České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Online aplikace pro zdravotnickou ordinaci

Jakub Zelenka

Softwarové technologie a management
Web a multimédia

Vedoucí práce: Ing. Martin Komárek

9. ledna 2018

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jakub Zelenka

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Online aplikace pro zdravotnickou ordinaci

Pokyny pro vypracování:

Navrhněte, vytvořte a otestujte online aplikaci pro podporu chodu zdravotnické ordinace. Vývoj provádějte vhodnou agilní metodikou. Pokud se na konzultacích s doktory a zdravotními sestrami konkrétní zdravotnické ordinace nedomluvíte jinak, tak aplikace bude především umožňovat správu pacientských karet, objednávání pacientů, zaznamenání provedených úkonů, tisk receptů a poukazů, zobrazení časového harmonogramu lékařů a automatickou notifikaci pacientů o blížícím se termínu sjednané lékařské prohlídky. Dbejte na přívětivost uživatelského rozhraní a zabezpečení citlivých údajů o pacientech.

Seznam odborné literatury:

SOMMERVILLE, Ian. Software engineering. 9th ed. Boston: Pearson, c2011. ISBN 9780137053469.
COBB, Charles G. The project manager's guide to mastering agile: principles and practices for anadaptive approach. ISBN 9781118991046.

Vedoucí: Ing. Martin Komárek

Platnost zadání: do konce zimního semestru 2018/2019

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 4.4.2017

Poděkování

Děkuji vedoucímu práce, Ing. Martinovi Komárkovi, za velkou trpělivost. Dále děkuji ředitelce urologické kliniky, Editě Řezáčové, za dohled nad prací. V neposlední řadě děkuji svým rodičům za podporu během studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů.

V Praze dne 9. ledna 2018 .....

Odkaz na tuto práci

Zelenka, Jakub. *Online aplikace pro zdravotnickou ordinaci*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2018.

Abstrakt

Cílem této bakalářské práce je navrhnout a vytvořit online aplikaci pro podporu chodu zdravotnické ordinace. Aplikace bude navrhována na základě konzultací s doktory a zdravotními sestrami, které ji budou následně využívat pro správu pacientských karet, objednávání pacientů či zaznamenání provedených operací a zobrazení časového harmonogramu lékařů. Vývoj bude probíhat vhodnou agilní metodikou a samotné programování bude provedeno ve frameworku nad programovacím jazykem PHP.

Abstract

The goal of this bachelor's thesis is to design and create an online application for operational support in a medical clinic. Design of the application will be based on consultations with doctors and nurses that will use it for management of patient cards, patient ordering, recording the operations and doctor's time schedule. For developing will be used suitable agile method and programming will be done in framework based on PHP programming language.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Na úvod, před počáteční analýzou, bych rád upřesnil, že aplikace by i přes zaměření na konkrétní urologickou kliniku měla být navrhována obecněji pro možnost případného následného využití i v jiných lékařských klinikách/ordinacích obdobného rozsahu s podobnou organizační strukturou. Dále bych si dovolil na přání manažerky dané urologické kliniky ponechat tuto kliniku v anonymitě, proto dále toto konkrétní lékařské zařízení budu nazývat jen „klinika“ či „urologická klinika“.

1.1 Počáteční analýza

Počáteční analýza má přesněji popisovat aktuální fungování urologické kliniky, a to z pohledu pracovníků kliniky, toku informací uvnitř kliniky a klientů, jež kliniku navštěvují.

1.1.1 Uživatelé a jejich role na klinice

Provoz kliniky zajišťuje celkem devět zaměstnanců, z toho jedna ředitelka, dva doktoři, dvě sestry (jedna vrchní), dvě recepční, jedna administrativní pracovnice a externí IT pracovník.

Recepční Recepční se stará o komunikaci s klienty, spravuje klientské karty, sděluje klientům výsledky a předává vystavené recepty, je tedy jakousi spojkou mezi klienty a zbytkem ordinace.

Sestra Stejně jako recepční i sestra spravuje klientské karty, zajišťuje objednávky pacientů a sděluje výsledky pacientům. Mimo to ale provádí některé úkony, dle doktora požadavku, například odběry či různá měření (uroflowmetrii). Dále také prochází laboratorní výsledky a sděluje klientům, pokud jsou v pořádku (když dokáže zhodnotit sama, jinak předá na doktora).

Vrchní sestra Dělá to samé co sestra, navíc spravuje kalendář doktorů.

Doktor Není zatížen zbytečnou agendou (ačkoliv kroky sestry či recepční může provádět také) ale stará se výhradně o kontrolu a vyšetření klientů, píše výsledky vyšetření a tiskne lékařské zprávy, vybírá léčiva a vypisuje na ně recepty. Určuje pacienta diagnózu a přidává provedené výkony, na základně nichž poté zdravotní pojišťovny vyplácí ordinaci nasmlouvané částky.

Ředitelka Zajímá ji časový plán doktorů a počet klientů za časové období, případně další výstupy týkající se chodu ordinace. Občasně také spravuje kalendář doktorů.

Administrativní pracovnice Výjmečně také objednává klienty, zajímá ji rovněž časový plán (kalendář doktorů).

IT pracovník Stará se o technické zázemí ordinace, pomáhá ostatním pracovníkům s problémy kolem počítačů i programů v nich.

Klient Dochází do ordinace na prohlídky, vyšetření, objednává se přes recepci a dostává výsledky laboratorních vyšetření.

1.1.2 Proces fungování kliniky z pohledu klienta

Registrace do kliniky Klient se chce registrovat do kliniky, provede to buď telefonicky, emailem nebo osobně, nejčastěji s recepční.

Objednání k návštěvě Klient sdělí svůj problém. Dle akutnosti, závažnosti a časového vytížení doktorů je mu (nejčastěji prostřednictvím recepční) nabídnut termín u preferovaného doktora či možnost navštívit v bližším termínu doktora druhého.

Dostavení se na prohlídku či vyšetření Při příchodu na kliniku s předchozím objednáním je klient na recepci identifikován a poslán do čekárny u příslušné ordinace. v případě příchodu bez objednání je zváženo jeho přijetí na základě závažnosti a časové vytíženosti doktorů.

Příchod k doktorovi Po vyzvání doktorem klient dochází do samotné ordinace, kde sdělí svoje obtíže a je mu provedeno příslušné vyšetření či operace. Jsou jim předepsány příslušné léky a na vyžádání vydána lékařská zpráva.

Objednání na příští kontrolu Ihned po návštěvě doktora je klient zpravidla vyzván k návštěvě sestry v sesterně, která ho objedná na další prohlídku (zpravidla za 3 nebo 6 měsíců). Sestra případně na žádost doktora provede vedlejší vyšetření jako uroflowmetrii či odběr krve nebo moči.

Zjištění výsledků Pokud byl klientovi odebrán vzorek pro laboratorní test, je klinikou do několika dnů informován ohledně výsledků. v případě potřeby je vyzván k další návštěvě (zde se opakuje objednání) či k vyzvednutí předepsaných léků.

