FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Modulární webový rezervační systém pro ordinaci
Student:	Jakub Hamza
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je vyvinout modulární webový rezervační systém usnadňující objednávání pacientů a související analyticko-administrativní činnosti ordinace (lékařské, psychologické, veterinární či jiné).

1. Zmapujte a popište činnosti ordinací různého typu relevantní pro rezervační systém s použitím konceptuálního modelování.
2. Proveďte stručnou rešerši současných řešení.
3. Navrhněte vlastní řešení umožňující efektivní objednávání pacientů ve formě modulární webové aplikace. Jednotlivé moduly budou podporovat dílčí administrativní a analytické činnosti s cílem optimalizace výkonnosti ordinace. Během návrhu zohledněte základní principy a koncepty Normalized Systems Theory.
4. Implementujte a následně otestujte navržené řešení. Zdůvodněte výběr programovacího jazyka i dalších technologií a nástrojů (například webového frameworku).
5. Zhodnoťte ekonomicko-manažerské náklady a přínosy vlastního řešení a porovnejte jej s již existujícími. Ve zhodnocení se zaměřte také na evolvabilitu řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 7. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Modulárny webový rezervačný systém pre ambulancie

Jakub Hamza

Katedra softwarového inženýrství
Vedúci práce: Ing. Marek Suchánek

15. mája 2018

Poďakovanie

V prvom rade by som rád poďakoval svojmu vedúcemu Ing. Marekovi Suchánkovi za pomoc a vedenie pri písaní tejto práce. Ďalej by chcel poďakovať svojej rodine a kamárátom za neustálu podporu. Na záver by som rád poďakoval Ing. Miroslavovi Nottnému za poskytnuté konzultácie ohľadom ASP.NET a IIS.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 15. mája 2018 ……

Odkaz na túto prácu

Hamza, Jakub. *Modulárny webový rezervačný systém pre ambulancie.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cieľom tejto práce je analýza, návrh a implementácia aplikácie, ktorá bude slúžiť ako rezervačný systém na objednávanie pacientov. Aplikácia využíva trojvrstvovú architektúru a je rozdelená do nezávislých modulov podľa funkčnosti. Výstupom je web, ktorý umožňuje užívateľovi rýchlo a efektívne vyhľadať svojho lekára, prípadne ambulanciu a objednať sa na vyšetrenie. Táto aplikácia je napísaná v jazyku C# s využitím ASP.NET MVC Frameworku. Na prácu s databázou je použitý Entity Framework, ktorý je podobne ako ASP.NET produktom firmy Microsoft.

Klíčová slova webová aplikácia, databáza, modularita, rezervačný systém, C#, .NET Framework, MVC 5, Razor, Entity Framework, MS SQL, IIS.

Abstract

The purpose of this thesis is to analyze, design and implement a web application which will serve as a booking system for patients. The application uses a 3-layerd architecture and it is divided into several independent plugins based on their functionalities. Output is a website that allows the user to quickly and efficiently find his doctor or doctor‘s office and to book an appointment. This application is written in language C#, using the ASP.NET MVC Framework. Entity Framework is used for operations with the database. Both Enity Framework and ASP.NET are products of Microsoft.

Keywords web application, database, plugins, C#, .NET Framework, MVC 5, Razor, Entity Framework, MS SQL, IIS.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V dnešnej dobe dostupnosti služieb na internete už vieme takmer všetko vyriešiť z pohodlia domova. Medicína, a zdravotníctvo všeobecne, je však odvetvie, v ktorom tieto služby nie sú až tak bežne dostupné. Každý z nás minimálne raz v živote zažil, keď musel k svojmu lekárovi vstávať skoro ráno, aby bol v čakárni ordinácie ešte pred jej otvorením a získal čo najlepšie poradové číslo, aby nemusel zbytočne čakať na vyšetrenie celý deň, prípadne nebol vyšetrený vôbec. Ďalším príkladom je objednávanie cez telefón. Človek sa snaží dovolať do ambulancie celý deň, no linka je stále obsadená. Keď sa linka konečne uvoľní a ohlási sa sestra, pravdepodobne mu oznámi, že najbližší voľný termín je o pol roka, ale že to môže skúsiť u iného lekára. A celý proces sa opakuje.

Jednou z možností, ako tieto problémy riešiť, je vytvorenie informačného systému vo forme webovej aplikácie. Keďže sa doba neustále posúva vpred, a s ňou aj požiadavky pacientov a lekára, je dôležité, aby sa tieto požiadavky do informačného systému dali jednoducho pridávať a upravovať. Riešením je modulárna architektúra, ktorá nám umožňuje spravovať jednotlivé moduly za behu aplikácie.

V prvej kapitole, si povieme čo je cieľom tejto práce a bližšie si popíšeme, ako by takáto aplikácia mala vyzerať a čo všetko by mala obsahovať a spĺňať.

Analýza bude zameraná na procesy, ktoré v ambulanciach prebiehajú a mohli by byť uľahčené použitím informačného systému. Ďalšia časť analýzy sa zameriava na vybranú vzorku súčasných rezervačných systémov, na ich výhody a nevýhody.

V návrhu si popíšeme jednotlivé prípady užitia systému, princípy Normalized System Theory a architektúru aplikácie.

Kapitola realizácia je rozdelená na implementačnú a testovaciu časť. Implementačná časť detailnejšie popisuje implementované časti aplikácie. Druhá zo spomenutých sa venuje testovacím scenárom a ich výsledkom.

Zhodnotenie aplikácie je kapitola, kde sa pozrieme na ekonomické náklady prevádzky aplikácie a jej manažerské prínosy. Druhou témou tejto kapitoly

bude evolvabilita aplikácie.

Kapitola 1

Cieľ práce

Hlavným cieľom tejto práce je vytvoriť aplikáciu, ktorá bude riešiť problém objednávania pacientov na vyšetrenie a zároveň uľahčí prácu sestrám a lekárom. Mala by byť navrhnutá tak, aby sa v budúcnosti dala jednoducho rozširovať podľa potrieb. Keďže sa aplikácia zameriava na všetky druhy užívateľov (od počítačovo zdatných tínedžerov až po menej zdatných dôchodcov), mala by byť čo najjednoduchšia a užívateľsky prívetivá.

Ďalšie ciele:

- Zmapovanie a popis činností ambulancie pomocou konceptuálneho modelovania, ktoré by informačný systém mohol uľahčovať.
- Stručná rešerš súčasných riešení.
- Zhodnotenie ekonomicko-manažerských nákladov a prínosov.

Kapitola 2

Analýza

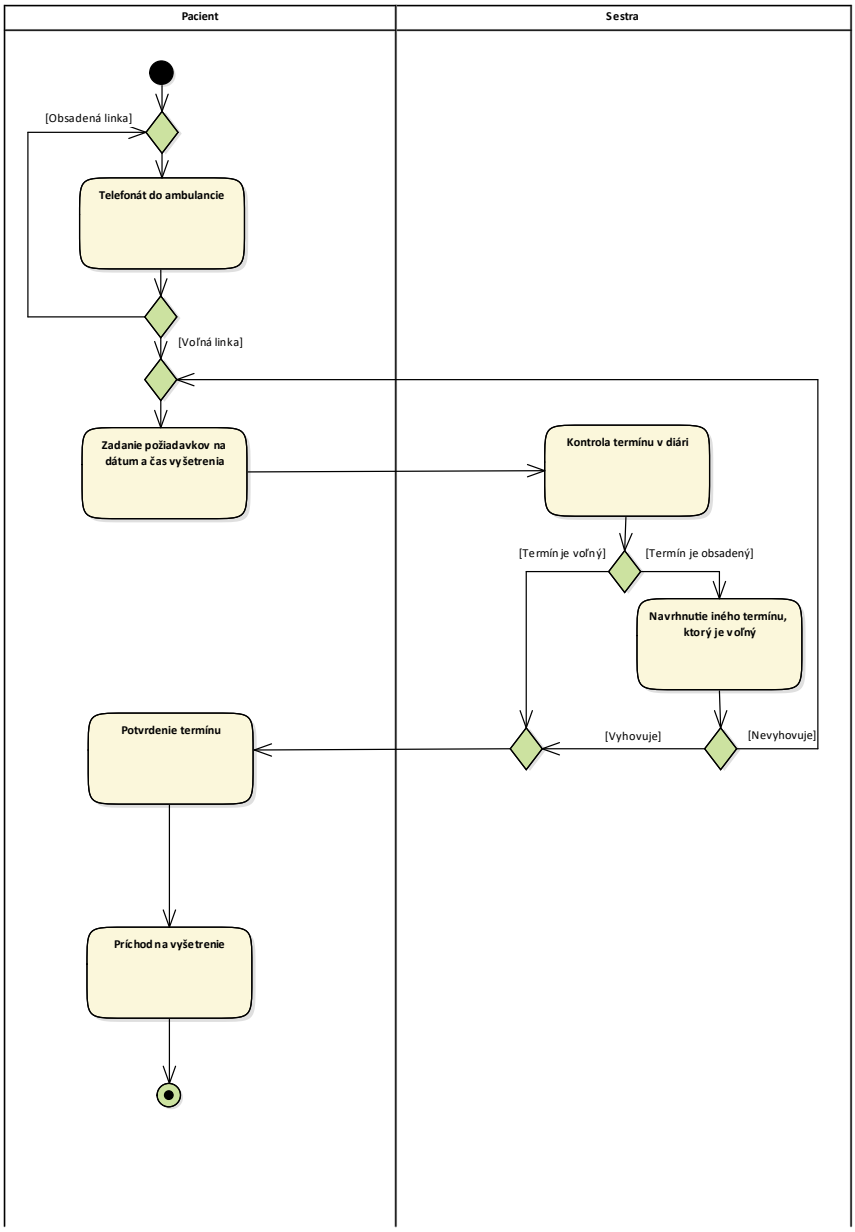
2.1 Analýza činností ambulancie

Táto analýza obsahuje popis procesov, ktoré prebiehajú počas bežného fungovania ambulancie. Každý proces je popísaný podľa toho, ako prebieha v ambulancii teraz a následne, ako by prebiehal, keby bol súčasťou informačného systému.

2.1.1 Objednanie cez telefón

Objednať sa na vyšetrenie je jeden z najhlavnejších procesov, ktoré v ambulancii prebiehajú.

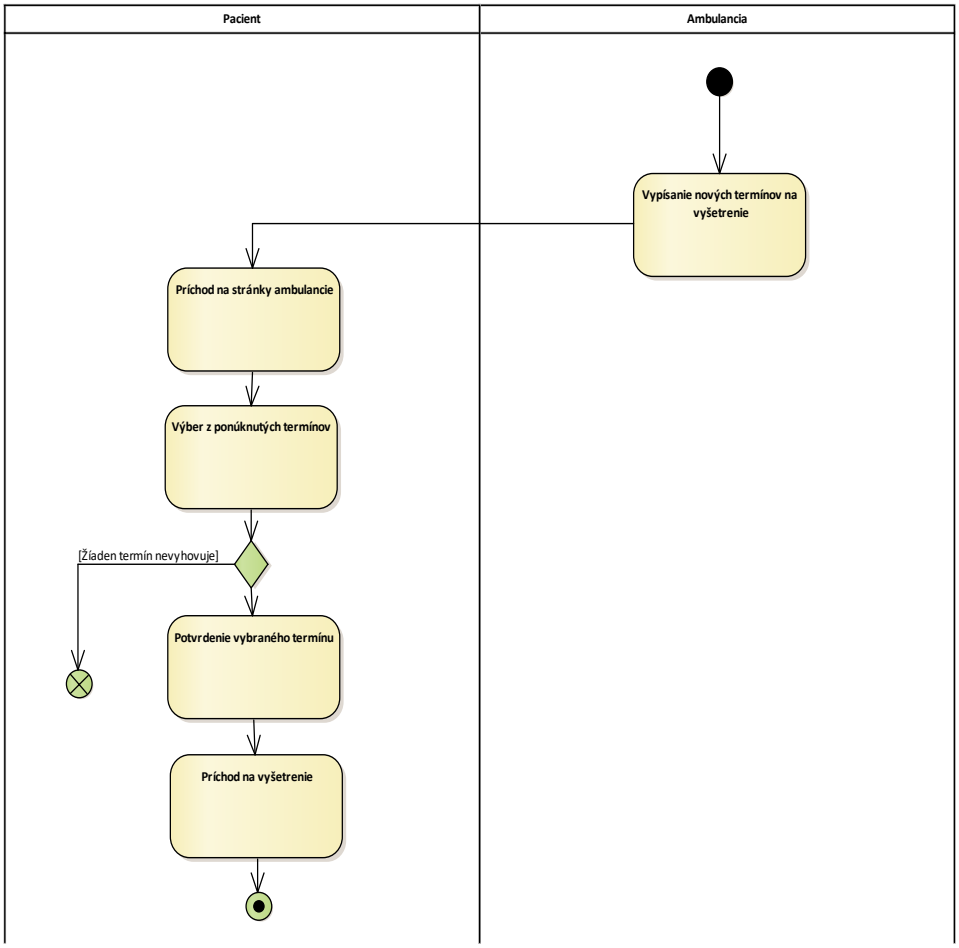
Po tom, ako sa pacient dovolá do ordinácie, snaží sa vybaviť si termín, ktorý mu najviac vyhovuje (napríklad počas dovolenky alebo mimo svoju pracovnú dobu). Bohužial, nie vždy je požadovaný termín voľný a tak mu sestra musí oznámiť, že mu nemôže vyhovieť. Následne mu ponúkne iný termín, ktorý je voľný a podobá sa pôvodnej požiadavke. Ten však pacientovi nemusí vyhovovať a tak povie nový dátum a čas, čím sa celý proces opakuje. Jeden takýto telefonát môže trvať až niekoľko desiatok minút, čo znemožnuje objednať sa ďalším ľuďom, ktorí sa snažia do ambulancie dovolať, no linka je stále obsadená. Takéto objednávanie je preto veľmi zdĺhavé a neefektívnem,ako pre pacienta, tak aj pre ambulanciu.



