VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM ZUBNÍ AMBULANCE

INFORMATION SYSTEM OF A DENTAL CLINIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — KLÁRA MIHALÍKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19692/2016/xmihal05

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Mihalíková Klára**
Obor: Informační technologie
Téma: **Informační systém zubní ambulance**
Information System of a Dental Clinic
Kategorie: Informační systémy

Pokyny:

1. Prostudujte aktuální technologie pro vývoj informačních systémů na platformě Windows.
2. Seznamte se s požadavky na informační systém zubní ambulance se zaměřením na slovenské prostředí a s existujícími systémy v této oblasti.
3. Na základě analýzy požadavků navrhněte systém zahrnující evidenci pacientů, zdravotní dokumentaci, objednávkový kalendář, komunikaci se zdravotní pojišťovnou a další funkce.
4. Implementujte navržený systém pomocí vhodně zvolených technologií.
5. Implementujte komunikaci s pacienty pomocí SMS nebo e-mailu.
6. Proveďte testování systému a zhodnoťte dosažené výsledky.

Literatura:

- Petzold, C.: Mistrovství ve Windows Presentation Foundation, Computer Press, 2010
- Dále dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

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

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

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

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cieľom tejto bakalárskej práce je návrh a implementácia informačného systému zubnej ambulancie. Jedná sa o desktopového klienta pre platformu Windows. Informačný systém je postavený na databázovom serveri typu MySQL a je implementovaný objektovo orientovaným jazykom C# s využitím .NET a Entity frameworku. Uživateľské rozhranie je implementované technológiou Windows Presentation Foundation(WPF).

Abstract

The purpose of this bachelor thesis is design and implementation of an information system of a dental clinic. It is a desktop client designed for the Windows platform. The information system is build on the MySQL database server and is implemented in C# object oriented language, using .NET and Entity framework. The user interface is implemented using the Windows Presentation Foundation (WPF) technology.

Kľúčové slová

informačný systém, IS, C# , .NET, Entity Framework, MySQL, WPF, Windows Presentation Foundation, dentálna ambulancia, zubná ambulancia

Keywords

information system, IS, C# , .NET, Entity Framework, MySQL, WPF, Windows Presentation Foundation, dental clinic

Citácia

MIHALÍKOVÁ, Klára. *Informační systém zubní ambulance.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Informační systém zubní ambulance

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Radka Burgeta, Ph.d. a uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

. . . . . .
Klára Mihalíková
11. mája 2017

Poďakovanie

Touto cestou by som rada poďakovala svojmu vedúcemu, Ing. Radkovi Burgetovi, Ph.D, za pomoc pri riešení tejto bakalárkej práce. Ďalej by som rada poďakovala Mudr. Máriusovi Melušovi a Adriáne Lukačovičovej, za poskytnuté informácie a konzultácie, a ďalšiu odbornú pomoc pri riešení dentálnych častí projektu.

Obsah

Kapitola 1

Úvod

V dnešnej dobe je pojem informačný systém vykladaný automaticky ako webový informačný systém. Je to spôsobené tým, že desktopové informačné systémy idú pomaly do úzadia a sú nahrádzané práve spomenutými webovými informačnými systémami, a to najmä z dôvodu multiplatformnej podpory (keďže informačný systém beží vo webovom rozhraní a nemusí nás tým pádom zaujímať, na akom operačnom systéme ho bude chcieť zákazník spúšťať) a prístupu prakticky od hocikiaľ. Je dobré spomenúť aj fyzicky nepotrebné úložisko (program väčšinou beží na prenajatom serveri o ktorého nechybovosť sa stará sprostredkovateľ).

V tejto bakalárskej práci sa však takýmto informačným systémom zaoberať nebudeme. Práve naopak, cieľom práce je vytvoriť desktovopý informačný systém. Dôvodov prečo bola zvolená implementácia desktopového informačného systému namiesto toho webového je hneď viacero.

V prvom rade sa jedná o zákaznícky orientovaný projekt. Celá idea vytvorenia dentálneho, alebo stomatologického, informačného systému vznikla na základe požiadavkov zákazníka. Takže pri jeho tvorbe je potrebné klásť dôraz na samostatné zákazníkové požiadavky, kde jedným z hlavných požiadavkov je práve desktopová aplikácia a úplné vyhnutie sa webovému rozhraniu. Dôvodom prečo na tom zákazník trvá je väčšia bezpečnosť dát, ktorá, keďže informačný systém bude uchovávať veľmi citlivé dáta o pacientoch, je v tomto prípade veľmi podstatná.

Ďalším dôvodom sú vyššie licenčné poplatky za webový informačný systém, spôsobené práve vyššie spomenutým hostingom u tretej strany. A tiež je vhodné spomenúť, že zákazník je majiteľom fyzického servera, na ktorom bude môcť aplikácia bežať.

Bakalársku prácu je možné rozdeliť na dve časti, teoretickú a praktickú. Zaťiaľ čo v teoretickej časti je popísaný rozbor zadania a príprava na praktickú časť, v praktickej časti ide o samotnú implementáciu a jej popis.

Teoretická časť bakalárskej práce je popísaná v kapitolách 2 a 3. V kapitole 2 sa rozoberá problematika projektu do detailov. Na začiatku sú preštudované existujúce stomatologické systémy, pre lepšiu predstavu, čo je potrebné implementovať. Ďalej je kladený dôraz na samotné požiadavky zákazníka, ktoré sú podrobne rozobraté. Nakoniec sú všetky tieto informácie použité na výber vhodných metód a technológií.
V kapitole 3 sa ďalej venuje samotnému návrhu informačného systému, a to z hľadiska návrhu úložiska dát, teda v našom prípade relačnej databázy. Táto kapitola môže byť pokladaná za čiastočne praktickú, keďže odkazuje na reálny MySQL skript využitý pri tvorbe databázy.

Praktickej časti bakalárskej práce sú venované kapitoly 4, 5 a čiastočne aj kapitola 3 (vysvetlené vyššie). Kapitola 4 popisuje implementáciu dentálneho informačného systému. Najskôr je projekt opísaný z všeobecného hľadiska a neskôr sa detailnejšie spomínajú jeho zaujímavé implementačné časti.
Ďalej sa kapitola 5 venuje samotnému testovaniu implementovanej časti projektu z hľadiska správnosti výstupov a intuitívnosti grafického užívateľského rozhrania, kde je kladený dôraz na zákazníkové reakcie a dojmi.

V kapitole 6 je obsiahnuté zhrnutie všetkých častí riešenia tejto bakalárskej práce. Okrem toho táto kapitola obsahuje ešte popis ďalších funkcií a metód, ktoré bude potrebné na informačný systém v budúcnoti aplikovať.

Kapitola 2

Rozbor problematiky

Táto kapitola obsahuje detailný popis cieľov projektu, informácie potrebné k jeho implementácii a zvolené implementačné technológie.

2.1 Stomatologické informačné systémy

Stomatologické informačné systémy majú na slovenskom trhu početné zastúpenie, ale napriek tomu je v ňom možné nájsť medzeru. Pre lepšie porozumenie problematike boli preštudované niektoré existujúce stomatologické informačné systémy popísané v nasledujúcich podkapitolách.

2.1.1 PC DENT

Staršia verzia programu bola doteraz využívaná zákazníkom. Bol to komplexný DOS program s množstvom funkcií, ktoré sa ale stávali stále menej a menej vyhovujúce.

Nová verzia programu[1] je prístupná len pre Českú republiku, ale ak by aj PC DENT ponúkal rozširujúci modul pre Slovenskú republiku, nebol by vyhovujúci už z dvoch dôvodov. Aplikácia je stále v rozhraní DOS, grafické uživateľské rozhranie aplikácie sa tým pádom príliš nevyčistilo a stále je komplexné, a z ponúkaných funkcií je ich stále dosť pre zákazníka nepotrebných. Čo má PC DENT vyriešené dobre, je používanie fráz. Frázy sú užívateľmi opakovane písané texty, zapisované do lekárskych správ. V PC DENTe je možné si zadefinovať vlastné frázy a neskôr ich jednoduchým výberom vložiť do textu a prípadne len dopísať zvyšok.

2.1.2 Správca Ordinácie Stomatológa pre Windows (SOSW)

SOSW[2] je podobne ako predošlý informačný systém aplikácia DOS, čiže má rovnako zastaralé grafické uživateľské rozhranie, avšak oveľa prehladnejšie, čo je spôsobené výberom najdôležitejších operácií a ich správneho spracovania. Napriek tomu, že mne sa zdalo rozhranie SOSW prívetivejšie, zákazníkovi robil ešte väčšie problémy ako PC DENT. Tiež ma upozornil na nedostačujúci zubný kríž.

[1] Viac o IS na stránkach http://www.pcdent.cz/

[2] Viac o IS na stránkach http://www.stomatolog.pap.sk/home

2.1.3 XDENT

XDent[3] je moderný, webový stomatologický informačný systém s prehľadným a intuitívnym grafickým uživateľským rozhraním. Jedná sa o produkt Českej republiky, od konca minulého roku prístupný aj pre Slovenkú republiku. V čase získania projektu síce ešte nebol plne prispôsobený pre slovenský trh, teda vykazovanie poisťovniam nebolo možné, ale v tejto chvíli už by mali byť všetky časti programu plne funkčné. Napriek tomu má stále vlastnosti, ktoré klienta odrádzajú. V prvom rade sa jedná o webový informačný systém, čo si klient výhradne neželá a za druhé sú to vysoké licenčné poplatky.

2.2 Požiadavky zákazníka

Požiadavky zákazníka sú špecifikované na základe osobných konzultácií. Spolu sme prechádzali v súčasnosti existujúce stomatologické informačné systémy, spomenuté v kapitole 2.1, ku ktorým sa zákazník vyjadroval. Určil, ktoré operácie sú pre neho dôležité, prípadne ako by ich chcel upraviť, a tiež vyškrtol pre neho nepotrebné funkcie. Výsledkom je jasný, až odborný popis programu, obsahujúci 9 najdôležitejsích operácií:

1. Operácie nad lokálnou databázou pacientov,
2. evidovanie zdravotnej dokumentácie jednotlivých pacientov,
3. tlač a export zdravotných dokumentov,
4. vykazovanie poisťovniam,
5. grafický plánovač,
6. vyhľadávanie v databáze registrovaných liekov,
7. SMS, či emailovú komunikáciu s pacientmi,
8. grafický zubný kríž obsahujúci aj operácie pre zubnú hygienu,
9. prepojenie so softwarom Florida Probe.

Výsledný produkt by mal byť jednoducho ovládateľný desktopový informačný systém prepájajúci lokálnu databázu pacientov so štyrmi firemnými počítačmi v rôznych režimoch prístupu (viď podkapitola 2.2.1), implementujúci vyššie definované operácie a ich podoperácie s moderným grafickým uživateľským rozhraním. V tejto bakalárskej práci sa budeme venovať riešeniu bodov 1 až 7, ktoré sú detailnejšie popísané v kapitole 2.2.2. Plánované body 8 a 9 sú popísané v kapitolách 6.1.1 a 6.1.2.

2.2.1 Režimy prístupu

Dentálny informačný systém bude možné spustiť v troch rôznych režimoch, konkrétne v *Dentálnom*, *Hygienickom* a *Personálnom* režime.

Dentálny režim - pracovný režim určený pre zubárov, umožňujúci im vykonávať takmer všetky operácie ponúkané programom, okrem zasahovania do "hygienickej" časti.

[3] Viac o IS na stránkach https://www.xdent.sk/

Hygienický režim - pracovný režim určený pre zubných hygienikov, umožňujúci im vykonávať operácie nad "hygienickou" časťou programu.

Personálny režim - pracovný režim určený pre recepciu a iný personál, umožňujúci prístup k programu iba vo forme prehliadania záznamov.

2.2.2 Detailnejší popis hlavných funkcií

Táto podkapitola obsahuje detailnejší popis zákazníkom definovaných hlavných funkcií (kapitola 2.2), ktoré budú základom navrhovaného dentálneho informačného systému.

Operácie nad databázou pacientov

Vytvorenie, editácia a mazanie pacientov s možnosťou rýchleho vyhľadávania a s odkazom do zdravotnej karty daného pacienta

Evidovanie zdravotnej dokumentácie jednotlivých pacientov

Zdravotná dokumentácia je evidovaná v podobe *lekárskych správ* (so zreteľným oddelením dentálnej časti od hygienickej), *anamnézy* a *BOP indexom krvácania.* Tiež je potrebné umožniť vyhľadávanie v lekárskych správach, vyhľadávanie v databáze registrovaných liekov a oddelenie výkonov vykonaných na pacientoch podľa typu výkonu na *konzervačné* výkony, *endodonciu*, *chirurgiu* a *protetiku.*

Lekárska správa obsahuje chronologický písomný záznam o vyšetrení a liečbe pacienta, ktorý zahŕňa anamnézu a sťažnosti pacienta, fyzické zistenia lekára, výsledky diagnostických testov a postupov, lieky a terapeutické postupy.[5]

V programe by bolo vhodné zobrazovať minimálne tri posledné lekárske správy v hlavnom okne po otvorení zdravotnej dokumentácie pacienta.