1.2 Současné řešení

Klinika v současné době využívá online aplikaci cizího zdravotnického zařízení na základě smlouvy, tato aplikace je sdílena mezi několik zdravotnických zařízení. Aplikace je vyvíjena přes 20 let a to s sebou přináší řadu výhod i nevýhod.

Výhody současného řešení

- **Multiplatformní použití**
  Aplikace je online a funguje ve webových prohlížečích, a protože je vyvíjena řadu let, je odladňována pro správné zobrazení na v nejrůznějších prohlížečích a to i v jejich starších verzích.

- **Velké množství formulářů a tiskopisů**
  Jelikož je aplikace používána ve velkém lékařském zařízení, které sdružuje ordinace nejrůznějších zaměření, obsahuje velké množství formulářů a tiskopisů k vyplnění (ať už jsou to lékařské recepty, zprávy či nejrůznější žádanky).

- **Sdílená databáze klientů**
  v aplikaci je zhruba 200 tisíc aktivních klientů a ačkoliv má urologická klinika svých klientů přibližně 11 tisíc, při příchodu klienta z jiného zařízení, které sdílí tuto aplikaci, může využít jeho kartu. Vidí tedy případné alergie, vydané recepty, lékařské zprávy a další údaje zapsané od ostatních lékařů a pracovníků z jiných zařízení.

Nevýhody současného řešení

- **Téměř neresponzivní webový design**
  Téměř žádná stránka aplikace nevyužívá celou šířku monitoru, což není ideální, protože se na urologii využívají monitory s FullHD rozlišením.

- **Obrovský přebytek nepodstatných funkcí**
  Protože je doposud využívaná aplikace značně univerzální, nabízí nepřeberné množství funkcí i informací, které pracovníci urologické kliniky nevyužijí.. Jsou to například rezervace místností, nemocničních přístrojů atp. Tyto funkce dělají systém méně přehledný a hůře použitelný.

- **Složité vyhledávání klientů**
  Současně používaná aplikace umožňuje vyhledávání klientů dle několika klíčů (například rodné číslo, příjmení, jméno), je však potřeba zadávat klíč do správného pole. Pokud si pracovník není jistý, zda se jedná o příjemní či jméno (což se stává u zahraničních klientů), musí zkusit zadat slovo postupně do více polí. Vzhledem k velikosti sdílené databáze klientů najde pracovník s velkou pravděpodobností klienty, které vůbec nehledal.

- **Nepřehledný seznam návštěv**
  Ať už je pracovník v samotné kartě klienta a zobrazuje jeho minulé a budoucí návštěvy, či je na denním přehledu doktora, mimo návštěvy, které ho zajímají, vidí ve stejné tabulce také laboratorní výsledky, informace o odeslaných upozorněních (zvací e-mail a sms) a další úkony, které jsou pro něho v tu chvíli naprosto irelevantní a měli by být z přehledu návštěv odstraněny.

- **Zdlouhavé plánování návštěv**
  Při plánování nové návštěvy musí pracovník ručně procházet dlouhé seznamy ordinačních budov a doktorů, ačkoliv má urologická klinika pouze jednu budova a dva doktory. To přináší zbytečnou časovou zátěž při tak častém úkonu, jako je objednání či přeobjednání klienta.

- **Nízké zabezpečení aplikace**
  Jelikož je aplikace primárně určena pro výše zmíněné velké zdravotnické zařízení, kde je provozována pouze jako interní aplikace, a pouze pro potřeby urologické kliniky je dostupní na veřejné webové adrese, nemá příliš dobře řešené zabezpečení. Uživatelům jsou generována krátká a heslo, téměř shodná s uživatelskými jmény (doslova jen dvě přidané číslice za uživatelské jméno). Dále nemá aplikace certifikát podepsaný důvěryhodným vystavovatelem, což budí v pracovnících značnou nervozitu, přeci jen se jedná o velmi citlivá data pacientů. Dále aplikace umožňuje po expiraci hesla zadat znova to samé, již tak nepříliš bezpečné heslo.

Současné řešení tedy má své nevýhody ale i několik výhod. Největším problémem je ale samotný návyk uživatelů, kteří ačkoliv aplikaci neradi používají, a stěžují si, jak je složitá a nepřehledná, jsou po i několikaletém používání vlastně zvyklí, vědí jak a kde co dělají a mohou těžce nést změnu.

1.3 Jiná existující řešení

Na trhu je samozřejmě několik aplikací od různých dodavatelů pro podporu chodu zdravotnické kliniky, problémem bývá složitost, zastaralost, použití pouze lokálně (bez možnosti vzdáleného přístupu) a hlavně nemožnost využít aplikaci jak na počítači s operačním systémem Windows tak na počítačích Apple s operačním systémem MacOS, které jsou na klinice využívány.

1.4 Důvody pro vytvoření vlastního řešení

Ačkoliv existujících řešení není malé množství, nejsem spokojen s tím, jak fungují, je to dáno hlavně délkou jejich vývoje, který je mnohdy i desítky let. Mám tedy pocit, že je zde do budoucna značná mezera na trhu, která se dále bude rozšiřovat s tím, jak tyto léta zaběhnuté aplikace budou stárnout a stávat se novým, převážně mladším uživatelům méně použitelné vzhledem k progresivnímu vývoji ostatních aplikací kolem nich.

Na výše zmíněné urologické klinice mohu nejet čerpat jejich dosavadní procesy fungování, seznámit se s jinak veřejnosti nedostupnou aplikací, kterou dosud využívají, ale i testovat své řešení v reálném prostředí.

V budoucnu je pak samozřejmě možné takovou aplikaci nabízet jako službu i pro další menší kliniky, což by bez řádného zaběhnutí v reálném prostředí nebylo příliš jednoduché.

Kapitola 2

Návrh aplikace

2.1 Požadavky na aplikaci

2.1.1 Business požadavky

Uvádím zde business požadavky uživatelů, které jsem sesbíral při počátečních pohovorech s jednotlivými pracovníky a hlavně s manažerkou kliniky, u každého požadavku je také uvedena kategorizace dle takzvaného Kano modelu, který vyjadřuje touhy uživatele u jednotlivých požadavků a také vyjadřuje uspokojení jeho potřeb v případě (ne)přidání požadované funkce do aplikace.[1]

2.1.1.1 Recepční

- potřebuje snadno vyhledávat registrované klienty a také nově registrovat doposud nezaregistrované klienty *(nutná vlastnost)*
- potřebuje rychle a intuitivně objednávat klienty na návštěvy k doktorům, případně rušit návštěvy *(nutná vlastnost)*
- měla by mít možnost snadno informovat sestry a doktory, že pacient již došel a čeká v čekárně *(požadovaná vlastnost)*
- má snadný přístup k ohodnoceným laboratorním výsledkům a lékařským výsledkům z kontrol *(požadovaná vlastnost)*