Obr. 2.1: Objednanie cez telefón (activity diagram)

2.1.2 Objednanie cez IS

Pomocou informačného systému vo forme webebovej aplikácie sa celý proces objednávania dá veľmi uľahčiť. Stačí, aby sestra alebo lekár vypísali nové termíny a pacient sa podľa toho, či mu vyhovujú, prihlási na jeden z nich. Vďaka tomu sa ambulancia zbaví neustále vyzváňajúceho telefónu, na ktorý volajú pacienti ohľadom objednania, a linka tak bude voľná pre iné naliehavé situácie.



Obr. 2.2: Objednanie cez IS (activity diagram)

2.2 Analýza súčasných riešení problému

V analýze sa zameriame na najznámejšie webové aplikácie, ktoré momentálne riešia problém objednávania pacientov na vyšetrenie u lekára v Českej republike a na Slovensku. Pozrieme sa na ich silné stránky, ale zároveň poukážeme aj na chyby, ktorým by bolo vhodné sa v našej aplikácii vyvarovať.

2.2.1 Česká republika

V Českej republike momentálne existuje mnoho webových stránok, ktoré nám ponúkajú možnosť objednať sa online. Väčšina z nich však nie je zameraná na objednávanie ako také, ale slúžia hlavne ako informačné stránky (napr. poliklinikaippavlova.cz), ktoré ako jednu zo služieb pre návštevníkov umožňujú aj online objednávanie k ich lekárom. Analýza tejto práce sa však skôr zameriava na webové portály, ktorých hlavná úloha je objednávanie na vyšetrenie.

2.2.1.1 ZnamyLekar.cz

Najznámejšou českou stránkou, ktorá slúži na objednávanie pacientov na vyšetrenie, je znamylekar.cz. Po načítaní stránok sa nám okamžite naskytne možnosť vyhľadať lekára. Vyhľadávací panel sa skladá z dvoch častí. Do textového poľa sa zadáva meno alebo odbornosť lekára a následne sa v drop-down liste vyberá lokalita, na ktorú chceme zúžiť naše vyhľadávanie. Po kliknutí na tlačidlo *Vyhľadať* nám je ponúknutý zoznam výsledkov, ktoré majú nejakú zhodu so zadaným textom. V tomto zozname môžeme vidieť jednotlivé hodnotenia vyfiltrovaných lekárov a tiež je nám ponúknutý odkaz na jednotlivé recenzie. Po tom, ako si vyberieme konkrétnu ambulanciu, sú nám zobrazené detaily ambulancie (poskytovateľ zdravotnej starostlivosti, ordinačné hodiny, adresa) a možnosť objednať sa na vyšetrenie na jeden z vypísaných voľných termínov.

Silné stránky:

- **Jednoduchý a prehľadný web** – jednotlivé stránky sú navrhnuté tak, aby pacient okamžite našli to, čo potrebujeme.
- **Rýchle a efektívne vyhľadávanie** – dá sa zúžiť na určenú lokalitu.
- **Nenáročnosť objednania** – celý proces, od príchodu na stránky až po objednanie, sa dá zvládnuť na najviac 5 kliknutí.
- **Hodnotenie** – možnosť čítania/pridania názorov na konkrétneho lekára/ambulanciu.
- **Zoznam mojich objednaní** - prehľad návštev a dohodnutých stretnutí s lekárom si pacient vie jednoducho nájsť vo svojom osobnom profile.

Slabé stránky:

- **Online poradenstvo** – jednou zo služieb, ktoré znamylekar.cz ponúka, je možnosť priamo sa spýtať na svoj problém. Myšlienka tejto služby nie je úplne zlá, ale problém nastáva pri jej zaisťovaní. Správca stránok na jednotlivé príspevky odpovedať nemôže, nakoľko nemá potrebnú kvalifikáciu a lekári, ktorí tieto stránky využívajú, nemajú žiadnu povinnosť na tieto príspevky reagovať, je to teda len na ich dobrej vôli. Preto sa môže stať, že na pacientov problém nikto neodpovie včas, prípadne žiadnú odpovede nedostane.

2.2.1.2 NavstevaLekare.cz

NavstevaLekare.cz je najväčší český portál na objednávanie. Úvodná stránka ponúka vyhľadávací panel, kde sa môže zadať meno alebo špecializácia lekára. Ďalšou z ponúk úvodnej stránky je priamy výber konkrétnej špecializácie ambulancie (napr. neurológia) alebo výber jedného z krajov. Či už je zadaný text do vyhľadávacieho panelu alebo sa klikne na kraj/špecializáciu, je zobrazený zoznam výsledkov, kde možno vidieť jednotlivé ambulancie s odkazom na detaily o ambulancii a zároveň je vidieť, či daný lekár umožňuje objednávanie cez internet.

Silné stránky:

- **Nenáročnosť objednania** – celý proces, od príchodu na stránky až po objednanie, sa dá zvládnuť na najviac 5 kliknutí.

- **Nepotrebná registrácia** – aby sa pacient mohol na vyšetrenie objednať, nepotrebuje mať vytvorený účet, stačí len vyplniť formulár, do ktorého zadá telefonné číslo a ďalšie potrebné údaje.

Slabé stránky:

- **Neefektívne vyhľadávanie** – samotná stránka ponúka možnosť vyhľadať lekára podľa odbornosti, mena a kraja, ale nevie tieto možnosti spájať/kombinovať. Takže keď si pacient zvolí napríklad kraj Praha, tak zoznam výsledkov zobrazí všetkých lekárov, ktorí majú svoje ambulancie v tomto kraji, ale ďalej nevie zúžiť výber na konkrétneho špecialistu.

- **Chýbajúci prehľad dohodnutých návštev** – navstevalekare.cz síce umožňuje objednať sa na vyšetrenie bez registrácie, ale zároveň neumožňuje nájsť všetky pacientové vyšetrenia, ktoré má naplánované. Takže človek, ktorý používa tento portál, si alebo musí pamätať všetky svoje plánované vyšetrenia, alebo sa musí prehrabávať vo svojich emailoch, aby našiel, kde všade je vlastne objednaný.

- **Mätúce informácie v detailoch** – okrem základných informácií o ambulancii sa zobrazujú doplnkové informácie o tom, či lekár prijíma nových pacientov, s akými zdravotnými poisťovňami má podpísane zmluvy... Problém však nastáva, ak hodnoty týchto informácií nie sú vyplnené. Tieto informácie nie sú vypisované formou tabuľky, kde jeden stĺpec je názov a druhý je hodnota, ale sú vypisované do riadku, a keď je hodnota nedefinovaná alebo prázdna, tak je vyobrazená mätúca informácia (napr. Noví pacienti Zmluvy s poisťovňami).

2.2.2 Slovensko

Slovensko, na rozdiel od Českej republiky, nemá možnosti objednávania sa cez internet príliš rozvinuté. Danú problematiku momentálne riešia asi len 2 až 3 webové portály.

2.2.2.1 MojLekar.eu

MojLekar.eu je prvým pokusom o online objednávanie pacientov na Slovensku. Na úvodnej stránke sa nachádza vyhľadávací panel, ktorý sa skladá z troch častí. Prvá časť je textové pole, do ktorého pacient môže zadať meno lekára. Druhá a tretia časť sú drop-down listy, v ktorých môže špecifikovať požiadavky vyhľadávania na konkrétnu špecializáciu a mesto. Výsledkom vyhľadávania je zoznam ambulancií so stručnými informáciami a s odkazom na kalendár, kde sa môže pacient objednať na jeden z vypísaných termínov.

Silné stránky:

- **Prvý riešiteľ problému na území Slovenska** – na začiatku veľké množstvo užívateľov, pretože nikto iný neposkytoval podobné služby.
- **Nenáročnosť objednania** – celý proces, od príchodu na stránky až po objednanie, pacient vie vybaviť na najviac 5 kliknutí.

Slabé stránky:

- **Zastaralý a neatraktívny web** – vzhľad stránok sa od ich vytvorenia nezmenil aj napriek tomu, že sa doba výrazne zmenila a dizajn je dôležitým faktorom ovplyvňujúci návštevnosť, ktorý spôsobil stratu záujmu mnohých užívateľov.
- **Veľmi zlé vyhľadávanie** – výsledky vyhľadávania nám nájdu aj lekárov, ktorí síce reálne existujú, ale nemajú s MojLekar.eu nič spoločné a nie sú o nich ani žiadne ďalšie informácie ako ordinačné hodiny, adresa...

- **Zlá správa rolí** – pri registrácii si užívateľ vyberá, či chce vytvoriť účet lekára alebo účet pacienta. Pokiaľ si však vytvorí účet ako lekár a chcel by využiť služby stránky k objednaniu sa k inému špecialistovi, tak mu táto možnosť nebude umožnená. Musí sa odhlásiť a prihlásiť sa z iného účtu, ktorý bol vytvorený ako pacient. Až potom mu systém umožní objednať sa.

2.2.2.2 ObjednatVysetrenie.sk

ObjednatVysetrenie.sk je súčasť portálu e-VÚC, ktorý slúži k objednávaniu pacientov. Úvodná stránka obsahuje okrem vrchného menu, ktoré je spoločné pre všetky stránky, len vyhľadávací panel a jednoduchý návod, ako sa objednať. Po vyhľadaní textu, ktorý sa zadal, sa zobrazí zoznam výsledkov. Ak pacient zadal mesto alebo nenapísal žiaden text, tak sa mu zobrazia všetci poskytovatelia zdravotnej starostlivosti s ich ambulanciami v danom meste. V ostatných prípadoch je výsledkom len zoznam ambulancií, ktoré sú nejakým spôsobom spojené s hľadaným textom. V oboch prípadoch je ponúknutý prechod na detaily o jednotlivých ambulanciách. V detailoch sú zobrazené informácie a ak je umožnené objednávanie online, tak je zobrazený kalendár s vypísanými termínmi, na ktoré sa môže pacient prihlásiť. V opačnom prípade je mu oznámené, že lekár takéto objednávanie neumožňuje.

Silné stránky:

- **Atraktívny dizajn** – webové stránky sú navrhnuté veľmi moderne a zároveň jednoducho. Potrebné veci vie pacient veľmi ľahko nájsť a mnohokrát je hneď ponúknutý návod, ako na to.

- **Nenáročnosť objednania** – celý proces, od príchodu na stránky až po objednanie, sa dá vybaviť na najviac 5 kliknutí.

