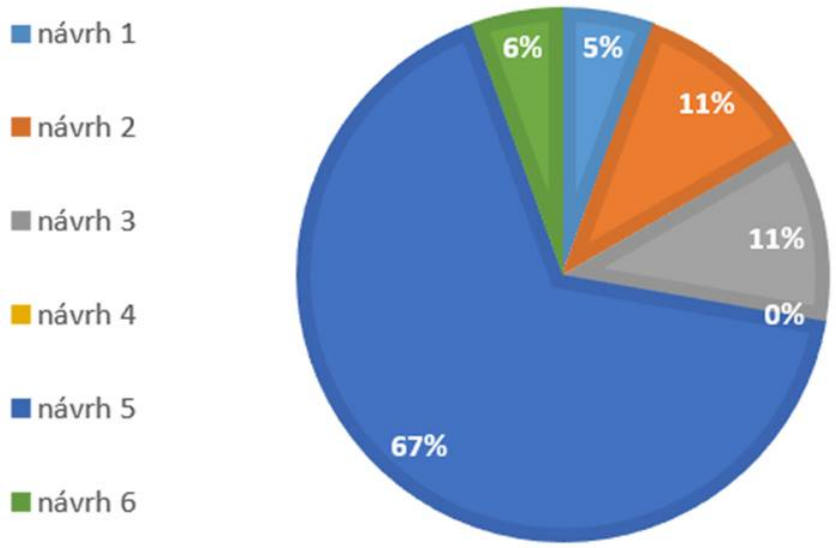
Jestli se má aplikace dále vyvíjet, je podstatné ji neustále testovat s uživateli. Z důvodu vytíženosti lékařské skupiny se první testovací návrhy aplikace zaměřily na uživatele, kteří testovali návrhy rozhraní pro mobilního klienta.Testování probíhalo na skupině uživatelů ve věkové kategorii 18-55 let, kde každému z nich byly představeny 3 rozdílné paralelní návrhy, ze kterých byly vybrány 2 nejlepší, jež budou postupně podrobené iterativnímu testování.

Vzhledem k velkému věkovému rozsahu bylo testovacích osob více, aby bylo obsáhnuto více věkových kategorií. Později se počet zredukoval na 5 osob podle studie Jakoba Neilsena, která je zmíněna v kapitole 2.4

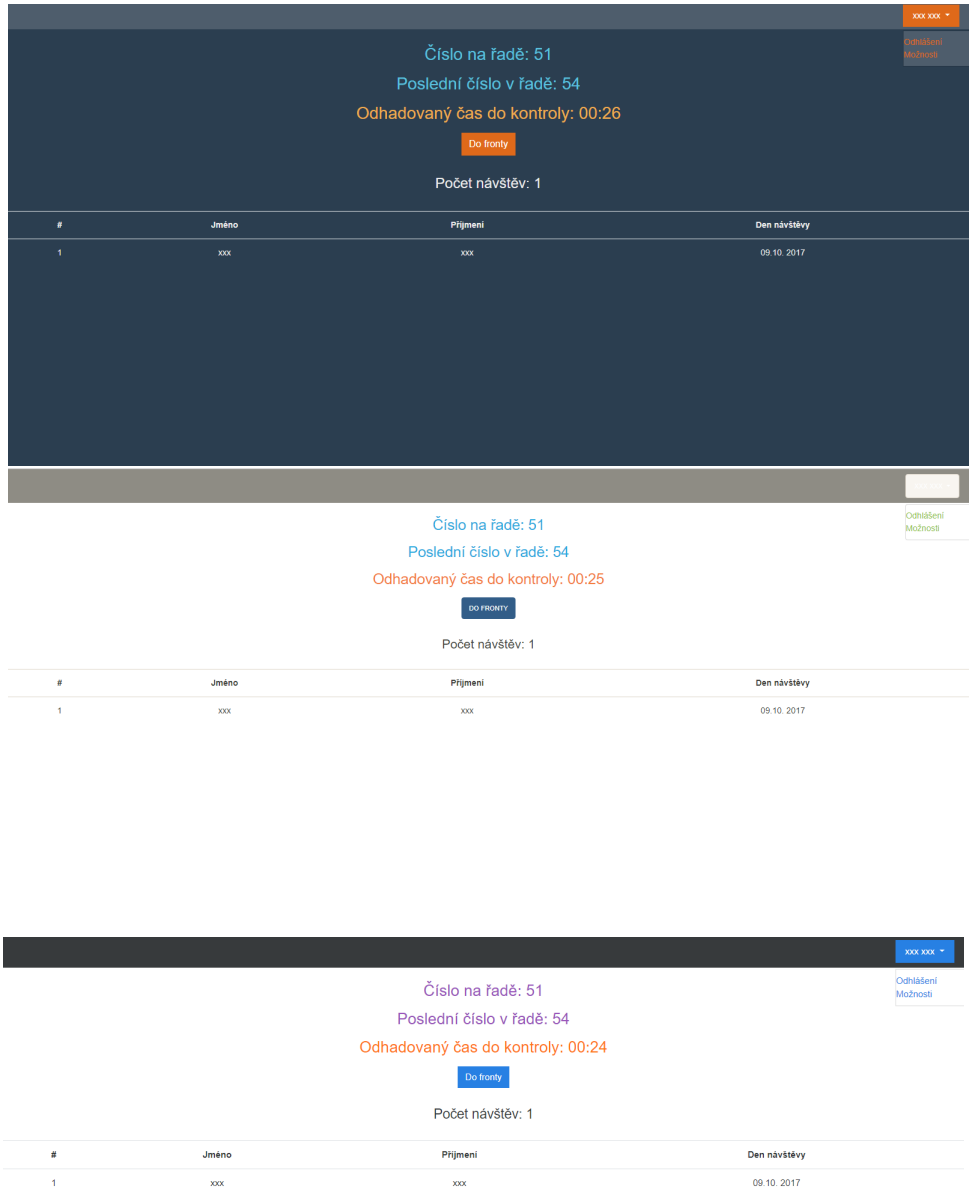
Při pozdějším iterativním testování se cíl návrhu zaměřil prvotně na přehlednost a spokojenost uživatelů, kteří poté zpětnou vazbou ve formě osobních konzultací či dotazníku s jednoduchými otázkami předali poznatky o mezerách a kladech, které byly zohledňovány v dalších iterativních modifikacích a testech.

### 3.4.2 Výsledky testování návrhů

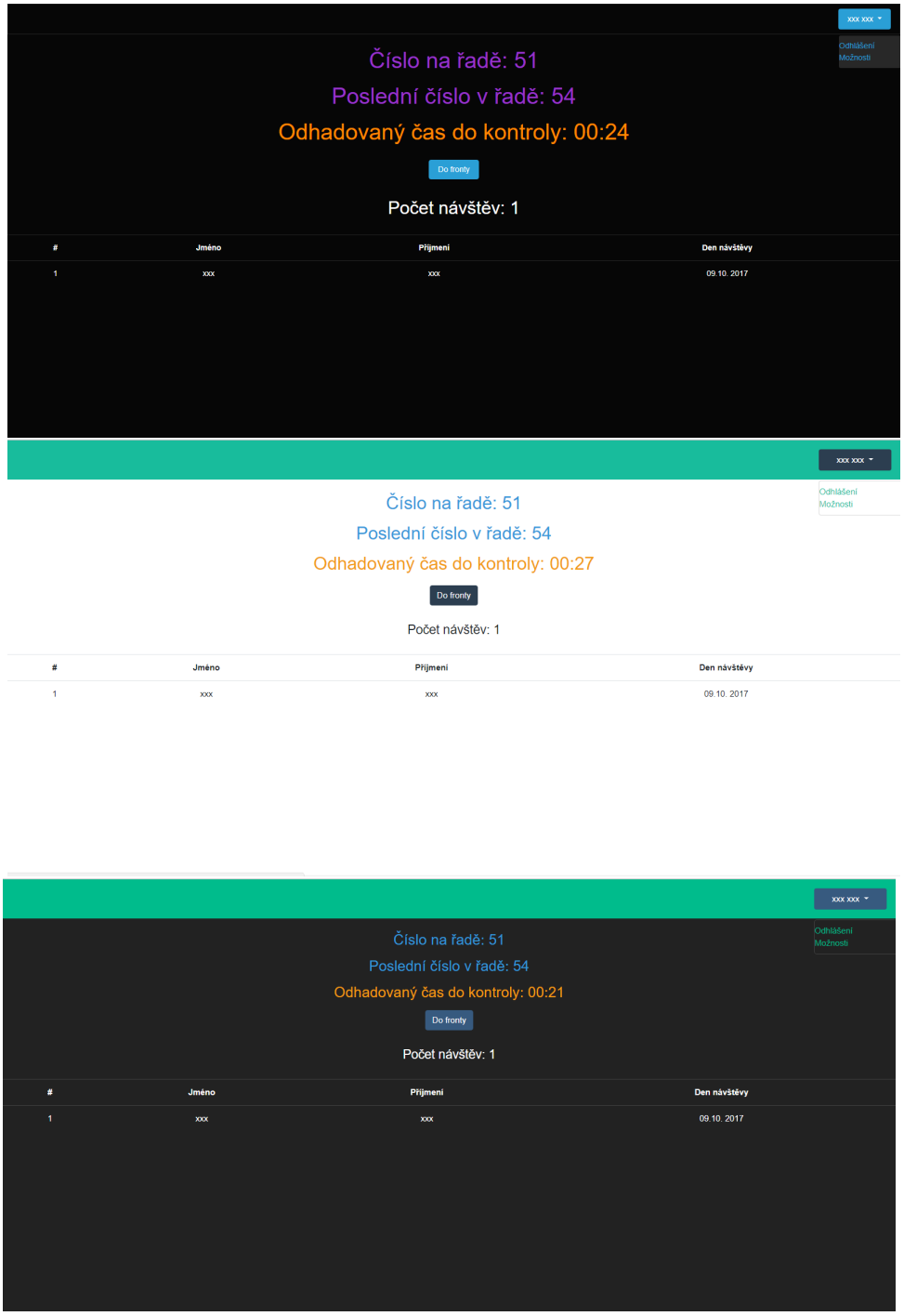
Nejdříve bylo podrobeno paralelnímu testování několik různých návrhů, z nichž dva nejpřívětivější posloužily jako odrazový můstek k dalšímu iterativnímu testování. Cílem tohoto testování bylo zjistit ideální rozložení informací na stránce, stejně tak jako přehlednost a grafickou podobu aplikace. Dalším krokem po vytvoření elektronických návrhů bylo navrženo a otestováno 6 různých barevných variant znázorněny na obrázcích 3.5,3.6. K testování grafických návrhů bylo přijato 18 lidí ve věkovém rozložení 18-62 let. Všichni byli seznámení s funkcionalitou aplikace a měli za úkol rozhodnout o přehlednosti a vybrat jeden návrh rozhraní, který podle sebe uznali nejvhodnějším. Výsledky jsou znázorněné na grafu 3.4, kde drtivá většina zvolila variantu se světlým podkladem a zeleným okrajem.




Obrázek 3.4: Výsledky testování výběru rozhraní




Obrázek 3.5: Návrh uživatelského rozhraní pro mobilního klienta typ 1,2,4




Obrázek 3.6: Návrhy uživatelského rozhraní pro mobilního klienta typy 2,5,6

# Kapitola 4

# Návrh a implementace programové části

V této kapitole bude představeno, co by měla celá aplikace obsahovat. Jednoduchost a preciznost jsou aspekty, na které se implementace zaměří, aby uživatelé neztráceli čas učením a přizpůsobováním nového systému. Vše by mělo být hned na první pohled jasné a snadno dostupné.

## 4.1 Implementace databáze

Prvním krokem k vytvoření aplikace je v prvé řadě databáze, ve které se budou uchovávat data a informace o přehledu čekárny a pacientech. Pro vytvoření samostatné databáze bylo použito grafické rozhraní phpMyAdmin, pomocí kterého byly vytvářeny základní relace naší aplikace. V tab 4.1 jsou představeny základní tabulky společně s jejich významem.