2.1.1.2 Sestra

- je jí umožněno vykonávat to, co vykonává recepční, může ji i zastat *(požadovaná vlastnost)*
- po obdržení laboratorních výsledků je ji umožněno na tyto výsledky reagovat směrem ke klientovi či je předat na doktora k vyřízení, pokud je vyhodnotit nedokáže *(požadovaná vlastnost)*

2.1.1.3 Vrchní sestra

- má možnost vykonávat všechno to, co vykonává sestra *(požadovaná vlastnost)*
- potřebuje spravovat pracovní dobu a volno doktorů *(požadovaná vlastnost)*

2.1.1.4 Doktor

- může dělat všechno to, co vrchní sestra *(požadovaná vlastnost)*
- může psát a editovat lékařskou zprávu u každé návštěvy klienta *(nutná vlastnost)*
- k návštěvě mu musí být umožněno přidávat diagnózy, k nim provedené výkony a také léčiva a k nim tisknout recepty *(nutná vlastnost)*
- k návštěvě by měl mít možnost přidat také soubory (například obrázek z ultrazvuku, výstupný dokument z uroflowmetru atp.) *(zbytná vlastnost)*
- může vypisovat léčiva a tisknout recepty ke klientům i bez návštěvy (tj. přímo) *(požadovaná vlastnost)*

2.1.1.5 Administrativní pracovnice

- může dělat vše to, co recepční *(požadovaná vlastnost)*
- měla by mít přehled o provedených výkonech *(zbytná vlastnost)*
- systém ji umožní vygenerovat seznam provedených výkonů za měsíční období (tzv. dávky) pro zdravotní pojišťovny *(zbytná vlastnost)*

2.1.1.6 Ředitelka kliniky

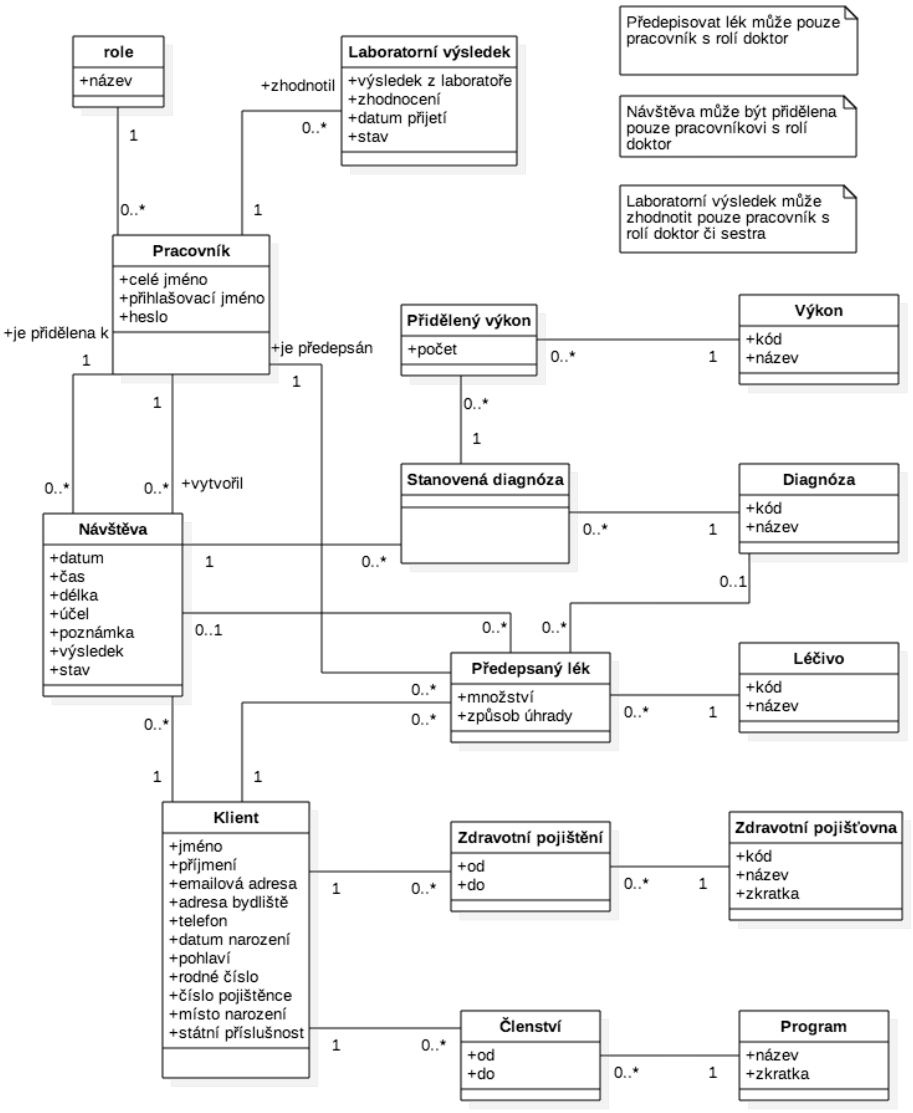
- může dělat vše, co recepční *(požadovaná vlastnost)*
- může spravovat pracovní dobu a volno doktorů stejně, jako vrchní sestra *(požadovaná vlastnost)*
- měla by mít přehled o provedených výkonech stejně jako administrativní pracovnice *(zbytná vlastnost)*
- měla by mít statistický přehled o celkové klientele (počet nových klientů) a o jednotlivých zaměstnancích (například produktivita - počet klientů u doktora za hodinu) *(zbytná vlastnost)*

2.1.2 Kvalitativní požadavky

- **Jednoduchost**
  Aplikace musí být jednoduchá a přehledná, aby se v ní uživatelé dobře orientovali a měli zobrazené všechny nezbytné informace k právě vykonávané akci.
- **Multiplatformní použití**
  Aplikace by měla být schopna běžet na počítačích s operačním systémem Windows ale i MacOS od Apple, není ale podstatné podpora různých prohlížečů a jejich starších verzí, bohatě stačí plná funkčnost v nejnovější stabilní verzi prohlížeče Chrome od Google.
- **Zabezpečení**
  Aplikace bude zpracovávat a uchovávat velmi citlivá data uživatelů, proto je důležité nejen zajistit bezpečné přihlašování do aplikace, ale zaměřit se na zabezpečení aplikace ve všech jejich částech.
- **Stabilita**
  Celá aplikace musí fungovat stabilně, přeci jen je používána uživateli po celou pracovní dobu (až 12 hodin denně, 5 dní v týdnu), chod kliniky je na aplikaci zcela závislý.
- **Zacílení na hlavní roli uživatele**
  I přesto, že dva uživatelé rozdílných rolí mohou mít dostupné stejné funkcionality, uživatel by měl mít nejsnadněji přístupné ty funkcionality a informace, které jsou v jeho pracovní roli nejčastěji využívány.
- **Zobecnění**
  Aplikace by měla být navrhována a implementována se značnou obecností pro možnost budoucího rozšíření i na jiné kliniky.
- **Rozšiřitelnost**
  Aplikace musí být do budoucna rozšiřitelná a je třeba s tím počítat již od začátku návrhu.