- **Nepotrebná registrácia** – aby sa pacient mohol na vyšetrenie objednať, nepotrebuje mať vytvorený účet, stačí len vyplniť formulár, do ktorého zadá svoje telefónné číslo a ďalšie údaje o sebe.

- **Služby pre lekárov aj pacientov sú zadarmo** – ide o projekt samosprávnych krajov Slovenska, ktoré tieto služby poskytujú pre svojich občanov bez poplatku.

- **Aktuálnosť** – poskytované informácie sú priamo z registra zdravotníctva.

- **Pripomenutie termínu** – aby sa pacient nezabudol dostaviť na vyšetrenie, dostane pripomienku deň vopred formou SMS.

Slabé stránky:

- **Obmedzené pole užívateľov** – ObjednatVysetrenie.sk je projektom VÚC a je zameraný len na lekárov, ktorí pracujú v Žilinskom alebo Trenčianskom samosprávnom kraji.
- **Rušenie objednania** – podobne ako u NavstevaLekare.cz ani tu nie je potrebné byť prihlásený na to, aby sa pacient u lekára mohol objednať. Ak sa chce objednať, stačí si vybrať termín, vyplniť potrebné údaje a zadať svoje telefónne číslo, aby mu prišla správa s číslom jeho objednania a s potvrdzovacím kódom, ktorého zadaním dokončí celý proces. Problémy nastávajú, keď si chce pacient svoje vyšetrenie zrušiť. Systém od neho požaduje len číslo objednania. Toto číslo je však obyčajná n-tica čísel, pravdepodobne integer a má nastavený autoincrement +1, a je ľahko uhádnuteľná. Po zadaní tohto čísla, ak je vyšetrenie ešte aktuálne, sa pacientovi zobrazia detailnejšie informácie vyšetrenia s možnosťou zaslania potvrdzovacieho kódu na zrušenie objednávky. Problém je, že nie je ošetrené zasielanie kódu na zrušenie. Zasielanie správy s týmto kódom nie je obmedzené ani časom ani počtom odoslaní. Takže ak sa niekomu podarí odhadnúť číslo objednania, vie pacienta zasypať správami o zrušení termínu vyšetrenia a on nebude vedieť, čo sa vlastne deje.

2.3 Doménový model

Doménový model je jeden zo spôsobov, ako zachytiť entity reálneho sveta a ich vzťahy, ktoré kolektívne opisujú problémovú doménu. Je zameraný len na zobrazenie základnej štruktúry aplikácie a teda neobsahuje objekty súvisiace so samotnou aplikáciou. Triedy tohoto modelu sú odľahčené, neobsahujú metódy ani dátové typy.

2.3.1 Ambulancia

Trieda reprezentuje skutočnú ambulanciu a uchováva jej meno, adresu, zameranie a kontakt.

2.3.2 Užívateľ

Predstavuje skutočnú osobu. Trieda uchováva prihlasovacie meno, email a rolu.

2.3.3 Vyšetrenie

Dátum a čas dohodnutého vyšetrenia.

2.3.4 Výsledky vyšetrenia

Lekársky nález, ktorý bol počas vyšetrenia zistený.

2.3.5 Lekár

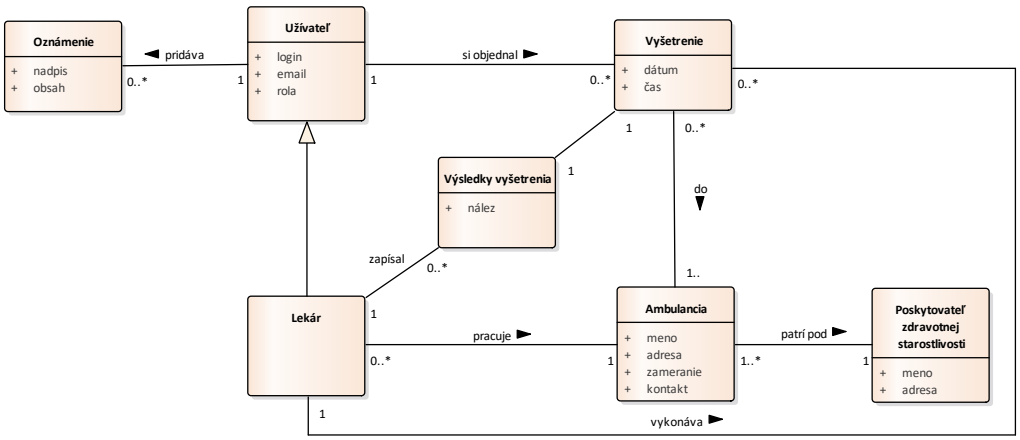
Osoba, ktorá pracuje v ambulancii, vykonáva vyšetrenia a zapisuje výsledky.

2.3.6 Poskytovateľ zdravotnej starostlivosti

Fyzická osoba alebo právnická osoba, ktorá poskytuje zdravotnú starostlivosť.

2.3.7 Oznámenie

Trieda reprezentuje oznámenie a uchováva nadpis a obsah.



Obr. 2.3: Doménovy model (class diagram)

2.4 Požiadavky

Analýza požiadaviek slúži k získaniu predstavy o tom, čo má systém obsahovať, spĺňať a ponúkať. Ako užívateľ systému pre túto prácu bol identifikovaný lekár, pacient, sestra a administrátor. Primárne je aplikácia určená pre lekárov a pacientov. Prebehlo viacero stretnutí s MUDr. Mariana Hamzová (Neurim, s.r.o.) a MUDr. Margita Grambličková (MEDIING, s. r. o.), kde ako lekárky s dlhoročnými skúsenosťami argumentovali svoje požiadavky na systém. Rola sestry bola konzultovaná s Bc. Simona Dolníková, ktorá pracuje ako zdravotná sestra v Žilinskej fakultnej nemocnici. Pacienta a administrátora som zastupoval ja sám. Výsledok týchto konzultácii viedol k vytvoreniu funkčných a nefunkčných požiadaviek.

2.4.1 Funkčné požiadavky

Definujú, čo má systém vedieť a poskytovať.

2.4.1.1 Vyhľadávanie lekárov a ambulancií

Systém bude umožňovať vyhľadať akéhokoľvek lekára alebo ambulanciu, ak využíva služby aplikácie. Pokiaľ si užívateľ nie je úplne istý celým menom, stačí zadať len časť názvu/mena. Po úspešnom vyhľadaní musí systém umožňovať zobrazenie detailnejších informácií.

2.4.1.2 Registrácia

Systém bude umožňovať vytvorenie účtu pre nového užívateľa. Užívateľ si bude musieť zvoliť vhodné meno, bezpečné heslo a zadať svoj email.

2.4.1.3 Prihlásenie do systému

Anonymnému užívateľovi budú sprístupnené len základné možnosti. K zisku ďalších bude potrebné prihlásenie.

2.4.1.4 Správa oznámení

Lekárovi systém umožní pridávať nové oznámenia a zároveň upravovať a mazať tie, ktoré už raz pridal. Administrátor ako správca systému bude môcť okrem pridávania upravovať a mazať akékoľvek oznámenia.

2.4.1.5 Čítanie oznámení

Systém umožní čítanie všetkých oznámení o prípadných dovolenkách, zmenách či prípadných odstávkach serveru.

2.4.1.6 Správa ambulancie

Lekár bude môcť upravovať údaje o svojej ambulancii. Administrátorovi budú udelené práva pre spravovanie všetkých.

2.4.1.7 Online objednávanie

Pokiaľ ambulancia poskytuje službu online rezerváciu vyšetrení, systém bude musieť umožniť užívateľovi prihlásiť sa na voľný termín. Zároveň musí kontrolovať, aby sa na jeden termín neprihlásilo viacero ľudí.

2.4.1.8 Prehľad objednaní

Pokiaľ má užívateľ dohodnuté vyšetrenia alebo nejaké už absolvoval, musí si ich vedieť zobraziť.

2.4.1.9 Správa obsahu a užívateľov

Administrátorova úloha má byť spravovanie aplikácie. Preto musí mať práva k jej správe.

2.4.2 Nefunkčné požiadavky

Ide o požiadavky, ktoré priamo nesúvisia s funkčnosťou systému, ale napriek tomu sú pre systém dôležité.

2.4.2.1 Modularita

Systém musí byť rozdelený podľa funkčnosti do jednotlivých modulov a musí byť ľahko rozšíriteľný/upraviteľný.

2.4.2.2 Webová aplikácia

Systém musí fungovať ako webová aplikácia a musí poskytovať plnú funkčnosť pre prehliadače Google Chrome 66.0.3359.139 a Mozilla Firefox 59.0.2.

2.4.2.3 Užívateľská prívetivosť

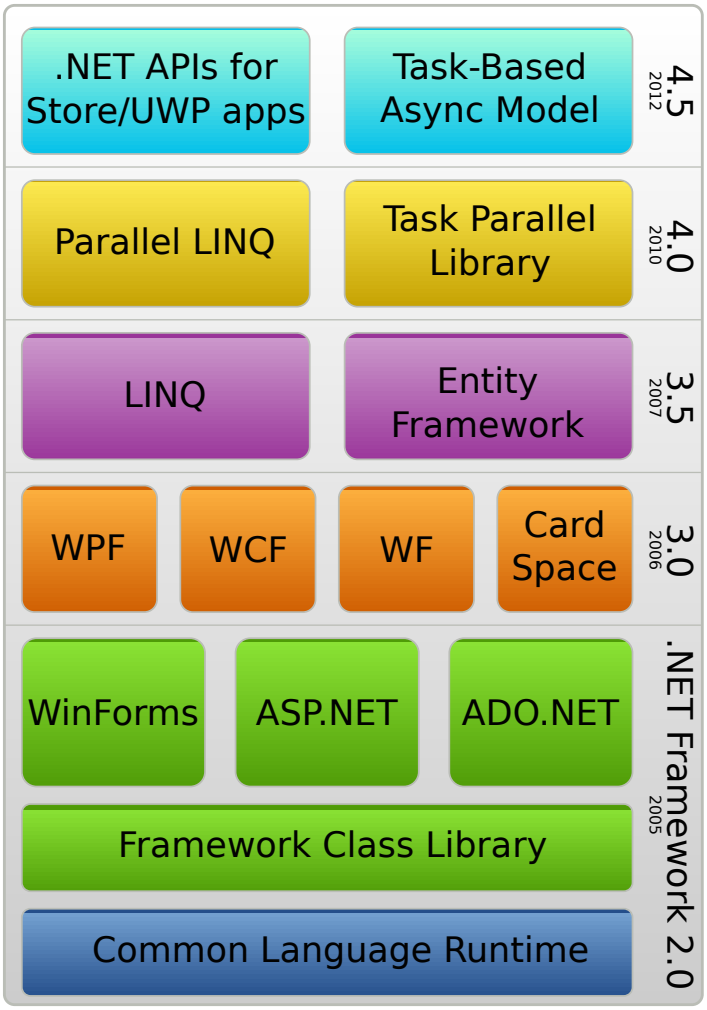
Systém musí byť navrhnutý tak, aby užívateľ mohol jednoducho nájsť čokoľvek, čo potrebuje. Zároveň musí byť navrhnutý tak, aby svojím dizajnom neodrádzal užívateľov od jeho používania.

2.5 Použité technológie

Technológií, ktoré by mohli byť použité na tvorbu a prevádzku webovej aplikácie, existuje mnoho. Najznámejšia sada nástrojov je LAMP (operačný systém Linux, webový server Apache, databázový server MySQL, skriptovací jazyk PHP alebo Python). Pre účely tejto práce sú použité nástroje od firmy Microsoft a to konkrétne Windows OS, webový server IIS, Microsoft SQL Server a programovací jazyk C#.