Anamnéza je subjektívny popis ťažkostí pacienta, získaný dotazmi kladenými priamo pacientovi. Môžeme ju rozdeliť do viacerých skupín, ale v dentálnom informačnom systéme budeme využívať iba:

- **osobnú anamnézu (OA)** - dotaz na liečené choroby, prekonané operácie a úrazy, vhodne tiež na poslednú hospitalizáciu[2]
- **farmakologickú anamnézu (FA)** - dotaz na trvalú aj príležitostne užívanú medikáciu, najlepšie vrátane sily liekov a rozpisu užívania[2]
- **Alergologická anamnéza (AA)** - dotaz na alergie, najmä na lieky a ich prejavy[2]

Anamnézu je potrebné zahrnúť do zdravotnej dokumentácie pacienta tak, aby bola hneď jasne viditelná (ako v dentálnej, tak hygienickej časti), vypĺňaná pomocou preddefinovaného formulára, ktorý je možné upravovať.

Bleeding on probing (BOP) je index označujúci krvácanie po sondovaní. Meria sa pomocou parodontologickej sondy, ktorá sa zavedie na dno defektu, a tam sa s ňou pohybuje. Tento index je jednoduchší ako index *PBI (Papillary Bleedin Index)*, hodnotí sa

iba, či krváca, alebo nie. Výsledok sa zaznamenáva v percentách, pričom sa využíva zlomok: súčet krvácajúcich miest x 100 a výsledok sa podelí celkovým počtom meraných miest.[3]

V programe je potrebné zaznamenávať BOP jednotlivých pacientov v tvare `x/y` spolu s dátumom vytvorenia záznamu, aby ich bolo možné porovnávať.

Tlač a export zdravotných dokumentov

Program musí byť schopný vytlačiť nasledujúce dokumenty:

- *Lekársku správu,* ktorá je definovaná vyššie. Lekárska správa je tlačená na samostatný hárok formátu A5, doplnená o dátum a čas prehliadky ku ktorej bola vytvorená, meno a priezvisko pacienta, a meno a priezvisko ošetrujúceho lekára.
- *Dotlač do zdravotnej dokumentácie* podľa smerníc udávaných Ministerstvom zdravotníctva Slovenskej republiky, musí byť zdravotná dokumentácia pacienta vedená aj fyzicky(papierovo), nie len elektronicky. Jedná sa o rovnaký text, ktorý obsahuje lekárska správa, doplnený o dátum a čas zdravotnej prehliadky a tiež meno a priezvisko pacienta. Fyzická zdravotná dokumentácia je vedená na papier formátu A4 a dotláča sa do textu z predchádzajúcich prehliadok, až pokým sa celý papier nezaplní. Je potrebné, aby sa toto "odriadkovanie" vykonávalo automaticky.
- *Prehľad výkonov za dané obdobie* exportované vo forme csv súboru. Exportované dokumenty je možné filtrovať na základe viacerých kritérií, kontrétne zdravotnej poisťovňe, kódu diagnózy, kódu lokalizácie, kódu výkonu a pacientov.

Vykazovanie poisťovniam

Vyfakturovanie jednotlivých úkonov vykonaných na poistených pacientoch pre zmluvné poisťovne. Je potrebné postupovať podľa nového metodického usmernenia *Úradu pre dohľad nad zdravotnou starostlivosťou (ďalej ÚDZS)* číslo *5/1/2015* o spracovaní a vykazovaní zdravotných výkonov poskytovateľov zdravotnej starostlivosti elektronickou formou, ktoré nadobúda platnosť dňa 1.1.2017. Program musí byť schopný vygenerovať jeden typ dávok spomenutého metodického usmernenia, a to konkrétne dávku *751a Dátové rozhranie - Vykazovanie výkonov v ambulantnej zdravotnej starostlivosti.*

Vykazovanie prebieha formou elektronického dokumentu, bližšie špecifikovaného v prílohe k *MU 5/1/2015*, konkrétne sa jedná o prílohu č.1, dostupnej na stránkach ÚDZS[4] .

Grafický plánovač

Plánovač zobrazujúci objednaných pacientov popisujúci typ ich návštevy, konrétne teda či ide o preventívnu prehliadku, chirurgický zákrok, zubnú hygienu a podobne. Bolo by dobré, ak by bolo možné nahliadnuť do kalendára kolegu, avšak bez možnosti jeho editácie.

Vyhľadávanie v databáze registrovaných liekov

Ako bolo už vyššie spomenuté, vyhľadávanie je umiestnené v zdravotnej dokumentácii pacietna. Je potrebné vyhľadávať medzi aktuálnymi informáciami dostupnými na stránkach

[4] http://www.udzs-sk.sk/metodicke-usmernenia?inheritRedirect=true

Štátneho ústavu pre kontrolu liečiv[5]. Toto vyhľadávanie bude využívané pri zisťovaní kontraindikácie liečiv a neskôr aj pri predpise liekov.

SMS, či emailová komunikácia s pacientmi

Komunikácia s pacientmi prebieha v zmysle upozornenia pacienta na nadchádzajúcu prehliadku (deň vopred), alebo pri vytvorení nového záznamu.

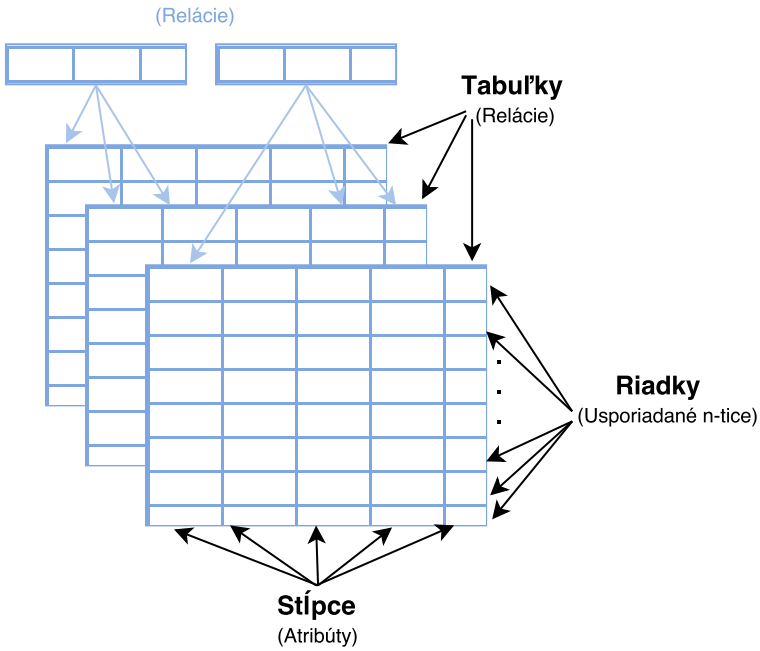
2.3 Zvolené metódy a technológie

Po dôkladnom preštudovaní požiadavkov zákazníka spomenutých v predchadzajúcej kapitole bolo rozhodnuté o následnom postupe. Zákazník je vlastníkom fyzického serveru a tým pádom bude k uchovávaniu perzistentných dát využitá relačná databáza, ktorá bude implementovaná jazykom MySQL. Definované operácie budú implementované jazykom C# s pomocou .NET frameworku. Pre prístup k dátam bude využitý Entity Framework a grafické uživateľské rozhranie bude implementované pomocou Windows Presentation Foundation.

Jednotlivé technológie budú bližšie priblížené v nasledujúcich podkapitolách.

2.3.1 Relačná databáza

Relačný model dát je taký, ktorého základná abstraktná jednotka je relácia, v prostredí databáz zobrazovaná formou tabuľky. Pre lepšiu predstavu sa môžeme pozrieť na obrázok 2.1.



Obr. 2.1: Model relačnej databázy

Relácie (alebo tabuľky) sú základnými prvkami relačného dátového modelu. Relácia sa skladá z atribútov (stĺpec tabuľky) a z usporiadaných n-tíc. (údaje - riadok tabuľky).

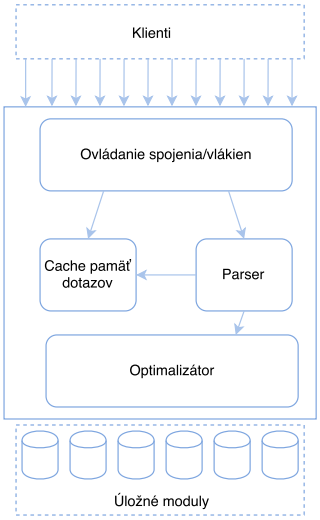
[5] http://www.sukl.sk/sk/databazy-a-servis/databazy/vyhladavanie-v-databaze-registrovanych-liekov?page_id=242

Relačné operácie vytvárajú ďalšie relácie - tieto opreácie znamenajú manipuláciu medzi dátami. Manipulácia s dátami je založená na operáciach relačnej algebry. Tieto operácie zahrňujú množinové operácie (zjednotenie, prienik, rozdiel, kartézsky súčin) a tzv. špeciálne relačné operácie ako sú selekcia, spojenie, delenie a projekcia[6].

MySQL

Databázový server a zároveň formálny dotazovací jazyk, ktorý umožňuje formulovať dotazy nad dátami uloženými v relačnej databáze.
Najneobvyklejším a najdôležitejším prvkom MySQL je jeho architektúra s pamäťovými mechanizmami, ktorej dizajn oddeľuje spracovanie dotazov a iné úlohy servera od ukladania dát a získavania údajov. Toto oddelenie vám umožňuje vybrať, na báze tabuliek, ako sa vaše dáta ukladajú, s akým výkonom, funkciami a ďalšími vami zvolenými vlastnosťami.[1]

Dobrú predstavu o fungovaní MySQL komponentov pomôže pochopiť MySQL server. Obrázok 2.2 zobrazuje logický pohľad na architektúru MySQL.



Obr. 2.2: Architektúra MySQL servera, prevzaté z [1]

Najvyššia vrstva obsahuje služby, s ktorými sa môžeme stretnúť u väčšiny sieťových klient-server nástrojou, ako je manipulácia s pripojením, autentizácia, bezpečnosť atď.
Druhá vrstva obsahuje najdôležitejšie MySQL služby, vrátane parsovania dotazov, optimalizácie, ukladania do pamäte a všetkých vstavaných funkcií.
Tretia vrstva obsahuje pamäťové moduly. Tie sú zodpovedné za ukladanie a načítanie všetkých údajov uložených v MySQL[1].

Dôležitou vlastnosťou MySQL, pre ktorú bol zvolený do projektu, okrem výhody open source zdielania, je jeho konektor Net 6.8, ktorý integruje podporu pre Entity Framework 5 (EF5), a to ako verziu Code First (Entity model generovaný z kódu) tak Database First (Entity model generovaný z existujúcej databázy). Viac o entity frameworku v podkapitole 2.3.4(ADO.NET a Entity Framework).

2.3.2 Jazyk C# a LINQ

Vysokoúrovňový programovací jazyk vyvinutý spoločnosťou Microsoft v rámci iniciatívy vývoja .NET, schválený štandardom Ecma (ECMA-334) a ISO (ISO/IEC 23270:2006).

Jazyk C# poskytuje plnú podporu objektovo orientovaného programovania vrátane zapúzdrenia, dedičnosti či polymorfizmu.

LINQ (Language-Integrated Query)[6] je súbor funkcií, ktoré rozširujú dotazovacie možnosti jazyka C#. Spoločnosť LINQ zaviedla štandard jednoducho zapamätateľných dotazovacích vzorov využívaných na vyhľadávanie alebo aktualizáciu údajov, pričom obsahuje podporu pre takmer každý druh dátových úložísk. .NET framework umožňuje využívanie LINQ pri práci s kolekciami, SQL serverom, ADO.NET a XML dokumentmi.

V projekte je využívaný pri vyhľadávaní a parsovaní XML súboru a dokonca pri vyhľadávaní a parsovaní HTML dokumentu.

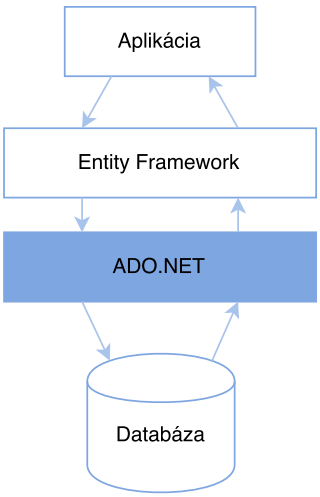
2.3.3 .NET Framework

.NET Framework poskytuje komplexný programovací model pre vytváranie všetkých druhov Windows aplikácií, od mobilného rozhrania, cez web až po desktop. .NET Framework sa skladá z dvoch častí:

- **Framework Class Library (FCL)** - poskytuje jazykovú interoperabilitu, inými slovami, každý programovací jazyk využíva iný kód a FCL ho "prekladá"
- **Common Language Runtime (CLR)** - virtuálny stroj aplikácie, v ktorom sa spúšťajú programy napísané pre .NET

2.3.4 Entity Framework a ADO.NET

Entity Framework je "Objektový relačný mapovač", z anglického Object Relational Mapper (ORP), ktorý umožňuje vývojárom aplikácie pracovať s relatívnymi dátami ako s modelmi. Toto uľahčuje prístup k dátam a oslobodzuje vývojárov od písania podobných prístupových kódov ako pri ADO.NET[4]. Na obrázku 2.3 môžeme však vidieť že Entity framework je písaný nad ADO.NET frameworkom, a vo vnútri teda stále využíva ADO.NET metódy a triedy pre vykonávanie dátových operácií.