2.2 Doménový model

Počáteční doménový model má za úkol nastínit, jakým směrem vývoje se bude aplikace dále ubírat, pro jakou metodiku vývoje se rozhodnu, a jak budou pravděpodobně vypadat základní třídy v aplikaci.



Obrázek 2.1: Diagram modelu tříd

Kapitola 3

Metodika vývoje

Agilní metodiky si zakládají na přírůstkovém přístupu programové specifikace, programování i dodání k zákazníkovi. Hodí se pro aplikace, kde se mění požadavky již během vývoje. Jsou určeny k rychlé dodávce funkční aplikace k zákazníkovi, který po té navrhne změny či nové funkce, jež budou zahrnuty v dalších verzích aplikace. Cílem agilních metodik je omezit procesní byrokracii například odstraněním dokumentace, která se nikdy nepoužije, či omezením práce s nejasnou dlouhodobou návratností.

3.1 Příklady agilních metodik

Agilních metodik je velké množství a je běžné, že se při vývoji agilní metodika přizpůsobí potřebám daného projektu dle rozsahu, složitosti či časové náročnosti některé z nich zde představím a krátce popíši.

Scrum Scrum, neboli česky "mlýn", je patrně nejpoužívanější[2] agilní metodika nejen pro vývoj softwaru. Spočívá v rozdělení práce na malé, lépe spravovatelné části, které tým vykonává v časově vymezené iteraci nazývané "sprint". Scrum předpokládá tři role účastníků v týmu, a to produktového vlastníka, Scrum mastera a vývojáře a čtyři předepsaná setkání - plánování Sprintu, denní setkání, zhodnocení Sprintu a retrospektiva.

Extrémní programování (XP) Extrémní programování (zkráceně XP) se zaměřuje na zajištění kvality dodávaného softwaru a předepisuje postupy, jak toho dosáhnout. Vývojový tým se schází při plánování vydání a plánování iterace, pracuje ve velmi krátkých cyklech, takže změny požadované zákazníkem mohou být rychle zahrnuty. Díky využití velkého množství základních postupů, jako je třeba Testem řízený vývoj (TDD), Párové programování či Testování zákazníkem, vzniká kvalitní produkt který reaguje na změny požadované zákazníkem.

Kanban Kanban byl vyvinut jako součást produkčního systému firmy Toyota. V Kanbanu je pracovní postup vizualizován, projekt je rozdělen na malé (nezávislé) části - úkoly, ty jsou napsány na karty a nalepeny na tabuli. Tabule je rozdělena na sloupce, které značí postup daného úkolu v projektu (jednoduše například sloupce: nápady, rozpracováno a dokončeno). Podstatné je, že je striktně limitován počet úkolů, které jsou rozpracovány, ale výběr úkolu, na kterém se bude pracovat, může být zcela náhodný.

3.2 Výběr agilní metodiky

Při výběru agilní metodiky, kterou použiji pro programování této aplikace, jsem se rozhodoval dle svých schopností dodržet časové intervaly, pracovat pravidelně a hlavně potřebu soustředit se na nejlépe jen jeden, úzce specifikovaný problém v danou chvíli, a také jsem zohlednil velikost týmu (jednočlenný). Rozhodl jsem se tedy použít Kanban, který navíc doplním o přibližně týdenní intervaly, ve kterých se budu scházet s ředitelkou kliniky, vždy si ukážeme stav úkolů zadaných na předchozí schůzce a specifikujeme úkol(y) do dalšího setkání, přitom si budeme vést výše zmíněnou tabuli toho, co je hotovo, rozpracováno a které úkoly jsou zatím v čekací frontě.

Kapitola 4

Výběr technologií

Technologie, které budou použity jak na vývoj, tak na provoz samotné aplikace, jsem vybíral na základě kvalitativních požadavků, zkušeností a také vlastního zájmu a dané technologie.

4.1 Architektura aplikace

Před samotným výběrem technologií nejprve určím, jak by měla vypadat architektura aplikace.

Jelikož má být aplikace provozována jako webová, rozhodl jsem se na klientské straně pro použití takzvané SPA (Single Page Application), ta bude uživateli aplikace zobrazovat obsah, odesílat formuláře a přijímat data.

Na druhé straně bude serverová část aplikace, jež bude přijímat a zpracovávat informace od klientské části a posílat do ní požadovaná data.

Klientská a serverová část aplikace spolu budou komunikovat přes protokol HTTP a informace budou předávány buď v těle zprávy ve formátu JSON nebo přímo v dotazovacím řetězci jako součást URL. [3]

4.2 Framework Symfony

Symfony je PHP framework pro vývoj webových aplikací. Celý framework či jeho komponenty jsou využívány ve velkém množství volně dostupných i proprietárních webových aplikacích a projektech. To přímo napovídá velké komunitě uživatelů a snadno dohledatelným návodům či řešení problémů. Je vydáván pod MIT licencí.

4.3 Framework Vue.js

Vue je JavaScriptový framework s dalšími volitelnými nástroji pro tvorbu uživatelského rozhraní. Mezi jeho výhody patří malá velikost, detailní dokumentace, snadnější vývoj díky poměrně přesnému zobrazování a upozorňování na chyby, snadná integrace, flexibilita (šablony je možné psát v HTML ale i v JavaScriptu) a dvojcestná komunikace pro snadnou manipulaci HTML bloků. v této aplikaci na něm budu stavět klientskou část aplikace.[4] Je vydáván pod MIT licencí.

4.4 Bootstrap

Bootstrap je elegantní, intuitivní a výkonný front-end framework pro rychlejší a snadnější vývoj web aplikací.[5]

4.5 CoreUI Bootstrap šablona

CoreUI Bootstrap Template je šablona pro administraci, která přímo integruje framework Bootstrap do frameworku Vue, rozhodl jsem se ji použít pro usnadnění začátku vývoje aplikace, pro její čistý kód a pro odstínění vývoje aplikace od návrhu designu uživatelského prostředí, který je v této šabloně pro začátek zcela dostačující, ale pro konzistentní značky a přehlednou strukturu není problém později aplikaci snadno přestylovat. Je také vydávána pod licencí MIT.

Kapitola 5

Implementace aplikace

V této kapitole se budu věnovat samotné implementaci a popisu fungování aplikace.

5.1 Serverová část aplikace

Serverová část aplikace je, jak jsem již zmiňoval ve vybraných technologicíh, psaná v PHP za použití frameworku Symfony, konkrétně ve verzi 3.2.13.