2.5.1 .NET

.NET je softwarový framework, vyvinutý spoločnosťou Microsoft. Obsahuje veľkú knižnicu s názvom Framework Class Library (FCL) a poskytuje jazykovú interoperabilitu (každý jazyk môže používať kód napísaný v iných jazykoch). Programy napísané pre .NET Framework sa spúšťajú v softwarovom prostredí s názvom Common Language Runtime (CLR). Ide o virtuálny stroj aplikácie, ktorý poskytuje služby ako zabezpečenie, správa pamäte a spracovanie výnimiek. FCL spoločne s CLR tvoria základ tohoto frameworku.[15]



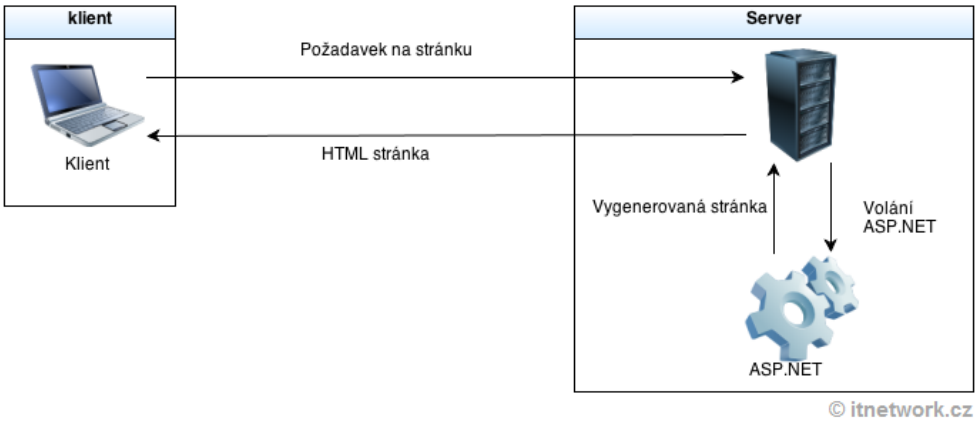
Obr. 2.4: Komponenty .NET Frameworku

[12]

2.5.2 ASP.NET

ASP.NET je súčasť .NET frameworku, vytvorený na tvorbu webových služieb a aplikácií. Je nástupcom ASP (Active Server Pages) a jeho knižnice poskytujú riešenia mnohých problémov, ktoré pri vývoji webových stránok nastávajú (bezpečnosť, autentifikácia, práca s databázou, správa formulárov, atď).

Technológia je založená na architektúre klient-server. ASP.NET teda beží na serveri, kde prijíma požiadavky od jednotlivých klientov. Na ich základe im vracia výsledky vo forme HTML stránok.[15]

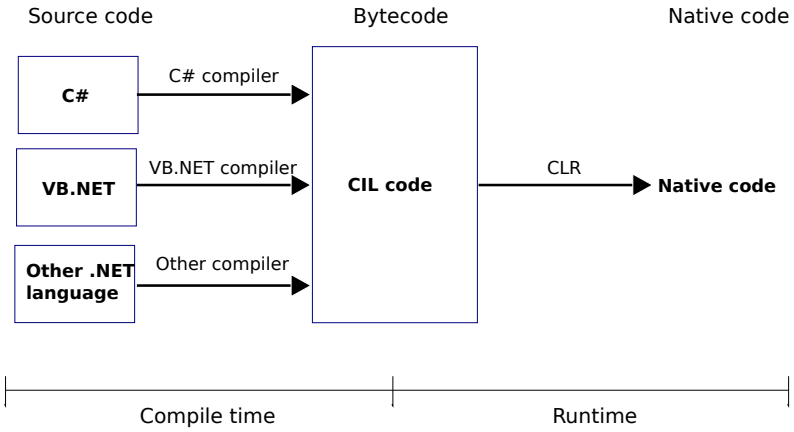


Obr. 2.5: ASP.NET klient-server model

[3]

2.5.3 C#

C# je silno-typový, objektovo-orientovaný programovací jazyk vyvinutý spoločnosťou Microsoft. Podobne ako Java, patrí medzi jazyky s virtuálnym strojom. Zdrojový kód sa najskôr preloží do CIL (Common Intermediate Language), ktorý slúži ako medzikód. Jeho úlohou je zjednodušenie inštrukcií, ktoré sú následne prevedené CLR (Common Language Runtime) na strojový kód. Programy napísané v tomto jazyku teda pobežia na akomkoľvek zariadení, na ktorom sa nachádza virtuálny stroj.[15]



Obr. 2.6: CLR Exexution Model

[5]

2.5.4 Microsoft SQL Server

Microsoft SQL Server patrí do skupiny RDBMS (**r**elational **d**ata**b**ase **m**anagement **s**ystem). Bol navrhnutý pre zvládnutie veľkého množstva operácií, preto je vhodný napríklad na online objednávky, účtovníctvo či výrobu. Ako jazyk používa T-SQL (Transact-SQL), ktorý je verziou jazyka SQL, navrhnutý Microsoftom.

Od verzie Microsoft SQL Server 2005 obsahuje komponentu s názvom SQL CLR ("Common Language Runtime"), prostredníctvom ktorej sa integruje s .NET Frameworkom. Na rozdiel od väčšiny iných aplikácií, ktoré používajú tento framework, SQL Server samotný, spúšťa jeho runtime. Požiadavky na správu pamäte, vlákien a riadenia zdrojov teda nie sú spúšťané operačným systémom, ale systémom SQLOS. Vďaka nemu môžu byť uložené procedúry a triggery napísané v ľubovoľnom jazyku .NET ako napríklad C# alebo Visual Basics. Spravovaný kód môže byť tiež použitý na definovanie UDT (user-defined types), ktoré môžu pretrvávať v databáze. Spravovaný kód je zostavený do zostáv CLI (Command Line Interface) a po overení pre bezpečnosť typu uložený v databázi.[11]

Pri písaní kódu pre jazyk SQL CLR sa dáta uložené v databázach SQL Server sprístupňujú pomocou ADO.NETu.

2.5.5 ADO.NET

ADO.NET (**A**ctiveX **D**ata **O**bjects for **.NET**) je technológia, ktorá zabezpečuje komunikáciu medzi relačnými a nerelačnými systémami prostredníctvom spoločného súboru komponentov. Tieto komponenty umožňujú prístup k údajom a dátovým službám databázy. Hlavným využitím ADO.NETu je prístup a úprava údajov uložených v relačných databázach.[2]

2.5.6 Entity Framework

Entity Framework je open-source, ORM (object-relational mapping) framework pre ADO.NET. Ide o súbor technológií, ktoré podporujú vývoj dátovo orientovaných softwarových aplikácií. Entity Framework umožňuje vývojárom pracovať s údajmi vo forme objektov a vlastností špecifických pre doménu bez toho, aby sa museli zaoberať ich databázovými tabuľkami a stĺpcami, v ktorých sú tieto údaje uložené. Jeho výhodou je, že umožňuje vyššiu úroveň abstrakcie pri práci s dátami a tak znižuje množstvo potrebného kódu.[9]

2.5.6.1 LINQ to Entities

LINQ to Entities je súčasťou Entity Frameworku a slúži ako API(application programming interface), ktoré umožňuje širokú definíciu modelov doménových objektov a ich vzťahov k mnohým rôznym poskytovateľom dát. Umožňuje kombinovať množstvo rôznych dodávateľov databáz, aplikačných serverov

alebo protokolov a navrhnúť agregované zmiešanie objektov, ktoré je konštruované z rôznych tabuliek, zdrojov a služieb.[10]

2.5.7 IIS

IIS (Internet Information Services alebo Internet Information Server) je softwarový webový server s kolekciou rozširujúcich modulov, vytvorený spoločnosťou Microsoft pre operačné systémy Windows. V dnešnej dobe zaujíma druhé miesto v celkovom používaní serverov za Apache HTTP Server na internete. Používa sa hlavne na podporu ASP.NET, pretože žiadna z konkurencie ho nepodporuje.[13]

2.5.7.1 IIS Express

Služba IIS Express je odľahčená a samostatná verzia IIS optimalizovaná pre vývojárov. Táto služba uľahčuje používanie najnovšej verzie služby IIS na vývoj a testovanie webových stránok. Má všetky základné funkcie služby IIS 7 a vyššie, plus obsahuje funkcie určené na uľahčenie vývoja webových stránok.[7]

2.5.8 jQuery

jQuery je knižnica JavaScriptu určená na zjednodušenie skriptovania HTML na strane klienta. Je to bezplatný, open-source software pod licenciou MIT. Táto knižnica bola vydaná Johnom Resigom v januári 2006 na newyorskom BarCampu.[1]

Syntax jQuery je navrhnutá tak, aby uľahčila navigáciu dokumentu, vytváranie animácií, spracovanie udalostí a vývoj aplikácií Ajax. jQuery tiež poskytuje vývojárom schopnosť vytvárať doplnky nad knižnicou JavaScript. Modulárny prístup ku knižnici jQuery umožňuje vytváranie výkonných dynamických webových stránok a webových aplikácií.

Kapitola 3

Návrh

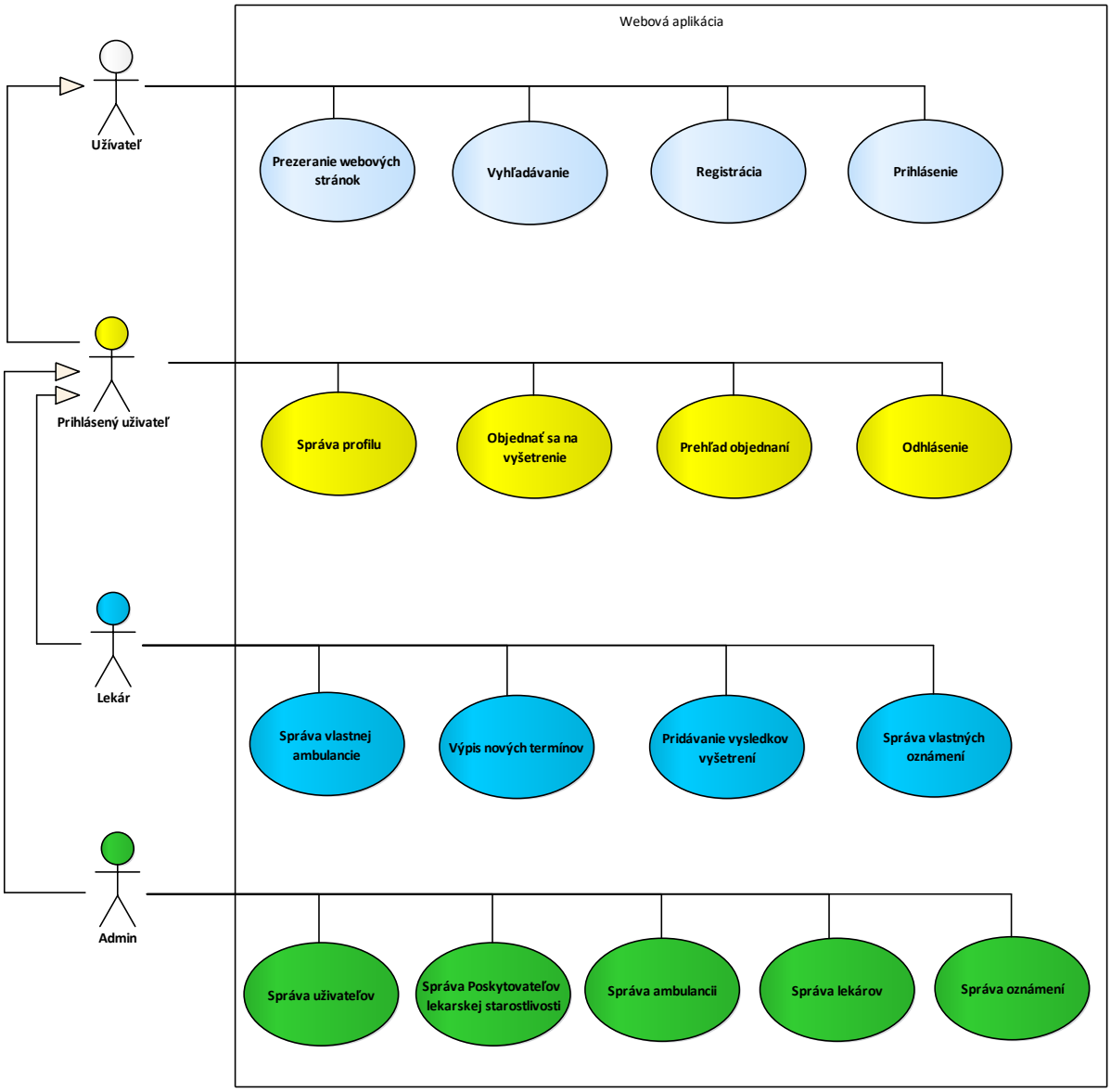
3.1 Prípady užitia systému

Dôležitou časťou každého návrhu aplikácie sú prípady užitia. Ide v podstate o jednotlivé funkcie, ktoré bude systém poskytovať svojím užívateľom. Preto je dôležité identifikovať všetkých užívateľov systému a rozdeliť funkcionalitu systému podľa ich rolí.