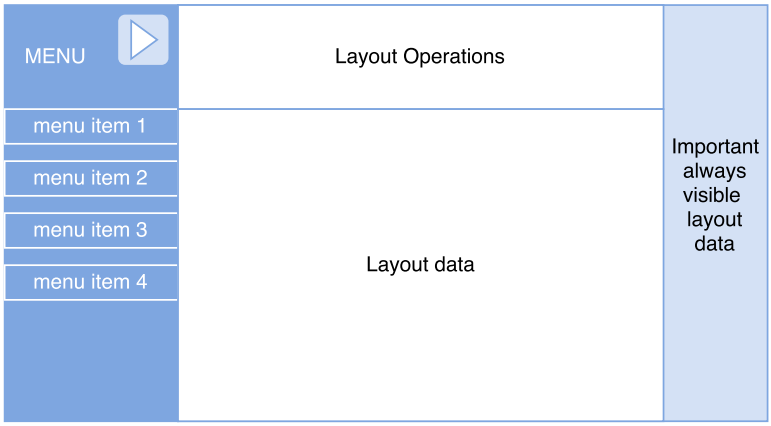
Obr. 2.3: Vizualizácia architektúry Entity Frameworku

[6] Viac o LINQ na stránkach https://msdn.microsoft.com/en-us/library/bb308959.aspx#linqoverview_topic1.

Ako bolo už spomenuté v podkapitole 2.3.1(MySQL), pre vytváranie Entity Modelu sa využívajú dva prístupy - Code First (Entity model generovaný z kódu) a Database First (Entity model generovaný z existujúcej databázy). V programe bude využitý Database First prístup, pretože návrh a vytvorenie funkčnej databázy je ťažiskom informačného systému.

2.3.5 Grafické uživateľské rozhranie (GUI)

Grafické uživateľské rozhranie je pri informačných systémoch veľmi dôležité. Nielen preto, že tvorí prvý dojem, ale aj preto, že je často práve ono dôležitým faktorom pri nalákaní alebo úplnom odradení zákazníka. Aj keď sú všetky operácie 100% funkčné, staviť na ikonami a operáciami presýtený layout nie je najlepší nápad (Ako už bolo poukázané v podkapitole 2.1). Preto bolo pri návrhu informačného systému stavené na jednoduchosť layoutu, obrázok 2.4.



Obr. 2.4: Návrh GUI pre stomatologický informačný systém

Ako môžeme na návrhu vidieť, aplikácia má layout rozdelený na menu a hlavný layout, ktorý je ďalej delený na tri časti - operácie, dáta a dôležité, vždy viditeľné, dáta layoutu.

Menu - tlačidlom skryteľné/otvoriteľné menu je akýmsi navigátorom po najdôležitejších operáciách. Po kliknutí na tlačidlo položky menu nastáva načítanie nového layoutu vo vnútri rovnakého okna a skrytie toho pôvodného. Zakrývaním a odkrývaním layoutov dosiahneme ilúziu zapamätávania si posledných operácií.

Layout operácií - je umiestnený navrchu hlavného layoutu. Je vždy viditeľný, čo umožňuje rýchly prístup k potrebným operáciám.

Layout dát - je umiestnený pod layoutom operácií a zobrazuje požadované informácie/dáta. Layout by mal byť skrolovateľný, kedže predpokladáme veľké množstvo zobrazovaných dát.

Layout dôležitých dát - nie je potrebné ho zobrazovať vo všetkých layoutoch, pravdepodobne bude mať zastúpenie iba v zdravotnej karte. Obsahuje dáta, ktoré musia byť vždy viditeľné a teda ich nestačí mať zastúpené v "Layoute dát", ale je potrebné ich aj extrahovať do tohoto panelu.

Windows presentation foundation (WPF)

Windows Presentation Foundation, knižnica tried od spoločnosti Microsoft, je určená na vykresľovanie GUI v aplikáciách bežiacich na platforme Windows. WPF oddeľuje užívateľské rozhranie od ostatnej logiky, čím sa snaží poskytnúť konzistentný programovací

model. Využíva značkovací jazyk XAML (podobný XML formátu), ale je možné vytvárať a volať grafické objekty priamo z kódu.

Kapitola 3

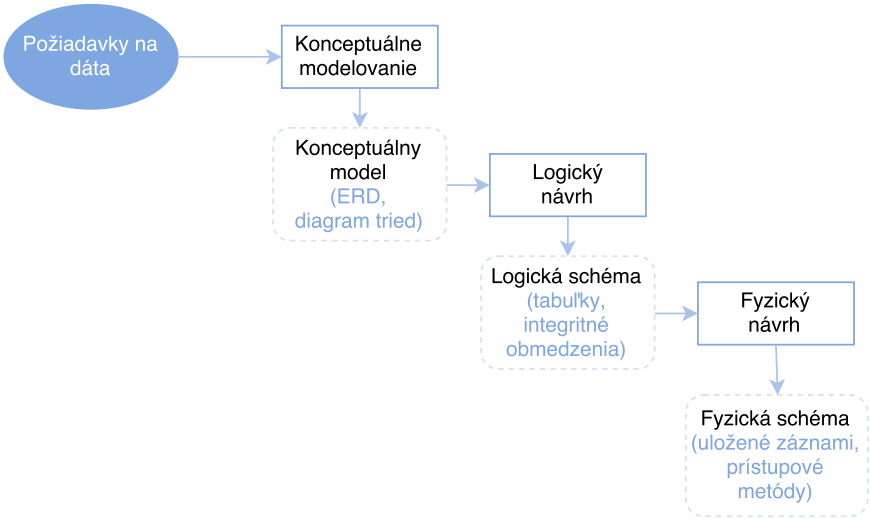
Návrh informačného systému

Návrh informačného systému je modelovaný na báze *konceptuálneho modelovania*, pričom ide o fázu dátovej, prípadne objektovej analýzy, využívajúc modely založené na objektoch aplikačnej domény (viac v podkapitole 3.1). Postup návrhu relačnej databázy s využitím konceptuálneho modelovania môžeme pozorovať na obrázku 3.1.

Po prijatí požiadavok a ich "dešifrácii" nastupuje konceptuálne modelovanie, ktorého výsledkom je *konceptuálny model*. Ako už bolo spomenuté, problematike konceptuálneho modelovania je venovaná podkapitola 3.1.

Po konceptuálnom modelovaní je na rade logický návrh (popísaný v kapitole 3.2). Výsledkom logického návrhu je *logická schéma*.

Posledným krokom návrhu relačnej databázy je fyzický návrh (popísaný v kapitole 3.3), ktorého výsledkom je *fyzická schéma*.



Obr. 3.1: Postup návrhu relačnej databázy, prevzatý z [7]

3.1 Konceptuálne modelovanie

Konceptuálne modelovanie je súčasťou rady metodologií vývoja softwarových systémov. Prebieha vo fáze analýzy požiadavkov. Jeho cieľom je vytvorenie modelu konceptov aplikačnej domény, s ktorými bude vyvíjaný systém pracovať. Nás budú z hľadiska návrhu

databázy zaujímať tie koncepty a vzťahy medzi nimi, ktorých reprezentácia bude uložená v databáze[7].

Najčastejšie používanou modelovacou technikou využívanou pre návrh relačnej databázy je entitne-vzťahové modelovanie, ktorého výsledkom je entitne-vzťahový diagram (z anglického Entity Relation Diagram). Tento model bude bližšie priblížený a aplikovaný na riešenie projektu v kapitole 3.1.1.

3.1.1 Entity Relation Diagram (ERD)

Entity Relation Diagram, alebo v preklade entitne-vzťahový diagram, je výsledkom entitne-vzťahového modelovania. Názov tejto modelovacej techniky, ktorá vznikla v polovici 70. rokov minulého storočia (autor P.P.Chen), odráža dva základné prvky vytváraného modelu:

- **Entity** = entita, objekt reálneho sveta, odlišný od ostatných objektov, o ktorom chceme mať v databáze uchované informácie,
- **Relation\Relationship** = vzťah, asociácia medzi dvoma a viacerými entitami.

Tretím základným prvkom modelu sú *atribúty*.[7]
Atribút je vlastnosť entity, ktorú chceme v databáze uchovávať. Okrem týchto pojmov obsahuje definícia ER diagramu ešte nasledujúce dôležité pojmy, vďaka ktorým je diagram lepšie špecifikovaný:

- **Identifikátor entitnej alebo vzťahovej množiny** - atribút, ktorého hodnota je v rámci danej množiny jedinečná (primárny kľúč),
- **Kardinalita** - maximálny počet vzťahov daného typu, ktorý môže jedna entita obsiahnuť,
- **Členstvo** - minimálny počet vzťahov daného typu, ktorých sa musí jedna entita účastniť.

Entitne-vzťahové modelovanie chápe modelovanú aplikačnú doménu ako množinu entít, medzi ktorými môžu existovať určité vzťahy. Dôležitým rysom ER modelu je, že popisuje dáta "v pokoji", teda nezobrazuje, aké operácie budú s týmito dátami vykonávané.[7]

Na základe týchto informácií bol vytvorený ER diagram aplikácie, ktorý je priložený v prílohe C.

3.2 Logické modelovanie

Cieľom logického modelovania je navrhnúť štruktúru databázy tak, aby bolo možné reprezentovať požiadavkami definované informácie bez existencie *redundancie* a s platnými *integritnými obmedzeniami*, vyplývajúcimi zo závislosti medzi hodnotami uloženými v databáze a ich reprezentáciou v reálnom svete.

Redundancia

Redundancia je duplikácia dát uchovávaných v databáze, ktorej dôsledkom je nefunkčnosť dotazov, či nesprávnosť výsledkov získaných dotazmi nad databázou. Dá sa jej predísť

správnym zvolením primárneho kľúča, poprípade kľúčov, jednotlivých tabuliek.

Integritné obmedzenia

Integritné obmedzenia sa viažu na objekty reálneho sveta a ich reprezenáciu v databáze. Ide o kontrolu správnosti formátu uchovávaných dát, napríklad tvar rodného čísla, mobilného čísla, atď.

3.2.1 Logická schéma aplikácie

Popis jednotlivých tabuliek databázy. MySQL skript, pomocou ktorého boli vytvorené je priložený na priloženom CD, viac v prílohe A (database.sql).

Personál

id_pnl	Meno	Priezvisko	datum_narodenia	adresa	tel_cislo	
e-mail	prava	poistovne	kod_lekara	kod_ambulancie	kod_zubnej_amb	

Tabuľka 3.1: Tabuľka Personál

Tabuľka personál obsahuje všetky potrebné informácie o lekároch, zubných hygienikoch a ďalších zamestnancoch, ktorý budú pracovať s informačným systémom. Primárnym kľúčom je `id_lekara`. Tabuľka ďalej obsahuje osobné informácie danej osoby, ako meno, priezvisko, adresu, email a telefónne číslo.

Stĺpec práva môže byť nastavený na 3 rôzne príznaky:

- **D** - osoba má práva prístupu v "dentálnom" režime
- **H** - osoba má práva prístupu v "hygienickom" režime
- **P** - osoba má práva prístupu v "personálnom" režime

Ďalší stĺpec tabuľky, poistovne, obsahuje päťznakový reťazec v implicitnom stave nastavený na samé nuly. V inom prípade je nastavený na jedna na tom indexe poisťovne, s ktorou má lekár zmluvu. Tieto príznaky budú overované iba v prípade, že budú mať lekári odlišné zmluvné poisťovne.

Posledné tri stĺpce, kod_lekara, kod_ambulancie a kod_zubnej_amb, identifikujú lekára. Tieto informácie sú využívané najmä pri vykazovaní poisťovniam a spolu so stĺpcom poistovne sú relevantné iba pri právach "D".

Všetky stĺpce tabuľky majú dátový typ `varchar`, okrem dátumu narodenia, ktorý má dátový typ `date`.

Pacient

Meno	Priezvisko	**rodne_cislo**	datum_narodenia	adresa	email	
		tel_cislo	poistovna	anamneza		

Tabuľka 3.2: Tabuľka Pacient

Tabuľka pacient ukladá osobné informácie pacienta, a to konkrétne meno a priezvisko, datum narodenia, adresu, telefónne číslo, kód zdravotnej poisťovne a pacientovu anamnézu. Všetky polia okrem dátumu narodenia a anamnézy ukladajú informácie s dátovým typom `varchar`. Dátum narodenia využíva dátový typ `date`.

Keďže anamnéza musí byť dostupná vo forme formulára a jej položky je potrebné občas zmeniť, je do databázy uladaná vo forme xml dokumentu, do stĺpca tabuľky s dátovým typom `blob`.

Tabuľka obsahuje cudzí kľúč poisťovne odkazujúci do tabuľky zmluvné poisťovne. Primárny kľúč je `rodné číslo`.

Zmluvné poisťovne

kod_poistovne	nazov	bodovy_kvocient	identifikator

Tabuľka 3.3: Tabuľka Zmluvné poisťovne

Primárnym kľúčom tabuľky zmluvne_poistovne je `kod_poistovne`. Ďalej obsahuje textový oficiálny názov poisťovne, jej bodový kvocient a identifikátor.

Bodový kvocient je číselná hodnota (môže byť v desatinnom tvare), ktorou sa delí bodové ohodnotenie zdravotných výkonov, ktorých hodnoty sú dané ministerstvom zdravotníctva. Tento prepočet sa využíva pri vykazovaní poisťovniam, prepočte pacientovho doplatku a podobne.

Identifikátor je celočíselná hodnota, určujúca index(poradie) príznaku zmluvných poisťovní, ktorý odkazuje na hodnotu uchovávanú v tabuľke personál v stĺpci poisťovne. Čo znamená, že identifikátor môže obsahovať hodnoty od 0 po 4, keďže reťazec poisťovne má znakov päť.

Z popisu celkom jasne vyplýva, že prvé dve hodnoty, kód poistovne a jej názov, sú vedené s dátovým typom `varchar`, a naopak bodový kvocient má dátový typ `float` a identifikátor `integer`.

Lokality

id_lokality	nazov_lokality

Tabuľka 3.4: Tabuľka Lokality

Tabuľka lokality obsahuje zoznam lokalít ústnej dutiny. Využívajú sa pri vykazovaní poisťovniam a tiež v texte správ zdravotnej dokumentácie. Neskôr budú potrebné aj pri tvorbe zdravotného kríža.

Primárnym kľúčom tabuľky je `id_lokality`, okrem ktorého obsahuje už iba položku nazov_lokality. Obe hodnoty sú dátového typu `varchar`.

Diagnózy

Táto tabuľka, podobne ako tabuľka vyššie, obsahuje oficiálne definovaný zoznam, teraz konkrétne zdravotných diagnóz, definovaný podľa Medzinárodnej klasifikácie chorôb[1].

[1] Dostupné na stránkach Národného centra zdravotníckych informácií: http://www.nczisk.sk/Standardy-v-zdravotnictve/Pages/Medzinarodna-klasifikacia-chorob-MKCH-10.aspx

id_diagnozy	nazov_diagnozy	popis_diagnozy

Tabuľka 3.5: Tabuľka Diagnózy

Ďalej sa podobne ako tabuľka lokality využíva pri vykazovaní poisťovniam a v zdravotnej dokumentácii.

Tabuľka diagnózy má tri položky, a to `id_diagnozy`, ktorá je zároveň primárnym kľúčom tabuľky, nazov_diagnozy a popis_diagnozy. Id a názov sú dátového typu `varchar` a pamäťovo náročnejší popis je typu `text`.

Výkony

id_vykonu	typ	popis_vykonu	poznamka	body	akutnost	
						priznak_vykazovanie	cena	zisk

Tabuľka 3.6: Tabuľka Výkony

Tabuľka výkony definuje úkony na pacientoch, ktoré zdravotník môže vykonávať. Tieto výkony sú definované na stránkach Ministerstva zdravotníctva Slovenskej republiky[2].

Na stránkach ministerstva sú definované jednotlivé výkony spolu s jedinečným kódom výkonu, jeho bodovým ohodnotením a prípadne poznámkou. Tieto štyri hodnoty sú v tabuľke reprezentované hodnotami id_vykonu, popis_vykonu, body a poznamka. `Id_vykonu` je zároveň aj primárnym kľúčom tabuľky.

Okrem týchto stĺpcov obsahuje tabuľka ďalšie položky, ktoré si už ale definuje zákazník sám. Sú to typ, akútnosť, príznak vykazovania, cena a percentuálny zisk.

Položka typ môže nadobúdať štyri možnosti, definujúce typy stomatologických zákrokov:

- **Konzervačné výkony**
- **Endodonciu**
- **Chirurgiu**
- **Protetiku**

Id výkonu, typ, akútnosť a príznak vykazovania sú dátového typu `varchar` a popis výkonu a poznámka sú `text`, kvôli obsahovej náročnosti. Body a cena sú dátového typu `float` a zisk je celočíselná hodnota, uchovávajúca percentuálny podiel krát 100 a má dátový typ `integer`.

Správy

rc_pacienta	**id_lekara**	**datum**	typ	text_spravy	riadky

Tabuľka 3.7: Tabuľka Správa

[2]Dostupné na stránkach ministerstva: http://www.health.gov.sk/Hladat?q=zoznam-vykonov-a-ich-bodove-hodnoty-20041218.xls

Táto tabuľka ukladá jednotlivé lekárske správy pacientov. Správy môžu byť pridávané od viacerých lekárov a hygienikov a tiež je potrebné vedieť odlíšiť či ide o dentálnu alebo hygienickú správu. Tento problém rieši položka id_lekara, ktorá odkazuje do tabuľky personál.

Ďalším dôležitým krokom je previazanie správy s pacientom, ktorého sa daná správa týka. O toto sa postará iná položka, konkrétne rc_pacienta, ktorá ukazuje do tabuľky pacient.
Oba odkazy sú vedené na základe primárnych kľúčov cudzích tabuliek.

Pre vytvorenie jedinečného kľúča bolo potrebné využiť viac záznamov a teda vytvoriť viacnásobný primárny kľúč. Primárny kľúč `id_spravy` je teda tvorený pomocou dvoch vyššie zmienených stĺpcov, id_lekara a rc_pacienta, doplnené o dátum, dátového typu `datetime` (obsahuje aj časovú zložku). Je potrebné riešiť to týmto spôsobom, keďže pacient bude mať od daného lekára viac správ a tým pádom by sa mohla vytvárať redundancia. Pridaním dátumu sa však každý záznam tabuľky stáva jedinečným.

Ďalšie položky tabuľky sú typ, pre rozlišovanie lekárskej správy od predpísaných receptov, samotný text správy a riadky. Riadky sú veľmi dôležité, keďže vďaka nim sa bude vypočítavať odriadkovanie pre dotlač do zdravotnej dokumentácie spomenutej v kapitole 2.2.2.

Rodné číslo pacienta, id lekára a typ správy sú v tabuľke zastúpené dátovým typom `varchar`. Text správy je vzhľadom na jej mohutnosť typu `text` a riadky sú typu `integer`.

Frázy

id_lekara	**skratka_frazy**	definicia

Tabuľka 3.8: Tabuľka Frázy

Tabuľka frázy je využívaná výhradne pri písaní lekárskych správ. Keďže potrebujeme zaručiť rýchle vyhľadávanie medzi užívateľmi zadefinovanými frázami, budeme ich identifikovať pomocou užívateľmi zadefinovaných kódov. Tieto kódy sú zastúpené v stĺpci skratka_frazy.

Avšak aby sme sa vyhli redundancii, pretože je možné, že dvaja užívatelia si zvolia rovnaké skratky pre svoje frázy, je táto tabuľka identifikovaná nielen pomocou id_frazy, ale aj id_lekara. Tieto dve položky tvoria viacnásobný primárny kľúč `id_frazy`.

Ďalej tabuľka obsahuje už iba definíciu, čiže text frázy. Definícia má dátový typ `text`, zatiaľ čo id lekára a skratka frázy `varchar`. Id lekára zastupuje cudzí kľúč ukazujúci do tabuľky personál.

BOP

id_lekara	**rc_pacienta**	**datum**	hodnota

Tabuľka 3.9: Tabuľka BOP

Tabuľka BOP(Bleeding On Probing, viď. kapitola 2.2.2) uchováva hodnoty BOP indexov. Táto tabuľka je využívaná výhradne v hygienickej časti programu.

Pre správnu orientáciu je pri jednotlivých hodnotách potrebné uchovávať informáciu o lekárovi a pacientovi, a tiež dátum, kvôli možnosti porovnávania histórie jednotlivých

záznamov. Tabuľka má tým pádom štyri položky, a to: id_lekara, rc_pacienta, dátum a hodnotu. Všetky položky okrem dátumu (`datetime`), majú dátový typ `varchar`.

Pre jedinečnosť záznamu je potrebné zobrať do úvahy ako lekára a pacienta, tak i dátum. Tieto tri položky teda tvoria aj primárny kľúč tabuľky, `bop_id`.

Kalendár

id_lekara	**rc_pacienta**	**datum**	cas_zaciatku	cas_konca	
		typ_osetrenia	poznamka		

Tabuľka 3.10: Tabuľka Kalendár

Jednotlivé záznamy objednávkového kalendára sú vedené v tabuľke kalendár. V kalendári je potrebné rozlíšiť nielen hygienickú a dentálnu časť, ale aj jednotlivých lekárov a hygienikov. Preto jednou z položiek kalendára je id_lekara, odkazujúca do tabuľky personál.

Okrem lekára je samozrejme potrebné uchovávať informáciu o tom, ktorého pacienta sa daný záznam týka a tiež dátum s časovým rozmedzím prehliadky alebo zákroku. O toto sa postará položka rc_pacienta, odkazujúca do tabuľky pacient, a tiež položky datum, cas_zaciatku a cas_konca.

Jednotlivé záznamy je možné doplniť o typ ošetrenia a poznámku.

Id lekára, rodné číslo pacienta a typ ošetrenia sú typu `varchar`. Dátum je dátového typu `date` a čas začiatku a čas konca `time`. Poznámka je typu `tinytext`. Primárny kľúč je opäť viacnásobný, `id_zapisu`, skladajúci sa z položiek id_lekara, rc_pacienta a datumu.

Vykazovanie

datum	**lekar**	**pacient_rc**	**kod_diagnozy**	**kod_vykonu**	**kod_lokality**	
		pocet	typ_uhrady	doplatok_pacienta		

Tabuľka 3.11: Tabuľka Vykazovanie

Tabuľka vykazovanie sa využíva pre uchovávanie záznamov o úkonoch vykonaných na pacientoch. Uchováva nielen informáciu o kóde vykonaného úkonu, ale aj o lokalite na ktorú bol aplikovaný a diagnózu, ktorá bola pacientovi diagnostikovaná. Tieto výkazy sa raz do mesiaca elektronicky vykazujú (vyfakturujú) zmluvným poisťovniam.

Vykazovanie sa pomocou cudzích kľúčov prepája s mnohými inými tabuľkami:

Stĺpec	=>	**Tabuľka**
lekar	=>	Personál
pacient_rc	=>	Pacient
kod_diagnozy	=>	Diagnózy
kod_vykonu	=>	Výkony
kod_lokality	=>	Lokality

Všetky vyššie uvedené položky doplnené o dátum tvoria viacnásobný primárny kľúč, `id_vykazu`.

Okrem neho je tabuľka doplnená ešte o počet daných výkonov, typ úhrady (celé prepláca poisťovňa, spoluúčasť poisťovne, pacient hradí všetko sám...) a výšku prípadného doplatku pacienta.

Všetky stĺpce tabuľky sú dátového typu `varchar`, okrem dátumu, počtu výkonov a doplatku pacienta, pričom dátum je typu `datetime`, počet je typu `integer` a doplatok pacienta `float`.

3.3 Fyzické modelovanie

Cieľom fyzického modelovania je navrhnúť fyzické uloženie tabuliek (ktoré sú výsledkom logického návrhu) využitím konkrétneho *systému riadenia bázy dát* tak, aby boli dosiahnuté čo najlepšie výkonnostné parametre.

Fyzická schéma aplikácie je reprezentovaná MySQL serverom a entity modelom, generovaným z existujúcej databázy pomocou Entity Frameworku. Návrh tejto databázy bol popísaný v predchádzajúcich dvoch kapitolách (viď. 3.1 a 3.2).

3.4 Diagram prípadov užitia (Use Case diagram)

Model prípadov užitia jazyka UML popisuje požiadavky na operácie, ktoré nad dátami popísanými ER diagramom musia byť umožnené vykonávať. Tento diagram zobrazuje pohľad na operácie aplikácie a ich operácie nad dátami databázy z hľadiska koncových užívateľov.

V našom programe sa pri návrhu Use Case diagramu zameriame na režimy prístupu, popísané v kapitole 2.2.1, doplnené o rolu administrátora (správcu databázy). V kapitole 2.2.1 boli popísané tri rôzne režimy prístupu, takže predpokladáme tri odlišné charakteristiky zamestnancov. Zamestnanci, pracujúci priamo s informačným systémom, sa delia na dentistov (zubárov), zubných hygienikov a ostatný personál (recepcia). Dentisti a hygienici majú *read/write* práva nad databázou a konkrétnymi operáciami aplikácie vytýčenými v kapitole 2.2.1. Podľa nich bol vypracovaný diagram prípadov užitia priložený v prílohe B.

Kapitola 4

Implementácia

Stomatologický informačný systém bol implementovaný vo vývojovom prostredí `Microsoft Visual Studio`. `Microsoft Visual Studio` môže byť využité ako pre vývoj konzolových aplikácií, tak pre aplikácie s grafickým užívateľským rozhraním (náš prípad), či webové aplikácie. Je to veľmi príjemné vývojové prostredie, ktoré dokáže oddeliť implementáciu GUI od ostatného kódu, rovnako tak aj databázové schémy aplikácie. `Microsoft Visual Studio` obsahuje podporu pre všetky jazyky a technológie, ktoré boli zvolené pre implementáciu informačného systému (viď. podkapitola 2.3). Zároveň je možné jeho funkčnosť vylepšovať rôznymi rozšíreniami.

Samotná implementácia aplikácie sa dá rozdeliť na tri časti:

1. implementáciu grafického užívateľského rozhrania,
2. implementáciu vnútorných operácií,
3. implementáciu komunikácie s databázou.

Časť tohoto popisu je bližšie definovaná v nasledujúcich kapitolách.

4.1 Implementácia grafického užívateľského rozhrania

Grafické užívateľské rozhranie je implementované pomocou knižníc `Windows Presentation Foundation` (ďalej iba `WPF`). Tieto knižnice boli rozšírené ešte dvoma toolkitmi, konkrétne pomocou `Extended.Wpf.Toolkit` a `MahApps.Metro` toolkitom.

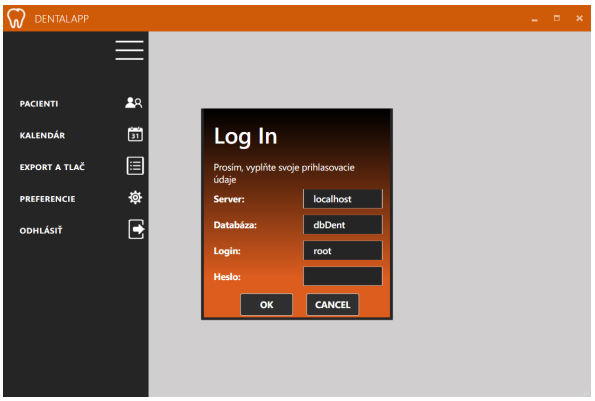
`MahApps.Metro` toolkit[1] je využitý pre implementáciu moderne vyzerajúceho GUI. Implementuje širokú škálu farebných kombinácií, vždy v dvoch témach, svetlej (light) a tmavej (dark theme).
Dentálna aplikácia bola implementovaná s rozhraním v oranžovej farbe s tmavou témou. Tmavé pozadie so svetlejšími farbami využitými pre jednotlivé objekty, dáva podstatné časti programu do popredia, čo umožňuje rýchlejšiu orientáciu na jednotlivých layoutoch a tým zvyšuje produktivitu koncového užívateľa. Okrem už spomenutej oranžovej je využitých približne iba päť ďalších farebných odtieňov a tým aplikácia nadobúda jednoduchý, prvkami nepresýtený vzhľad.

Okrem základného `MahApps.Metro` toolkitu je využité aj `MahApps.Metro.SimpleChildWindow`[2]. Pomocou neho sú implementované "podokná" vo vnútri "rodičovských okien".

[1] Bližší popis toolkitu na stránkach https://github.com/MahApps/MahApps.Metro
[2] Bližší popis na stránkach https://github.com/punker76/MahApps.Metro.SimpleChildWindow

Tento typ okien je využitý napríklad pre login (obrázok 4.1) a namiesto ďalších vyskakovacíh okien.

Narozdiel od `MahApps.Metro` toolkitu, ktorý definuje vzhľad a prípadne správanie jednotlivých grafických objektov, `Extended.Wpf.Toolkit` WPF roširuje priamo novými grafickými objektami. Celé rozšírenie je bližšie definované na internetových stránkach extended toolkitu[3]. Konkrétne v aplikácii sú využité objekty `CheckComboBox` a `TimePicker`.



Obr. 4.1: Prihlasovacie okno aplikácie implementované pomocou `MahApps.Metro.SimpleChildWindow`

Celkové GUI aplikácie bolo navrhnuté podľa návrhu v podkapitole 2.4. Od tohto návrhu sa nezišlo. Celá aplikácia je implementovaná vrámci jedného jediného hlavného okna, kde sa iba "odkrývajú" a "skrývajú" jednotlivé layouty. Okrem tohoto okna sú v programe implementované iba `SimpleChildWindow` spomenuté vyššie a `Message` či `Dialog Boxy` implementované s využitím `MahApps.Metro` toolkitu.
Vľavo v hlavnom okne je implementované "vysúvateľné" menu. Toto menu rozdeľuje aplikáciu na štyri hlavné časti, v našom prípade layouty:

- **Pacienti** - layout vypisuje zoznam pacientov, umožňuje vytvorenie nových, ich editáciu, zmazanie či vyhľadávanie podľa priezviska pacienta,
- **Kalendár** - layout kalendára umožňuje vkladanie nových záznamov a zobrazuje nielen kalendár, alebo možno lepšie povedané plánovač, prihlásenej osoby, ale aj ostatných dentistov a hygienikov, viac však ku kalendáru v podkapitole 4.3,
- **Export a tlač** - export a tlač layout je využívaný k najmä pri vykazovaní poisťovniam, ale aj pri tlači uzávierky či iných prehľadov spojených s úkonmi vykonanými na pacientoch (viac podkapitola 4.4),
- **Preferencie** - layout preferencií ponúka možnosť úpravy formulára anamnézy, úpravy dát vo vybraných tabuľkách databázy, vyhľadávanie v ich záznamoch a dokonca export celej tabuľky, či jej časti.

Aplikácia má ešte jeden layout, layout zdravotnej karty, na ktorý však neodkazuje menu, ale dá sa do neho dostať odkazom z `DataGridu` pacientov v layoute "pacienti".

[3] http://wpftoolkit.codeplex.com

4.2 Prihlasovanie k databáze

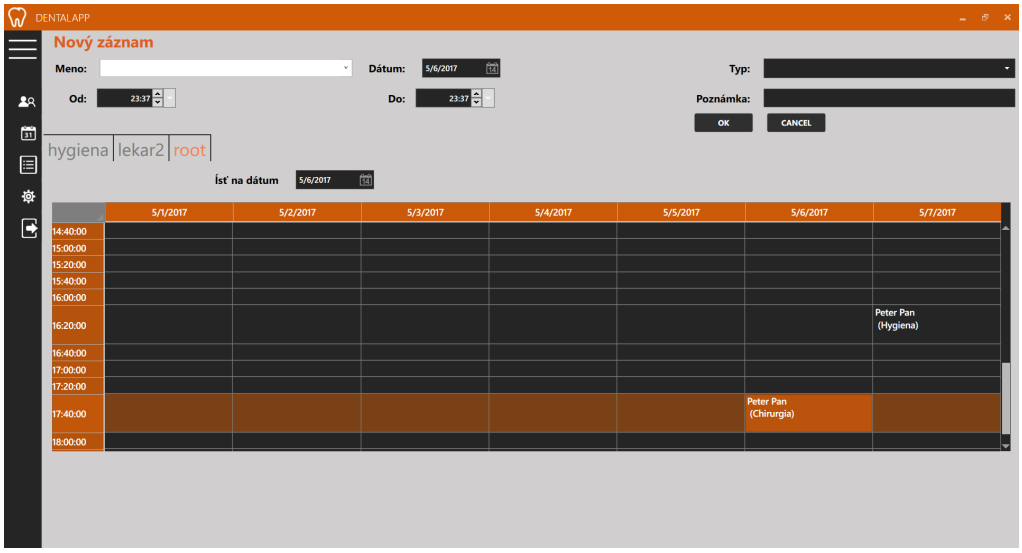
Komunikáciu aplikácie s dátami uloženými v databáze rieši Entity Model, vytvorený pomocou Entity Frameworku. Ako už bolo spomenuté, Entity Model bol vytváraný pomocou prístupu `DatabaseFirst`, ktorého postup je popísaný v knihe [4], na stránkach 12 až 21. Pri tomto type vytvárania je editovaný aj konfiguračný súbor aplikácie (`App.config`), ktorý je doplnený o `connectionString`. Avšak z dôvodu viacužívateľského využívania aplikácie a samotnej ochrane citlivých dát, je potrebné vyriešiť prihlasovanie k serveru a konktrétnej databáze inak.

`ConnectionString` je preto zmazaný z konfiguračného súboru aplikácie a miesto neho je vytvorená nová trieda `dbEntities`, ktorá dedí vlastnosti `DbContext`, a v ktorej je vytváraný nový `connectionsString`. Tento `string` je doplnený o prihlasovacie informácie prevzaté pri logine užívateľa aplikácie (priamo zo `SimpleChildWindow` loginu, obrázok 4.1).
Po vytvorení validného prihlasovacieho reťazca je nadväzované skúšobné spojenie s databázou. Ak prebehne bez chyby, je aplikácia k databáze opätovne prihlasovaná, a je ukladaný adekvátny prihlásený užívateľ, podľa ktorého sa ďalej odvíja rozhranie aplikácie.

Celá spomínaná implementácia je obsiahnutá v triede `manageConnection.cs`.

4.3 Kalendár

Kalendár je implementovaný ako `DataGrid` s dvoma `headermi`, kde `header` stĺpcov viaže jednotlivé dni daného týždňa a `header` riadkov viaže čas v jednotlivých dňoch, a to od 8 ráno do 18 večer, vždy v dvadsať minútových intervaloch (predpokladaný čas jednotlivých prehliadok/zázkrokov).



Obr. 4.2: Kalendár/plánovač aplikácie

Samotné záznamy vkladané do kalendára sú dynamicky viazané pomocou dvojdimenzionálneho poľa. Táto operácia je umožnená rozšírením `Gu.Wpf.DataGrid2D`, ktoré má definovanú funkciu pre 2D viazanie `Gu.Wpf.DataGrid2D.ItemsSource.SetArray2D(DataGrid element, Array value)`. Okrem toho ďalej ponúka funkcie pre dynamické viazanie `headerov`, a to ako stĺpcov, `Gu.Wpf.DataGrid2D.ItemsSource.SetColumnHeadersSource(Data-`

`Grid element, IEnumerable value)`, tak i riadkov, `Gu.Wpf.DataGrid2D.ItemsSource.SetRowHeadersSource(DataGrid element, IEnumerable value)`.

Ako už bolo spomenuté v kapitole 4.1, GUI layout kalendára zobrazuje nielen `DataGrid` prihláseného užívateľa, ale aj ďalších dentistov, či hygienikov. Jednotlivé záznamy kalendára sú uchovávané v databáze v tabuľke `kalendar`. Tieto záznamy sú najskôr parsované podľa dátumu, pričom vyberané sú iba záznamy s dátumom v rozpätí pondelok až nedeľa daného týždňa. Neskôr sú tieto záznamy kontrolované, či sedia k danému užívateľovi, a podľa toho sú vkladané do `DataGridov` kalendára. Zároveň je možné pomocou `DatePickeru` preskočiť na vybraný dátum v `DataGride` jednotlivých kalendároch. Implementovaný kalendár je zobrazený na obrázku 4.2.

4.4 Vykazovanie poisťovniam

Vykazovanie poisťovniam je implementované podľa metodického usmernenia *ÚZDS* s číslom *MU 5/1/2015*, spomínané v kapitole 2.2.2 (Vykazovanie poisťovniam). Podľa tohoto usmernenia bolo naimplementované *"Vykazovanie výkonov v ambulantnej starostlivosti"*.

Jednotlivé dávky sú vykazované formou textového súboru bez formátovania a diakritiky. Ako oddeľovací znak je využívaný znak "|", ktorý sa používa aj na konci riadku.
Dávka sa skladá z troch častí:

- **Identifikácia dávky** - prvý riadok textového súboru, ktorý obsahuje nasledujúce položky:
  - charakter dávky (N,O,A...),
  - typ dávky (751a),
  - IČO odosielateľa dávky,
  - dátum odoslania dávky (vo formáte RRRRMMDD),
  - poradové číslo dávky,
  - počet dokladov (viet tela dávky),
  - počet médií (vykazovaných súborov),
  - číslo média (poradové číslo vykazovaného súboru),
  - kód zdravotnej poisťovne a pobočky, s ktorou má PZS uzatvorenú zmluvu (pr. 2509).
- **Záhlavie dávky** - druhý riadok vykazovacieho súboru s nasledujúcimi ôsmimi položkami:
  - identifikátor PZS (v databáze ako kód ambulancie),
  - kód PZS (v databáze ako kód zubnej ambulancie),
  - kód lekára (pod rovnakým názvom aj databáze),
  - úväzok lekára k špecializovanej zdravotnej starostlivosti, za ktorú sa dávka vykazuje (pr. 50% zapisovaných ako 0.50),
  - zúčtovacie obdobie v tvare RRRRMM
  - typ starostlivosti (843 za zubného lekára),
  - číslo faktúry (len numerické znaky, max 10 znakov),

– mena (EUR).

- **Vety tela dávky** - následuje za prvými dvoma riadkami a obsahuje 24 položiek, z ktorých sú pri zubnej starostlivosti vypĺňané nasledujúce:

  1. poradové číslo riadku,
  2. deň v mesiaci,
  3. rodné číslo poistenca,
  4. meno poistenca (v tvare priezvisko meno),
  5. kód diagnózy,
  6. kód výkonu,
  7. počet výkonov,
  8. kód lokality zubu.

  Nasledujúce položky sú buď nepovinné alebo pre súkromnú dentálnu kliniku nepotrebné a teda vo výkaze budú prázdne.

Podľa tejto metodológie sú vytvárané vykazované súbory, vždy s príponou udávajúcou poradie výkazu pre danú poisťovňu. Exportovaný vykazovací súbor potom môže vyzerať následovne:

```
  N|751a|318438**|20170509|1|4|1|1|25|
P91999000001|B12345678|P91999|1.00|201703|843||EUR|
1|16|985602****|Kratochvilová Terka|K10.9|V 20|1|16|||||||||||||||||
2|16|985602****|Kratochvilová Terka|K10.9|V 31|1|20|||||||||||||||||
3|16|985602****|Kratochvilová Terka|K10.9|V 41|1|16|||||||||||||||||
4|16|985602****|Kratochvilová Terka|K10.9|V 41|1|20|||||||||||||||||
```

V aplikácii je export výkonov implementovaný pod layoutom “export a tlač”, kde teda nie je možné využiť pre filtrovanie žiadne iné položky, okrem výberu jednej poisťovne a dátumov od a do. Tento layout je zobrazený na obrázku 5.1.

4.5 Vyhľadávanie v databáze liekov

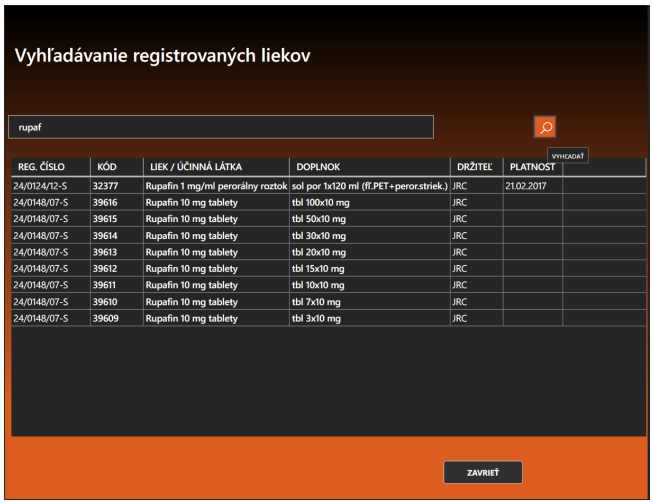
Prístup do registrovanej databázy liekov je implementovaný cez stránku www.sukl.sk, čiže stránku Štátneho ústavu pre kontrolu liečiv (ďalej iba SÚKL).

Najskôr je získaný `HTML kód` zo stránok SÚKL, ktorý je odpoveďou na dotaz kladený na tieto stránky, obsahujúci klientom vyhľadávaný reťazec. Zhoda k tomuto reťazcu je vyhľadávaná ako podreťazec v názvoch alebo účinných látkach registovaných liekov. O toto vyhľadávanie sa však stará samotná stránka SÚKL.
Táto operácia je v aplikácii implementovaná vo funkcii `List<drugInfo> getSearchedDrugs (string searchStr, MetroWindow mainWin)`, nachádzajúcej sa v triede `getDrugsInfo.cs`, ktorej sú predávané 2 parametre, a to reťazec ktorý klient vyhľadáva (prevzatý z `TextBoxu` na obrázku 4.3) a ukazateľom na hlavné okno aplikácie. Vo vnútri funkcie je najskôr pripravený dotaz na stránky SÚKL v tvare `string`, ktorý je doplnený o vyhľadávaný reťazec.

Po vytvorení dotazu je zaslaný `HttpWebRequest` obsahujúci práve spomenutý dotaz, a následne sa čaká na prijatie `HttpWebResponse`. `HttpWebResponse` je ďalej parsovaný a je z neho získaný `stream` pomocou funkcie `GetResponseStream()`, ktorý je vzápätí prečítaný do `stringu`.

`HttpWebRequest` a `HttpWebResponse` sú definované knižnicou `System.Net`. Väčšia časť funkcie `getSearchedDrugs(string searchStr, MetroWindow mainWin)` je prevzatá z internetu, zo stránky http://stackoverflow.com/questions/16642196/get-html-code-from-website-in-c-sharp.

String obsahujúci `HTML kód` je ďalej parsovaný funkciou `List<drugInfo> parseHtmlString(string htmlStr)`, ktorej je predávaný ako jediný parameter. V tejto fukncii sú z neho "vyťiahnuté" iba požadované informácie.
V tejto funkcii je využívané rozšírenie `HtmlAgilityPack`, ktoré je vlastne `.NET` knižnicou, umožňujúcou parsovať `HTML` súbory. Najskôr je teda potrebné vytvoriť zo `stringu` obsahujúceho `HTML` kód validný `HTML` súbor, pomocou funkcie `LoadHtml(string html)`. Až nad validným `HTML` súborom môžeme vykonávať parsovacie operácie, ku ktorým je využitý dotazovací jazyk LINQ.
Najskôr sú vyhľadané všetky tabuľky s ich nasledovníkmi, ktoré sú získané pomocou funkcie `DocumentNode.DescendantsAndSelf(string name)`, ktoré jednotlivé zhody vrátia s typom `IEnumerable<HtmlAgilityPack.HtmlNode>`. `IEnumerable` rozhranie uchováva informácie formou jednoduchého iterátora nad kolekciou určeného typu, v našom prípade `HtmlAgilityPack.HtmlNode`.
Ďalej sa prechádza jednotlivými položkami iterátora, až pokým neprídeme k tabuľke s triedou `searchTable`, ktorá obsahuje nami požadované informácie. Po vnorení do tejto tabuľky prechádzame jej jednotlivé prvky z ktorých vyberieme maximálne prvých pätnásť alebo menej, a tie následne ukladáme do zoznamu s elementami typu `drugInfo`.



REG. ČÍSLO	KÓD	LIEK / ÚČINNÁ LÁTKA	DOPLNOK	DRŽITEĽ	PLATNOSŤ
24/0124/12-S	32377	Rupafin 1 mg/ml perorálny roztok	sol por 1x120 ml (fľ.PET+peror.striek.)	JRC	21.02.2017
24/0148/07-S	39616	Rupafin 10 mg tablety	tbl 100x10 mg	JRC	
24/0148/07-S	39615	Rupafin 10 mg tablety	tbl 50x10 mg	JRC	
24/0148/07-S	39614	Rupafin 10 mg tablety	tbl 30x10 mg	JRC	
24/0148/07-S	39613	Rupafin 10 mg tablety	tbl 20x10 mg	JRC	
24/0148/07-S	39612	Rupafin 10 mg tablety	tbl 15x10 mg	JRC	
24/0148/07-S	39611	Rupafin 10 mg tablety	tbl 10x10 mg	JRC	
24/0148/07-S	39610	Rupafin 10 mg tablety	tbl 7x10 mg	JRC	
24/0148/07-S	39609	Rupafin 10 mg tablety	tbl 3x10 mg	JRC	

Obr. 4.3: Vyhľadávanie v databáze liekov

`DrugInfo` je implementovaná štruktúra obsahujúca sedem elementov typu `string`, uchovávajúca informácie o liekoch. Zoznam (`LIST<T>`), kde `T` je `drugInfo`, je využívaný ako zdroj emelentov (`ItemSource`) `DataGridu` zobrazujúceho vyhľadané lieky, ktorý je výsledkom funkcie `List<drugInfo> parseHtmlString(string htmlStr)`. Okrem `DataGridom` zobrazených informácií uchováva `drugInfo` ešte odkaz na stránku s detailnejším popisom

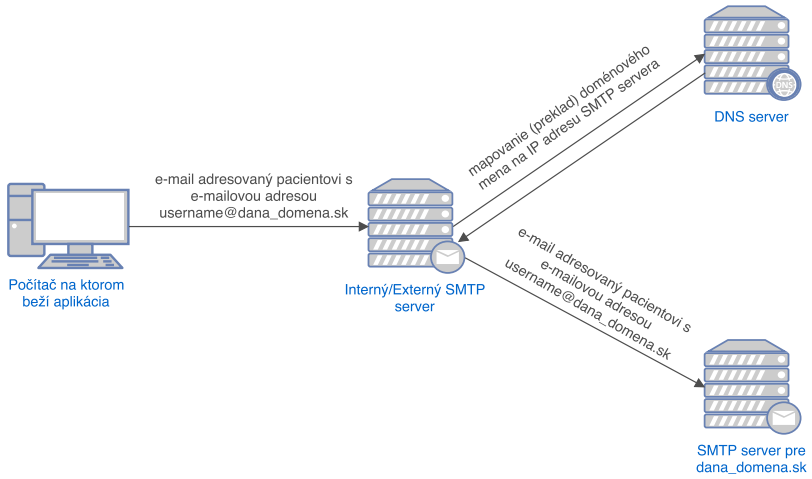
daného lieku, ktorá je načítaná predvoleným prehliadačom zákazníka, po dvojkliku na daný element `DataGridu`.

Implementované `SimpleChildWindow` pre vyhľadávanie liekov z rozparsovaným html kódom je zobrazené na obrázku 4.3.

4.6 Komunikácia so zákazníkmi

Implementovaná je zatiaľ iba e-mailová komunikácia so zákazníkmi, SMS komunikácia bude doplnená neskôr.

E-mailová komunikácia je implementovaná s využitím `smtp` protokolu. Tento protokol je definovaný knižnicou `System.Net.Mail`. Zasielanie správ pomocou tejto knižnice je zobrazené na obrázku 4.4.



Obr. 4.4: Zasielanie e-mailov pomocou knižnice `System.Net.Mail.SmtpClient`

Najskôr je potrebné vytvoriť `smtp klienta` pomocou `new SmtpClient(string host)`. Po vytvorení `klienta` je potrebné nastaviť parametre:

- `UseDefaultCredentials` - tento parameter je potrebné nastaviť na `false`, aby sme v ďalšom kroku mohli nastaviť vlastné preferencie,
- `Credentials` - nastavenie prihlasovacích údajov k e-mailovej adrese, z ktorej budú správy pre pacientov odosielané,
- `EnableSsl` - nastavenie šifrovania spojenia nastavíme na `true`,
- `Port` - nastavenie `čísla portu` na `587`, čo je `defaultný port` pre odosielanie pošty.

Po nastavení týchto parametrov prechádzame k vytvoreniu samotnej správy, ktorú chceme odoslať.

Odosielaná správa musí byť typu `MailMessage`, so správne nastavenou e-mailovou adresou odosielateľa a tiež príjemcu. Po inicializovaní správy nastavíme jej `Subject`, čiže

predmet správy a tiež `Body`, čo pokrýva vlastný text správy. Do textu správy je vložený dátum a čas prehliadky pacienta. Keď je správa vytvorená, odosiela sa pomocou `SmtpClient.Sent(MailMessage message)`.

K odosielaniu správ sa v aplikácii dostávame pri odhlásení, kde sa tesne pred odhlásením zobrazí `MessageBox` zisťujúci, či si klient želá odoslať upozornenia k nasledujúcemu dňu. Po kliknutí `OK` dochádza k odoslaniu správ.

Kapitola 5

Testovanie

Testovanie je podstatnou časťou vývoja informačného systému, ktoré vývojárom pomáha implementovanú aplikáciu zlepšovať a zdokonalovať, a tiež pomáha dosiahnuť zákazníkové predstavy a požiadavky presnejšie, bez zbytočného mrhania času nad vývojom niečoho nechceného.
Aplikácia implementovaná vrámci tejto bakalárskej práce je vyvýjaná s dôrazom kladeným na požiadavky zákazníka, a teda môžeme pri jej návrhu hovoriť o User-Centred design(e), ďalej spomínanom iba pod skratkou UCD. UCD je zamerané na optimalizáciu produktu s ohľadom na to, ako užívatelia chcú, či potrebujú produkt využívať. V tomto prípade sú budúci koncoví užívatelia zapojený do návrhu a vývoja projektu, a to najmä v procese testovania.

Testovanie s užívateľmi prebehlo viackrát, pričom sa testovala najmä GUI stránka aplikácie (podkapitola 5.1). Pri testovaní správnosti výstupov, podkapitola 5.2, nebol zákazník prakticky potrebný, keďže popis požiadavkov bol zreteľný a podložený reálnymi tlačenými či elektronickými dokumentami.

5.1 Testovanie z hľadiska GUI

Dá sa povedať, že testovanie z hľadiska GUI je podstatnou časťou testovania UCD. Práve nemu sú venované nasledujúce podkapitoly.

5.1.1 Testovanie použiteľnosti

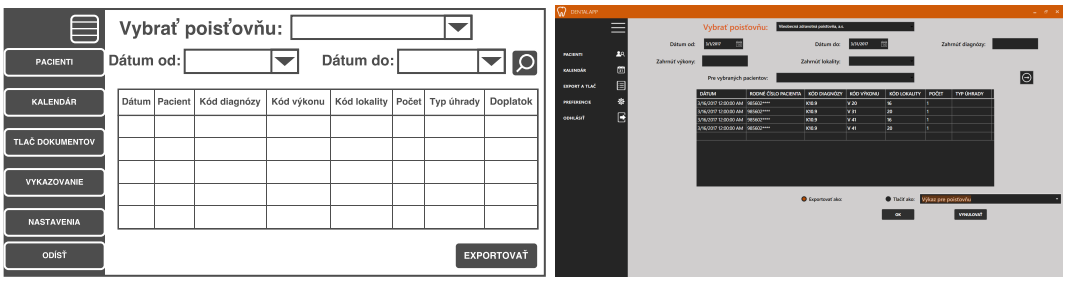
Za testovanie použiteľnosti môžeme pokladať testovanie prototypov, či mockup návrhov na reálnych užívateľoch. Toto testovanie je zamerané na porozumenie navrhnutému rozhraniu.
Využitá testovacia technika je zvyčajne takzvaný Think-aloud protokol. Tento protokol je presne tým, čo už jeho názov prezrádza. Testovaný užívateľ nahlas komentuje, čo práve robí, aké kroky sa snaží podniknúť, poprípade aké dáta získať.

V našom prípade boli testovaný dvaja užívatelia, pričom obaja budú mať v dokončenej aplikácii úlohu dentistov. Boli vybratý pre výrazný vekový rozdiel a tiež preto, že sa jedná o muža a ženu, a vďaka týmto rozdieľom bola očakávaná veľká diverzita reakcií.
Užívateľom boli postupne odkrývané rôzne náčrty zobrazujúce rôzne situácie, ktoré môžu v aplikácii nastať. Tieto mockup náčrty boli kreslené ručne, ale grafická ukážka jedného z navrhnutých layoutov je prekreslená na obrázku 5.1. Mockupy boli užívateľovi predávané podľa toho, na ktorú "položku" návrhu práve "klikol". Užívatelia mali jedinú úlohu, a to ko-

mentovať všetko čo robia. Nedostali presné zadanie, ktoré museli splniť (v tomto ohľade im bola dopriata volnosť), keďže obaja vedeli čo presne budú chcieť s aplikáciou robiť.

Počas testovania sa dospelo k prekvapivému záveru, obaja užívatelia reagovali totižto na rovnaké operácie takmer zhodným spôsobom. Bolo to zrejme spôsobené tým, že napriek tomu, že rozhranie aplikácie je diametrálne odlišné od rozhrania v ktorom boli zvyknutí pracovať, malo navrhnuté rozhranie logickú následnosť a členenie prvkov.
Jediný objekt, ktorý zostal oboma testovanými užívateľmi neobjasnený, bolo tlačidlo na skrytie/vysunutie menu. Po vysvetlení jeho funkčnosti však boli obaja užívatelia jeho myšlienkou nadšený, a tak bolo toto tlačidlo ponechané.

Počas tohoto testovania tiež došlo k malej zmene layoutov. V prvotnom mockup návrhu obsahovalo menu položiek šesť, konkrétne pacienti, kalendár, tlač dokumentov, vykazovanie, nastavenia a odísť. Avšak vo finálnej verzii obsahuje menu už iba päť položiek. Pri testovaní totiž vysvitlo, že na niektoré operácie z layoutu tlač, je potrebné odkazovať priamo zo zdravotnej karty. Po ich presunutí do zdravotnej karty zostala v layoute tlač už iba jedna položka, tlač uzávierky. Po bližšiom priblížení práve spomínanej uzávierky bolo zistené, že je možné ju pridať do layoutu vykazovania, a vytvoriť tak layout export a tlač.



Obr. 5.1: Prekreslený originálny mockup layoutu vykazovanie a jeho finálna verzia

5.1.2 User experience testing

Narozdiel od testovanie použiteľnosti, boli pri tomto testovaní presne zadané úlohy, ktoré musel užívateľ splniť. Výpis jednotlivých úloh:

1. spustiť aplikáciu,
2. prihlásiť sa k databáze,
3. vytvoriť nového pacienta,
4. zmeniť e-mailovú adresu iného pacienta na vlastnú,
5. evidovať anamnézu pre novovytvoreného pacienta,
6. vytvoriť novú frázu,
7. exportovať tabuľku diagnózy,
8. tlačiť 5. až 10. element tabuľky výkony, zoradené zostupne podľa ID úkonu,
9. pridať novú správu do zdravotnej karty predtým vytvoreného pacienta, využívajúc novovytvorenú frázu,

10. v prípade, že ide o dentistu, evidovať výkon vykonaný na tomto pacientovi, a zároveň evidovať výkon u ďalších dvoch pacientov,
11. v prípade, že sa jedná o hygienika, evidovať vymyslený BOP index pacienta,
12. upraviť správu ktorá bola novému pacientovi vytvorená a vložená a dať ju vytlačiť ako "lekársku srávu",
13. v prípade, že ide o dentistu, vyhľadať kontraindikácie lieku "algifen",
14. v prípade, že ide o dentistu, vytvoriť výkazy pre všetky poisťovne z daného dňa,
15. objednať pacienta, ktorému sme menili e-mail, na nasledujúci deň,
16. zmazať novovytvoreného pacienta,
17. odhlásiť sa a zaslať upozornenia na prehliadky k nasledujúcemu dňu,
18. skontrolovať svoju e-mailovú adresu kvôli novému e-mailu.

V tomto testovaní užívateľ iba dostával zadania, inak mu nebolo nijak napomáhané pri ich riešení. Počas riešenia problémov boli na užívateľoch pozorované viaceré faktory, najmä teda čas, za ktorý sa užívateľovi podarilo jednotlivé zadania vyriešiť. Z časového rozsahu sa dá usúdiť, či je rozhranie aplikácie pre užívateľa zreteľné a či ho dokáže efektívne využívať. Ďalej boli pri testovaní sledované reakcie užívateľa - či mu aplikácia príde atraktívna alebo naopak, či má pocit, že má kontrolu nad produktom a nebojí sa vykonávať jednotlivé operácie s tým, že to "pokazí", atď.

Na konci testovania bol zaznamenaný feedback od jednotlivých užívateľov. Boli testovaný traja ľudia, dvaja dentisti a jeden hygienik. Výsledky tohoto testovania sú zaznamenané v tabuľke 5.1.

V tabuľke sú fajkou označené úlohy, ktoré boli vyriešené ihneď, a čas ich riešenia sa stával irelevantný. Prázdna množina zase označuje úlohy, ktoré pre daného testovaného užívateľa neboli určené.

Zo samostatných výsledkov tabuľky môžeme usúdiť, že aplikácia je z väčšej časti zrozumiteľná a jej rozhranie dostatočne intuitívne. Najväčšou prekážkou pri plnení úloh bola úloha číslo 7, kde mal užívateľ exportovať tabuľku diagnózy. Avšak táto operácia sa nenachádzala v layoute "export a tlač", ale v layoute "preferencie", kde je možné editovať jednotlivé tabuľky databázy. Tým pádom bol názov layoutu zavádzajúci, ako užívateľ s testovacím menom Dentista1 poznamenal vo svojom feedbacku a lákal užívateľa hľadať riešenie práve tam. Tento problém bol ihneď vyriešený. Názov bol zmenený na "Dokumenty a výkony", ktorý jasnejšie popisuje, čo daný layout obahuje.

Ostatné úlohy prebehli viac-menej bezproblémovo. Najdlhšie odozvy boli zaznamenané u hygienika, avšak jedná sa o osobu najmenej znalú počítačov. Riešenie niektorých úloh mu trvalo dlhšie, pretože užívateľ sa ešte nestretol s podobným riešením daného problému, a tak bol v určitej nevýhode oproti ďalším dvom testujúcim. Napriek tomu je vyhodnotenie tohoto užívateľa veľmi dobré a dá sa teda predpokladať, že pri dlhodobejšom používaní aplikácie nebude čeliť podobným problémom.

5.2 Testovanie správnosti výstupov

Ako bolo už vyššie spomenuté, testovanie správnosti výstupov prebehlo takmer bez účasti zákazníka. Jednotlivé časti programu boli priebežne testované a porovnávané s ukáž-

Úlohy\Užívatelia	Dentista1	Dentista2	Hygienik
1	✓	✓	✓
2	✓	✓	✓
3	✓	✓	✓
4	✓	✓	✓
5	✓	✓	✓
6	do 1 minúty	✓	1-2 minúty
7	2 a viac min	2 a viac min	2 a viac min
8	✓	✓	1-2 minúty
9	✓	✓	do 1 minúty
10	✓	✓	∅
11	∅	∅	✓
12	✓	✓	✓
13	✓	✓	∅
14	✓	do 1 minúty	∅
15	do 1 minúty	✓	1-2 minúty
16	✓	✓	✓
17	✓	✓	✓
18	✓	✓	✓
Feedback	+ aplikácia je prehľadná, netreba sa preklikávať cez 1000 okien k jednotlivým operáciám − zmeniť názov položky menu export a tlač, je zavádzajúci	+ dávam do popredie nápovedy pri jednotlivých operáciách, sú veľmi nápomocné, pokým si na ne človek nezvykne − neviem	+− veľmi príjemný program, treba si naň však zvyknúť

Tabuľka 5.1: Tabuľka nameraných hodnôt získaných pri user experience testovaní

kovými dokumentami, či ich oficiálnym popisom. Chybové výstupy boli ihneď opravované a optimalizované. Po ukončení testovania jednotlivých častí, boli zákazníci o výstupoch týchto častí buď osobne informovaní, počas osobnej konzultácie, alebo v prípade exportovania dokumentov, boli zákazníci informovaní formou e-mailovej správy s adekvátnymi prílohami. Takto prebiehalo aj testovanie odosielania e-mailových notifikácií, ktoré boli zákazníkovi zasielané, až po finálnu verziu. V prípade, že zákazník s výstupom z akéhokoľvek dôvodu nesúhlasil, prešlo sa k upresneniu požiadavkov a opätovnej implementácii danej časti. Požiadavky zákazníka obsiahnuté v podkapitole 2.2.2 sú definované v ich finálnej verzii, nie je vidieť ich vývoj od prvotnej špecifikácie.

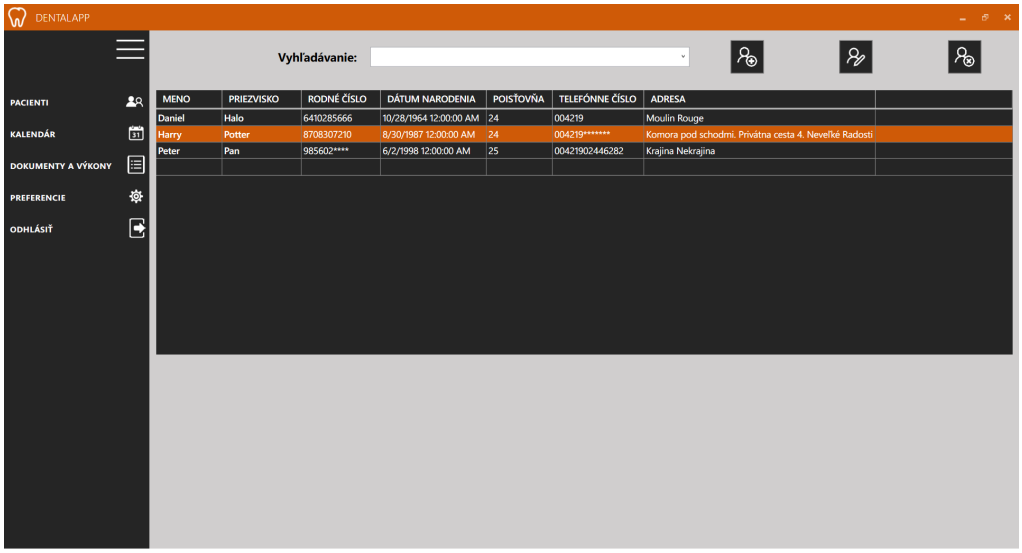
Kapitola 6

Záver

Zámerom tejto bakalárskej práce bolo navrhnúť a implementovať informačný systém zubnej ambulancie. Pre lepšie porozumenie problematike bolo potrebné nielen podrobne preštudovať požiadavky zákazníka, ale aj všeobecné požiadavky kladené na stomatologické informačné systémy vytvárané pre slovenský trh, a tiež už existujúce informačné systémy v tejto oblasti.

Z celkového hľadiska sa dá skonštatovať, že zadanie bakalárskej práce bolo splnené. Aplikácia je momentálne vo fáze dlhodobejšieho testovania, zameraného na doladenie posledných detailov, aby bolo možné v jej implementácii pokračovať. Od začiatku septemra 2017 je plánované reálne využívanie aplikácie zákazníkom a to s plne funkčnými doimplementovanými a dôkladne otestovanými rozšíreniami, popísanými v nasledujúcich podkapitolách.

Finálna verzia aplikácie po testovaní popísanom v kapitole 5 a následných úpravách, je zobrazená na obrázku 6.1.

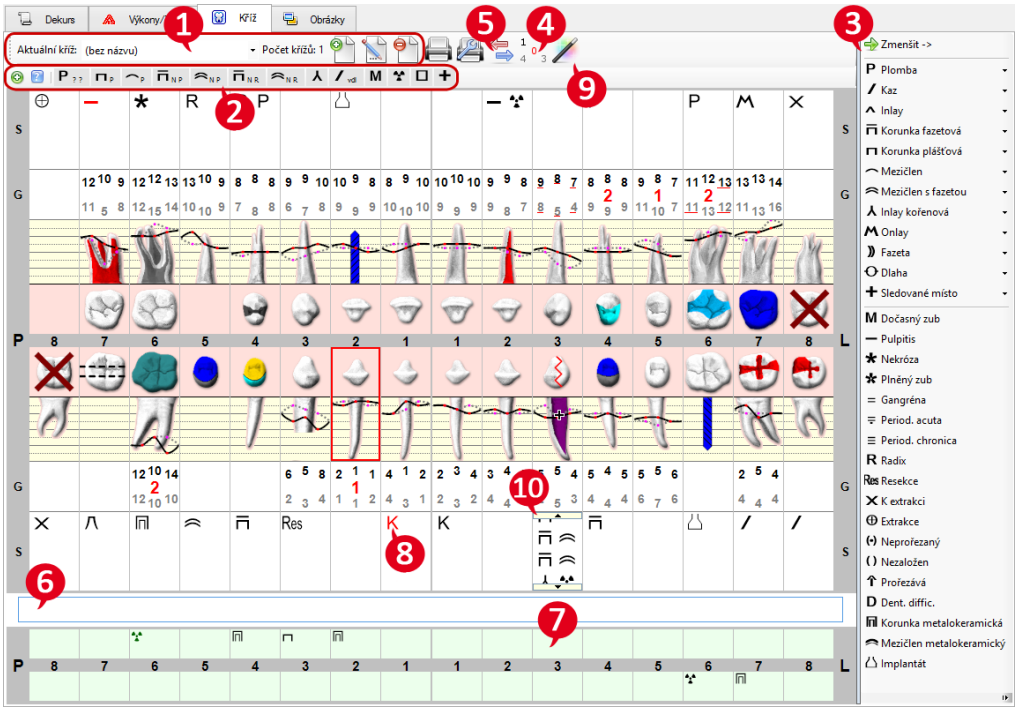


Obr. 6.1: Finálna verzia aplikácie zobrazujúca layout “pacienti”

6.1 Plánované rozšírenia

6.1.1 Grafický zubný kríž

Zubný kríž je textové, či grafické značenie zubov. V aplikácii je požadované grafické zobrazenie zubného kríža, zaznamenávajúceho stav chrupu jednotlivých pacientov. Grafický zubný kríž bude potrebné môcť upravovať rôznymi vyfarbeniami, popismi, či značkami, a to nielen v dentálnom, ale aj v hygienickom móde. Možno by bolo najlepšie viesť oddelený zubný kríž zvlášť pre hygienickú a zvlášť pre dentálnu časť aplikácie. Pre ukážku reálneho grafického zubného kríža je využitý zubný kríž stomatologického informačného systému DENTIST+[1], ktorý je prevzatý zo stránok manuálu DENTIST+ a je zobrazený na obrázku 6.2.



Obr. 6.2: Zubný kríž stomatologického informačného systému DENTIST+

6.1.2 Prepojenie so systémom Florida Probe

Počítačový integrovaný systém Florida Probe je kompletným systémom na parodonálne snímanie a mapovanie. Poskytuje interaktívne testovanie, ktoré rapídne zvyšuje akceptáciu parodontálnej liečby[2].

Florida Probe obsahuje software aj hardware. Hardware sa skladá z počítačovej sondy a footswitchu. Ich využívanie a komunikácia s vlastným softwarom vytvára výstup parodontálnej skúšky, ktorý bude potrebné pri jednotlivých pacientoch evidovať. Toto rozšírenie tak zlepší organizovanosť záznamov a zároveň značne zvýši produktivitu dentálneho hygienika.

[1] Webové stránky stomatologického IS DENTIST+ http://www.dentist.cz/
[2] Viac na stránke http://www.floridaprobe.com/fptv_fpsystem.htm

Literatúra

[1] Balling, D. J.; Lentz, A.; Zawodny, J. D.; aj.: *High Performance MySQL, 2nd Edition.* O'Reilly Media, Inc., 2008, ISBN 9780596101718.

[2] Bydžovský, J.: *Tabulky pro medicínu prvního kontaktu.* Praha: Triton, 2010.

[3] Dundálková, P.: *Dentální hygiena v parodontologii.* Diplomová práce, Masarykova univerzita, 2013.

[4] Singh, R. R.: *Mastering Entity Framework.* Packt Publishing Ltd., 2015, ISBN 9781784391003.

[5] Stedman, T. L.: *Stedman's Medical Dictionary.* Lippincott Williams & Wilkins, 2005, ISBN 9780781733908.

[6] Szabó, R. P.: *Databázové a informačné systémy - Databázová terminológia - relačný dátový model.* 2005, [Online; navštívené 18.4.2017].
URL http://spseke.sk/tutor/prednasky/dbs/rdm.html

[7] Zendulka, J.; Rudolfová, I.: *Databázové systémy. IDS. Studijní opora.* FIT VUT v Brně. 2006(rev. 2016).

Prílohy

Príloha A

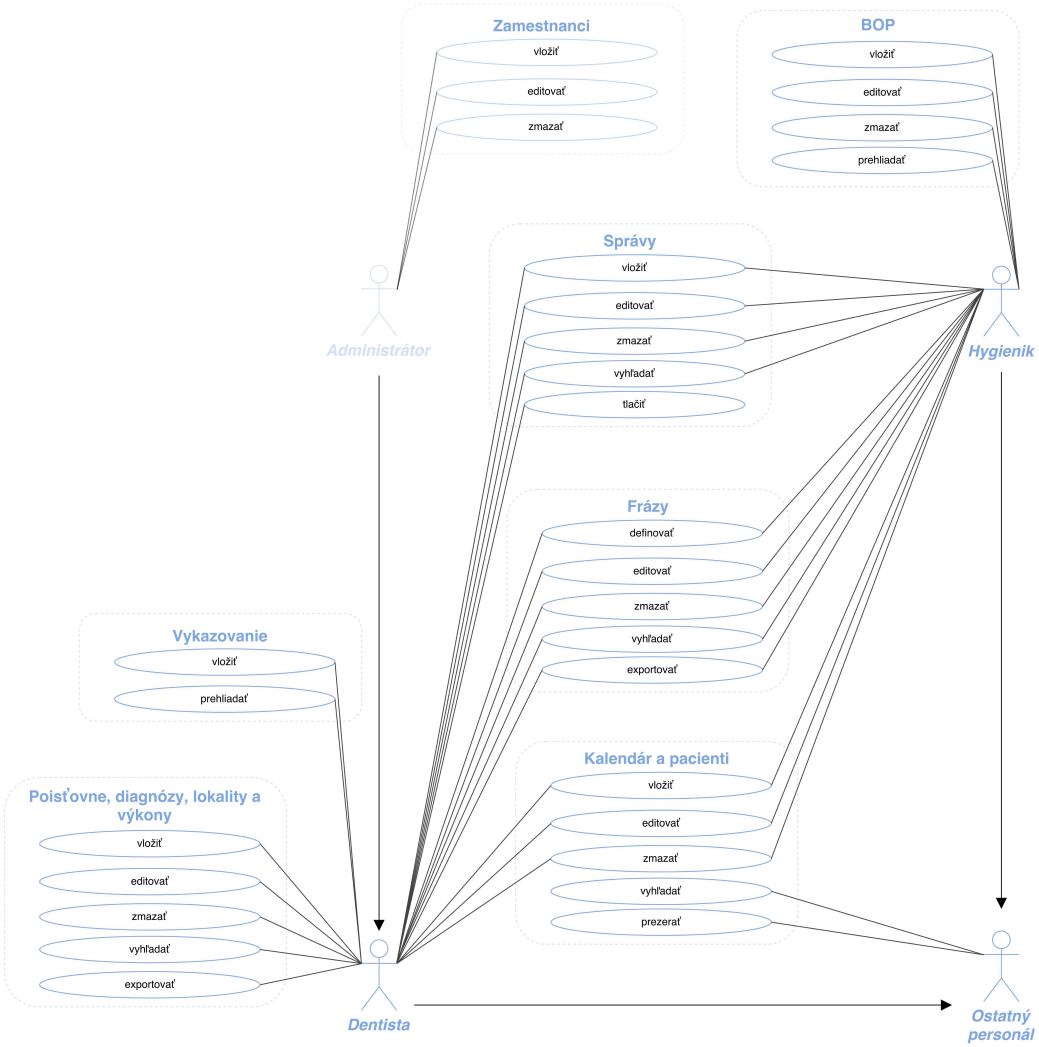
Obsah priloženého pamäťového média

Priložené CD má nasledujúcu adresárovú štruktúru:

- **Zdrojové súbory**
  - **aplikácia** - zdrojové súbory aplikácie
  - `database.sql` - sql súbor využitý k vytvoreniu databázy a jej jednotlivým tabuľkám
- **Dokumenty**
  - **sablona** - zdrojové kódy LaTeX šablóny využitej pre vytvorenie BP.pdf
  - `BP.pdf` - text bakalárskej práce vo formáte pdf
  - `README` - súbor popisujúci postup potrebný pre správne spustenie a fungovanie aplikácie

Príloha B

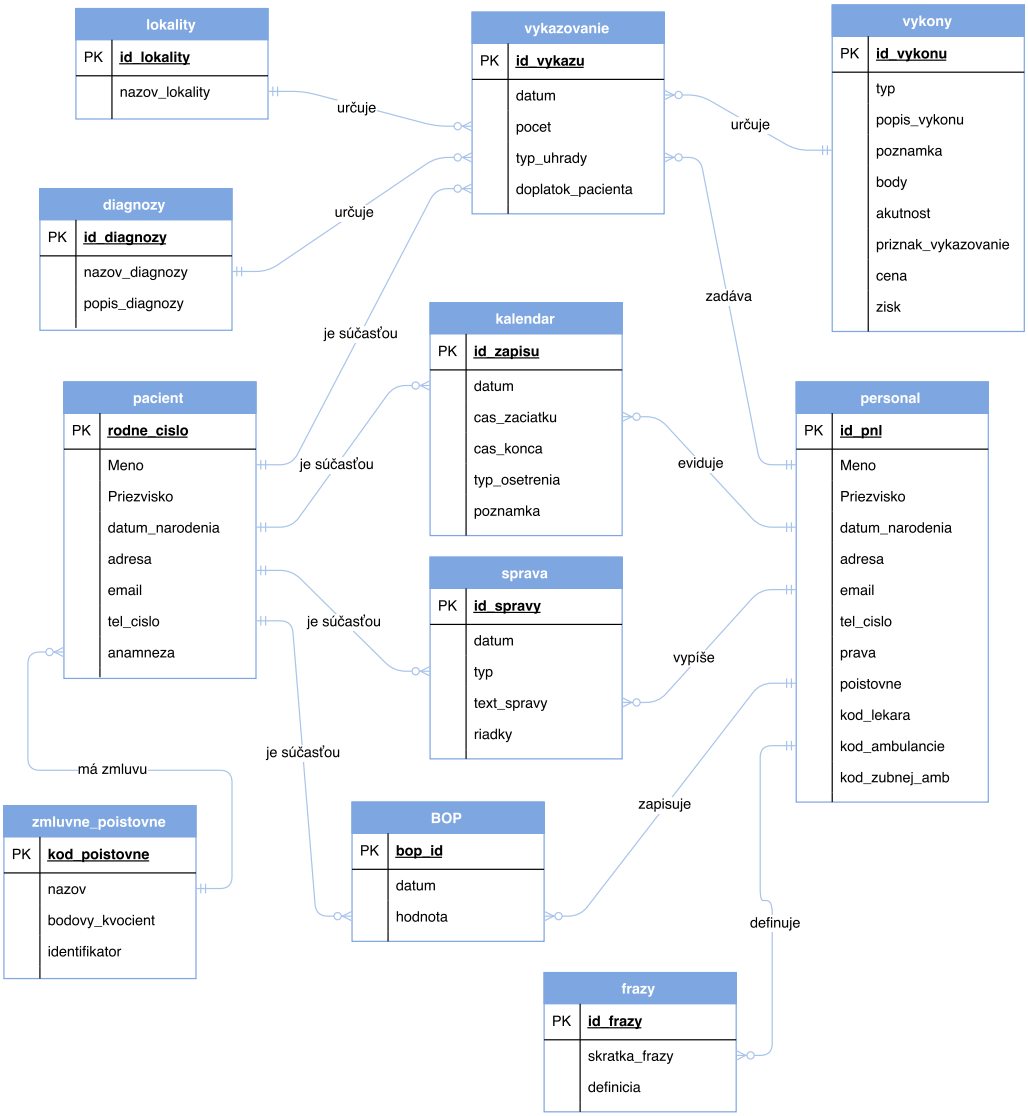
Use Case diagram



Obr. B.1: Use Case diagram využitý pri návrhu databázy informačného systému

Príloha C

Entity Relation Diagram



Obr. C.1: ERD využitý pre návrh relačnej databázy informačného systému