5.1.1 Hlavní část aplikace

Hlavní část aplikace je v balíčku pojmenovaném AppBundle *(/src/AppBundle)*, ten obsahuje controllery, entity, managery, repository (repozitáře) a service (služby), kterým se zde budu blíže věnovat. Nechci rozebírat každou třídu, protože se v nich často opakuje to samé, účelem je probrat různé druhy tříd a u nich zmínit podstatné techniky a funkčnost. Při přidávání nové funkce do aplikace jsme obvykle postupoval tak, že jsme vytvořil novou entitu, přidal controller a manager, případně doplnil repository a service.

5.1.1.1 Entita

Entit je v balíčku AppBundle celkem 24. Generoval jsem je po jedné pomocí příkazu *php bin/console generate:doctrine:entity*, který v příkazové řádce vyvolá interaktivní zadávání nastavení entity a jednotlivých atributů. Pro nastavení mapování jsem využil anotace. Po dokončení je vygenerována třída entity s potřebnými atributy, ORM anotacemi pro vygenerování tabulky a sloupců v databázi, gettery a settery.

Poté u entit ve většině případů přidávám také anotace z balíčku JMS\Serializer. Ten zajišťuje poměrně snadnou serializaci objektů i celých kolekcí vytažených z databáze na objekty JSON. Konkrétně jde o tyto anotace:

- **Groups**
  Umožňuje atributy v entitách přiřazovat do různých skupin, to usnadní výběr dat, která se budou do výsledného JSONu serializovat, funguje to i napříč entitami, které jsou spolu v relaci.

- **Type**
  Určuje typ atributu, který chceme serializovat. Velmi užitečné například pokud entita obsahuje atribut typu *DateTime*, tímto ho rovnou můžeme do výsledné JSONu zobrazit ve formátu *DD.MM.YYYY*.

- **VirtualProperty**
  Slouží k vytvoření virtuálního atributu. Příklad využití je například u věku klienta, který nikde uložený není, nicméně je v databázi uložen datum narození. Tato anotace mi vytvoří atribut z funkce a serializuje do výsledného JSONu.

- **SerializedName**
  Slouží k pojmenování či přejmenování atributů ve výsledném JSONu, tuto anotaci využívám hlavně u virtuálních atributů.

U některých entit také přidávám anotaci *SoftDeletable* z balíčku Gedmo, touto anotací označím atribut, který bude označovat smazaný řádek v tabulce (datem smazání), atribut musí být typu *DateTime* a musí být nulovatelný (*nullable=true*). Dále tento balíček zajišťuje, že mohu používat klasické mazání objektů, nicméně se nesmažou, pouze se označí jako smazané (takzvaný soft delete).

Dále využívám u entit anotaci Assert, která je součástí komponenty Validator, přímo obsažené ve frameworku Symfony. Touto anotací lze určovat typy, podmínky a hodnoty, kterých musí atribut nabývat, aby byl validní. Také umožňuje zadávat zprávy při porušení validačních podmínek. Touto validací by postupně měly být doplněny všechny atributy, jež zadává uživatel aplikace.

5.1.1.2 Controller

Controller v aplikaci zpracovává příchozí HTTP požadavky a vrací HTTP odpovědí s příslušným stavovým kódem a případně s daty ve formátu JSON či s popisem chyby.

Obvykle controller obsahuje 5 funkcí

- **Výpis všech objektů**
  Tzv. indexAction, je reakcí na HTTP požadavek GET při zadání URL bez konkrétního id.
  například */client*

- **Výpis jednoho objektu**
  Tzv. idAction, je reakcí na HTTP požadavaek GET při zadání URL s požadovaným id obejktu.
  například */client/6*

- **Přidání nového objektu**
  Tzv. postAction, je reakcí na HTTP požadavek POST při zadání URL bez požadovaného id, které ve chvíli vytváření objektu neznáme.
  například */client*

- **Úprava jednoho objektu**
  Tzv. putIdAction, je reakcí na HTTP požadavek PUT při zadání URL s požadovaným id, které při úpravě objektu známe.
  například */client/8*

- **Smazání jednoho objektu**
  Tzv. deleteIdAction, je reakcí na HTTP požadavek PUT při zadání URL s požadovaným id, které při mazání objektu známe.
  například */client/15*

Nicméně některé controllery všechny z výše uvedených funkcí neobsahují, buď protože je výpis úplně všech objektů irelevantní, v takovém případě naopak obsahují funkci pro získání všech objektů dle jednoho jiného objektu (například všechny návštěvy daného klienta), nebo jsou některé funkce zatím nepodstatné a budou doplněny při dalším rozšiřování aplikace.

U funkcí, jež reagují na GET je struktura obvykle stejná, najde se požadovaný objekt nebo kolekce (při složitějším dotazu je dotaz přesunut do manageru, ze kterého se volá), provede se případná úprava a je vrácena odpověď. Funkce může vrátit HTTP odpověď s kódem 200 či 404.

Funkce, které reagují na POST nejprve vytvoří požadovaný objekt a pomocí deserializace (zde používám komponentu Serializer, která je přímo součástí frameworku Symfony). Funkce obvykle vrací odpovědi s kódem 201 či 204.

Funkce reagující na PUT nejdříve vyhledají objekt, který se bude upravovat a poté ho přes deserializaci změní (změní se jen nově přijaté atributy). Funkce obvykle vrací odpovědi s kódem 200, 404 či 422.

Funkce, které reagují na DELETE nejdříve vyhledají objekt, který se má smazat, poté ho odstraní (pokud se jedná o objekt, jehož entita má nastavený soft delete, nesmaže se úplně, jen se přidá záznam o smazání). Funkce obvykle vrací odpovědi s kódem 204 či 404.

5.1.1.3 Manager

Manager se stará o odstínění (zatím jen) složitějších dotazů z Conrolleru pro jeho zjednodušení.

5.1.1.4 Repository

Sem patří funkce pro sestavování netriviálních dotazů do databáze, obvykle prostřednictvím DQL.

5.1.2 Vedlejší části aplikace

Ačkoliv to nebylo z úvodní analýzy po rozhovorech s pracovníky urologické kliniky zřejmé, v aplikaci musí být také data z externích číselníků a databází, která jsou nezbytná pro zadávání léků, diagnóz či provedených výkonů. Pro lepší orientaci v aplikaci jsem se rozhodl umístit je v rámci aplikace do samostatných balíčků (bundlů), konkrétně se jedná o ICDBundle (*/src/ICDBundle*), SUKLDrugBundle (*/src/SUKLDrugBundle*) a SZVBundle (*/src/SZVBundle*). Všechny tři jsou tedy jakousi implementací externích databází a seznamů pro potřeby aplikace.

5.1.2.1 ICDBundle

Slouží k vyhledávání mezinárodního klasifikace nemocí (MKN-10, anglicky ICD-10). Obsahuje 4 entity - kapitolu, sekci, podsekci a nemoc.