Užívateľské role:

- **Užívateľ**
- **Prihlásený užívateľ**
- **Sestra** (nie je súčasťou tejto práce)
- **Lekár**
- **Admin**

Užívateľ je bežný návštevník webových stránok a disponuje základnými prípadmi užitia systému (napr. prezeranie webových stránok alebo vyhľadávanie ambulancií). Ostatní užívatelia sú nadstavbou tejto role, teda využívajú okrem základných aj funkcionality potrebné pre svoju úlohu v rámci systému.



Obr. 3.1: Prípady užitia systému (usecase diagram)

3.1.1 Užívateľ

3.1.1.1 Prezeranie webových stránok

Umožňuje navštíviť webové stránky a povoľuje prístup ku všetkým verejným informáciám, oznámeniam a textom, ktoré sú ich obsahom.

3.1.1.2 Vyhľadávanie

Sprístupnením vyhľadávacieho panelu umožňuje vyhľadať lekára, ambulanciu, zdravotné stredisko alebo poskytovateľa zdravotnej starostlivosti. Ak text zadaný do tohoto panelu neobsahuje aspoň 4 znaky, je zobrazená chybová hláška a následne sú zobrazené všetky lekárske ambulancie, ktoré momentálne využívajú služby aplikácie. V prípade, že text spĺňa požadovanú dĺžku, sú zobrazené všetky výsledky, ktoré tento výraz obsahujú.

3.1.1.3 Registrácia

Ponúka možnosť vytvorenia nového účtu. Po spustení procesu registrácie sa zobrazí formulár umožňujúci vyplniť údaje potrebné k registrácii (prihlasovacie meno, heslo, email), ale aj doplnkové údaje ako meno, priezvisko, telefónne číslo, národnosť a pohlavie. Po vyplnení týchto údajov systém skontroluje, či majú správny formát (napr. minimálna dĺžka hesla). Ak je všetko v poriadku, je odoslaná požiadavka na vytvorenie nového účtu. Aby mohol byť účet vytvorený, prihlasovacie meno a email musia byť unikátne. Systém preto skontroluje, či sa už zadané meno alebo email v databáze užívateľov nenachádza. Ak sa nenašla žiadna zhoda, nový účet je vytvorený a na zadaný email je odoslaná správa s validačným kódom. Potvrdením odkazu v tejto správe je registrácia úspešne dokončená.

3.1.1.4 Prihlásenie

Umožňuje užívateľovi prihlásiť sa a tak sprístupniť ďalšie funkcionality systému, ktoré nie sú bez autorizácie prístupné. Po stlačení tlačidla *Prihlásiť* systém zobrazí stránku s prihlasovacím formulárom. Obsahuje len textové pole na vyplnenie prihlasovacieho mena a hesla. Ak si užívateľ svoje heslo nepamätá, má možnosť zaslať si správu s odkazom na vytvorenie nového hesla. Po odoslaní žiadosti o prihlásenie systém skontroluje, či uživateľske meno existuje a či je zadané heslo správne. Ak je prihlásenie úspešné, ale užívateľ nemá potvrdený email, je mu zobrazené upozornenie, že pre sprístupnenie ďalších služieb je potrebné tento účet validovať kliknutím na odkaz, ktorý bol odoslaný na jeho email pri registrácii. Ak je účet validovaný, po prihlásení je užívateľ presmerovaný na domovské stránky.

3.1.2 Prihlásený užívateľ

3.1.2.1 Správa profilu

Ak užívateľ pri registrácii doplnkové údaje nevyplnil alebo sa odvtedy zmenili, má možnosť upraviť ich podľa potreby vo svojom profile.

3.1.2.2 Objednať sa na vyšetrenie

Ak sa užívateľ chce objednať na vyšetrenie a jeho lekár umožňuje objednávanie online, stačí, aby prešiel na detaily ambulancie a z kalendára si vybral voľný termín, ktorý mu vyhovuje.

3.1.2.3 Prehľad objednaní

Okrem zmeny údajov si vie užívateľ vo svojom profile zobraziť všetky objednania, ktoré ho v budúcnosti čakajú a vie si aj pozrieť zápis od lekára, ku vyšetreniam, ktoré už absolvoval.

3.1.2.4 Odhlásenie

Umožní užívateľovi odhlásiť sa.

3.1.3 Lekár

3.1.3.1 Správa vlastnej ambulancie

Umožňuje lekárovi upravovať informácie o svojej ambulancii (kontakt, adresu, pracovnú dobu, zameranie, popis, objednávanie cez webovú aplikáciu).

3.1.3.2 Výpis nových termínov

Ak má lekár povolené objednávanie pacientov online, táto funkcionalita slúži k vypísaniu nových voľných termínov, na ktoré sa pacienti môžu prihlásiť.

3.1.3.3 Pridávanie výsledkov vyšetrení

Po vyšetrení pacienta, ktorý bol objednaný pomocou navrhnutého systému, môže lekár zapísať výsledky nálezu do jeho osobnej zložky. Lekár, ktorý neskôr vyšetruje tohoto pacienta, si vie tento záznam jednoducho pozrieť a nemusí ho hľadať v zdravotnej karte.

3.1.3.4 Správa vlastných oznámení

Umožňuje lekárovi pridávať a upravovať vlastné oznámenia (zástupy, dovolenky, zmena adresy, atď.).

3.1.4 Admin

3.1.4.1 Správa užívateľov

Administrátorovi dáva prístup k zobrazeniu a úprave údajov užívateľov, prípadne k ich zmazaniu. Ak chce upraviť údaje, systém mu zobrazí formulár s aktuálnymi informáciami, ktoré sú editovateľné. Po vykonaní zmien a odoslaní žiadosti k ich aktualizácii, systém skontroluje, či nedošlo k nejakým chybám. Ak je všetko v poriadku, dáta sú aktualizované a administrátor je presmerovaný na hlavnú stránku správy užívateľov. V prípade chyby sa zobrazí chybová hláška.

3.1.4.2 Správa poskytovateľov lekárskej starostlivosti

Umožňuje administrátorovi vytvoriť, zobraziť údaje, upraviť poskytovateľov lekárskej starostlivosti, prípadne ich zmazať. Ak chce údaje upraviť, systém postupuje rovnako ako u správy užívateľov. V prípade, že chce poskytovateľa zmazať, mal by byť veľmi opatrný, pretože jeho zmazaním odstráni okrem vybraného poskytovateľa aj všetky jeho ambulancie a lekárov, ktorí v nich pracujú.

3.1.4.3 Správa ambulancií

Administrátor môže pridávať, zobrazovať a upravovať lekárov. Pri vytváraní ambulancie však administrátor musí zabezpečiť, aby poskytovateľ zdravotnej starostlivosti, pod ktorého ambulancia spadá, už v databáze bol. Pri vypĺňaní položky poskytovateľa systém totiž ponúka len drop-down list existujúcich. Pri mazaní ambulancie sa zmažu aj všetci lekári, ktorý v nej pracujú.

3.1.4.4 Správa lekárov

Systém poskytuje možnosť správy lekárov. Podobne ako u vytvárania novej ambulancie, tak aj u nového lekára administrátor musí zaručiť, že ambulancia, v ktorej má pracovať, už v systéme existuje.

3.1.4.5 Správa oznámení

Umožňuje pridávať, zobrazovať, upravovať a mazať oznámenia.

3.2 Normalized System Theory

Pojem – Normalized System Theory (Teória Normalizovaných Systémov) – popisuje spôsob vývoja agilných a evolvability-schopných softwarov, definovaním poučiek a návrhových vzorov pre architektúru aplikácií.

Tvrdí, že aplikácia by sa mala neustále meniť podľa aktuálnych požiadaviek. V teórií chápe evolvabilitu (vývojaschopnosť) ako absenciu kombinačného efektu. Tento efekt je zadefinovaný ako: *„a change of which the impact is not solely related to the kind of the change, but also to the size of the system it is applied on"*[14, s. 4761], čo vo voľnom preklade znamená, že je to zmena, ktorej vplyv nie je iba o type zmeny, ale aj o veľkosti systému, na ktorej je vykonaná.

Predpokladom je, že software v priebehu času podlieha neustálemu vývoju (dodatočné a zmenené požiadavky na systém) a kombinačný efekt má v značnej miere nepriaznivý vplyv na tento vývoj.

Teoretickým základom argumentácie je koncept stability systému zo systémovej teórie, ktorý hovorí, že pre ohraničený vstup (požiadavky) by mal byť aj ohraničený výstup (zmeny v aplikácii).[14]

Opiera sa o 4 základné princípy:

- **Separation of Concerns (SoC)** – rozdelenie aplikácie na rôzne časti, ktoré sa z hľadiska funkcionality prekrývajú čo najmenej.
- **Data Version Transparency (DvT)** – každá dátova entita môže byť aktualizovaná bez ovplyvnenia entít, ktoré ju používajú ako vstupný alebo výstupný parameter.
- **Action Version Transparency (AvT)** – action entita (entita, ktorá poskytuje nejaké akcie) musí mať možnosť vylepšenia bez ovplyvnenia komponentov, ktoré ju volajú.
- **Separation of States (SoS)** – každý krok v pracovnom procese je včas oddelený a uchovaný.

3.3 Architektúra

Každá aplikácia má svoju štruktúru. Pre tento projekt je vhodné použiť trojvrstvovú architektúru (Three-tier architecture). Ide o klient-server architektonický vzor, ktorý rozdeľuje aplikáciu do troch vrstiev (prezentačná, business, dátová). Bol vyvinutý Johnom J. Donovanom v Open Environment Corporation (OEC). [6]

Rozdelením aplikácie na vrstvy sa využíva prvý z princípov Normalized System Theory, konkrétne Separation of Concerns. Každá z týchto vrstiev predstavuje nezávislý modul alebo skupinu modulov.

3.3.1 Dátová vrstva

Vrstva je zložená z dátového skladu (databázový server, obyčajný súbor(file)), objektovo-relačným maperom a od neho odvodených tried.

3.3.2 Business vrstva

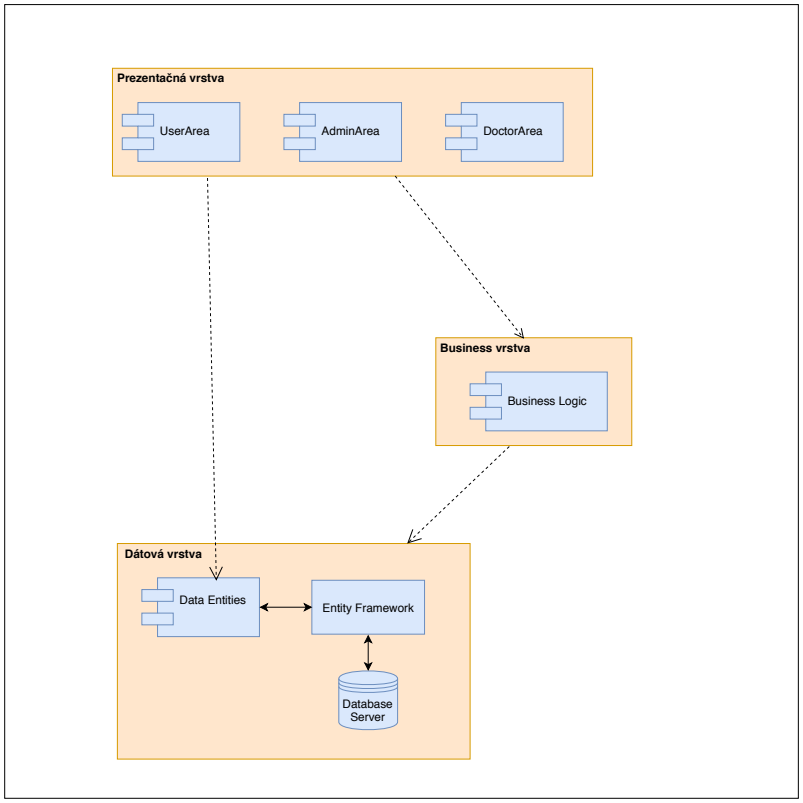
Úlohou tejto vrstvy je starať sa o funkčné požiadavky systému. Tvorí jadro celej aplikácie, pretože práve tu prebieha celá logika.