Tabulka 4.1: Relace a jejich funkce

| **Relace** | **Popis** |
|---|---|
| users | jméno mobilního pacienta, jeho číslo v čekárně |
| visit | počet návštěv, datum návštěvy |
| drstats | ordinační hodiny, průměrný čas na jednoho pacienta |
| guests | jméno a číslo pacienta v čekárně, |

## 4.2 Jazyk programu

Společně s MySQL popsanou v kapitole 2.3 tvoří jazyk PHP nerozlučnou dvojici při tvorbě a správy dynamických webů. Programovací jazyk PHP patří mezi skriptovací programovací jazyky a ačkoliv je jednoduchý, poskytuje funkce pro správu složitější projektů. PHP byl původně jednoduchý systém pro webové stránky, který se později vyvinul v komplexní programový jazyk pro dynamické aplikace [3].

PHP je také ceňeno proto, neboť se v něm dají projekty rychle vytvořit. Tento z funkcí složený jazyk, jejíž příkazy algoritmus vykonává krok za krokem, je možné vkládat přímo do zdrojového kódu HTML [4].

## 4.3 Framework

Nejen na základě popularity, ale také na předešlých zkušenostech byl pro tvorbu virtuální čekárny zvolen framework Laravel 5, který využívá pro chod aplikace architekturu MVC. Dále si představíme základní modely, které nás propojují s databázovými tabulkami, ovladače jejímiž úkoly jsou funkcionalita aplikace a pohledy, které znázorňují výslednou podobu internetové stránky.

### 4.3.1 Models

Model je ve výpočetní technice aktivní reprezentací abstrakce ve formě dat. Je to vrstva, která tvoří základ celé aplikace obsahující aplikační logiku a data. Modely propojují aplikační vrstvu s databázovými tabulkami. Dovolují, aby aplikace jejich prostřednictvím získávala informace z tabulek stejně tak, jako do nich vkládata nové data [6].

Nejdříve budou rozebrány třídy pro mobilního klienta a lékaře. Tyto dvě třídy spolu úzce spolupracují a vytváří kostru pro chod celé aplikace. Obecně řečeno se bude třída pro pacienta starat o to, aby vytvářela a ukládala informace, které pak bude třída pro lékaře potřebovat pro vyhodnocení a chod čekárny.

#### Patient class

Samostatná třída Patient, která rozšiřuje základní vrstvu Model, poskytuje vitální funkce, které jsou pro běh aplikace ze strany pacienta nepostradatelné. Základní informaci, kterou uživatel mobilního klienta potřebuje je znát své pořadové číslo ve frontě. Číslo, které je další v pořadí získá z funkce *getNumber()*. Stejně tak je podstatné vědět, kolik pacientů ve frontě ještě čeká před

námi. Tuto informaci předávají funkce, jejichž úkoly je znázorňovat číslo pacienta, který je na řadě společně s informací, která udává poslední a zároveň nejvyšší čekající číslo pacienta v čekárně. Také byla navržena a napsána funkce, která určí odhadovaný čas čekání viz kód 4.2, což je podstatná informace pro odhad, kdy pacient zatím není fyzicky přítomen v čekárně. Tato funkce vypočítává odhadovaný čas na základě počtu pacientů v čekárně a průměrné doby, kterou lékař stráví nad vyšetřením jednoho pacienta.

```
1 public static function getNumber(){
2     $PNU = DB::table('users')->max('pickedNum');
3     $PNG = DB::table('guests')->max('pickedNum');
4     if($PNU < $PNG) return $PNG+1;
5     else return $PGU+1;
6 }
```

Listing 4.1: Získání čísla pro pacienta

```
1  public static function time_to_wait($id){
2      $CN = DB::table('users')->max('currentNum');
3      $Time_For_Patient_m = DB::table('drstats')
4                                ->max('timeForPatient');
5      $Time_For_Patient = $Time_For_Patient_m*60;
6      $LPN = Patient::PickedNum($id);
7      $BPN = Patient::BiggestPickedNum();
8      $time = gmdate('H:i:s',$Time_For_Patient*($LPN-$CN));
9      $upd = DB::table('users')->max('updated_at');
10     $secs = strtotime($time)-strtotime("00:00:00");
11     $end = date("Y-m-d H:i:s",strtotime($upd)+$secs);
12     $start = Carbon::now('Europe/Prague')
13                       ->format('Y-m-d H:i:s');
14     $ts1 = strtotime($start);
15     $ts2 = strtotime($end);
16     $seconds_diff = $ts2 - $ts1;
17     $x = date("Y-m-d H:i:s",$seconds_diff);
18     $Hours = Carbon::createFromFormat("Y-m-d H:i:s",$x)
19                           ->format('H');
20     $Minutes = Carbon::createFromFormat("Y-m-d H:i:s",$x)
21                           ->format('i');
22     if($seconds_diff < 0) return "Čas vypršel.";
23     else  return $Hours.":".$Minutes;
24 }
```

Listing 4.2: Získání času čekání

### Dr class

Důležitou částí, která úzce spolupracuje s třídou, jež má na starost mobilního klienta, je třída Dr, která už podle názvu napovídá, že slouží výlučně pro lékařské účely. Základním kamenem a funkcí, bez které se lékařské prostředí neobejde, je zjištění, který pacient z čekárny je právě na řadě. K tomu musí

znát jméno a jeho pořadové číslo. Číslo pacienta získá zavoláním funkce *nextPatient()*, jež vybere pacienta, který je v pořadí a hlavně je přítomen ve frontě.

```
1 public static function NextPatient(){
2    $CN = Dr::getCurrentNumber();
2        $NumG = DB::table('guests')
3              ->where([['pickedNum','>',$CN],['confirmation','=','1']])
4              ->min('pickedNum');
5    $Num = DB::table('users')
6              ->where([['pickedNum','>',$CN],['confirmation','=','1']])
7              ->min('pickedNum');
8   if($Num == "")
9       return $NumG;
10  else if($NumG == "")
11      return $Num;
12  else if($NumG < $Num)
13      return $NumG;
14  else return $Num;
15 }
```

Listing 4.3: Získání čísla dalšího pacienta

### Visit class

Třída Visit poskytuje funkce, které sbírají a ukládají informace, kdy byl pacient naposled u lékaře a vytváří tak historii návštěv. Ta je pacientovi zobrazena jako tabulka s vyšetřeními.

### Guest class

Pro tuto třídu je podstatné jen čištění databáze od pacientů, kteří nezískali číslo přes aplikaci virtuální čekárny, ale přímo v čekárně. Čištění probíhá tak, že po sedmi dnech bude databázová tabulka prázdná a nebude zaplňovat paměť. Ostatní důležité vlastnosti pro funkcionalitu jsou obsaženy v řadiči pro tuto třídu GuestController.

## 4.3.2 Conrollers

Controllers neboli také řadiče, jsou ve frameworku s MVC architekturou nepostradatelné. Jejich úkolem je propojení mezi pohledem a modelem. Z pohledu zpracovává požadavky uživatele a ty předá modelu k vykonání, následně opět požádá pohled o vykreslení dat [5].

Nyní budou představeny řadiče, které spravují uživatelské prostředí aplikace. V úvodu bude zmíněno, jak jsou tyto řadiče propojeny funkcionálně

mezi sebou a poté bude rozebrán každý řadič zvlášť. Controller, který zasahuje a ovlivňuje ostatní je GuestController, který spravuje prostředí pro fyzickou čekárnu. Podílí se na potvrzení, zda uživatel mobilního klienta je již v čekárně a také, jestli s ním má lékař počítat nebo jej přeskočit. Řadiče lékaře a mobilního pacienta jsou mezi sebou propojeny pouze informacemi, které lékaře upozorní, že pacient je v čekárně a informováním pacienta od lékaře, že se dostal na řadu.

### PatientController

PatientController je třída, která rozšiřuje řadič Controller a spravuje uživatelské prostředí mobilního klienta. Využívá předdefinovaných a vytvořených funkcí v modelech Dr a Patient. Při spustění se inicializuje funkce obecně známá jako *index()*, ta obsahuje funkce, které poskytnou uživateli informace vzájemně propojené s čekárnou, kterými jsou například pořadové číslo a kolik zbývá odhadem času k přístupu na řadu. Důležitou funkcí, která je zde více než potřebná, je funkce *getNum()*. Ta po zavolání přidělí mobilnímu klientovi číslo do fronty ve virtuální čekárně.

```
1 public function index(){
2     $update = Patient::update_after_time();
3     $lastPickedNum = Patient::LastPickedNum(\Auth::user()->id);
4     $biggestPickedNum = Dr::getMaxPickedNum();
5     $patient = User::all();
6     $numOfVisit = Visit::numberOfVisits(\Auth::user()->email);
7     $visits = Visit::individualVisits(\Auth::user()->email);
8     $time = Carbon::now('Europe/Prague')->format('Y-m-d H:i');
9    $timeForButton = Carbon::now('Europe/Prague')->format('H:i');
10    $updTMP = DB::table('users')->max('updated_at');
11    $upd = Carbon::createFromFormat('Y-m-d H:i:s',$updTMP)
12                  ->format('Y-m-d H:i');
13    $diff_patient=Patient::time_diff_patient(\Auth::user()->id);
14    $diff_wr=Patient::time_diff_wr(\Auth::user()->id);
15    $SWBT1=DB::table('drstats')->max('TimeToStartButton');
16    $SWBT2=DB::table('drstats')->max('TimeToEndButton');
17    $startWaitingButtonTime = Carbon::createFromFormat
      ('H:i:s', $SWBT1)->format('H:i');
18    $stopWaitingButtonTime = Carbon::createFromFormat
      ('H:i:s', $SWBT2)->format('H:i');
19    return view('patient.index')
20            ->with('timeForButton',$timeForButton)
21            ->with('stopWaitingButtonTime',$stopWaitingButtonTime)
22            ->with('startWaitingButtonTime',$startWaitingButtonTime)
23            ->with('diff',$diff_patient)
24            ->with('diff_wr',$diff_wr)
25            ->with('visits',$visits)
26            ->with('time',$time)
27            ->with('upd',$upd)
28            ->with('numOfVisit',$numOfVisit)
```

```
29            ->with('patient',$patient)
30            ->with('biggestPickedNum',$biggestPickedNum)
31            ->with('lastPickedNum',$lastPickedNum);
32 }
```

Listing 4.4: Funkce pro načtení úvodní stránky mobilního klienta

```
1 public static function getNum(){
2     $Max=Patient::getNumber();
3     DB::table('users')
4          ->where('id',\Auth::user(->id)
5          ->update([
6                'confirmation'=>0,
7                'pickedNum'=>$Max,
8              ]);
9     $patient = User::all();
10    return view('patient.showL')
11            ->with('patient',$patient);
12 }
```

Listing 4.5: Získání čísla dalšího pacienta

### DrController

Pro lékaře je připravena třída DrContoller, která obsahuje funkce pro práci s lékařským prostředím. Stejně tak jako jiné třídy rozšiřující řadič controller i tato obsahuje základní funkci pro načtení úvodní stránky *index()*. V ní jsou obsaženy funkce, které sdělí, který pacient a s kterým číslem je právě na řadě, stejně tak kolik dalších pacientů ještě čeká v čekárně. Obdobně jako řadič pro mobilního klienta obsahuje funkci, který mu přidělí nové číslo, tak i pro lékaře je vytvořena funkce *nextPatient()*, která vyvolá dalšího pacienta v řadě z čekárny.

```
1 public function nextPatient(){
2     $newCurrNum = Dr::NextPatient();
3     DB::table('users')
4          ->update([
5              'currentNum'=>$newCurrNum,
6              'updated_at' => Carbon::now('Europe/Prague')
7     ]);
8     DB::table('guests')
9          ->update([
10              'currentNum'=>$newCurrNum,
11              'updated_at' => Carbon::now('Europe/Prague')
12    ]);
13    $curN = Dr::getCurrentNumber();
14    $PN = Dr::PatientToServeName();
15    $PS = Dr::PatientToServeSurname();
16    $MaxPickedNum = Dr::getMaxPickedNum();
17    $guests = DB::table('guests')
18                 ->select('name','surname','pickedNum',
```

```
    'local_online','confirmation');
19   $patients = DB::table('users')
20                  ->select('name','surname','pickedNum',
    'local_online','confirmation')
21                  ->union($guests)
22                  ->orderBy('PickedNum','asc')
23                  ->get();
24      return view('dr.nextpatient')
25              ->with('patients',$patients)
26              ->with('MaxPickedNum',$MaxPickedNum)
27              ->with('PN',$PN)
28              ->with('PS',$PS)
29              ->with('curNum',$curN);
30 }
```

Listing 4.6: Funkce pro zavolání dalšího pacienta do ordinace

### GuestController

Třídy pro lékaře i mobilního klienta byly již představeny, teď zbývá třída, která pracuje s nemobilním klientem přímo v čekárně. Stejně tak jako v předchozích řadičích, tak i zde máme opět funkci `index()`, která je úvodní stránkou pro nemobilního klient, respektive stránkou pro terminál umístěný v čekárně. Aby bylo možné zařadit pacienta do fronty, který právě přišel a začíná čekat přímo ve fyzické čekárně, musí být pro něj připravené funkce, které mu vytvoří dočasný profil a uloží jej do databáze, odkud si jej načte lékařský řadič a přidá ho do fronty. Přesně k tomu slouží funkce *create()* a *store()*.

Posledními funkcemi, které řadič čekárny poskytuje, jsou zjištění svého čísla v případě, že není k dispozici tiskárna lístků pro pacienta v čekárně a také funkce, která potvrdí příchod uživatele mobilního klienta do čekárny. K tomu slouží funkce *confirmOnlineWaiting()* a *confirmed()*, jež nalezne podle přihlašovacího emailu klienta, který čeká online a potvrdí mu příchod do čekárny, bez kterého by nebyl přijat k lékaři.

```
1 public function confirmOnlineWaiting(){
2    $alert = false;
3    return view('guest.confirmation')
4             ->with('alert',$alert);
5 }
6 public function confirmed(Request $request){
7   $exist=DB::table('users')
8                ->where('email','=',$request->email)
9                ->get();
10   $curN = Dr::getCurrentNumber();
11   $ifinQueue = DB::table('users')
12                 ->select('name')
13                 ->where([
```

```
14                      ['pickedNum','>',$curN],
15                      ['email','=',$request->email]])
16                  ->get();
17    $alert = false;
18    if($exist == '[]' || $ifinQueue == '[]'){
19        $alert = true;
20        return view('guest.confirmation')
21                  ->with('alert',$alert);
22    }
23    DB::table('users')
24         ->where('email','=',$request->email)
25         ->update([
26               'confirmation'=>true,
27    ]);
38    $visit = new Visit;
29    $visit->email = $request->email;
30    $visit->->save();
31    return view('guest.confirmed');
32 }
```

Listing 4.7: Funkce pro potvrzení čísla v čekárně

#### RegisterController

Základním kamenem pro webovou aplikaci pracující s uživatelským přístupem je potřeba mít třídu s funkcemi pro registraci uživatelů. Takovou třídou je RegisterController, který vytváří uživatele, jež chce virtuální čekárnu používat online. Pomocí funkce *validator()* je nový uživatel ověřen, zda vyplnil registrační údaje správně a jestli není doposud zaregistrovaný jiný uživatel pod stejnou emailovou adresou, která musí být unikátní.

### 4.3.3 Views

View neboli pohled je v MVC architektuře vrstva, která se stará o zobrazení výsledku požadavku. Ke každému Modelu je připojen jeden nebo více pohledů, z nichž je každý pohled schopen zobrazení jednoho nebo více obrazových reprezentacích modelu. Pohled je také schopen provádět operace nad Modelem, který je logicky spojený s tímto Pohledem [6].

#### Blades

Blade je jednoduchý, avšak výkonný šablonový engine (nástroj) poskytovaný Laravelem. Na rozídíl od jiných PHP šablonových nástrojů, Blade nás neomezuje v používání čistého PHP kódu v pohledech. Soubory zobrazení blade používají příponu souboru blade.php a jsou obvykle uloženy v adresáři resources\views [15].

### Auth views

Pohledy ve skupině views\auth patří mezi první, se kterými se při spuštění této aplikace uživatelé setkají a budou potřebovat, aby mohli především aplikaci používat. Obsahují pohledy, díky kterému si vytvoříme nový účet odkazem na registrační formulář, který obsahuje povinné údaje jako emailová adresa, jméno, heslo apod. Samozřejmě je připraven i pohled *Login*, přes který se uživatel přihlásí ke svému již vytvořenému účtu.

### Patient views

Každý uživatel mobilního klienta musí mít na své stránce znázornění, v jakém stavu se čekárna nachází. Především je nutné znát kolik osob se v čekárně vyskytuje, respektive které číslo je současně na řadě a přiřazené číslo posledního pacienta. Další informaci, která uživateli pomůže k orientaci stavu čekárny je čas čekání, který je znám ještě před přihlášením do fronty a zobrazován či aktualizován během doby čekání. Poslední informací, jež je známa je buď upozornění na stav, kdy je potřeba potvrdit místo v čekárně, anebo informace, kdy je čekání ve frontě potvrzené. Podoba pohledu, která se zobrazí při rozhraní o velikosti mobilního telefonu je zobrazena na obrázku 4.2.

### Dr views

Lékař potřebuje při vyvolání pacienta mít přehled o tom, kteří pacienti se nacházejí ve čekárně. Tudíž bylo pro přehled stavu čekárny použito abstrakce tabulky, ve které jsou znázorněny záznamy o pacientech, kteří se nacházejí ve frontě. Navíc jsou tam zobrazení ti, kteří čekají online pomocí naší aplikace a nepotvrdili si ještě přítomnost v čekárně. Z této tabulky jsou vytyčeny informace o tom, který pacient a se kterým číslem je právě na řadě. K tomu, aby mohl lékař vyvolávat pacienty, je vytvořeno tlačítko, které vyjme pacienta z fronty a zobrazí jeho číslo. Dále má možnost lékař upravovat ordinační hodiny, které omezí pacientovi se přihlásit do fronty a průměrný čas, který stráví vyšetřováním jednoho pacienta. Celkový náhled je zobrazen na obrázku 4.1.

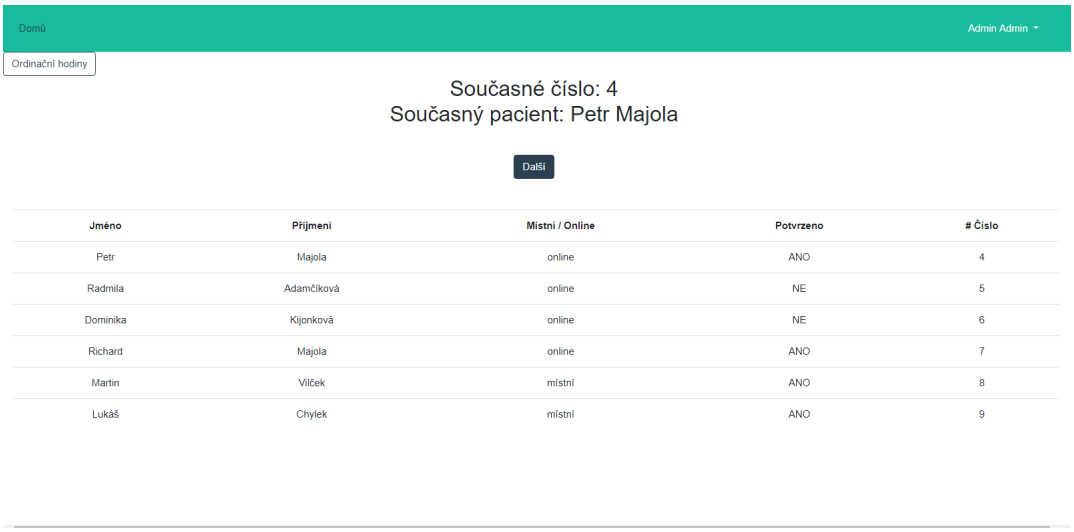
### Guest views

Pohled úvodní stránky, která bude ve fyzické čekárně se bude rozdělovat na dvě základní části. V prvé řadě je to zobrazení vyvolávaného číslo pacienta. Tou druhou jsou tři odkazovací tlačítka pro manipulaci s prostředím virtuální čekárny. První odkaz je pro pacienty, kteří nepoužívají online aplikaci a

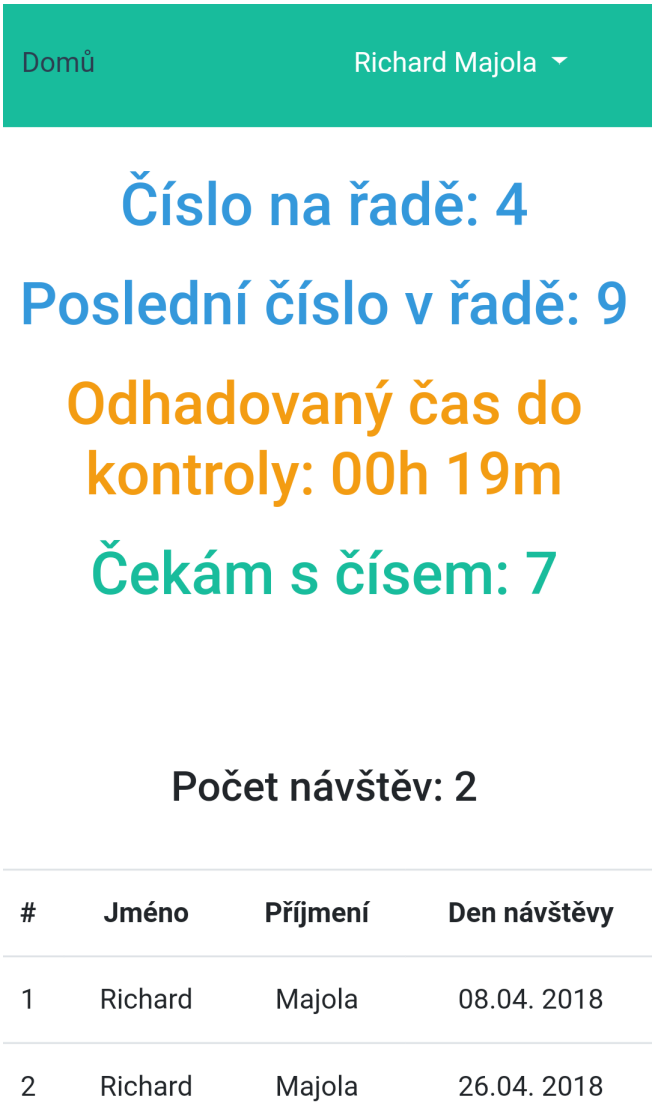
potřebují získat pořadové číslo do čekárny. Odkazovaná stránka představí pacientovi prostor pro zadání svého jména a potvrzení pro získání čísla. Druhý odkaz je připraven pro pacienty, kteří využili možnosti online čekání, jedná se tak o potvrzovací formulář. V tomto formuláři uživatel mobilního klienta zadá svou emailovou adresu jako identifikaci pro potvrzení svého čekajícího čísla a potvrdí tak příchod do čekárny. Třetím a posledním odkazem, který je připraven pro případ neexistující tiskárny lístků, je pro pacienty, kteří čekají v čekárně a zapomněli svoje číslo. Dává jim možnost napsat své jméno a číslo, které se kterým čekají, je jim znovu ukázáno.

## 4.4 Shrnutí implementace

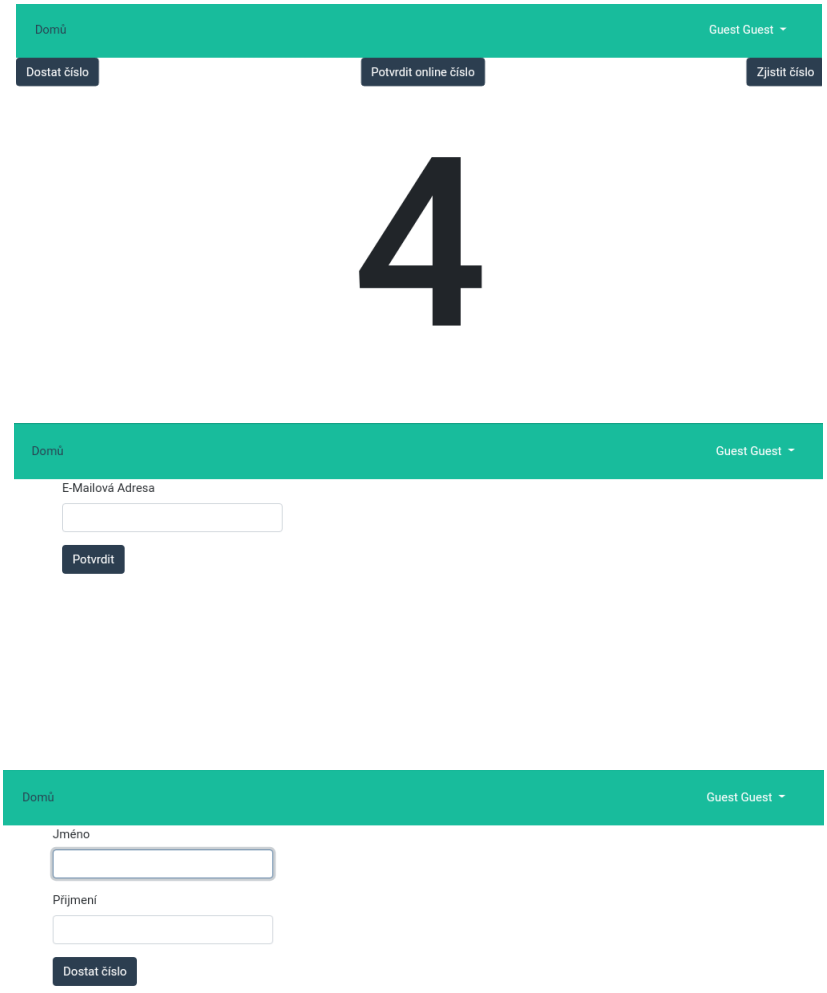
Celá aplikace je implementována tak, že cílem bylo předně aplikovat jednoduchost, která by uživateli na první pohled hned prozradila, jak aplikace pracuje. Lékařské prostředí obsahuje přehled o stavu čekárny a lékař s použitím tlačítka "Další" vyvolává dalšího pacienta z fronty do ordinace. Rozhraní pro mobilního klienta se ovládá stroze získáním čísla do fronty tlačítkem "Do fronty". Ovládání rozhraní v čekárně se dělí pro uživatele, který použil online aplikaci za použití odkazovacího tlačítka "Potvrdit online číslo" a pro pacienty, kteří číslo potřebují právě získat tlačítkem "Dostat číslo". Aplikace je dostupná na stránce www.virtualni-cekarna.cz.




Obrázek 4.1: Výsledná podoba grafického rozhraní lékaře




Obrázek 4.2: Výsledná podoba grafického rozhraní mobilního klienta




Obrázek 4.3: Výsledné podoby grafického rozhraní fyzické čekárny

# Kapitola 5

# Testování

Po vytvoření návrhů rozhraní a implementace větší části funkcionality nastal čas, aby byla aplikace virtuální čekárny otestována. Testování probíhalo na dvě etapy Alfa testování a Beta testování.

## 5.1 Alfa verze

Nejdříve by bylo vhodné si říct, co to vlastně alfa verze znamená, alfa verze je v podstatě první relativně funkčního nového IT produktu, která je testována zpravidla samotným autorem, či skupinou vývojářů, která se na vývoji podílela.

### 5.1.1 Alfa testování

Cílem tohoto testování je vyhodnotit kvalitu produktu a zajistit jeho připravenost se zaměřením na hledání nejvažnějších chyb, a jestli produkt funguje. Produkt je testován ve fázi, kdy je dostatečně stabilní, ale ještě není kompletně dokončen.

Prvním krokem během testování nového produktu bylo zjistit, jestli každé prostředí je funkcionálně správě naprogramované a nenastávají nežádoucí chybové stavy, do kterých by se aplikační program mohl dostat. Hned v úvodu bylo třeba otestovat registraci a zajistit zda není možné zadávat nesmyslné hodnoty. Testování lékařského prostředí bylo zaměřeno, jestli jsou pacienti správně vyvolávání z čekárny a zadávání konzultačních hodin. V prostředí mobilního klienta bylo za úkol získání čísla pro čekání ve frontě a následně si jej potvrdit v prostředí klienta pro fyzickou čekárnu. V ní bylo potřeba také otestovat, zda získávání pořadových čísel pro pacienty bez aplikace je synchronizováno s virtuální čekárnou.

### 5.1.2 Výsledky alfa testování

Během testování bylo zaznamenáno plno přechodů do chybových stavů, které nastávaly obzvlášť při zadávání nesmyslných hodnot u registračního formuláře, úpravě času konzultačních hodin a potvrzování online čísla v čekárně. Tyto nedostatky byly řádně ošetřeny, stejně tak byly ošetřené výskyty, při kterých se zobrazovalo více národnostních jazyků.

## 5.2 Beta verze

Beta verzí produktu je odladěná alfa verze od většiny chyb, která je téměř hotová a dostupná pro testování cílovými uživateli. Jedná se o testovací verzi výsledného produktu.

### 5.2.1 Beta testování

V beta testování je cíleno na skupinu zákazníků, pro které je naše aplikace vytvářena. Předně se testování zaměří na jejich spokojenost s výsledným produktem. Případně odladit grafické nesrovnalosti, které by jim byly nepříjemné.

K testování bylo přizváno pět testovacích subjektů, kdy všichni byli pod dohledem pro případné poznámky a návrhy. Jeden ze subjektů měl za úkol funkcionalitu a grafické přizpůsobení lékařského prostředí, zbylí čtyři získali na starost testování pacientské rozhraní, dva s použitím mobilního klienta a dva použití platformy pro fyzickou čekárnu. Rozdělení úkolů pro subjekt testující lékařské rozhraní bylo hledat chybové stavy, do kterých se může aplikace, ať už úmyslně či náhodně dostat a zaznamenat je pro jejich odladění. Pacienti mobilní aplikace měli pro sebe připravené úkoly:

- Registrace a následné znovu příhlášení,
- čekání ve frontě,
- potvrzení čísla na platformě čekárny.

Pacienti bez aplikace měli za úkol získat číslo v čekárně a otestovat, jestli se dá číslo vyhledat podle zadaných údajů (jméno a příjmení)

## 5.3 Výsledky testování

Každý uživatel podal zprávy o přehlednosti aplikace ke každé platformě, na které ji zkoušel (PC, mobil, tablet). Pro menší rozhraní se grafická úhlednost

nejevila příjemně, nebo se k některým atributům nebylo možné dostat. Co však bylo schváleno při použití PC platformy, byla přehlednost a grafická uspořádanost rozhraní.