5.1.2.2 SUKLDrugBundle

Slouží k vyhledávání léčiv prodávaných a schválených na území ČR, jde o otevřená data, která pochází jsou dostupná na webových stránkách Státního ústavu pro kontrolu léčiv (SUKL). Nezbytně nutný je jedinečný 7místný kód léčiva, název, síla, velikost balení a forma podání (tablety, injekce atp.), tyto informace jsou nutné pro uvedení na vydávaném receptu. Já jsem se ale rozhodl tento balíček udělat kompletnější a připravit ho i na budoucí potřeby, například informace o dostupnosti, omezení výdeje, informaci o ceně, obsahu látek vyvolávajících závislost, doping, hormony či indikační skupiny. Entit je tedy celkem 11, některé jsou využívány již nyní, další jsou připraveny.

5.1.2.3 SZVBundle

Slouží k vyhledávání zdravotních výkonů, jde o data dostupná pro širokou veřejnost, spravovaná Ústavem zdravotnických informací a statistiky České republiky. Tyto výkony slouží jako podklad pro kalkulaci výše úhrady zdravotních výkonů ze zdravotního pojištění a podmínky jejich vykazování.[6] SZVBundle má celkem 4 entity: samotný výkon, omezení místem, kategorie a odbornost.

5.1.3 Použité HTTP odpovědi, které serverová část aplikace vrací a jejich význam

Serverová část aplikace posílá zpět odpověď, která obsahuje třímístný číselný kód, který klientské aplikaci říká, jak byl dotaz zpracovaný.

5.1.3.1 Úspěšná odpověď

Úspěšné odpovědi mají stavový kód ve formátu 2xx, server je odesílá pokud vše proběhlo v pořádku.

200 OK Server zpracoval dotaz a vše proběhlo v pořádku. Př.: Tuto odpověď posílá server při požadavku na vypsání pacientů, a do těla odpovědi přidá nalezené pacienty.

201 Created Server zpracoval požadavek na vytvoření entity a vše proběhlo v pořádku, tato odpověď je odeslána například při vytvoření nového pacienta. v těle odpovědi odešle JSON s vytvořenou entitou.

204 No content Odpověď na korektně zpracovaný dotaz ale bez nalezených záznamů. Př.: Server dostal požadavek na vypsání všech návštěv klienta, žádná ale nebyla nalezena.

5.1.3.2 Chyba na straně klienta

Tyto chybové odpovědi předpokládají, klient zaslal chybný požadavek, mají formát 4xx.

401 Unauthorized Tato odpověď je serverem odeslána, pokud klient nepřiloží do hlavičky požadavku autorizační token a autorizace je na dané adrese serverem vyžadována, nebo pokud platnost tokenu, přiloženého v hlavičce, již vypršela.

403 Forbidden Server odešle tuto odpověď v případě, že se klient dotazuje na adresu sice s platným tokenem, ale s nedostatečným oprávněním (nemá roli, která je pro danou adresu vyžadována).

404 Not Found Server odesílá tuto odpověď v případě, že se klient dotáže na adresu, jež server neobstarává, nebo se dotazuje metodou, která není na dané adrese použita (př.: záměna GET za POST).

404 Conflict Server odesílá tuto odpověď v případě, že je poslán požadavek na smazání objektu, jež je v relaci s jinými objekty, a smazání by porušilo logiku.

422 Unprocessable Entity Tato odpověď je odeslána, pokud klient pošle požadavek POST (případně PUT) na vytvoření objektu, ale údaje nejsou kompletní (či neprojdou validací). Součástí odpovědi ze strany serveru by měl být výpis chybných či chybějících parametrů a důvod, proč neprošly validací. Stane se tak, pokud například přidáme nového klienta a zapomeneme vyplnit údaj "Datum narození".

5.1.3.3 Chyba na straně serveru

Takový druh odpovědi by server správně neměl nikdy odeslat, nastane v případě, že vznikla interní chyba na serveru. Pro příklad: než jsem ošetřil dekódování tokenu výjimkou a odchycením chyby při vypršení, server vracel takovou odpověď. Tyto odpovědi jsou ve formátu 5xx.

5.2 Klientská část aplikace

Klientská část aplikace je vyvíjena v JavaScriptovém frameworku Vue. Ten umožňuje díky takzvaným komponentám dělit aplikaci na menší části, ve kterých se lze lépe orientovat. Umožňuje také snadnou integraci dalších rozšíření.

Mimo komponenty, které již byly obsaženy v balíčku CoreUI (jako hlavička, patička a další obvyklé prvky webové aplikace) jsem přidal vlastní, jedná se o složitější komponenty, které nazývám pohledy (views), v aplikaci jich je celkem sedm:

5.2.1 Denní přehled

Denní přehled je téměř shodný ve dvou pohledech, a to *HomeDoctor* a *HomeReceptionist*, zatím jsou tam jen drobné rozdíly. Je to hlavně příprava do budoucna, kdy bude domovská stránka co nejvíce uzpůsobena jednotlivým rolím a budou přidány i pohledy dalších, již zmíněných rolí (ředitelka, administrativní pracovnice, sestra), které zatím mohou vykonávat stěžejní práci ve stejném prostředí jako recepční.

Pracovníci v něm vidí všechny návštěvy, které náleží danému dni, podstatné je zde filtrování, a to podle doktorů, data či stavu návštěvy. Nastavení filtrování se ukládá do cookies a zůstane nastavené až do odhlášení z aplikace.

5.2.1.1 Stavy návštěvy

'scheduled','failed','excused','arrived','canceled','completed','elaborated' Návštěva má celkem 7 možných stavů:

- **naplánovaná** - *scheduled*
  Jde o výchozí stav návštěvy, je ji přiřazen po vytvoření nové návštěvy,

znamená, že je klient na daný termín objednán ale zatím se nedostavil ani se objednávka nijak nerušila.

- **klient dorazil** - *arrived*
  Je stav návštěvy, kdy klient dorazil na kliniku a recepční ho identifikovala. Znamená tedy, že je klient již v čekárně.

- **rozdělaná** - *elaborated*
  Je stav návštěvy, který se nastaví automaticky po jakékoliv úpravě doktora u návštěvy (přidání léku, zapsání diagnózy, rozepsání lékařské zprávy). Značí, že je klient již u doktora, ale objednávka ještě není zcela dokončena.

- **dokončená** - *completed*
  Tímto stavem doktor označí návštěvu a je tedy brána jako hotová.

- **klient se nedostavil, s omluvou** - *excused*
  Je stav, kterým pracovníci označí návštěvu, u které se klient omluvil, že se nedostaví.

- **klient se nedostavil, bez opmluvy** - *failed*
  Je stav, kterým pracovníci označí návštěvu, u které se klient v daný den nedostavil a neomluvil se. Slouží orientačně pro přehled klientů, kteří se často nedostaví bez omluvy a blokují tím termíny v kalendáři.