3.3.3 Prezentačná vrstva

Vrstva určená pre komunikáciu medzi užívateľom a systémom. Jej úlohou je zobrazenie dát do užívateľského rozhrania (v našom prípade webová stránka).

3.4 Moduly

DLL je skratka pre anglický výraz *„dynamic-link library"*. Ide o implementáciu konceptu zdieľanej knižnice pre operačné systémy Microsoft Windows. Poskytuje štandardné výhody zdieľaných knižníc, ako je modularita. Tá umožňuje vykonávať zmeny kódu a dát, v jedinej samostatnej knižnici DLL, zdieľanej niekoľkými aplikáciami alebo inými knižnicami, bez vplyvu na ne. Ide o využitie princípov Data a Action Version Transparency z Normalized System Theory. [8]



Obr. 3.2: Relaxovaná trojvrstvová architektúra

Moduly jednotlivých vrstiev:

- **Dátová vrstva** – tvorená modulom DataEntities, ktorý bude obsahovať entitné triedy.
- **Business vrstva** – modul Business Logic, ktorý obsahuje všetky action entity (triedy popisujúce logiku aplikácie).
- **Prezentačná vrstva** – obsahuje moduly UserArea, AdminArea a DoctorArea. Moduly su navrhnuté pre užívateľov, ktorí ich budú používať, čo môže vyplývať už aj z ich názvov.

Kapitola 4

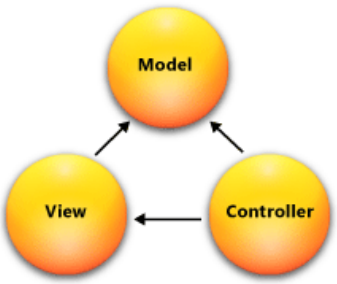
Realizácia

4.1 Implementácia

Aplikácia je vyvíjaná v ASP.NET frameworku a ten nám ponúka 2 možnosti.

Prvou možnosťou je WebForms. Ide o pokus prenesenia štandardných formulárových aplikácií, ktoré poznáme z desktopu, na web. Hlavnou myšlienkou WebForms je v dizajnéri vytvoriť formulár s objektami, ktorým sú pridané rôzne eventy.

Druhou z možností je MVC (Model-View-Controller). Je to novší koncept a rozdeľuje aplikáciu na tri nezávislé komponenty, ktoré vyplývajú z názvu. Hlavnou výhodou je, že modifikácia jednej z nich má minimálny vplyv na ostatné. Pre túto prácu bolo zvolené MVC, kvôli odľahčenosti a jednoduchej testovateľnosti.



Obr. 4.1: Návrhový vzor MVC

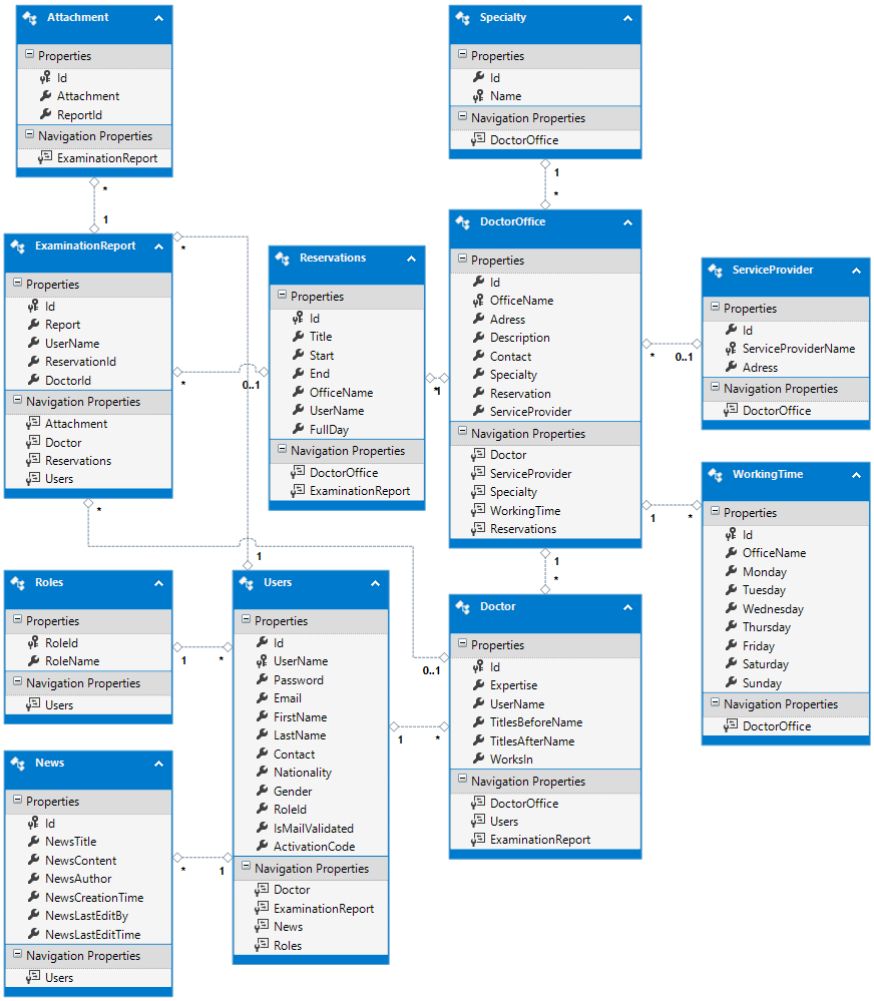
[4]

4.1.1 Dátová vrstva

Dátovú vrstvu tvorí databáza Microsoft SQL Server 2016 a entitné triedy, na ktoré sú dáta z nej mapované pomocou Entity Frameworku. Pre každú tabuľku

databázy existuje práve jedná entitna trieda, ktorej atribúty odpovedajú jej stĺpcom. Tieto triedy sa nachádzajú v module DataEntities.

Štruktúra databázy je popísaná v nasledujúcom obrázku 4.2.



Obr. 4.2: Databázový model

4.1.1.1 Tabuľky

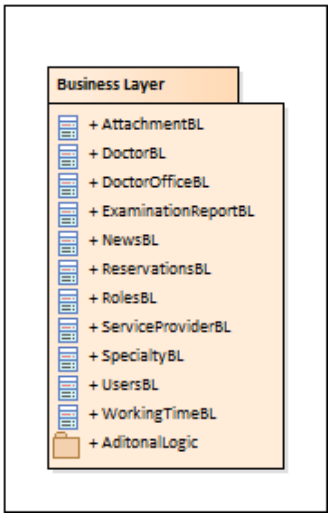
Databáza je zložená z 11 tabuliek:

- **Users** – Obsahuje zoznam užívateľov a informácie o nich.
- **Roles** – Uchováva užívateľské role, ktoré systém rozoznáva.
- **News** – Slúži pre ukladanie oznámení a noviniek.

- **Doctor** – Dodatočné informácie o užívateľoch, ktorí sú lekármi.
- **DoctorOffice** – Drží základné informácie o ambulanciách.
- **WorkingTime** – Slúži na ukladanie pracovnej doby pre ambulancie.
- **ServiceProvider** – Zoznam poskytovateľov zdravotnej starostlivosti a ich adries.
- **Specialty** – Obsahuje všetky druhy lekárskych ambulancií.
- **Reservation** – Úloha tejto tabuľky je držať údaje o objednaní na vyšetrenie.
- **ExaminationReport** – Tabuľka určená k uchovávaniu záznamov z lekárskych vyšetrení.
- **Attachment** – Ak chce lekár okrem textového záznamu evidovať aj ďalšie veci (grafické výsledky CT mozgu, röntgenové snímky, atď), môže tak urobiť a tieto záznamy sú uložené v tejto tabuľke.

4.1.2 Business vrstva

Pre každú entitnú triedu z dátovej vrstvy je vytvorená trieda, ktorá nesie jej názov + BL, čo je skratka pre Business Logic (napríklad NewsBL), ktorá popisuje všetky operácie, ktoré sú s danými dátami v aplikácii vykonávané (obrázok 4.3).



Obr. 4.3: Business Layer

Triedy RoleBL, SpecialtyBL a AttachmentBL neimplementujú žiadne metódy, pretože aplikácia momentálne s entitnými triedami Role a Specialty nijako nepracuje. Sú vytvorené a pripravené pre rozširovanie funkcionality systému v budúcnosti. Ostatné triedy implementujú okrem iných aj metódy Create, Edit a Delete, ktorých úlohou je pomocou Entity Frameworku pracovať s databázou. Tieto metódy sú typu void, takže nemajú žiadne návratové hodnoty.

4.1.2.1 DoctorBL

Ďalšie implementované metódy:

- **GetDoctorById** – Vstupným parametrom je id (int). Metóda vracia objekt typu Doctor. Ten drží informácie o lekárovi, ktorému toto id patrí.
- **GetDoctorByName** – Vstupným parametrom je UserName (string). Úloha tejto metódy je rovnaká ako u GetDoctorById, s tým rozdielom, že vracia informácie o lekárovi na základe jeho užívateľského mena.
- **AllDoctors** – Vracia list všetkých lekárov.
- **SearchedDoctors** – Metóda vráti zoznam lekárov (List<Doctor>), ktorí nejako súvisia so vstupným atribútom (string).
- **DoctorName** – Úlohou tejto metódy je vrátiť meno a priezvisko lekára (string). Vstupným parametrom je objekt typu doctor.
- **GetDoctorWorkingPlace** – Na základe užívateľského mena lekára (string) vráti názov ambulancie (string), v ktorej pracuje.
- **IsDoctor** – Zisťuje, či osoba, ktorej užívateľské meno je predané ako parameter, je lekár alebo nie. Návratové hodnoty su True alebo False.

4.1.2.2 DoctorOfficeBL

Ďalšie implementované metódy:

- **GetDoctorOfficeById** – Vracia objekt typu DoctorOffice, ktorý drží informácie patriace ambulancii s id, ktoré bolo predané ako vstupný parameter (int).
- **GetDoctorOfficeByName** – Vracia informácie o ambulancii (DocorOffice), ktoré sú asociované s menom (string), ktoré bolo metóde predané.
- **GetDoctorOfficeByProviderName** – Táto metóda vracia list ambulancií (List<DoctorOffice>), ktoré patria pod poskytovateľa lekárskej starostlivosti, ktorého meno (string) metóda dostala.

- **GetAllDoctorOffices** – Zoznam všetkých ambulancií s informáciami o nich (List<DoctorOffice>).

- **GetSearchedOffices** – Metóda vracia zoznam ambulancií (List<DoctorOffice>), ktoré nejako súvisia s textom (string), ktorý bol vstupným parametrom.

- **AdvancedDetails** – Metóda vracia objekt, ktorý drží, okrem základných údajov o ambulancii, aj dodatočné informácie.

4.1.2.3 NewsBL

Ďalšie implementované metódy:

- **GetNews** – Metóda bez vstupných parametrov, vracia list so všetkými oznámeniami (List<News>).

- **GetNewsById** – Na základe vstupného parametra id (int), vracia konkrétnu správu (News).

- **GetSelectedNews** – Vracia 10 správ (List<News>), ktoré po sebe nasledujú vzhľadom na čas vytvorenia, zo zoznamu všetkých správ. Ktoré správy má vybrať určuje vstupný atribút (int).

- **GetSearchedNews** – Metóda vráti zoznam správ (List<News>), ktoré nejak súvisia s hľadaným textom (string).

- **GetOfficeNews** – Vracia zoznam správ (List<News>), ktoré boli pridané zamestnancami konkretnej ambulancie (string).