Po finálním odladění chybových stavů, bylo dalším krokem zpřehlednit stránky tak, aby se automaticky přizpůsobovaly velikosti rozlišení při používaní jiných platforem jakou jsou mobily či tablety.

## 5.4 Možnosti rozšíření

Do budoucna by se dala aplikace rozšířit k použití pro více lékařů a dokonce zaměřit se i na jejich specializaci. Další aspekt, který by se dal v budoucnu vylepšit, je adaptivní čas čekání v závislosti na typu vyšetření. Možnost, kterou by lékaři uvítali, byl nápad s prioritním výběrem pacientů z fronty.

# Kapitola 6

# Závěr

Cílem praktické části této bakalářské práce bylo vytvořit aplikaci virtuální čekárny jako webovou aplikaci, která zaujímá funkci čekání ve frontě u lékaře. Výsledným produktem je aplikace, která umožní pacientům zjistit si přehled čekárny a čas, který v ní stráví. Podstatné je, že uživatel aplikace má možnost čekat již z domu nebo z práce a ne až ve chvíli, kdy přijde do čekárny, což mu ušetří drahocenný čas čekání.

Tato práce splnila všechny body svého zadání. V první části byly přestaveny technologie, které tato aplikace využívá. Další část se zaměřila na návrh aplikace, součástí toho bylo získat informace, jež by výsledný produkt měl obsahovat. V implementační části práce udává, jak byla aplikace vytvářená a z kterých základních částí se skládá. Popisuje také funkce, které celou aplikaci řídí. Při testování výsledného produktu došlo k odhalení mnoha chyb a grafických nedostatků, které byly postupně odstraněny.

Na začátku bylo důležité si projekt správně navrhnout. Dalším krokem bylo navržení a implementace správné databáze, která byla vytvořena v prostředí phpMyAdmin, které pracuje s jazykem SQL. Celý program byl dále vyvíjen v prostředí Laravel s použitím jazyka PHP. Velkou problematikou bylo navrhnout výsledný design tak, aby byl jednoduchý na okamžité pochopení a manipulaci s ním.

Během průběžného vývoje bylo zjištěno, že možnost používání aplikace je opravdu výhodná a pacienti by určitě jistou modernizaci uvítali.

# Literatura


[1] *Naramore E.* PHP5, MySQL, Apache: Vytváříme webové aplikace. Vyd. 1. Brno: Computer Press, 2006, 813 s., ISBN 80-251-1073-7. [cit. 2017-12-05]

[2] *Zendulka J.* Databázové systémy IDS, Studijní opora. [online] 2006, [cit. 2017-12-02] Dostupné z: `https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIDS-IT%2Ftexts%2FIDS_predn.pdf&cid=9969`

[3] *Bráza J.* PHP5: začínáme programovat Praha: Grada Publishing, 2005, 224 s., ISBN 80-247-1146-X [cit. 2017-12-05]

[4] *Kofler M.* PHP5, MySQL: průvodce webového programátora. Vyd. 1. Brno: Computer Press, 2007, 607 s., ISBN 978-80-251-1813-9 [cit. 2017-12-05]

[5] MVC aplikace & presentery. [online]. [cit. 2017-12-08]. Dostupné z: `https://doc.nette.org/cs/2.2/presenters`

[6] *Reenskaug T.* Models-Views-Controllers [online]. 1979 [cit. 2017-12-08]. Dostupné z: `https://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf`

[7] *10 PHP Frameworks For Developers – Best of* [online]. 2015 [cit. 2017-11-29]. Dostupné z: `http://www.hongkiat.com/blog/best-php-frameworks/`

[8] *Nielsen J.* Parallel & Iterative Design + Competitive Testing = High Usability [online]. 2017 [cit.2017-12-02]. Dostupné z: `https://www.nngroup.com/articles/parallel-and-iterative-design/`

[9] *Nielsen J.* Why You Only need to Test with 5 Users? [online]. 2017 [cit. 2017-11-30]. Dostupné z: `https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`

[10] *Nielsen J.* How Many Test Users in a Usability Study? [online]. 2017 [cit. 2017-11-30]. Dostupné z:
`https://www.nngroup.com/articles/how-manytest-users`

[11] *Laravel Introduction.* [online]. [cit. 2017-11-28]. Dostupné z:
`http://laravel.com/docs/4.2/introduction`

[12] *Symfony* [online]. 2017 [cit. 2017-11-29]. Dostupné z:
`http://symfony.com`

[13] *Nette* [online]. 2017 [cit. 2017-11-29]. Dostupné z:
`https://doc.nette.org/cs/2.4/`

[14] *Kadlec Elektro* Vyvolávací systémy [online]. 2017 [cit. 2017-12-10]. Dostupné z:
`http://www.kadlecelektro.cz/produkty/vyvolavaci-systemy/`

[15] *Laravel* Blade Templates. [online]. 2017 [cit. 2017-12-10]. Dostupné z:
`https://laravel.com/docs/5.5/blade`

[16] *DeLisle M.* PhpMyAdmin, efektivní správa MySQL Brno: Zoner Press, 2004. 264 s., ISBN: 80-86815-09-9

[17] *Kubrická K., Kubrický J.* Základy tvorby www stránek a jednoduchých www aplikací [online]. 2017 [cit. 2017-12-10]. Dostupné z:
`http://www.kteiv.upol.cz/frvs/ict-kubricky/inc/TWWW/Metodicke-Listy-Bootstrap.pdf`

[18] *Composer* [online]. 2018 [cit. 2018-04-12]. Dostupné z:
`https://getcomposer.org/doc/00-intro.md`

[19] *Frameworks* [online]. 2015 [cit. 2017-11-28]. Dostupné z:
`https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/`

# Obsah CD

Přiložení CD obsahuje následující složky a soubory:

- wr_src - složka se zdrojovými kódy aplikace,
- latex_src - složka se zdrojovými kódy pro LaTeX,
- Demo - složka s demonstračním videem s titulky,
- bakalarska_prace.pdf - tato práce v PDF,
- README.txt - popis instalace aplikace.