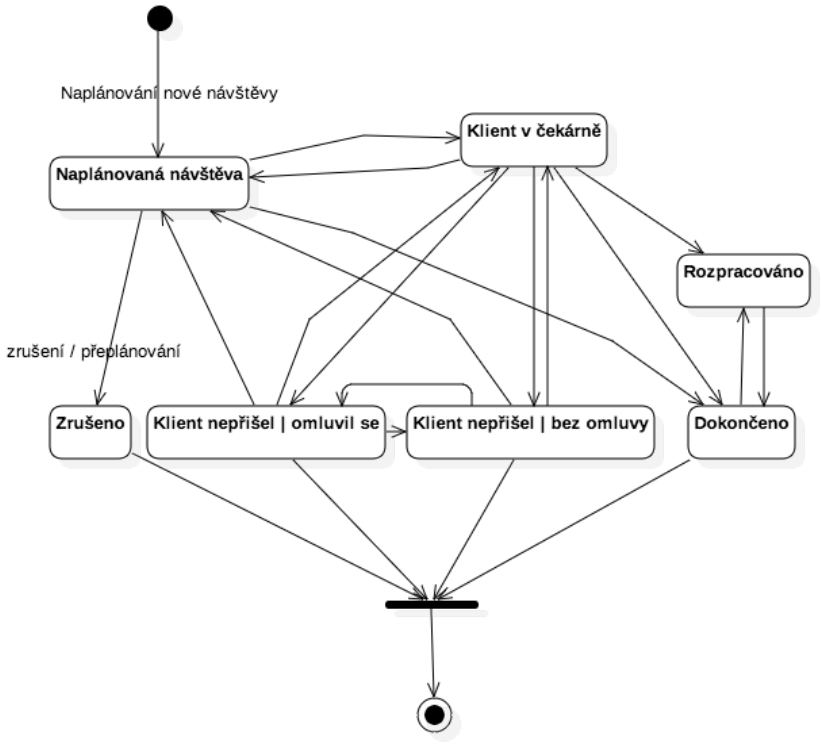
- **zrušená** - *canceled*
  Tímto stavem pracovníci označí návštěvu, která je z nějakého výše neuvedeného důvodu zrušena. Automaticky se také nastaví k návštěvám, které jsou přeobjednány na jiný termín (původní se nesmaže ale nastaví se mu tento stav).

5.2.1.2 Funkce a informace u každé návštěvy

Mimo změnu stavu návštěvy se v každém řádku zobrazují nejdůležitější informace o návštěvě (datum, čas, délka návštěvy, důvod a poznámka), klientovi (rodné číslo, pojišťovna, členství v programu) a možnost upravit poznámku a změnit doktora. Pracovníci s rolí recepční mohou zobrazit výsledky kontroly (lékařská zpráva, provedené diagnózy, výkony a předepsané léky), doktoři mohou do návštěvy vstoupit a zmíněné informace přidávají.

5.2.2 Úkolník

Tento pohled zobrazuje frontu úkolů, pro snadnou orientaci je vytvořený v podobném stylu, jako návštěvy. Zatím slouží pro zobrazení a vyřešení výsledků z laboratoře. Dále do něho budou přibývat další typy úkolů a filtrace, která bude velmi podobná té, která je na hlavní stránce (denní návštěvy).



Obrázek 5.1: Diagram stavů návštěvy a povolených přechodů

U každého laboratorního výsledku je vidět datum přijetí do systému, klient, ke kterému laboratorní výsledek patří (v případě, že ho bylo možné přiřadit dle rodného čísla či čísla pojištěnce) a doktor, který výsledek zadal. Dále je zde tlačítko pro zobrazení výsledku a formulář pro odpověď. Jelikož zatím v aplikaci není doimplementován emailový server, tlačítko "Odpovědět"odkáže pomocí *mailto* do výchozího emailového klienta s předvyplněným předmětem a zadaným tělem zprávy. Navíc je možné kliknout pravým tlačítkem myši kamkoliv do bloku obsahujícího výsledek z laboratoře, tím se text automaticky zkopíruje do schránky, je tedy snadné ho následně do emailu vložit.

5.2.3 Kalendář

v tomto pohledu je zobrazen kalendář doktora, zajímavá je samotná komponenta kalendáře, kde po výběru jednoho z doktorů zobrazuji v týdenním / 24hodinovém rozsahu 15minutová okénka poskytující informaci, zda má doktor v tomto časovém okénku pracovní dobu, naplánovaného klienta (či více klientů), dovolenou či žádný plán dle barevného značení:

- **zelená**

Označuje časová okénka s neobsazenou pracovní dobou (bez návštěv).

- **oranžová**
  Označuje časová okénka s obsazenou pracovní dobou (jsou v ní naplánovány návštěvy).

- **modrá**
  Značí časová okénka volna či dovolené.

- **červená**
  Označuje časová okénka s návštěvami mimo pracovní dobu či v době volna.

Přidávání pracovní doby Pracovní dobu lze přidávat na jednotlivé dny v týdnu (lichý, sudý či každý), od pondělí určitého data do neděle. Rozumné je nastavení od prvního pondělí v prvním týdnu kalendářního roku do neděle posledního pracovního týdne téhož roku s časovým intervalem od - do.

Přidání volna Volnou či dovolenou lze naopak přidávat vždy jen na jeden konkrétní datum buď s časovým intervalem od - do, nebo s volbou „celý den“.

Priority a překrývání v kalendáři Samozřejmostí je, že se volno, pracovní doba či návštěva navzájem překrývají, jak aplikace toto řeší? Nejvyšší prioritu má informace, že je v daném okénku návštěva. Druhou nejvyšší prioritu má vytvořené volno (dovolená). Nejnižší prioritu má pracovní doba. Bez jakékoliv priority zůstává okénko bez pracovní doby/volna či návštěvy. z toho vyplývá, že tvoříme pracovní dobu, kterou následně překrýváme volnem (dovolenou). Lze na ni pohlížet jako na určitou výjimku v opakující se pracovní době.

Zobrazení návštěv v kalendáři v kalendáři lze zobrazit výpis návštěv, obdobně, jako je tomu na hlavní stránce. Můžeme zobrazit buď všechny návštěvy, které probíhají v časovém intervalu jednoho konkrétního okénka (klikem na konkrétní okénko), či celý den klikem na datum v levém sloupci.

5.2.4 Plánování návštěvy

Komponenta pro plánování návštěvy je podobná, jako kalendář doktora, na rozdíl od něho je zde zobrazen čas pouze od 8:00 do 20:00 a okénka mají velikost 5 minut. Počet označených okének se dynamicky mění s vybranou délkou návštěvy.

5.2.5 Registrace klienta

Registrace probíhá v pěti krocích, kdy každý krok sdružuje informace, které spolu přímo souvisí.

- **1. krok**
  v prvním kroku se zadávají základní informace o klientovi, tedy občasnství, rodné číslo, jméno, příjmení, pohlaví a datum narození. Při zvolení jiného než českého občanství je možné přidat také číslo pojištěnce. Při výběru českého občanství je po zadání devíti či desetimístného rodného čísla toto číslo serverem zvalidováno, a je z něho zjištěno pohlaví a vypočítán datum narození. Okamžitě po prvním kroku je klient registrován a dalšími informacemi se jen dále upravuje.
- **2. krok**
  Druhý krok slouží pro výběr zdravotní pojišťovny, je také možnost tento krok bez výběru přeskočit.
- **3. krok**
  Třetí krok umožňuje přidat e-mailové adresy klienta, jedna z nich může být nastavena jako výchozí a volitelně je možné přidat poznámku.
- **4. krok**
  čtvrtý krok je velmi podobný tomu třetímu, místo e-mailových adres slouží k přidání telefonních čísel.
- **5. krok**
  Pátý krok slouží k přidání adresy trvalého bydliště.

Po potvrzení každého kroku se načítají nové informace do lišty obsahující informace o registrovaném klientovi.

5.2.6 Návštěva

Pohled na návštěvu, který je dostupný pro doktora obsahuje lištu s informacemi o klientovi, jehož se návštěva týká. Dále obsahuje lištu s navigací na čtyři podstatné kroky:

- **Lékařská zpráva**
  Tato komponenta obsahuje textový editor pro zadání lékařské zprávy, text se automaticky ukládá po 1 vteřině od každé úpravy textu. Pokud zpráva není prázdná, je možné tuto zprávu otevřít v PDF (pro snadnější pochopení doktorů je tato akce označena jako tisk).
- **Diagnózy a výkony**
  Tato komponenta umožňuje vyhledat kódy nemocí, přidat diagnózy a k nim vyhledat a přidat provedené výkony.

- **Léky a recepty**
  Komponenta umožňující přidání léků a poté výběr a tisk receptů. Standardně je na každý recept vytisknut pouze jeden lék, je zde ale možnost vybrat druhý lék pomocí tlačítka "2".
- **Dokumenty**
  Tato komponenta je připravena pro možnost přidávat ke každé návštěvě také dokumenty, například fotky z ultrazvuku.

5.2.7 Vyhledávání

Vyhledávání je zobrazeno v horní liště od přihlášení do odhlášení, je stěžejní funkcí pro každou roli. Rozhodl jsem se tedy pro sofistikovanější řešení, které dokáže vyhledat jakéhokoliv klienta, protože aplikace nenabízí možnost volně listovat klienty.

Zadání jednoho slova Při zadání jednoho slova o délce minimálně 3 znaky je vyhledáváno pouze podle příjmení klienta, to má omezit příliš mnoho výsledků vyhledávání, které by způsoboval velký počet opakujících se křestních jmen.

Zadání dvou slov Pokud jsou zadána dvě slova oddělená mezerou, jsou klienti prohledány na základě jména i příjmení, přičemž nezáleží na pořadí slov.

Zadání čísla Při zadání čísla jsou prohledána jak rodná čísla a čísla pojištěnců, tak všechna telefonní čísla klienta, což umožňuje vyhledání klienta ještě před tím, než recepční zvedne telefon.

Zadání čísla a slova Při tomto zadání je číslo vyhledáno opět mezi rodnými čísly, telefonními čísly a čísly pojištěnce. Slovo, které je od čísla odděleno mezerou (nezáleží na pořadí), je vyhledáno mezi příjmeními i jmény.

Kapitola 6

Zabezpečení aplikace

Vzhledem k citlivé povaze dat by měla být každá část aplikace velmi dobře zabezpečena, a to i fyzicky provozem na serveru u ověřeného poskytovatele serverhostingu.

Veškerá komunikace mezi klientskou a serverovou částí by měla probíhat pouze šifrovaně přes zabezpečený HTTPS protokol. Jednotlivé routy (kromě přihlašování) zabezpečeny pouze pro přihlášené uživatele s požadovanou rolí. Ačkoliv to sníží přístupnost celé aplikace, je možné omezit přístup na server pouze pro vybrané IP adresy, což jsem řešil já jako provizorní řešení při nasazení aplikace do provozu v ordinaci.

Tokeny, které serverová část při přihlášení generuje mají aktuálně platnost omezenou časem, zde by bylo vhodné omezit každý token pouze na jedno použití a poté vytvořit jiný. Každý token bude mít tedy platnost vždy jen na jeden dotaz.

Závěr

Ačkoliv bylo nasazení aplikace do reálného provozu kliniky v plánu až po dobrém otestování a proškolení pracovníků, bylo nutné kvůli náhlému ukončení smlouvy mezi klinikou a dosavadním dodavatelem aplikaci spustit dříve i za předpokladu, že budou funkční jen stěžejní části aplikace.

Spuštění do ostrého provozu bylo i přes komplikace provedeno zdárně a po velmi krátkém souběžném provozu je již jedinou aplikací, která podporuje chod tohoto zdravotnického zařízení.

Některé požadavky zadání se mi zatím nepodařilo dopracovat do konečné fáze (například zmíněné zabezpečení), jinými jsem naopak předčil požadavky (databáze léčiv, nemocí, výkonů či import laboratorních výsledků).

Práce na této aplikaci mi rozšířila rozhled nejen v technologiích, které jsem do té doby neznal, ale také rozšířila znalosti do oblastí zdravotnictví, které úzce s programováním souvisí a jsou nezbytné pro práci na aplikaci v tomto odvětví, které považuji za velice zajímavé.

Literatura

[1] What is the Kano Model? [online]. [Cited 2017-11-29]. Dostupné z: https://www.microtool.de/en/what-is-the-kano-model/

[2] Agile: The World's Most Popular Innovation Engine [online]. [Cited 2017-12-06]. Dostupné z: https://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine/

[3] URI Generic Syntax [online]. [Cited 2017-12-05]. Dostupné z: https://tools.ietf.org/html/rfc3986#section-3

[4] What is Vue.js and What are its Advantages [online]. [Cited 2017-12-05]. Dostupné z: https://hackernoon.com/what-is-vue-js-and-what-are-its-advantages-4071b7c7993d

[5] Bootstrap Github [online]. [Cited 2017-12-19]. Dostupné z: https://github.com/twbs/bootstrap

[6] Databáze zdravotních výkonů [online]. [Cited 2017-12-01]. Dostupné z: https://szv.mzcr.cz/

Příloha **A**

Seznam použitých zkratek

HTTPS HyperText Transfer Protocol Secure

HTTP HyperText Transfer Protocol

URL Uniform Resource Locator

TDD Test-driven development

JSON JavaScript Object Notation

MKN Mezinárodní klasifikace nemocí a souvisejících zdravotních problémů

ICD International Classification of Diseases and Related Health Problems

SUKL Státní ústav pro kontrolu léčiv

HTML HyperText Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `server` ..... adresář se serverovou částí aplikace
  - `readme.md` ..... návod na spuštění/sestavení serverové části aplikace
- `client` ..... adresář s klientskou částí aplikace
  - `readme.md` ..... návod na spuštění/sestavení klientské části aplikace
- `text` ..... adresář s textem práce
  - `thesis.pdf` ..... text práce ve formátu PDF