4.1.2.4 ServiceProviderBL

Ďalšie implementované metódy:

- **GetProviderStruct** – Vracia objekt (ProviderStruct), ktorý drží informácie o poskytovateľovi lekárskej starostlivosti a všetkých ambulanciách, ktoré pod neho spadajú.

- **GetProviderById** – Návratovým atribútom metódy je objekt typu ServiceProvider, ktorý drží informácie o konkrétnom poskytovateľovi.

- **SearchedProviders** – Metóda vracia list poskytovateľov lekárskej starostlivosti (List<ServiceProvider>), ktorí nejako súvisia so vstupným atribútom (string).

4.1.2.5 UserBL

Ďalšie implementované metódy:

- **GetUserById** – Metóda má jeden vstupný atribút typu int, na základe ktorého vráti odpovedajúceho užívateľa (User).
- **GetAllUsers** – Vracia zoznam všetkých užívateľov (List<User>).
- **SearchedUsers** – Metóda vráti zoznam užívateľov (List<User>), ktorí sú nejako spojení so vstupným atribútom (string).

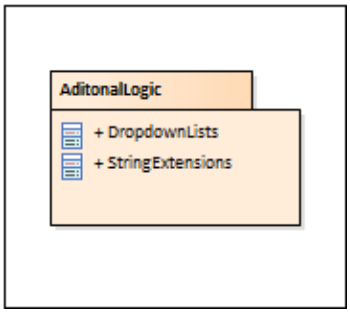
4.1.2.6 WorkingTimeBL

Ďalšie implementované metódy:

- **GetWorkingTime** – Vracia pracovnú dobu (WorkingTime) pre konkrétnu ambulanciu (string).

4.1.2.7 AditionalLogic

Okrem tried s logikou pre entity, business vrstva vrstva obsahuje priečinok s dodatočnou logikou (AditionalLogin - obrazok 4.4)). Nachádzajú sa tu triedy, v ktorých sa implementujú metódy, nesúvisiace s dátovou vrstvou.



Obr. 4.4: Aditional Logic

4.1.3 Prezentačná vrstva

Prezentačná vrstva je tvorená radičmi (controllers), ktoré používajú metódy tried business vrstvy a pohľadmi (views). Spoločne sú rozdelené do jednotlivých modulov.

Úlohou radiča (controller) je reagovať na požiadavky, ktoré sú kladené užívateľom na aplikáciu. Každá požiadavka je mapovaná na konkrétny radič, ktorý na ňu odpovedá. Jednou z takýchto odpovedí môže byť pohľad (view), ktorého meno odpovedá názvu metódy, ktorá ho volá, ale môže ísť napríklad aj o presmerovanie, obrázok, xml...

4.1.3.1 UserArea

Radiče:

- **HomeController** – Obsahuje metódy, ktoré súvisia s domovskou stránkou.

  Pohľady:

  - **Index** – Zobrazuje najnovšie oznámenia. Slúži ako domovská stránka.
  - **About** – Informácie o spoločnosti (čo je jej cieľom, história, atď)
  - **Contact** – Úlohou tohto pohľadu je zobraziť kontaktné údaje.

- **SearchController** – V tomto radiči sú metódy spojené s vyhľadávaním.
  Pohľady:

  - **Index** – Zobrazuje hľadané ambulancie.
  - **Details** – Detaily konkrétnej ambulancie. Zároveň zobrazuje kalendár s vypísanými termínmi.

4.1.3.2 DoctorArea

Radiče:

- **MyOfficeController** – Obsahuje metódy, ktoré súvisia so správou lekárovej ambulancie.

  Pohľady:

  - **Index** – Základné informácie o ambulancii.
  - **Edit** – Formulár pre úpravu informácií.
  - **WorkingTime** – Zobrazuje pracovnú dobu.
  - **WorkingTimeEdit** – Formulár pre zmenu pracovnej doby.

- **NewsController** – Metódy na pridávanie a spravovanie oznámení.

  Pohľady:

  - **Index** – Všetky oznámenia pridané ambulanciou.
  - **Create** – Formulár pre vytvorenie nového oznámenia.
  - **Details** – Zobrazuje dáta konkrétneho oznámenia.
  - **Edit** – Formulár pre upravenie oznámenia.

- **PatientController** – Spravuje požiadavky pre prácu s pacientami.

  Pohľady:

  - **Index** – Zobrazuje zoznam lekárskych záznamov.

- **CreateExaminationReport** – Formulár pre vytvorenie nového záznamu.
- **DetailsExaminationReport** – Detaily konkrétneho záznamu z vyšetrenia.
- **EditExaminationReport** – Formulár pre úpravy záznamu.
- **ReservationEvent** – Zobrazuje kalendár s vypísanými termínmi.

4.1.3.3 AdminArea

- **DoctorController** – Obsahuje metódy, ktoré súvisia so správou lekárov.
- **NewsController** – Metódy na pridávanie a spravovanie oznámení.
- **OfficeController** – Jeho úlohou je spravovať požiadavky pre prácu s lekárskymi ambulanciami.
- **ProvidersController** – Spravuje požiadavky pre prácu s poskytovateľmi lekárskej starostlivosti.
- **UserController** – Metódy pre prácu s užívateľmi.

Všetky radiče majú nasledujúce pohľady: Index, Create, Details, Edit, Search. Index a Search vždy zobrazujú nejaký zoznam (užívatelia, lekári, ambulancie, atď). Create a Edit sú formuláre, prvý na vytvorenie, druhý na úpravu záznamu (UserController nemá Create). Details zobrazuje jeden konkrétny záznam.

4.2 Testovanie

Testovanie aplikácie prebiehalo už počas samotného vývoja. Aplikácia bežala na IIS Express, ktorý je podporovaný Visual Studiom (IDE pre vývoj aplikácii v jazyku C#). Funkčnosť sa testovala cez webový prehliadač Google Chrome, verzie 66.0.3359.0. Boli testované jednotlivé prípady užitia a odhalené chyby boli okamžite odstránené. Po skončení implementácie boli vytvorené testovacie scenáre, ktoré ich nielen znova testovali, ale boli vytvorené situácie, pri ktorých by sa systém mohol správať netypicky alebo by mohol spadnúť. Tieto scenáre sú ďalej popísané.

4.2.1 Testovacie scenáre

4.2.1.1 Prezeranie webových stránok

Scenár simuluje situáciu, kedy chce užívateľ navštíviť domovskú stránku aplikácie, prečítať si novinky, ktoré sú tam zverené. Potom sa rozhodne zobraziť kontaktné informácie, po ktorých sa znova chce vrátiť na domovskú stránku.

4.2.1.2 Vyhľadávanie

Scenár simuluje situáciu, kedy chce návštevník stránok vyhľadať svojho lekára/ambulanciu. Do vyhľadávacieho panelu najskôr nezadá žiaden reťazec, potom reťazce o dĺžke 3 znakov (s/bez diakritky, špeciálne znaky), a následne dlhšie reťazce. Testuje sa správanie systému a správnosť výsledkov.

4.2.1.3 Registrácia

Scenár simuluje proces registrácie. Testuje sa, či systém neumožní vytvoriť 2 užívateľov s rovnakým menom alebo emailovou adresou, či umožní dokončenie registrácie, ak sa *heslo* a *heslo znova* nezhoduje, či bol odoslaný a doručený potvrdzovací email.

4.2.1.4 Prihlásenie

Scenár simuluje snahu o prihlásenie. Testuje sa, či prihlásenie prebehne v prípade, že užívateľské meno je zadané nesprávne, prípadne užívateľ neexistuje. Rovnako sa testuje pokus o prihlásenie so zlým heslom. Okrem toho sa zisťuje, či systém nedokončil prihlasovací proces v prípade, že užívateľ nemá účet potvrdený. Nakoniec sa overuje, či prihlásenie so správnym prihlasovacím menom a heslom prebehlo úspešne.

4.2.1.5 Správa profilu

Scenár simuluje proces zmeny osobných údajov. Testuje sa, či zmena údajov prebehne úspešne a či drop-down listy zobrazujú správne položky. Zároveň sa testuje, či systém zakáže prístup k tejto funkcii neprihláseným užívateľom.

4.2.1.6 Objednať sa na vyšetrenie

Simulácia procesu objednania na vyšetrenie. Testuje sa, či systém zabráni prihláseniu viacerých užívateľov na jedno vyšetrenie a či neumožní túto akciu neprihláseným užívateľom.

4.2.1.7 Prehľad objednaní

Testuje sa, či prihlásený užívateľ má možnosť zobraziť si svoje objednania a ich detailnejšie informácie.

4.2.1.8 Odhlásenie

Testovanie procesu odhlásenia – či prebehlo úspešne.

4.2.1.9 Test užívateľských rolí

Testuje sa, či systém správne rozoznáva užívateľské role a umožňuje prístup k jednotlivým funkcionalitám na ich základe.

4.2.1.10 Správa vlastnej ambulancie

Scenár simuluje prezeranie a úpravu informácií o ambulancii. Testuje sa, či sú informácie zobrazené správne, či lekár pracujúci v jednej ambulancii nemá právo upravovať údaje druhých.

4.2.1.11 Výpis nových termínov

Scenár testuje, či lekár môže vypisovať, upravovať a mazať nové termíny. Rovnako sa kontroluje, či sa správne ukladajú do databázy.

4.2.1.12 Pridávanie výsledkov vyšetrenia

Scenár simuluje proces pridávania výsledkov vyšetrenia. Testuje sa, či systém umožní lekárovi uložiť/vytvoriť záznam k užívateľovi, ktorý alebo neexistuje alebo je jeho meno zadané nesprávne. Rovnako sa testuje, či systém skúsi uložiť záznam bez akéhokoľvek obsahu.

4.2.1.13 Správa oznámení

Scenár simuluje situáciu pridávania nového oznámenia, jeho úpravu a zmazanie. Testuje sa, či formát oznámenia je zobrazovaný správne, či sa jeho úpravy ukladajú v databáze a či po jeho zmazaní zanikne. Testuje sa u lekára aj u administrátora. U lekára sa testuje, či môže spravovať len oznámenia spojené s jeho ambulanciou, kdežto u administrátora sa testuje správa všetkých.

4.2.1.14 Správa užívateľov

Simulácia správy užívateľov. Cieľom je zistiť, či administrátor môže vyhľadávať jednotlivých užívateľov a či pri vykonávaní zmien prebehnú úspešne.

4.2.1.15 Správa poskytovateľov lekárskej starostlivosti

Scenár simuluje proces od vytvorenia po zmazanie nového poskytovateľa. Testuje sa, či systém zabráni vytvoriť 2 poskytovateľov s rovnakým menom alebo bez vyplnenia povinných údajov. Ďalej sa testuje, či sa zmeny vykonané administrátorom naozaj uložia a či zmazaním poskytovateľa sa z databázy zmažú aj všetky ambulancie, ktoré pod neho spadali spoločne s lekármi, ktorí tam pracovali.

4.2.1.16 Správa ambulancií

Scenár simuluje proces od vytvorenia novej ambulancie, cez úpravu jej informácií až k jej zmazaniu. Testuje sa, či systém zabráni existencií 2 ordinácií s rovnakým menom, či sa zmena detailov uloží správne a či sa jej zmazaním odstránia z databázy aj jej lekári. Ďalej sa testuje, či má administrátor možnosť medzi ambulanciami vyhľadávať a či vyhľadávanie funguje správne.

4.2.1.17 Správa lekárov

Scenár simuluje proces od vytvorenia nového lekára. Testuje sa, či systém nedovolí vytvoriť lekára, ktorý už existuje. Okrem toho sa testuje úprava jeho údajov a či jeho zmazaním stratí prístup k správe ambulancie, kde pôvodne pracoval.

4.2.2 Výsledky testovania

Väčšina testov pomocou scenárov prebehla hladko a systém sa správal tak, ako mal. Zároveň však boli odhalené aj chyby v administrátorskej časti aplikácie. U vytvárania a editácie v správe poskytovateľov a ambulancií chýbali validátory, ktoré by zabraňovali žiadosťam o vytvorenie alebo aktualizáciu záznamov, bez potrebných údajov, kvôli ktorým systém padal. Zároveň chýbala kontrola duplicít, čím vznikali viacerí poskytovatelia a ambulancie s rovnakými menami. Tieto chyby boli okamžite odstránené.

Kapitola 5

Zhodnotenie aplikácie

5.1 Ekonomicko-manažérske zhodnotenie

5.1.1 Náklady prevádzkovania aplikácie

Aplikácia nebola vyvíjaná pre komerčné účely, takže náklady spojené s jej využívaním zahrňujú len poplatky za prenájom domény a webhostingové služby. Tabuľka 5.1 popisuje priemerné ročné náklady na prevádzku aplikácie v krajinách Slovenskej a Českej republiky (s DPH). Kvôli zachovaniu jednotnej meny, boli Eurá prepočítane na Korunu českú(Kč) podľa aktuálneho kurzu, suma bola následne zaokrúhlená na celé číslo.

	Slovenská republika	Česká republika
Prenájom domény	175 Kč	179 Kč
Webhosting	1983 Kč	2160 Kč
Celkové náklady	2158 Kč	2339 Kč

Tabuľka 5.1: Náklady na prevádzkovanie aplikácie za 1 rok.

Pod pojmom webhosting sa predstavuje prenájom serveru s operačným systémom Windows, na ktorom beží služba IIS a databázou MS SQL.

5.1.2 Prínosy aplikácie

Ambulancie, ktoré budú používať túto aplikáciu na objednávanie získajú nasledujúce prínosy:

5.1.2.1 Úspora času

Úspora času je najväčším prínosom, ktoré aplikácia ponúka. Jej používaním sa zabráni akejkoľvek inetrakcí medzi pacientom a zamestnancami ambulancie až do momentu, kedy príde na vyšetrenie. Vďaka tomu je setra menej zaťažená

procesom objednávania a môže viac času asistovať lekárovi, čím sa urýchlia jednotlivé vyšetrenia. Dôsledkom toho môžu nastať dva prípady. Prvým z nich je skončenie v práci skôr. Vďaka zrýchleniu priebehu vyšetrenia, ktoré nijako neutrpelo na kvalite sú objednaní pacienti vybavení rýchlejšie a o to môžu ísť zamestnanci ambulancie skôr domov. Druhým prípadom je vybavenie väčšieho množstva pacientov. Lekár spolu so sestrou stihnú za rovnaký čas stihnúť vyšetriť viac ľudí, čo vedie k spokojnosti na oboch stranách. Pacienti sú radi, že boli vyšetrení a lekár je rád, pretože viac zarobil.

5.1.2.2 Noví pacienti

Vďaka využívaniu aplikácie viacerými ambulanciami sa rozširuje aj základňa pacientov. Tí sa najskôr chcú objednať ku svojmu lekárovi, ktorý je špecialista v jednej oblasti, no následne potrebuje vyšetrenie od iného. Pokiaľ doposiaľ nemá konkrétneho, je vysoká pravdepodobnosť, že zvolí pohodlnejšiu cestu, a miesto toho aby začal obvolávať jednotlivé ambulancie, či môže prísť na vyšetrenie, skúsi najskôr vyhľadať takéhoto špecialistu na stránkach aplikácie, kde sa rovno môže aj prihlásiť na voľný termín.

5.2 Evolvabilita aplikácie

Aplikácia je implementovaná tak, aby sa dala upravovať jej funkcionalita počas toho ako bude nasadená. Všetky moduly sa načítavaju dynamicky za behu programu. Pokiaľ bude teda požiadavka na zmenu nejakej funkcionality, stačí len nahradiť strarú verziu modulu za novú. Aplikácia si túto novú verziu okamžite načíta. Rovnako to platí aj pri vytvorení nového modulu, ktorý rozširuje pôvodnú funkcionalitu systému. Stačí ho pridať len do adresára *bin*, z ktorého sa moduly načítavajú.

5.2.1 Možnosti rozšírenia funkcionality aplikácie v blízkej budúcnosti

Možností, ktoré by táto aplikácia mohla svojím užívateľom ponúkať je mnoho. Niektoré však boli identifikované, aj keď neboli priamo požadované. Pri návrhu s nimi bolo počítané a podľa toho bola navrhnutá databáza.

5.2.1.1 Implementácia modulu pre zdravotnú sestru

Momentálne aplikácia nerozlišuje zamestnancov v ambulancii. Sestra a lekár používaju pre úkony spojené s ambulanciu jeden spoločný účet. Nie vždy však lekár chce, aby mala sestra prístup k všetkým jeho funkcionalitám. Riešením je preto implementácia modulu, ktorý bude presne definovať právomoci sestry v rámci ambulancie. Systém už teraz má implementované nejake metódy, ktoré sú prípravou pre tento krok.

5.2.1.2 Prílohy k vyšetreniu

Lekár má momentálne možnosť pomocou aplikácie ukladať záznamy z vyšetrenia vo forme textového zápisu. Môže sa však stať, že bude chcieť evidovať záznamy, ktoré majú inú formu (súbor, obrázok, atď). Touto možnosťou systém zatiaľ nedisponuje, ale boli položené základy pre takéto rozšírenie. V databáze je už vytvorená tabuľka, ktorá bude tieto záznamy evidovať.

5.2.1.3 Dokončenie podpory užívateľov

Implementované metódy aplikácie momentálne neumožňujú užívateľovi zaslať žiadosť o reset hesla v prípade, že sa mu nedarí prihlásiť. Rovnako tak aplikácia neumožňuje zaslanie opätovného potvrdzovacieho mailu v prípade, že sa správa po ceste niekde stratila alebo bola zadaná zlá emailová adresa. Tieto funkcie nemajú priamy vplyv na jej hlavný účel, ale bez týchto funkcionalít sa podstatne komplikuje život jej užívateľom. V prípade, že sa niekto chce prihlásiť na vyšetrenie, no nevie si spomenúť na svoje prihlasovacie údaje, nemá veľa možností. Môže vyhľadať kontakt na ambulanciu a vybaviť si termín objednania telefonicky, čo je úplným protikladom hlavného účelu aplikácie. Druhou možnosťou je vytvorenie nového účtu, čím sa zbytočne zaplňuje databáza. Táto funkcionalita by mala byť integrovaná do systému čo najskôr.

5.2.1.4 Pripomenutie vyšetrenia

Ďalšou z možností rozšírenia je služba aplikácie, ktorá by užívateľom včas pripomenula dátum a čas vyšetrenia, aby sa naň nezabudli dostaviť. Systém je na podobnú službu pripravený. Prvým krokom by mohlo byť posielanie správy na emailovú adresu uživateľa. Ďalším krokom by mohlo byť pripomenutie formou SMS. Systém už teraz umožňuje svojím užívateľom pridať do svojích údajov aj telefónne číslo.

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať modulárny webový rezervačný systém, ktorý by uľahčil objednávanie pacientov na vyšetrenie u lekára.

Aplikácia bola navrhnutá a implementovaná tak, aby si užívateľ vedel čo najjednoduchšie nájsť svojho lekára/ambulanciu, detailnejšie informácie o nich, prípadne sa objednať na vyšetrenie, ak je to možné. Okrem objednávania má aj informačnú funkciu a pacient po príchode na stránky aplikácie môže vidieť najnovšie oznámenia pridané jednotlivými lekármi, sestrami alebo správcami.

Lekár využívajúci túto aplikáciu si vie pozrieť, koľko pacientov je k nemu na daný deň objednaných a vie ich presunúť na iný deň. Vie vytvárať, mazať, upravovať a vyhľadávať záznamy ako k užívateľovi, tak aj k objednaniu. Ďalšou z možností lekára je pridávať a upravovať oznámenia o prípadných zástupoch, dovolenkách...

Rolou administrátora je spravovanie webových stránok a ich užívateľov. Podľa toho boli implementované jeho funkcie. Vie upravovať roly jednotlivých užívateľov, pridávať/upravovať/mazať oznámenia, lekárske ambulancie, poskytovateľov zdravotnej starostlivosti.

Literatúra

[1] About jQuery. In:*jQuery, write less, do more* [online]. 26 Január 2015 [cit. 02.05.2018].Dostupné na: https://learn.jquery.com/about-jquery/

[2] ADO.NET Overview. In: *Microsoft Docs* [online]. Microsoft. 30 Marec 2017 [cit. 13.05.2018]. Dostupné na: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview

[3] ASP.NET MVC, Web API, and Web Pages (Razor) are Open Source Projects. In: *ASP.NET*[online]. Microsoft. 25 Júl 2017 [cit. 01.05.2018]. Dostupné na: https://www.asp.net/open-source

[4] ASP.NET MVC Overview. In: *Microsoft Developer Network* [online]. Microsoft. [cit. 14.05.2018]. Dostupné na: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[5] Common Language Runtime. In: *Wikipédia, slobodná encyklopédia* [online]. [cit. 25.04.2018]. Dostupné na: https://en.wikipedia.org/wiki/Common_Language_Runtime

[6] Concepts of Three-Tier Architecture. In: *Net-informations.com* [online]. [cit. 30.04.2018]. Dostupné na: http://net-informations.com/q/mis/3tier.html

[7] GOPALAKRISHNAN, Vaidy. IIS Express Overview. In: *Microsoft Developer Network* [online]. Microsoft. 6 Júl 2010 [cit. 29.04.2018]. Dostupné na: https://docs.microsoft.com/en-us/iis/extensions/introduction-to-iis-express/iis-express-overview

[8] HART, Johnson. *Windows System Programming*. Third Edition. Addison-Wesley, 2005. 609. ISBN 0-321-25619-0.

[9] KRILL, Paul. Microsoft open-sources Entity Framework. In: *InfoWorld* [online]. IDG. 20 Júl 2012 [cit. 25.04.2018]. Dostupné na: https://en.wikipedia.org/wiki/Entity_Framework

[10] KULKARNI, Dinesh et. al. LINQ to SQL: .NET Language-Integrated Query for Relational Data. In: *Microsoft Developer Network* [online]. Microsoft. Marec 2007 [cit. 14.05.2018]. Dostupné na: https://msdn.microsoft.com/en-in/library/bb425822.aspx

[11] Microsoft SQL Server. In:Wikipédia, slobodná encyklopédia [online]. [cit. 05.05.2018]. Dostupné na: https://en.wikipedia.org/wiki/Microsoft_SQL_Server

[12] .NET Framework. In: *Wikipédia, slobodná encyklopédia* [online]. [cit. 25.04.2018]. Dostupné na: https://en.wikipedia.org/wiki/.NET_Framework

[13] New Features Introduced in IIS 10.0 Version 1709. In: *Internet Information Services*. Microsoft. 24 Október 2017 [cit. 13.05.2018]. Dostupné na: https://docs.microsoft.com/sk-sk/iis/get-started/whats-new-in-iis-10-version-1709/new-features-introduced-in-iis-10-1709

[14] OORTS, Gilles, et al. Building evolvable software using normalized systems theory: A case study. In: *System sciences (hicss), 2014 47th hawaii international conference on*. IEEE, 2014. [cit. 10.05.2018]. p. 4760-4769. Dostupné na: https://pdfs.semanticscholar.org/cdff/850a88974c06446e5c02e2646256427d2563.pdf

[15] TROELSEN, Andrew. *C# and the. NET Platform* [online]. Apress . 2001. [cit. 13.05.2018]. Dostupné na: https://books.google.cz/books?hl=sk&lr=&id=7kUnCgAAQBAJ&oi=fnd&pg=PR16&dq=C%23+and+the.+NET+Platform.+Apress&ots=kIm9mtqOpy&sig=pf3skaGB7I_V_bd5fVg_W4yrTMM&redir_esc=y#v=onepage&q=clr&f=false.

Dodatok **A**

Zoznam použitých skratiek

GUI Graphical user interface

XML Extensible markup language

MVC Model–view–controller

VUC Vyšší Územný Celok

s.r.o. Spoločnosť s ručenínm omezeným

ORM Objektovo-relačné mapovanie

API Aplikačné programovacie rozhranie

IIS Internet Information Services

HTTP Typertext transfer protocol

HTML Hypertext mark-up language

MIT Massachusetts Institute of Technology

SoC Separation of Concerns

DvT Data Version Transparency

AvT Action Version Transparency

SoS Separation of States

OEC Open Environment Corporation

DLL Dynamic-link library

MS SQL Microsoft SQL Server

Dodatok **B**

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF