VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

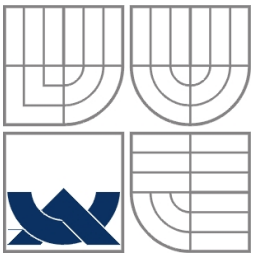
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZOVÝ ARCHÍV OBRAZOVÝCH MEDICÍNSKÝCH DAT

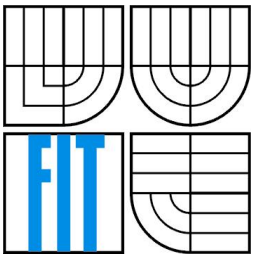
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Vítězslav Sára
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZOVÝ ARCHÍV OBRAZOVÝCH MEDICÍNSKÝCH DAT

DATABASE ARCHIVE OF IMAGE MEDICAL DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Vítězslav Sára

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Přemysl Kršek Ph.D.

BRNO 2008

Abstrakt

Práce se zabývá problematikou správy medicínských obrazových dat. Analyzuje problém extrakce metadat z medicínského formátu DICOM a jejich efektivního ukládání. Systém je navržen jako aplikace klient-server. Poskytuje rozhraní pro import dat ve formátu DICOM, rozhraní pro vyhledávání dat v databázi, úpravu dat a export. Systém úzce spolupracuje s kolaborativním systémem VCE. Implementace byla provedena v jazycích C++ a PHP s podporou databáze SQLite a MySQL.

Klíčová slova

obrazová medicínská data, uložení dat, DICOM, C++, PHP, databáze, aplikace klient-server, kolaborativní systém VCE, SQLite, MySQL

Abstract

This Thesis deals with the problems of medical image data management. It analyzes a metadata extraction problem from medical format DICOM and their efficient saving. System is designed as a client-server application. System provides an interface for DICOM data storing, interface for searching, data editing and exporting. System cooperatives with the collaborative system VCE. Implementation was carried out in C++ and PHP language with support of SQLite and MySQL database.

Keywords

Image medical data, data storage, DICOM, C++, PHP, client-server application, collaborative system VCE, SQLite, MySQL

Citace

Sára Vítězslav: Databázový archív obrazových medicínských dat. Brno, 2008, diplomová práce, FIT VUT v Brně.

Databázový archív obrazových medicínských dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Přemysla Krška, Ph.D., UPGM FIT VUT.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vítězslav Sára
1. 5. 2008

Poděkování

Chtěl bych poděkovat Ing. Přemyslu Krškovi, Ph.D., Ing. Michalu Španělovi,
Ing. Miroslavu Švubovi, Ing. Vítu Štanclovi, Ing. Ondřeji Šilerovi,
Ing. Tomášovi Kašpárkovi za jejich spolupráci, pomoc, náměty a věcné připomínky, bez kterých by tato práce v této podobě nemohla vzniknout. Práce vnikla za podpory sdružení CESNET, z. s. p. o.

Obsah

1 Úvod

S pokrokem ve vědě se neustále zvětšují objemy zpracovávaných dat. S tímto trendem ovšem souvisí problém jejich uchování a organizace. Ani lékařství není výjimkou. Lékařské přístroje produkují velké množství dat. Ať jsou to data elektronická či psaná na papíře (nebo na jiném mediu), je nutno je efektivním, a tedy organizovaným, způsobem uchovávat.

Patrně nejvhodnější formou informace pro uchovávání je její digitální podoba. Oproti např. papírové formě má nesporné výhody. A tak, pokud je to možné, provádí se její digitalizace.

Digitalizovaná data jsou přístupná vždy, když jsou potřeba, a v co možná nejkratším čase (čas je v medicíně obzvláště důležitý faktor). Pokud dojde k selhání systému, je většinou možné ztracená data znovu obnovit. Digitalizovaná data jsou navíc přístupná na více místech současně. Informace je stále aktuální, protože ji může obsluha (nebo automat) bezprostředně po její změně v reálném světě změnit. S tím souvisí další výhoda takovýchto dat – kontrola vstupních údajů při zadávání. Vstupní program může např. eliminovat překlepy, odhalí logicky chybné hodnoty nebo i záměrně zadané špatné hodnoty. Dále se zde mohou uplatnit i různé úrovně oprávnění přístupu k informacím.

Takový způsob uložení dat však přináší i řadu problémů, mezi které patří bezpečnost uložených dat. To platí zejména při přenosech po síti, ale i na samotném počítači, kde jsou data uložena.

Úložiště dat mohou být značně různorodá – od uložení informace v souborovém systému počítače až po databáze. Liší se například co do výkonnosti, schopnosti zvládat souběžný přístup více procesů a v dalších parametrech. I samotné databáze se mohou mezi sebou značně lišit – od lokálních databází situovaných v souborovém systému, ke které mají zpravidla přístup pouze programy běžící na tomtéž počítači, až po velké distribuované databáze, ke kterým přistupuje současně i několik set programů.

Aby bylo možné data sdílet mezi uživateli, kteří jsou často od sebe fyzicky velmi vzdálení, bývají tato úložiště přístupná z počítačové sítě. Každý uživatel musí mít na svém počítači nainstalovaného klienta, prostřednictvím kterého pak komunikuje s databází. Sdílení dat je pak základem vzdálené spolupráce. Odborníci mohou diskutovat o problému na základě sdílené informace, aniž by se osobně setkali.

Tato práce obsahuje studii a následný popis implementace systému, který spravuje obrazová medicínská data v síťovém prostředí, a tím úzce spolupracuje se systémem virtuálního kolaborativního prostředí.

2 Rozbor problematiky

Tato kapitola si klade za cíl seznámit čtenáře se základními pojmy problematiky, které se budou v dalším textu vyskytovat. Pokusí se také nastínit některá obecná řešení a problémy spojená s touto problematikou.

2.1 Formát medicínských obrazových souborů

Systém pro správu jakýchkoli dat musí bezpodmínečně znát formát spravovaných souborů. Nejinak je to i se systémy spravující medicínská data. Bez jasně definované struktury (formátu) se stávají soubory pouhými shluky bitů, které nenesou žádnou informaci.

2.1.1 Formát souborů

Formát souboru (také typ souboru) určuje strukturu uložené informace v elektronickém souboru. Protože na záznamová media (např. pevný disk) mohou být ukládány pouze jednotlivé bity, udává tato struktura jejich význam. Existuje mnoho různých formátů přizpůsobených k ukládání různých typů informace. Často existuje pro jeden typ dat i více formátů.

Některé formáty jsou navrženy pro uchovávání přesně daného typu dat (např. formát JPEG je navržen pro uchovávání obrazových dat). Jiné mohou uchovávat i více typů dat (např. formát GIF slouží pro uchovávání statických obrázků i krátkých sekvencí).

2.1.1.1 Specifikace formátu

Aby mohl být formát využíván co možná největším počtem uživatelů, vydává se k jednotlivým formátům jejich specifikace. Specifikace je dokument, který popisuje transformaci informace do souboru a její zpětnou extrakci. Mnoho formátů však takovou specifikaci nemá (např. formát souboru je považován za obchodní tajemství). V takovém případě je možné se pokusit formát souboru alespoň „odhadnout“ z dostupných materiálů (což je např. u binárních souborů téměř nemožné). Je však nutné si uvědomit, že takto získaná („odhadnutá“) specifikace formátu souboru nebude postihovat některé speciální případy vkládané informace, kterou by původní specifikace zakódovala jinak.

2.1.1.2 Obecná struktura souboru

Obsah souboru lze většinou logicky rozdělit na dva díly – hlavičku souboru a vlastní data.

Hlavička (pokud je přítomna – např. textové soubory hlavičku zpravidla nemívají) většinou obsahuje informace (tzv. metadata) o vlastních datech, které následují za ní. Typicky obsahuje minimálně typ (formát) souboru a velikost vlastní hlavičky (pokud není její velikost fixní). U obrázkových formátů obsahuje ještě např. velikost obrázku, jeho barevnou hloubku, způsob kódování

barev, kompresi a další. U hudebních souborů by to mohla být např. vzorkovací frekvence. Pomocí hlavičky může operační systém určit program pro jeho otevření (např. OS UNIX)

Datová část pak obsahuje vlastní přenášenou informaci, která je kódována podle pravidel uvedených ve specifikaci.

2.1.2 Medicínské obrazové soubory

Strukturu medicínského obrazového souboru můžeme rozdělit na tytéž základní části, a to na hlavičku a data souboru. Datová část souboru se logicky neliší od jiných obrazových souborů – její formát je popsán specifikací a doplněn daty z hlavičky. V hlavičce souboru však nalezneme větší množství informací. Informace se mohou týkat pacienta (např. jméno, věk), jeho stavu, zařízení, na kterém byl snímek vytvořen (např. výrobce, nastavení přístroje), informace o poloze pacienta při snímání, informace o vlastních datech (např. počet barev, rozlišení) a mnohé další.

2.1.3 Standard DICOM

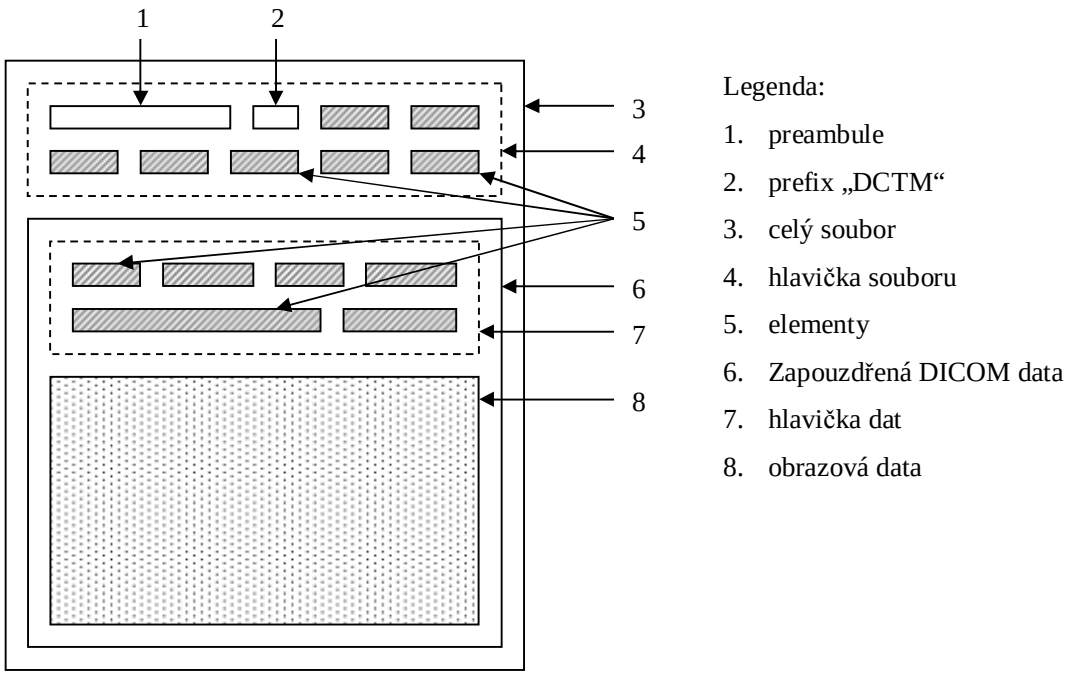
DICOM (Digital Imaging and Communication in Medicine) je komplexní množina standardů (detailně popsána v [1]) pro zpracování, ukládání a přenos medicínských obrazových studií. Má dvě komponenty – komunikační protokol a formát obrazových dat. DICOM představuje digitální zobrazování a komunikaci v medicíně. Jedná se o veřejný standard vyvíjený spojeným úsilím Americké vysoké školy radiologické ACR (American College of Radiology) a Federální asociace výrobců elektroniky NEMA (National Electronics Manufactures Association).

V další části budu popisovat pouze ty části standardu, které se zabývají formátem obrazových dat, protože celý standard je značně komplexní.

2.1.3.1 Popis formátu medicínských obrazových souborů DICOM

Na rozdíl od standardu ACR-NEMA definuje standard DICOM také offline verzi medicínských dat, a tedy i formát souborů.

Soubory se ukládají v binárním formátu a implicitním bitovém zarovnání typu „Little Endian“ (nejméně významný datový bit je uložen na nejnižší adrese)[2]. Standard definuje kódování metadat v hlavičce dokumentu. Je jím standard ISO-8859. Soubor lze opět logicky rozdělit na hlavičku, obsahující metainformace, a na zapouzdřená DICOM data (Obrázek 1).



Obrázek 1 Struktura souboru DICOM

Hlavička souboru se skládá z 128bajtové preambule. Tato preambule je vyplněna čísly 0x00 a nenese žádnou užitečnou informaci. Za ní následují 4 bajty prefixu. Tato část je vyplněna řetězcem „DICM“. Za prefixem se nacházejí další elementy.

Každý z těchto elementů (Obrázek 2) je uvozen dvojicí čísel - skupina tagů, které označují konkrétní element. Jejich velikost je stanovena standardem. Tato dvojce přesně identifikuje typ informace, který následuje. Za tímto identifikačním tagem může následovat informace o bitovém zarovnání dat elementu. Za ní následuje délka datové části elementu, protože (např. u popisných textů) nemůžeme délku předem určit. V jiných případech je délka informace dána standardem. Podle typu elementu pak následují vlastní data elementu. To umožňuje přidání dalšího druhu informace do standardu bez změny koncepce. Tato koncepce ovšem vyžaduje sekvenční čtení elementů (i když je

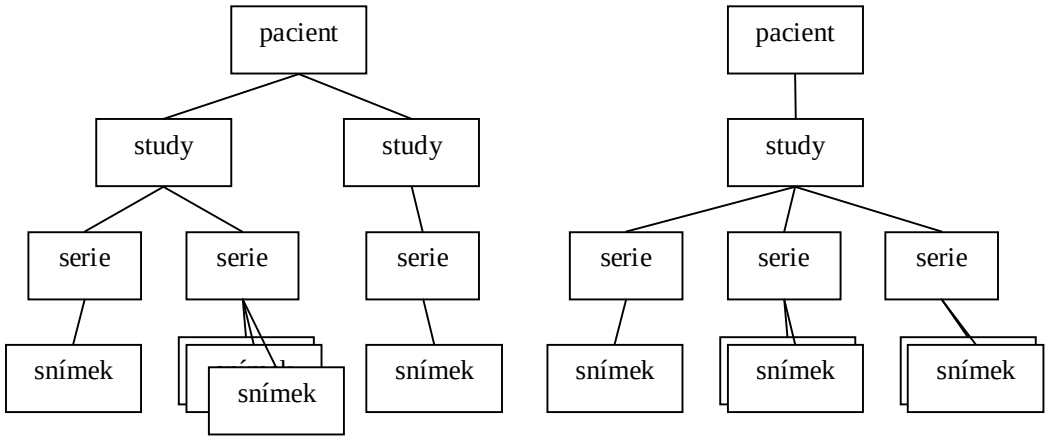
tag	bitové	délka dat	data elementu
číslo skupiny | číslo elementu	zarovnání		

Obrázek 2 Informační element formátu DICOM

samozřejmě možné přečtené neznámé elementy vynechat a přejit na další). Pokud daný program nebude nějaký z tagů podporovat, může tento tag vynechat (protože zná jeho délku) a číst další. Elementy souboru mohou být například tyto: délka elementů v hlavičce souboru, verze formátu souboru, kódování obsahu, identifikace souboru, identifikace autora a další.

Zapouzdřená DICOM data (vlastní obsah souboru) obsahuji další elementy, které popisují nesená data (v našem případě data obrazová). Vzhledem k tomu, že tento formát umožňuje nést obrazovou informaci různého typu, je počet elementů značně velký. Uvedu proto pouze několik typů informace nesených elementy: jméno pacienta, tloušťka řezu, datum pořízení snímku, pozice pacienta při snímání obrazu a další. Pomocí těchto elementů lze každý snímek jednoznačně identifikovat.

Pokud by byl počet snímků jednoho pacienta velký (což je dnes naprosto běžné), byla by orientace v nich velice obtížná. Standard proto umožňuje snímky řadit podle dalších hodnot elementů. Po jejich zařazení vznikne hierarchická struktura (Obrázek 3), ve které je již orientace snadná. Jsou to tzv. study (lze si ji představit jako návštěvu nemocnice), série (lze si ji představit jako vyšetření na oddělení). Série pak obsahuje samotné snímky.



Obrázek 3 Hierarchické uspořádání záznamů

O pacientovi jsou k dispozici následující informace: jméno, jednoznačné ID (minimálně v rámci nemocničního zařízení), datum narození, pohlaví pacienta, adresa pacienta, věk pacienta a další.

Study obsahuje následující informace: jednoznačné ID, ID jednoznačné v rámci pacienta, datum vyšetření, poznámku a další.

Informace o sérii zahrnují: jednoznačné ID, ID jedinečné v rámci study, datum a čas pořízení série, popis a tzv. modalitu. Modalita udává typ neseného obrazu (CT, MR, XR, US, …) a další.

Každý soubor také nese informaci o snímku v něm uloženém. Obsahuje informaci o projekci, barevné hloubce, rozměrech snímku a mj. i identifikační číslo jednoznačné v rámci série.

Všechny tyto informace jsou uloženy jako elementy v hlavičce souboru.

2.2 Úložiště dat

S postupujícím vývojem technologií produkují jednotlivá zařízení stále více informací. Vzniká tak potřeba ukládat tyto informace. Klade se důraz na efektivní ukládání a získávání dat z úložiště. Úložiště by také mělo být snadno spravovatelné, mělo by umožňovat souběžný přístup a mělo by být dobře chráněno proti vnějšímu neoprávněnému přístupu (tzn. kontrolovat přístup k uloženým datům). I úložiště medicínských souborů musí splňovat tyto obecné požadavky, navíc musí být dobře zabezpečené, protože obsahuje citlivá osobní data.

2.2.1 Souborový systém

Pravděpodobně nejstarším typem úložiště je ukládání dat programem přímo v souborovém systému operačního systému počítače.

Není zde téměř žádná abstrakce dat. Nositelem datových typů je sám program. To je velká nevýhoda, protože je zde velmi silná vazba mezi daty a zpracovávajícím programem. Program také zajišťuje veškeré vstupně-výstupní operace prostřednictvím služeb operačního systému (malá nebo žádná optimalizace těchto operací). Zabezpečení dat je realizováno operačním systémem. Možnost sdílení informace je na velmi nízké úrovni. Vzhledem k tomu, že informace je rozdělena do více souborů, hrozí nebezpečí nekonzistence (informace uložená v jednom souboru odporuje informaci uložené v jiném souboru) a porušení integrity dat (tj. správnost dat). Také riziko redundance dat je vysoké.

Souběžný přístup více procesů ke stejnému datovému souboru se také řeší velmi obtížně.

Možnou výhodou může být téměř nulové zatížení hostitelského systému - veškerý výpočetní výkon počítače lze využít k výpočtům. Tato výhoda je ovšem relativní, neboť program bude stejně provádět vstup-výstupní operace, které budou pravděpodobně nejpomalejší. Další výhodou je skutečnost, že není nutné instalovat žádný podpůrný software, který by datový sklad řídil.

Přístup k datům je omezen pouze operačním systémem. Tato omezení však zpravidla jsou příliš „hrubá“(nedokážou postihnout všechny potřebné přístupové případy). Navíc při práci v počítačové síti je nutné, aby každý, kdo přistupuje do úložiště, měl na tomto počítači systémový účet.

2.2.2 Databázový systém (DBMS)

Základním přínosem databázové technologie je dosažení jisté nezávislosti dat na uživatelských programech a naopak. Databázový systém se skládá z databáze (perzistentní data využívaná aplikacemi) a systému řízení báze dat. [3]

2.2.2.1 Systém řízení báze dat

Systém řízení báze dat (SŘBD) je programová vrstva řešící operace nad databází. Cílem SŘBD je odstínění aplikace od technických detailů. Systém řízení báze dat zajišťuje definici dat, manipulaci s nimi. Zajišťuje také bezpečnost a integritu dat. Dále souběžný přístup (transakční zpracování) a zotavení po chybách. Využíváním vyrovnávacích pamětí a optimalizací příchozích příkazů před jejich provedením zajišťuje maximální výkonnost systému.

2.2.2.2 Modely databázových systémů

Z hlediska způsobu ukládání dat a vazeb mezi nimi můžeme rozdělit databáze do základních typů:

- Hierarchická databáze
- Síťová databáze
- Relační databáze
- Objektově relační databáze
- Objektová databáze

Hierarchický model má záznamy organizovány do stromů. Záznamy jsou procházeny tzv. navigačním programováním.

Síťový model má záznamy organizovány do struktur (nemusí být hierarchické). Záznamy jsou provázány ukazateli, což mohou být fyzické adresy bloků na disku nebo čísla záznamů.

Relační model publikoval v roce 1970 E. F. Codd v článku "A relational data model for large shared data banks". Tento model má záznamy organizovány do relací. Relace je definována jako množina n-tic, která má stejné atributy. Hodnoty každého atributu jsou vybírány z tzv. domény a jsou atomické – dále vnitřně nestrukturovaná. Pro tento model byl standardizován dotazovací jazyk – SQL. Relační databáze je vnímána uživatelem (aplikací) jako kolekce časově proměnných normalizovaných relací. Veškerá data v relační databázi jsou reprezentována explicitní hodnotou (žádné ukazatele apod.). [zendulka05]

Objektově relační model je přechodovým modelem mezi relačním a objektovým modelem. Nejčastěji se do relační databáze přidávají prvky z databáze objektové.

Objektový model respektuje vývoj objektových programovacích jazyků a snaží se data ukládat jako objekty. V databázi existují vazby (podobně jako v databázi hierarchické), ale s tím rozdílem, že identifikátorem již není číslo fyzického bloku disku, ale jednoznačná generovaná hodnota. Odkazy na objekty v databázi se uchovávají v tzv. extentu. Ten je přístupovým bodem do celé databáze.

2.2.2.3 Architektury databázových systémů

Databázová aplikace zahrnuje dvě základní programové vrstvy – procesy na popředí a procesy na pozadí. Proces na popředí (front-end) je část aplikace, která využívá SŘBD. Proces na pozadí (backend) realizuje všechny základní funkce SŘBD.

Základní databázové architektury:

- architektura typu „mainframe“: front-end, backend i databáze jsou na jednom počítači
- architektura typu „PC file server“: front-end, backend jsou na jednom počítači, databáze je oddělena
- architektura typu „klient/server“: front-end je na klientském počítači, backend i databáze jsou na jednom počítači
- vícevrstvá architektura: na klientském počítači běží jen proces, který zobrazuje výsledky a zadává požadavky (např. internetový prohlížeč). Požadavek zpracuje aplikační server, který ho převede na databázový dotaz (např. webový server). Databázový server dotaz zpracuje a vrátí požadovaná data aplikačnímu serveru, který je pošle klientovi.

K těmto architekturám lze jako zvláštní případ přidat databáze vestavěné. Jedná se o databázi, která je zkompilovaná přímo v aplikaci. Tato konfigurace má tu výhodu, že databáze „běží“ pouze v případě, kdy je skutečně potřeba, a je typicky i méně náročná na systémové zdroje. Další výhodou může být možnost vytvářet pouze paměťové databáze (bez zápisu na pevný disk). Nevýhodou je pak menší výkon - zejména u datově aktivních aplikací. Menší výkon vyplývá z horšího zpracování dotazů (optimalizace dotazu) a malých paměťových cache (samostatná databáze může větší část dat uchovávat v paměti, což si vestavěná databáze většinou dovolit nemůže).

2.3 Síťová komunikace

Historie sítí sahá až do 60. let 20. století, kdy začaly první pokusy s komunikací počítačů. Od té doby se jejich velikost i význam znásobil. Prostřednictvím počítačové sítě můžeme komunikovat s lidmi na druhém konci světa, řídit stroje nebo třeba nakupovat.

Systém pro správu medicínských dat využívá počítačovou síť pro distribuci obrazových souborů. Data nashromážděná na jednom místě (na serveru) jsou prostřednictvím sítě přístupná odkudkoliv (pro autorizované uživatele).

2.3.1 Počítačová síť

Počítačová síť je souhrnné označení pro technické prostředky, které realizují spojení a výměnu informací mezi počítači. Umožňují tedy uživatelům komunikaci podle určitých pravidel, za účelem sdílení využívání společných zdrojů nebo výměny zpráv. [3]

2.3.2 Protokoly

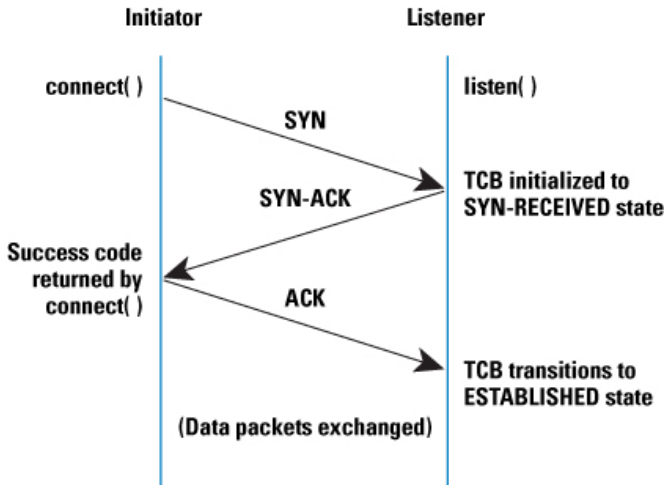
Protokoly jsou tvořeny souhrnem pravidel, formátů a procedur, které určují výměnu údajů mezi dvěma či více komunikujícími prvky na stejné logické vrstvě vrstvového modelu. Zahrnují proceduru

navázání spojení, adresování, přenos dat, zpracování chyb, řízení toku komunikace a přidělování prostředků.

2.3.2.1 Protokoly transportní vrstvy – TCP a UDP

Služby mohou být na úrovni transportní vrstvy spojované a nespojované.

Spojované jsou takové, které na začátku ustanoví spojení (Obrázek 4), jež se během komunikace udržuje a po skončení uzavře. Toto spojení taktéž zajišťuje bezpečný (ve smyslu doručitelnosti paketu) přenos. Pakety jsou v přenosu číslovány, a tak pokud dojde ke ztrátě paketu, může o tom být jak vysílající, tak přijímající strana informována. Po ukončení přenosu obě strany spojení uzavřou. Typickým zástupcem spojovaného transportního protokolu je v internetu protokol TCP (Transmission Control Protocol) [4]. Protokol TCP taktéž zaručuje, že pakety odeslané odesilatelem dorazí k příjemci ve stejném pořadí, ve kterém byly odeslány. Aby se minimalizovala režie při počátečním ustanovení spojení, je možné otevřené spojení zachovat a při dalším datovém přenosu znovu použít.



Obrázek 4 Ustavení spojení TCP

(TCB je struktura uvnitř OS, uchovávající mj. i stav spojení, převzato z [5])

Nespojované služby jsou takové, které žádné spojení neustanovují. Oproti spojovaným mají tu výhodu, že se nemusí ustanovovat spojení, jež může znamenat kritickou prodlevu. Na druhou stranu to znamená, že se pakety mohou být ztraceny. Taktéž je možné, že pakety přijdou v jiném pořadí než ve kterém byly vyslány. Toto všechno musí řešit aplikační protokol. Další výhodou je to, že pakety jsou menší a mohou rychleji „cestovat“ sítí. Tyto protokoly se požívají při komunikaci, kde je tolerovatelná jistá malá ztráta dat a důraz je kladen na rychlost přenosu (streamované audio a video). Typickým zástupcem nespojovaného transportního protokolu je v internetu protokol UDP.

2.3.2.2 Protokol aplikační vrstvy – HTTP

HTTP (Hypertext Transfer Protocol) je internetový protokol, který pracuje na aplikační úrovni internetového modelu (většinou nad protokolem TCP), určený původně pro výměnu hypertextových dokumentů ve formátu HTML. Používá obvykle port TCP/80, verze 1.1 protokolu je definována v [6].

V současné době je používán i pro přenos dalších informací. Pomocí rozšíření MIME (Multipurpose Internet Mail Extensions) umí přenášet jakýkoli soubor (tedy i soubory medicínských dat).

HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (URL, Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu (celosvětová počítačová síť).

K protokolu HTTP existuje také jeho zabezpečená verze HTTPS, která umožňuje přenášená data šifrovat, a tím chránit před odposlechem či jiným narušením pomoci SSL/TSL (Secure Sockets Layer/Transport Layer Security).

Protokol HTTP je bezstavový, což znamená, že neudržuje kontext mezi dvěma požadavky na server. Tuto „nevýhodu“ lze odstranit použitím tzv. cookies. Funkce cookies je definována v [7] pomocí HTTP hlaviček Set-Cookie a Cookie. Hlavička Set-Cookie je poslána v odpovědi serveru a obsahuje jednak data cookie (omezená prohlížečem, vyžadována je podpora alespoň pro 4096 byte), jednak informace o době platnosti, adresář na serveru, pro který cookie platí apod. Pokud má prohlížeč alespoň jednu cookie pro daný server (a daný adresář na něm), posílá s každým dotazem danému serveru i hlavičku Cookie, která obsahuje stejná data, která server původně poslal.

2.4 Kolaborativní prostředí

V praxi je stále častěji nutné, aby na jednom projektu spolupracovalo více odborníků. Není však vždy možné, aby se tito odborníci neustále scházeli, ať již z časových nebo zeměpisných důvodů. Proto vyvstává potřeba vzdálené spolupráce zprostředkované počítačovou sítí.

Navrhovaný databázový archiv obrazových medicínských dat je součástí kolaborativního prostředí VCE (Virtual Collaborative Environment). Pro toto prostředí zajišťuje synchronizaci medicínských dat a poskytuje i některé další služby spojené se správou prostředí (zejména správa VCE session), které se tykají databáze.

2.4.1 Funkce medicínského kolaborativního prostředí

Kolaborativní systém umožňuje pro zadaná obrazová data otevřít na serveru tzv. session. Aby se mohla obrazová data na připojených klientech prezentovat, je nutné zajistit synchronizaci dat mezi

těmito klienty a daty na serveru. Po této synchronizaci, která může proběhnout i za nevědomosti uživatele, se klienti mohou připojit do session.

Zakládající uživatel pak prezentuje problém svým spolupracovníkům prostřednictvím sdíleného modelu (v případě medicínského kolaborativního prostředí nejčastěji model kostí a tkání), který je synchronizován na všech klientech kolaborativního prostředí. S modelem je možno pohybovat a procházet různými úrovněmi detailu. Aby byla spolupráce efektivnější, obsahuje takové prostředí sdílené ukazovátko, kterým moderátor ukazuje problémové oblasti modelu, a značky, které je možné na model umísťovat kvůli snadnější orientaci.

Po skončení spolupráce se klienti od serveru odpojí a kolaborativní session zaniká.

Práce s modelem by měla být pokud možno interaktivní, aby nedocházelo k nedorozuměním. Pro snazší komunikaci mezi účastníky by mělo kolaborativní prostředí přenášet nejenom změny modelu, ale i hlas, popřípadě i video zainteresovaných. Pokud by nebylo technicky možné vést konferenci audiovizuálně, pak by mělo prostředí zajistit komunikaci alespoň pomocí textových zpráv.

2.4.2 Kolaborativní prostředí VCE

Kolaborativní prostředí VCE (Virtual Collaborative Environment) je vyvíjeno týmem pracovníků pod vedením ing. P. Krška na ústavu UPGM VUT FIT. Vývoj prostředí je podporován sdružením CESNET, z. s. p. o.

2.4.2.1 Specifikace kolaborativního prostředí VCE

Kolaborativní prostředí VCE je paravirtuální prostředí (synchronizace modelu neprobíhá úplně v reálném čase), které pracuje s medicínskými modely. Tyto modely jsou zkonstruovány z dat obsažených v souborech DICOM. Session jsou moderované (není povoleno více účastníkům současně manipulovat s modelem). Prostředí nezajišťuje žádnou formu hlasové komunikace mezi účastníky sezení (musí byt zajištěna externím programem např. Skype). Umožňuje ale do sdíleného modelu vkládat značky, které mohou být velmi důležité při orientaci v modelu.

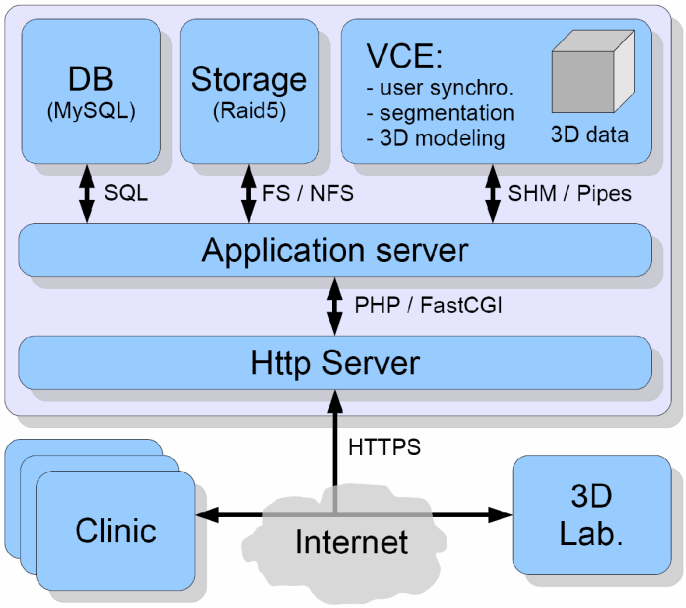
2.4.2.2 Schéma kolaborativního prostředí VCE

Kolaborativní prostředí VCE lze rozdělit na dvě základní části - klienta a server (obrázek 5). Tyto dvě části komunikují po síti standardním protokolem HTTP(S). Použití tohoto protokolu bylo podmíněno skutečností, že směrovače nemocnic obvykle jiný odchozí provoz z bezpečnostních důvodů blokují.

Klient zobrazuje modely uživatelům a aktivně si zajišťuje synchronizaci se serverem (klient zasílá výzvu na server, zda se stav modelu nezměnil, a server mu na ni odpovídá zprávou o změně stavu).

Server se skládá z několika spolupracujících částí. Příchozí požadavky na kolaborativní prostředí přijímá webový server (Apache httpd), který požadavky vyřizuje pomocí php skriptů (skripty FastCGI jsou v současné době jako výkonová záloha). Tyto skripty komunikují podle potřeby

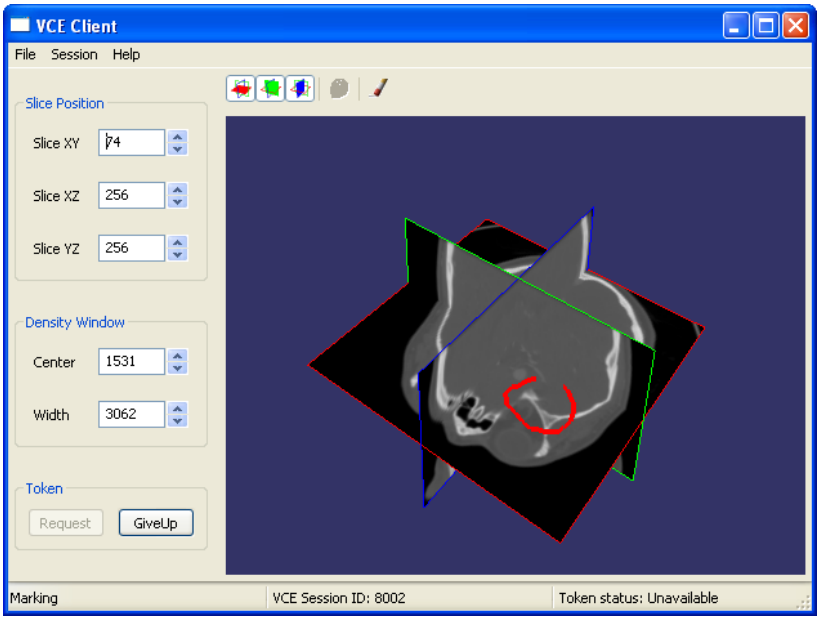
s databází (jsou v ní uložena vyextrahovaná data ze souborů), storage (úložiště fyzických souborů) nebo modulem VCE. VCE modul synchronizuje VCE klienty, kteří jsou připojeni v jednotlivých session (udržuje fronty požadavků, stav modelu)[8].



Obrázek 5 Schéma kolaborativního prostředí VCE

2.4.2.3 Klientské rozhranní kolaborativního prostředí VCE

Kolaborativní systém VCE má na straně klienta poměrně jednoduché uživatelské rozhraní, které ale obsahuje všechny důležité prvky pro ovládání modelu. Kolaborativní systém VCE umožňuje moderátorovi sezení vkládat do scény značky, které jsou pak viditelné všem účastníkům sezení.



Obrázek 6 Uživatelské rozhraní klientské části kolaborativního prostředí VCE

2.5 Návrhové vzory

Architektura těchto systémů představuje poměrně náročný problém a na jejím správném navržení většinou závisí úspěch celého systému. Architektura je jasné a srozumitelné sdělení, jakým způsobem budou realizovány požadavky od zadavatelů systému. Jasné musí být nejen pro uživatele, ale hlavně pro celý vývojový tým. Je určitě zajímavé a užitečné při jejím návrhu a tvorbě využívat standardních postupů, které ji zpřehledňují a usnadňují její implementaci.

2.5.1 Návrhový vzor

Návrhový vzor (design pattern) představuje obecné řešení problému, které se využívá při návrhu programů. Návrhový vzor není knihovnou nebo částí zdrojového kódu, která by se dala přímo vložit do našeho programu. Jedná se o popis řešení problému nebo šablonu, která může být použita v různých situacích. Objektově orientované návrhové vzory typicky ukazují vztahy a interakce mezi třídami a objekty, aniž by určovaly implementaci konkrétní třídy. Algoritmy nejsou považovány za návrhové vzory, protože řeší konkrétní problémy, nikoliv problémy návrhu.[9]

Návrhové vzory lze rozdělit do tří skupin:

- Tvořivé vzory – umožňují ovlivnit způsob vytváření objektů a jejich počet (Singleton, Factory Method, Abstract Factory, Builder)
- Strukturální vzory – popisují, jak jsou třídy a objekty složeny do větších struktur (Adapter, Bridge, Facade, Proxy, Decorator, Composite, Flyweight)
- Vzory chování – popisují rozdělení funkčnosti a zodpovědnosti mezi objekty (Template method, Interpreter, Mediator, Chain of Responsibility, Observer, Strategy, Command, State, Visitor, Iterator)

2.5.2 Architektonický vzor

Architektonické vzory jsou softwarové vzory, které nabízejí osvědčená řešení architektonických problémů v softwarovém inženýrství. Architektonické vzory vyjadřují základní schéma softwarového systému, který se skládá z podsystémů, popisuje jejich odpovědnost a vztahy. Na rozdíl od návrhových vzorů jsou architektonické vzory komplexnější.

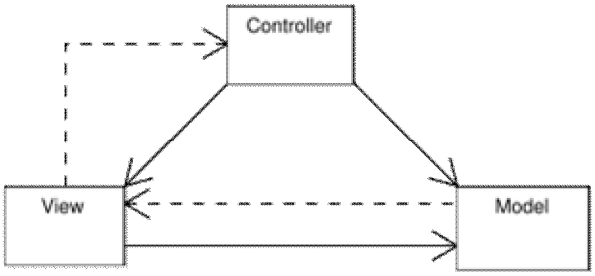
2.5.2.1 Model-view-controller

Model-view-controller je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní. [10]

Model má tři základní prvky:

- Model – doménově specifická reprezentace informacím, s nimiž aplikace pracuje.

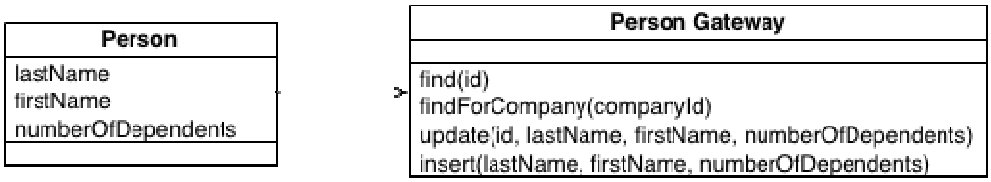
- Views (pohled) – převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli.
- Controller (řadič) – reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.



Obrázek 7 Vztahy mezi prvky MVC

2.5.2.2 Table Data Gateway

Table Data Gateway je architektonický vzor, který vznikl kvůli lepšímu oddělení procedurálního jazyka a jazyka SQL. Veškerý kód SQL je zapouzdřen uvnitř třídy (příkazy pro select, update, insert), což je výhodné pro lokalizaci případné chyby a ladění výkonu databáze databázovým administrátorem.



Obrázek 8 Architektonický vzor Table Data Gateway (převzato z [11])

3 Analýza požadavků

3.1 Neformální zadání

Navrhněte a implementujte řešení systému pro správu medicínských obrazových dat (soubory ve formátu DICOM), které bude spolupracovat s kolaborativním systémem (VCE – Virtual Collaborative Environment).

Doporučená architektura systému je klient-server. Nejmenší spravovatelnou jednotkou (podle hierarchie DICOM) je jedna série.

Klientská část bude implementována v jazyce C++. Bude to aplikace s přívětivým uživatelským rozhraním. Musí být schopná pracovat nezávisle na serverové části. Dále bude umožňovat vyhledávání jak v uložených souborech, tak v databázi serverové, a to minimálně podle následujících kritérií: jméno pacienta (vyhledat i neúplná jména), ID pacienta, pohlaví pacienta a popis (pacienta nebo vyšetření). Dále podle data vytvoření studie (minimálně od – do). Výsledek bude možné dále upřesnit podle modality obrazu.

Klientská část systému musí umožňovat import nových obrazových dat (souborů) do své databáze, a to jak z lokálních disků, tak ze serverové části systému. Při importu je možné provádět anonimizaci dat. Po importu dat do klientské databáze bude možno data odeslat na server.

Klient umožní ke každému vloženému snímku přiřadit komentář (na všech úrovních hierarchie vycházející z formátu DICOM).

Pokud by došlo k porušení databáze, bude klientská část systému schopna tuto databázi znovu obnovit (nebo alespoň částečně obnovit – závisí na stupni anonimizace dat).

Klientská část systému bude spolupracovat s kolaborativním prostředím dohodnutým aplikačním rozhraním (klient bude vystupovat v roli „Open dialog“). Se serverovou částí bude komunikovat přes protokol HTTP (kvůli omezením na firewallech).

Serverová část bude komunikovat s klienty dohodnutým protokolem nebo přímo s uživateli přes webové rozhraní. Bude poskytovat informace o pacientech a distribuovat jejich data. Na serverové části bude možné vyhledávat pacienty minimálně podle stejných parametrů jako na klientovi. Bude také umožněno importovat data přímo přes webové rozhraní.

Serverová část systému musí navíc spolupracovat s kolaborativním systémem (konkrétně s VCE superserverem), a to zahájením a ukončením kolaborativních session (u každé se bude uchovávat minimálně název a popis).

Celý systém by měl být přenositelný mezi více platformami (minimálně Windows a Linux). Pokud je to možné, měla by komunikace mezi klientem a serverem probíhat v souladu s platnými standardy. Klientská část systému by neměla zbytečně zatěžovat počítač, na kterém běží (předpokládá

se, že bude používána i na výkonově slabších počítačích). Serverová část by měla zvládnout paralelně obsluhovat alespoň 5 klientů.

Při výběru knihoven dbejte na otevřenost zdrojového kódu a licence, která umožní použití knihovny i na komerčních produktech.

3.2 Definice požadavků

3.2.1 Funkční požadavky

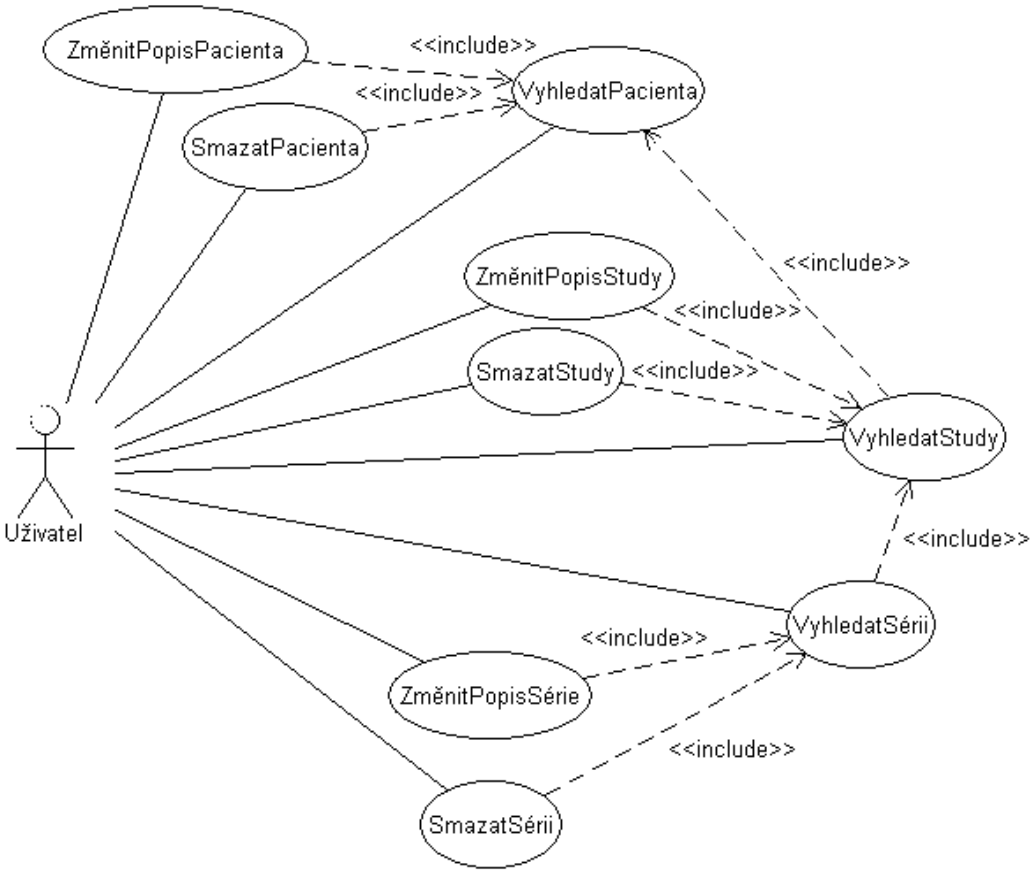
1. Klient bude umožňovat vyhledávání v lokální i serverové databázi minimálně podle jména pacienta, id pacienta, data vytvoření studie a může být upřesněn podle modality serie
2. Klient bude umožňovat import dat do lokální databáze
3. Klient bude umožňovat import dat ze serverové databáze
4. Klient bude umožňovat export dat z lokální databáze do databáze serverové
5. Klient bude umožňovat export dat z lokální databáze do vybraného adresáře
6. Klient bude umožňovat editaci poznámek k uloženým datům
7. Klient bude schopen alespoň částečně obnovit poškozenou databázi
8. Klient by mohl poskytovat nástroj pro anonimizaci importovaných dat
9. Server umožňovat vyhledávání ve své databázi
10. Server bude umožňovat import dat do své databáze
11. Server bude umožňovat zakládat a rušit kolaborativní sezení
12. Server bude schopen alespoň částečně obnovit poškozenou databázi

3.2.2 Nefunkční požadavky

1. Klient bude implementován v programovacím jazyce C++
2. Klient bude mít přehledné „okenní“ GUI
3. Klient nebude svou činností příliš zatěžovat hostitelský operační systém
4. Server bude zvládat souběžnou obsluhu minimálně 5-ti klientů
5. Klient i server budou implementováni s pomocí volně dostupných knihoven

3.3 Diagramy případů užití

3.3.1 Správa záznamů v klientské části systému



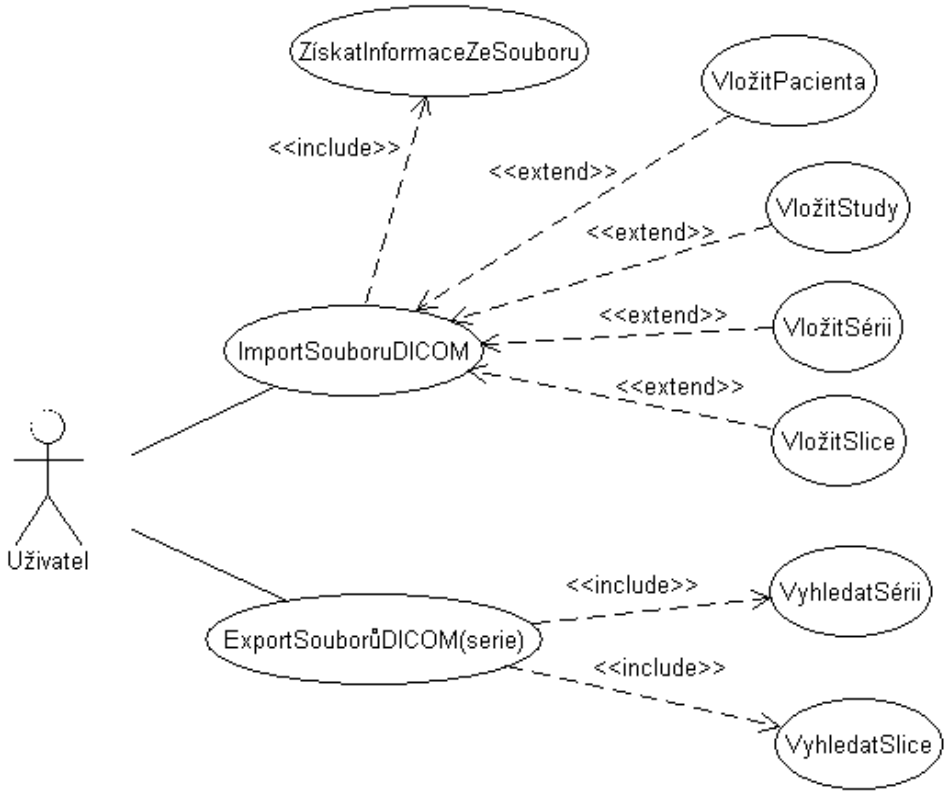
Obrázek 9 Diagram případů užití 1

ID: 1	Případ použití:	VyhledatPacienta
Stručný popis:	Systém provede vyhledání pacienta ve své databázi	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	žádné	
Hlavní tok:	1. Uživatel zadá kritéria pro hledání 2. Systém vyhledá odpovídající záznamy o pacientech 3. POKUD systém nenalezne žádné záznamy, bude o tom uživatele informovat varovným hlášením JINAK zobrazí nalezené záznamy o pacientech	
Následné podmínky:	Systém nalezl záznamy o pacientech	
Alternativní toky:	Žádné	

ID: 2	Případ použití:	ZměnitPopisPacienta
Stručný popis:	Systém změní popis pacienta ve své databázi	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	žádné	
Hlavní tok:	1. Uživatel vyhledá pacienta (VyhledatPacienta) 2. Změní popis pacienta 3. Potvrdí změnu popisu	
Následné podmínky:	Systém změnil popis pacienta	
Alternativní toky:	žádné	
ID: 3	Případ použití:	SmazatPacienta
Stručný popis:	Systém smaže pacienta ze své databáze. Pokud jsou v databázi pod pacientem uložena nějaká vyšetření, jsou také odstraněna	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	žádné	
Hlavní tok:	1. Uživatel vyhledá pacienta (VyhledatPacienta) 2. Potvrdí smazání záznamu o pacientovi z databáze	
Následné podmínky:	Systém smazal záznam o pacientovi	
Alternativní toky:	žádné	
ID: 4	Případ použití:	VyhledatStudy
Stručný popis:	Systém provede vyhledání study ve své databázi	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	Je znám pacient, ke kterému study patří	
Hlavní tok:	1. Uživatel zadá kritéria pro hledání 2. Systém vyhledá odpovídající záznamy o studiích podle zadaných kritérií a příslušnosti k pacientovi 3. POKUD systém nenalezne žádné záznamy, bude o tom uživatele informovat varovným hlášením Systém vypíše záznamy o nalezených study	
Následné podmínky:	Systém nalezl informace o pacientech	
Alternativní toky:	Žádné	

ID: 5	Případ použití:	ZměnitPopisStudy
Případ užití je sémanticky totožný s případem ID:2 ZměnitPopisPacienta		
ID: 6	Případ použití:	SmazatStudy
Případ užití je sémanticky totožný s případem ID:3 SmazatPacienta		
ID: 7	Případ použití:	VyhledatSérii
Případ užití je sémanticky totožný s případem ID:4 VyhledatStudy		
ID: 8	Případ použití:	ZměnitPopisSérie
Případ užití je sémanticky totožný s případem ID:2 ZměnitPopisPacienta		
ID: 9	Případ použití:	SmazatSérii
Případ užití je sémanticky totožný s případem ID:3 ZměnitPopisPacienta		

3.3.2 Import a export v klientské části systému



Obrázek 10 Diagram případů užití 2

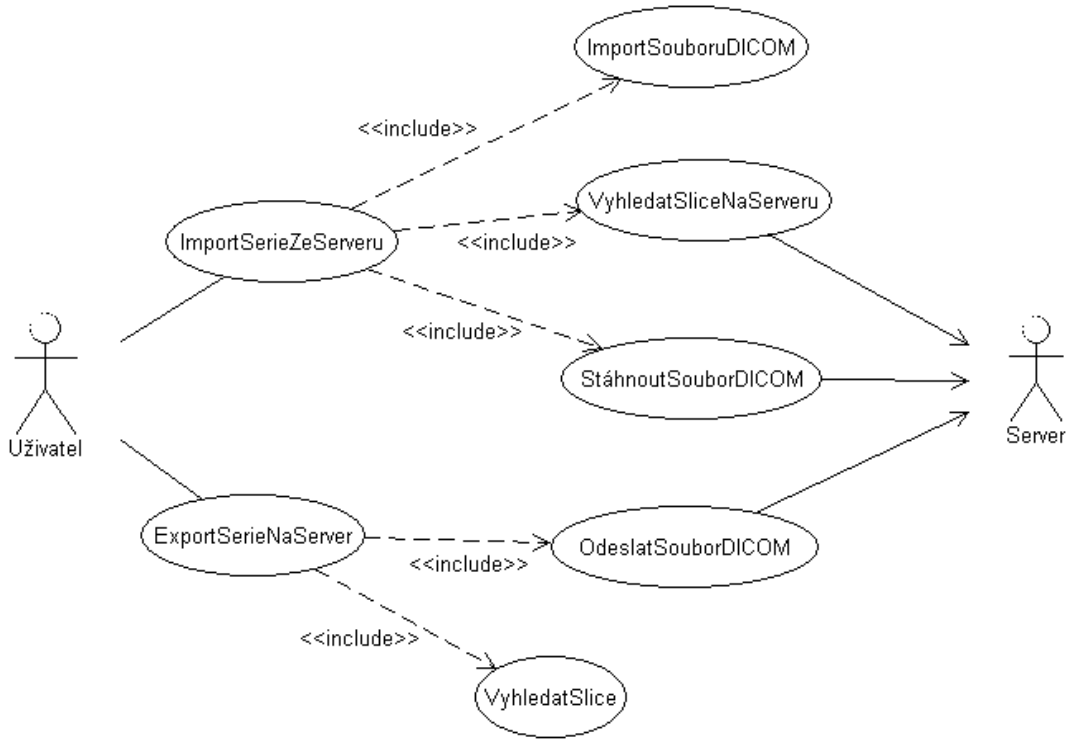
ID: 10	Případ použití:	ZískatInformaceZeSouboru
Stručný popis:	Systém zanalyzuje soubor DICOM a vyhledá v něm informace o pacientovi (jméno,DICOM ID, …), study (datum vyšetření, DICOM UID, …), sérii (modalita, DICOM ID, …) a řezu (DICOM ID).	
Primární aktéři:	Systém	
Sekundární aktéři:	žádní	
Předpoklady:	Jsou dostupný soubor pro analýzu	
Hlavní tok:	1. Systém zkontroluje, zda se jedná o platný soubor DICOM 2. Systém získá informace ze souboru	
Následné podmínky:	Jsou dostupné informace o pacientovi, study, sérii a řezu	
Alternativní toky:	Žádné	
ID: 11	Případ použití:	VložitPacienta
Stručný popis:	Systém vloží záznam o pacientovi ve své databáze	
Primární aktéři:	Systém	
Sekundární aktéři:	žádní	
Předpoklady:	Jsou dostupné informace o pacientovi (minimálně DICOM ID)	
Hlavní tok:	1. Systém vloží do databáze záznam o pacientovi podle dostupných informací	
Následné podmínky:	V databázi je záznam o pacientovi	
Alternativní toky:	Žádné	
ID: 12	Případ použití:	VložitStudy
Případ užití je sémanticky totožný s případem ID:11 VložitPacienta		
ID: 13	Případ použití:	VložitSérii
Případ užití je sémanticky totožný s případem ID:11 VložitPacienta		
ID: 14	Případ použití:	VložitSlice
Případ užití je sémanticky totožný s případem ID:11 VložitPacienta		
ID: 15	Případ použití:	ImportSouboruDICOM
Stručný popis:	Systém importuje soubor DICOM do své databáze	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	Je dostupný soubor pro import	

Hlavní tok:	1. Systém zanalyzuje vstupní soubor (ZískatInformaceZeSouboru) 2. POKUD jsou dostupné informace o pacientovi, study, sérii a slice 2.1. POKUD se záznam o pacientovi v databázi ještě nenachází 2.1.1. vloží záznam o pacientovi (VložitPacienta) do databáze 2.2. POKUD se záznam o study v databázi ještě nenachází 2.2.1. vloží záznam o study (VložitStudy) do své databáze 2.3. POKUD se záznam o sérii v databázi ještě nenachází 2.3.1. vloží záznam o sérii (VložitSérii) do své databáze 2.4. POKUD se záznam o slice v databázi ještě nenachází 2.4.1. vloží záznam o slice (VložitSlice) do své databáze JINAK informuje uživatele o neúspěšném importu
Následné podmínky:	V databázi je importován soubor DICOM
Alternativní toky:	Žádné

ID: 16	Případ použití:	VyhledatSlice
Stručný popis:	Systém vyhledá řez podle jeho ID a vlastnící série	
Primární aktéři:	Systém	
Sekundární aktéři:	žádní	
Předpoklady:	Je známa série, ke kterému slice patří	
Hlavní tok:	1. Systém vyhledá záznamy o řezech podle vlastnící série 2. POKUD systém nenalezne žádné záznamy, bude o tom uživatele informovat varovným hlášením	
Následné podmínky:	Systém nalezl informace o řezech	
Alternativní toky:	Žádné	

ID: 17	Případ použití:	ExportSouborůDICOM
Stručný popis:	Systém exportuje soubory DICOM ze své databáze do určeného adresáře	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	1. Je známa série, která se má exportovat 2. Je znám adresář, do kterého se budou exportovat DICOM soubory	
Hlavní tok:	1. Systém vyhledá řezy patřící pod danou sérii (VyhledatSlice) 2. POKUD jsou dostupný alespoň jeden řez Systém zkopíruje soubory DICOM ze své databáze do adresáře JINAK informuje uživatele o neúspěšném exportu	
Následné podmínky:	V cílovém adresáři jsou exportované soubory DICOM	
Alternativní toky:	Žádné	

3.3.3 Import a export v klientské části systému ze serveru



Obrázek 11 Diagram případů užití 3

ID: 19	Případ použití:	VyhledatSliceNaServeru
Stručný popis:	Systém vyhledá řezy na serveru	
Primární aktéři:	Uživatel	
Sekundární aktéři:	Server	
Předpoklady:	1. Je známa serie, do níž řezy patří	
Hlavní tok:	1. Systém vyhledá slice podle příslušnosti k sérii 2. POKUD systém nenalezne žádné záznamy, bude o tom uživatele informovat varovným hlášením	
Následné podmínky:	Byly vyhledány slice	
Alternativní toky:	Žádné	
ID: 20	Případ použití:	StáhnoutSouborDICOM
Stručný popis:	Systém stáhne ze serveru soubor DICOM	
Primární aktéři:	Uživatel	
Sekundární aktéři:	Server	
Předpoklady:	1. Je známý slice odpovídající stahovanému souboru DICOM	

Hlavní tok:	1. Systém stáhne soubor DICOM ze serveru 2. POKUD systém nebude moci soubor stáhnout, bude o tom uživatele informovat varovným hlášením
Následné podmínky:	Byl stažen soubor DICOM
Alternativní toky:	Žádné

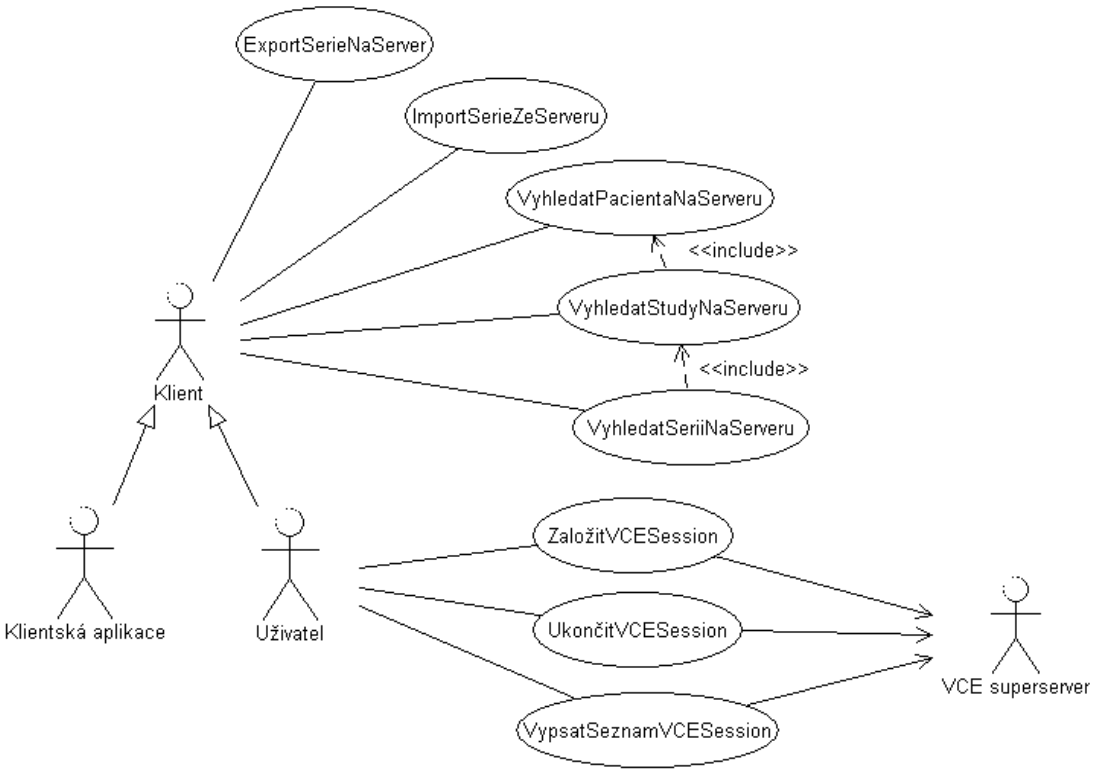
ID: 21	Případ použití:	ImportSérieZeServeru
Stručný popis:	Systém importuje sérii ze serveru	
Primární aktéři:	Uživatel	
Sekundární aktéři:	Server	
Předpoklady:	1. Je známa serie, která je určena k importu do klientské databáze	
Hlavní tok:	1. Systém zjistí seznam řezů, které patří pod danou sérii (VyhledatSliceNaServeru) 2. Systém stáhne DICOM soubory odpovídající seznamu řezů (StáhnoutSouborDICOM) 3. Systém importuje stažené soubory do klientské databáze (ImportSouboruDICOM) 4. POKUD se při importu vyskytne chyba, bude o tom systém uživatele informovat varovným hlášením	
Následné podmínky:	Systém importoval celou sérii	
Alternativní toky:	Žádné	

ID: 22	Případ použití:	OdeslatSouborDICOM
Stručný popis:	Systém odešle soubor DICOM na server	
Primární aktéři:	Uživatel	
Sekundární aktéři:	Server	
Předpoklady:	1. Je znám soubor k odeslání	
Hlavní tok:	3. Systém odešle soubor DICOM na server 4. POKUD systém nebude moci soubor odeslat, bude o tom uživatele informovat varovným hlášením	
Následné podmínky:	Byl odeslán soubor DICOM	
Alternativní toky:	Žádné	

ID: 23	Případ použití:	ExportSérieNaServer
Stručný popis:	Systém exportuje sérii z klientské databáze na server	
Primární aktéři:	Uživatel	
Sekundární aktéři:	Server	
Předpoklady:	1. Je známa serie, která se má exportovat do serverové databáze	

Hlavní tok:	1. Systém zjistí seznam řezů, které patří pod danou sérii (VyhledatSlice) 2. Systém odešle DICOM soubory odpovídající seznamu řezů na server (OdeslatSouborDICOM) 3. POKUD se při exportu vyskytne chyba, bude o tom systém uživatele informovat varovným hlášením
Následné podmínky:	Systém exportoval celou sérii do serverové databáze
Alternativní toky:	Žádné

3.3.4 Serverová část systému



Obrázek 12 Diagram případů užití 4

ID: 24	Případ použití:	VyhledatPacientaNaServeru
Případ užití je sémanticky totožný s případem ID:1 VyhledatPacienta		
ID: 25	Případ použití:	VyhledatStudyNaServeru
Případ užití je sémanticky totožný s případem ID:4 VyhledatStudy		
ID: 26	Případ použití:	VyhledatSeriiNaServeru
Případ užití je sémanticky totožný s případem ID:7 VyhledatSérii		

ID: 27	Případ použití:	ZaložitVCESession
Stručný popis:	Systém založí novou VCE session	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	Uživatel má právo vytvářet session	
Hlavní tok:	1. Uživatel zadá jméno a popis nové session 2. Uživatel potvrdí vytvoření session 3. Systém vytvoří novou session 4. POKUD se při vytvoření vyskytne chyba, bude o tom systém uživatele informovat varovným hlášením	
Následné podmínky:	Byla vytvořena nová session	
Alternativní toky:	žádné	
ID: 28	Případ použití:	UkončitVCESession
Stručný popis:	Systém změní popis pacienta ve své databázi	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	Uživatel má právo ukončit session	
Hlavní tok:	1. Systém zobrazí otevřená session (VypsatVCESession) 2. Uživatel vybere session, která má být uzavřena 3. Systém session uzavře 4. POKUD se při ukončení session vyskytne chyba, bude o tom systém uživatele informovat varovným hlášením	
Následné podmínky:	Session byla ukončena	
Alternativní toky:	žádné	
ID: 29	Případ použití:	VypsatVCESession
Stručný popis:	Systém změní popis pacienta ve své databázi	
Primární aktéři:	Uživatel	
Sekundární aktéři:	žádní	
Předpoklady:	Uživatel má právo obnovit seznam session	
Hlavní tok:	1. Systém obnoví seznam otevřených session 2. Systém vypíše tento seznam	
Následné podmínky:	Systém vypsal aktuální seznam otevřených session	
Alternativní toky:	žádné	

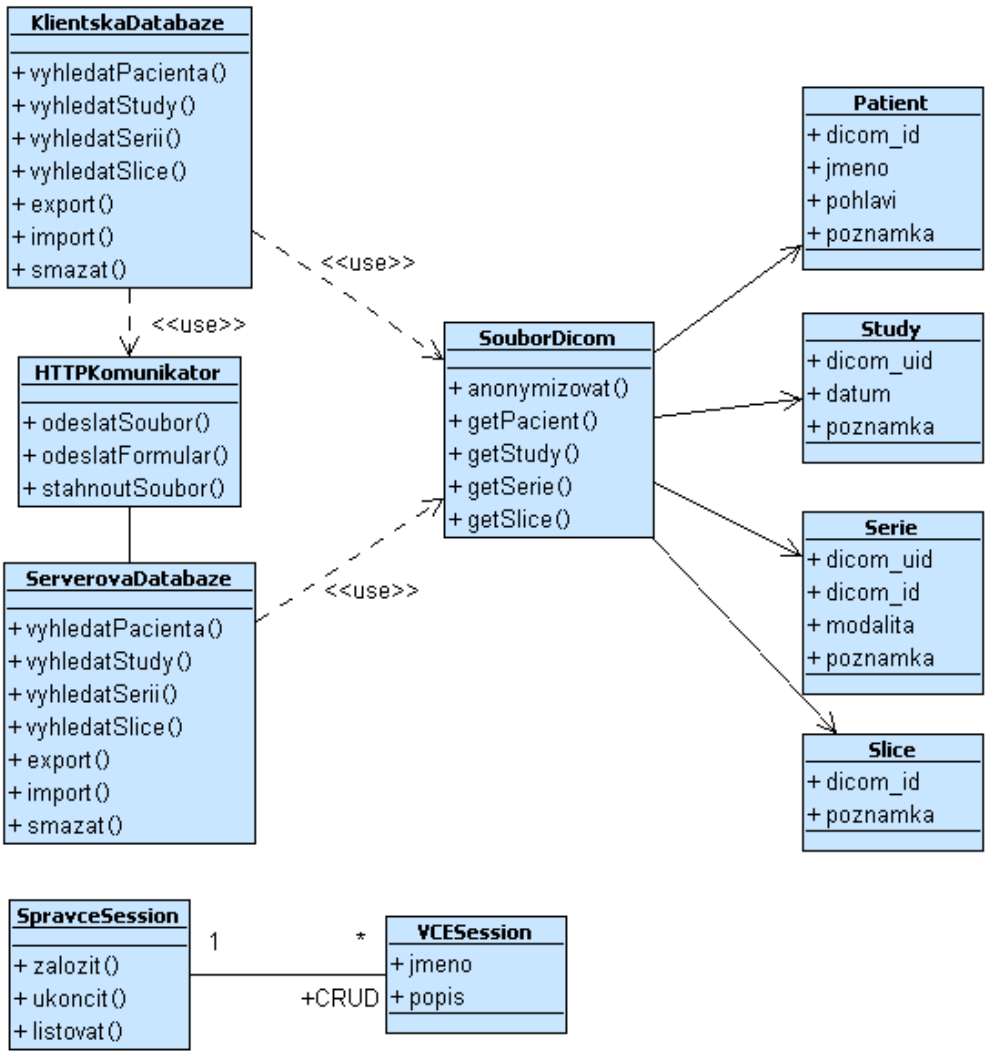
3.4 Slovníček pojmů

Pojem	Definice
Server	Serverová část systému, která se stará o synchronizaci dat Synonyma: serverová část systému
Klient	Klientská část systému, která se stará o synchronizaci dat Synonyma: klientská část systému
VCE klient	Klientská část kolaborativního systému (stará se o synchronizaci sdíleného modelu)
VCE server	Serverová část kolaborativního systému (stará se o synchronizaci klientů)
VCE superserver	Část serverového kolaborativního systému spravující otevřená kolaborativní session
VCE session	Otevřené kolaborativní prostředí, do kterého se připojují klienti se stejným sdíleným modelem Synonyma: session, sezení
Sdílený model	Předmět spolupráce (například 3D model kyčelního kloubu). Tento model je zkonstruován na základě dat ze souboru DICOM.
Soubor DICOM	Fyzický soubor, ve kterém jsou informace o pacientovi, studii, sérii, řezu a obrazová data
Patient	Záznam o pacientovi
Study	Záznam o studii
Serie	Záznam o sérii
Slice	Záznam o řezu
Serverová databáze	Databáze umístěná na straně serveru
Klientská databáze	Databáze umístěná na straně klienta

4 Analýza systému

Pro správné pochopení analytických diagramů je nutné si uvědomit, že jeden řez (slice) v hierarchickém modelu DICOM odpovídá právě jednomu souboru DICOM, v němž jsou kromě informací o samotném řezu také informace pacientovi (patient), studii (study) a sérii (serie).

4.1 Diagram analytických tříd



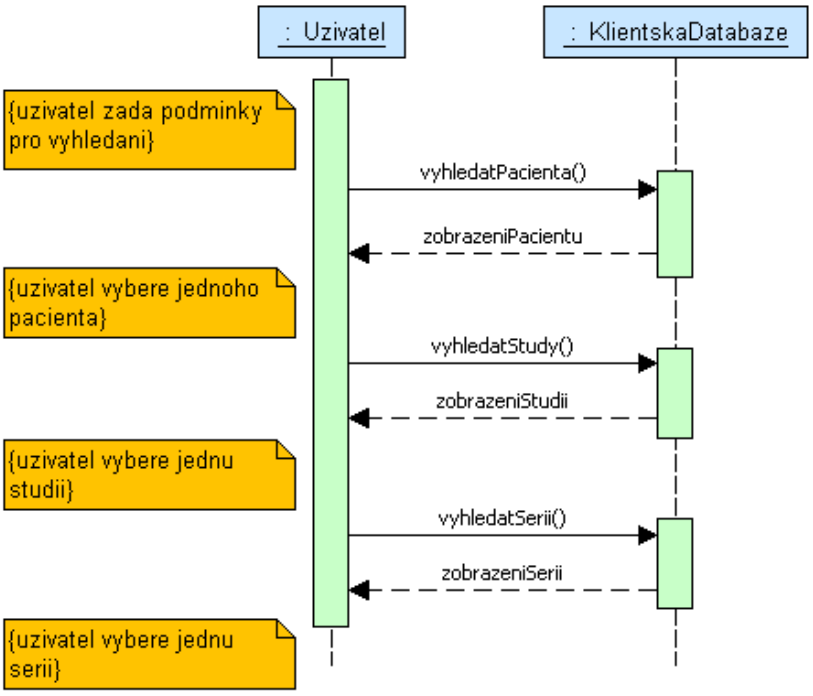
Obrázek 13 Diagram analytických tříd klienta

Analytická třída KlientskaDatabaze reprezentuje lokální klientskou databázi. Třída ServerovaDatabaze pak databázi na serveru. KlientskaDatabaze využívá pro komunikaci se střídou ServerovaDatabaze třídu HTTPKomunikator (třída reprezentuje přenos informací po síti). Obě

při importu používají třídu SouborDicom, která je jediným zdrojem entit (Patient, Study, Serie, Slice) v systému. Umožňuje taktéž data anonymizovat. Metoda smazat umožňuje odstranit jakoukoliv entitu z databáze s tím, že jsou odstraněny i entity, které pod ni patří. SpravceSession vytváří a ukončuje VCESession.

4.2 Realizace případů užití

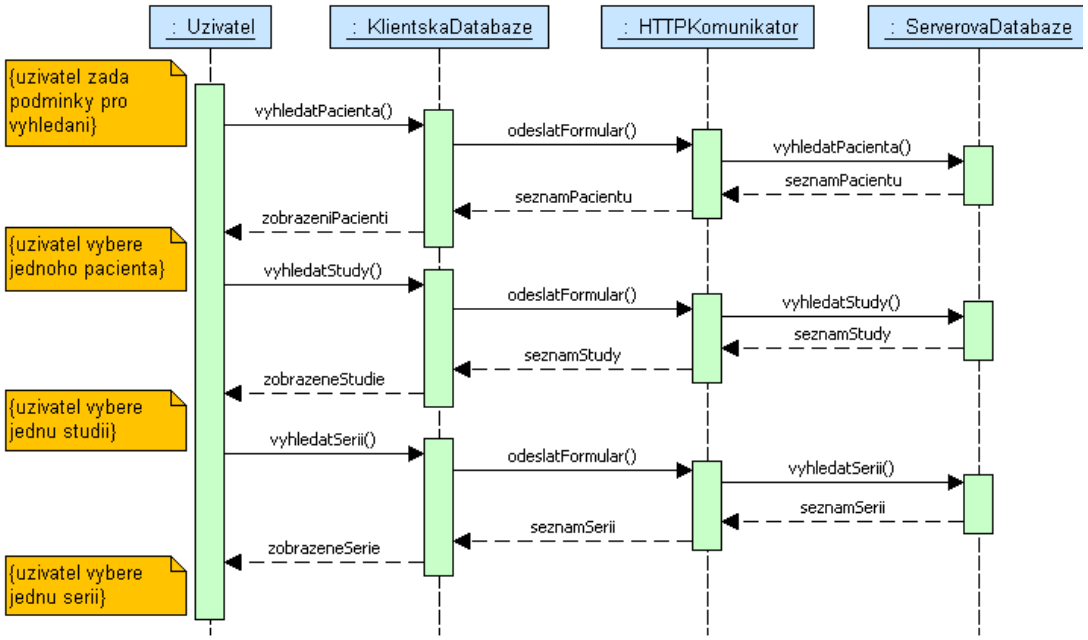
4.2.1 Vyhledání pacienta, studie a série



Obrázek 14 Diagram sekvence – Vyhledávání pacienta, studie a série

Podmínky pro vyhledávání jsou shrnuty ve funkčním požadavku 1. Diagram realizuje případy užití 1, 4, 7.

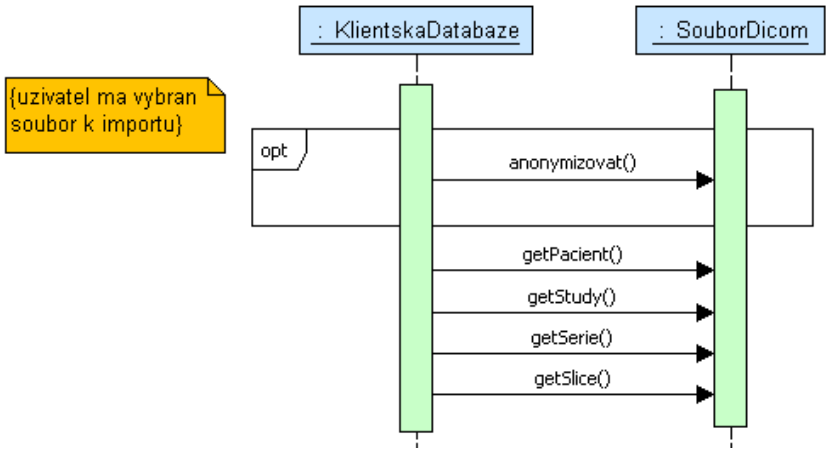
4.2.2 Vyhledání pacienta, study a série na serveru



Obrázek 15 Diagram sekvence – Vyhledávání pacienta, studie a série

Komunikaci mezi klientskou a serverovou částí zajišťuje HTTPKomunikator. Podmínky pro vyhledání přenáší jako formulář (realizace případů užití 24, 25, 26).

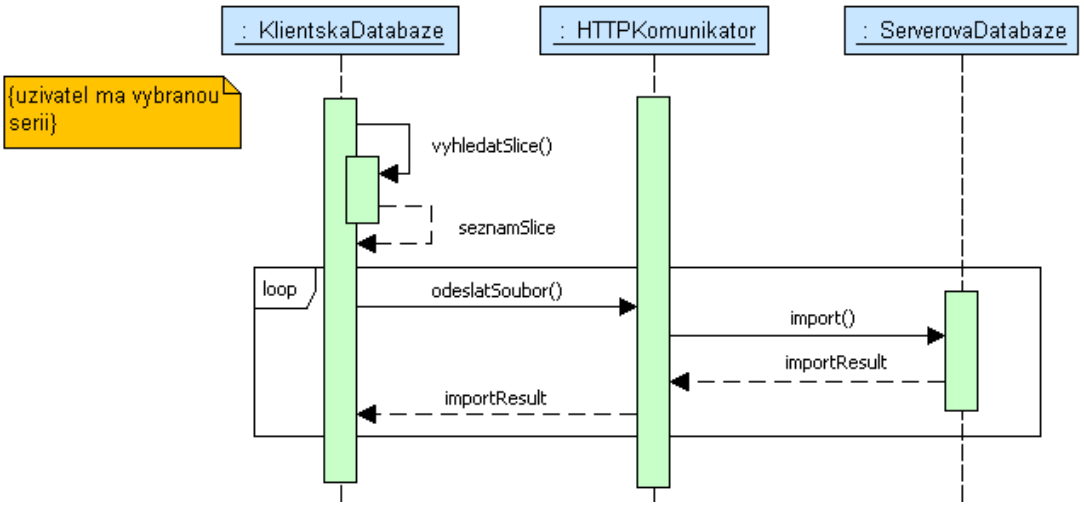
4.2.3 Import souboru do databáze



Obrázek 16 Diagram sekvence – Import souboru do databáze

Pokud nejsou záznamy o pacientovi, studii, sérii a řezu v databázi, jsou vloženy. Import souboru DICOM je do klientské i serverové databáze je sémanticky stejný (realizace případů užití 15, 21).

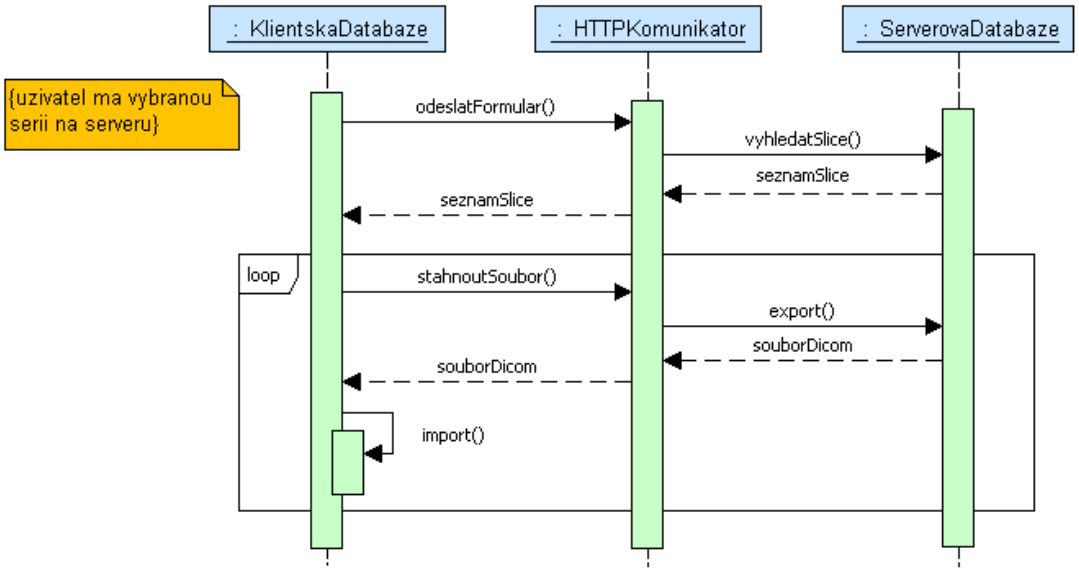
4.2.4 Import do serverové databáze



Obrázek 17 Diagram sekvence – Import souboru do databáze

Řezy patřící do jedné série jsou odeslány po jednotlivých souborech na server pomocí třídy HTTPKomunikator. Zde se z nich importem do databáze obnoví serie (realizace případu užití 15).

4.2.5 Import do klientské databáze ze serverové databáze



Obrázek 18 Diagram sekvence – Import souboru do databáze

Řezy patřící do jedné série jsou staženy po jednotlivých souborech ze serveru. Zde se z nich importem do databáze „opět vytvoří“ serie (realizace případu užití 21).

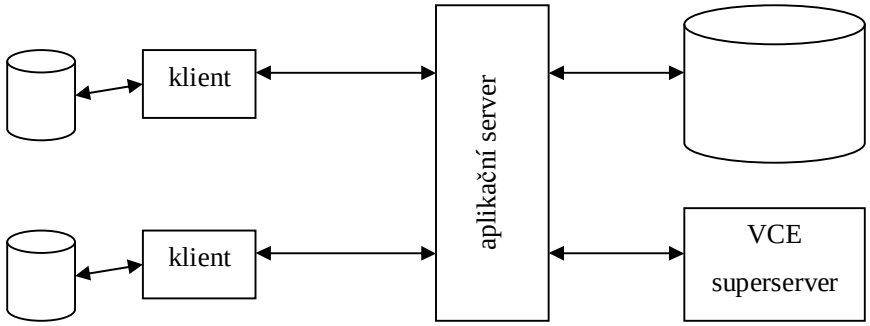
5 Návrh a implementace systému

5.1 Návrh architektury systému

Systém byl navržen jako třívrstvá architektura – klient, aplikační server, databázový server.

Klientská aplikace bude implementována jako klient, který bude spolupracovat se serverem (import export souborů), ale bude schopna pracovat i v režimu offline. Proto bude mít každý klient svou lokální databázi a v ní uložená data.

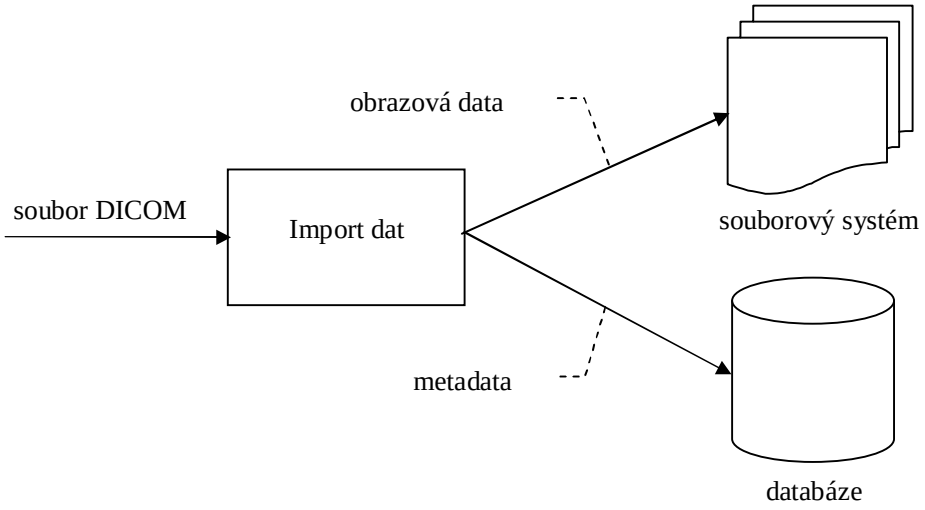
Serverová část bude poskytovat služby klientům (export a import souborů) a bude komunikovat s VCE superserverem (unixový démon, který spravuje otevřené VCE session). K dispozici bude mít robustní databázi s vlastním systémem zálohování.



Obrázek 19 Architektura systému

5.2 Návrh datového úložiště

Aby nebyla databáze (klientská a serverová) příliš zatížena objemovými obrazovými daty (jejich velikost je značná a není potřeba vyhledávat přímo v obrazových datech), jsou obrazová data uložena mimo databázi – v adresářové struktuře systému souborů (obrázek 20).



Obrázek 20 Uložení obrazových dat

5.3 Použité informace ze souboru DICOM

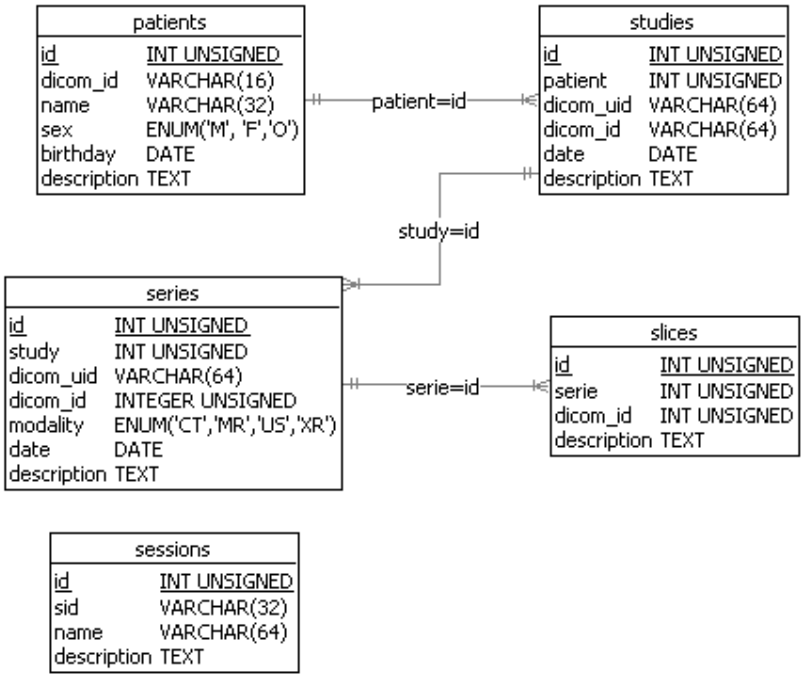
Soubor DICOM nese velké množství informací. Podle zadání však nemusíme všechna zpracovat. Například vyhledávání podle barevné hloubky obrazu nebudeme v aplikaci využívat. Třída, která bude implementovat extrakci dat ze souboru, bude číst informace pouze z elementů uvedených v tabulce. Tyto informace budou uloženy v databázi.

Název elementu	**(skupina, element)**	**Typ**	**Popis**
Jméno pacienta	(0010,0010)	text	Pacientovo jméno
Pacientovo ID	(0010,0020)	text	Jednoznačné identifikační číslo pacienta.
Datum narození pacienta.	(0010,0030)	číslo	Datum narození pacienta
Pohlaví pacienta	(0010,0040)	výčet	Pohlaví pacienta; M-muž, F-žena, O-neurčeno
Poznámka k pacientovi	(0010,4000)	text	Poznámka
Study UID	(0020,000D)	text	Jednoznačný identifikátor study
Study ID	(0020,0010)	číslo	ID jednoznačné v rámci pacienta
Datum study	(0008,0020)	číslo	Datum začátku study
Poznámka ke study	(0032,4000)	text	Komentář ke study
UID série	(0020,000E)	text	Jednoznačný identifikátor série
Číslo série	(0020,0011)	číslo	Číslo série
Modalita	(0008,0060)	výčet	Modalita obrazových dat; CT, MR, US, XR

Datum série	(0008,0021)	číslo	Datum série
Popis série	(0008,103E)	text	Komentář k sérii
Číslo snímku	(0020,0013)	číslo	Číslo snímku v sérii

5.4 Schéma relační databáze

Schéma databáze vychází z entit identifikovaných během analýzy a návrhu systému. Jsou to patient, study, serie a slice (hierarchie DICOM). V serverové části je navíc entita VCE session.



Obrázek 21 Entitně relační diagram

5.5 Návrh komunikačních protokolů

5.5.1 Komunikace klient – server

Data mezi klientem a http serverem jsou přenášena jako obsah standardního protokolu http.

Při vyhledávání (klienta, study, serie, slice) odesílá klient na server standardní webový formulář pomocí metody POST. Pokud je vstupní formulář platný (obsahuje všechna požadovaná data), odpovídá server seznamem nalezených záznamů ve formátu XML (příloha 1). Jinak odesílá popis chyb (XML).

Export souborů ze serveru je realizován standardním požadavkem GET s jedinečným identifikátorem souboru. Pokud server nalezne požadovaný soubor ve své databázi, odešle jej jako odpověď klientovi. Jinak vrací standardní chybový kód HTTP 404.

Import souboru na server je realizován standardní metodou HTTP POST (metoda PUT, nebyla vybrána z bezpečnostních důvodů). Server informuje klienta o výsledku importu pomocí XML souboru (příloha 1) zaslaného jako odpověď na požadavek importu.

5.5.2 Komunikace server – VCE superserver

Komunikace je postavena na principu dotaz – odpověď.

Zasláním příkazu „C“ (create) superserver vytvoří novou VCE session a vrátí její identifikátor. Příkazem „D“ (delete), zaslaným s identifikátorem VCE session, superserver ukončí příslušný VCE server. VCE superserver vrací znak „1“ – pokud byla session ukončena úspěšně – jinak „0“. Po zaslání příkazu „L“ (list) vrátí superserver seznam identifikátoru VCE serverů oddělených znakem„:“.

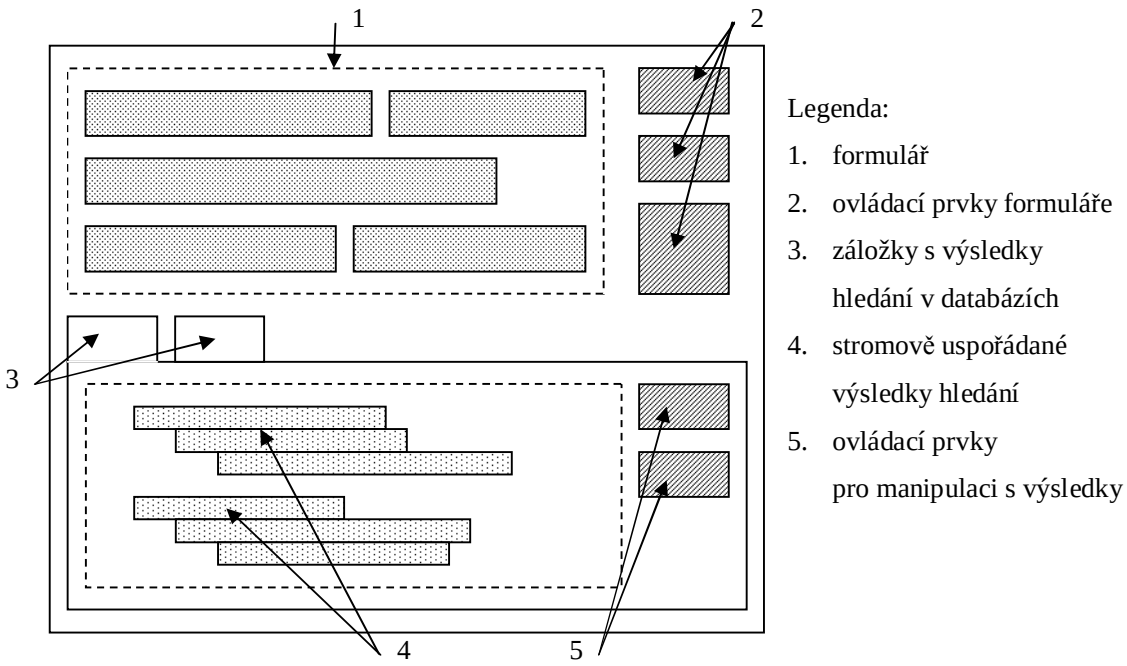
5.6 Návrh klientské části systému

Návrh je přizpůsoben implementaci v jazyce C++ a použití knihovny pro grafické uživatelské rozhraní wxWidgets. Ukládání bude zajišťovat databáze SQLite s C++ nadstavbou CppSqlite3. Komunikaci klienta se serverovou částí se bude starat komunikační knihovna libcurl. Parsování komunikačního protokolu v XML (kapitola 5.7 Návrh komunikačního protokolu) bude provádět knihovna TinyXml.

Úplná programová dokumentace klientské části se nachází v příloze 3.

5.6.1 Návrh uživatelského rozhraní klienta

Uživatelské rozhraní by mělo obsahovat formulář, do kterého bude uživatel zadávat omezující podmínky pro vyhledávání, a dále komponentu pro zobrazení výsledků vyhledávání. Protože je nutné zobrazovat výsledky vyhledávání z lokální i ze serverové databáze, bude toto vyřešeno pomocí „záložek“. Výsledky se budou zobrazovat ve stromové struktuře. Funkční prvky (tlačítka) pro manipulaci s výsledky budou při pravém okraji výsledkového stromu. Funkce asociované s konkrétní částí výsledku bude dostupné přes kontextové menu.



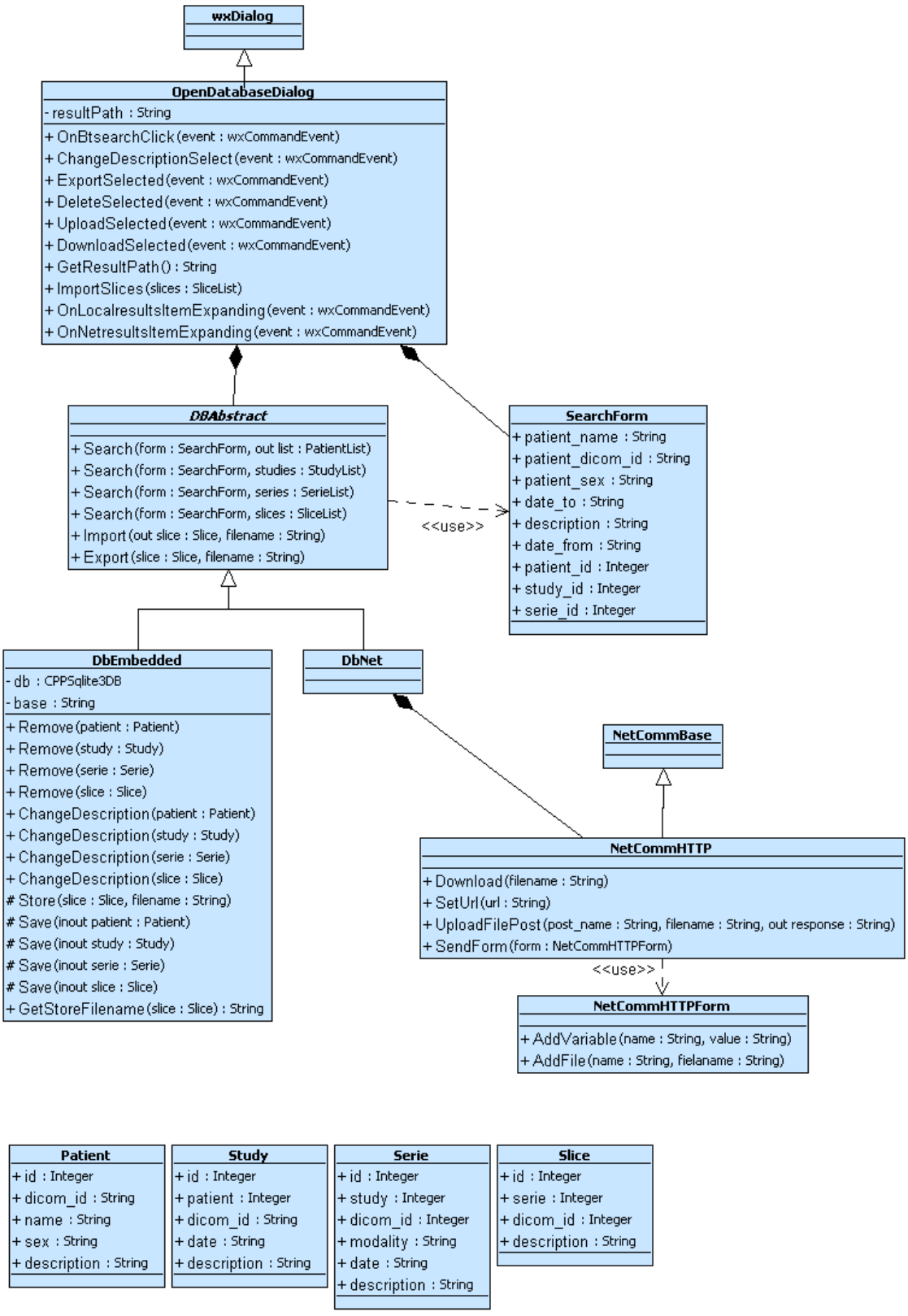
Obrázek 22 Návrh uživatelského rozhraní

5.6.2 Diagram návrhových tříd

Diagram obsahuje třídu uživatelského rozhraní OpenDatabaseDialog, která uchovává odkazy na třídy lokální (DbEmbedded) databázi a databázi připojenou přes síť (DbNet). Obě třídy implementují rozhraní DbAbstract.

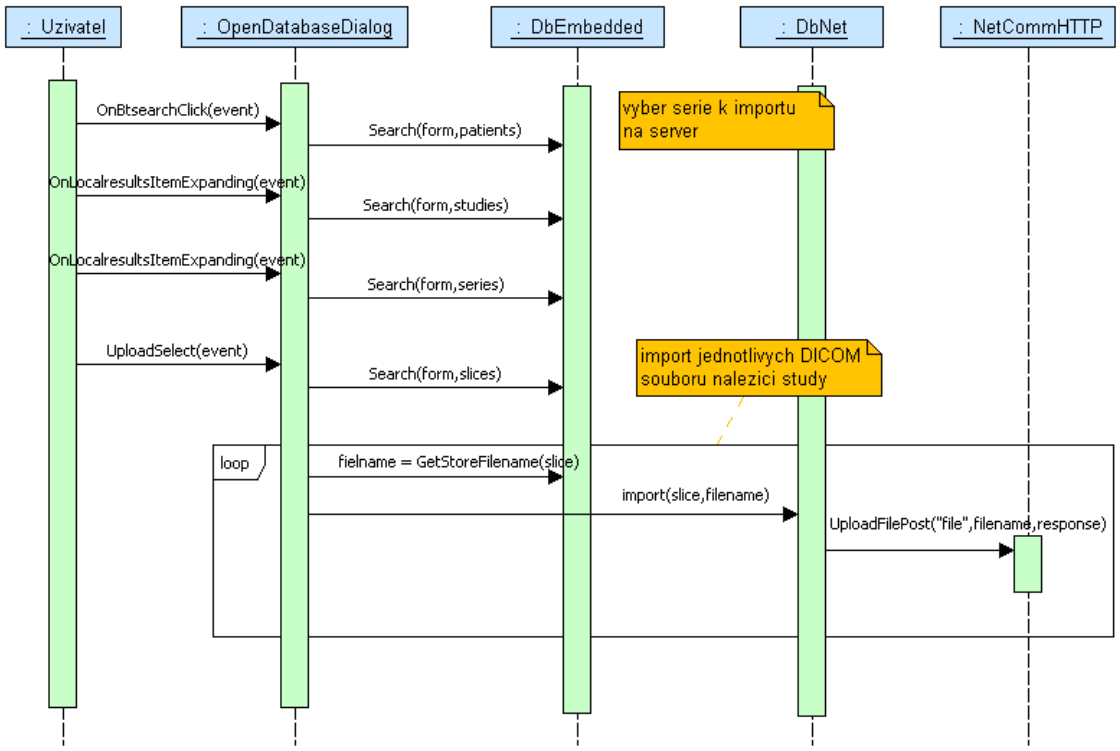
DbNet využívá pro komunikaci se serverovou databází třídu NetCommHttp. Její bázová třída (NetCommBase) používá interně knihovnu libcurl. Třída NetCommHttpForm se požívá při odesílání vyhledávacích parametrů i při odesílání souboru na server (realizováno metodou POST).

Třídy Patient, Study, Serie a Slice jsou entitními třídami.



Obrázek 23 Diagram návrhových tříd klienta

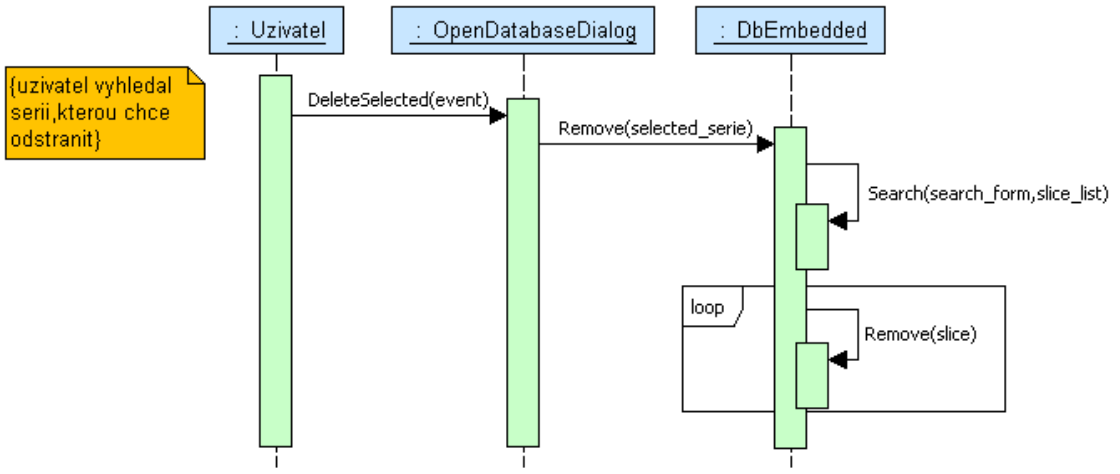
5.6.3 Import série na server



Obrázek 24 Diagram sekvence – Export série na server

Uživatel nejprve vyhledá sérii, která má být importována na server. Aplikace potom zjistí seznam řezů, které patří pod vybranou sérii, a po souboru je importuje na server.

5.6.4 Odstranění série z klientské databáze



Obrázek 25 Diagram sekvence – Odstranění série z klientské databáze

5.7 Návrh serverové části systému

Návrh serverové části systému je přizpůsoben implementaci ve skriptovacím jazyce PHP. Při implementaci serverové části systému bude použit Zend framework. Funkci databáze na straně serveru bude vykonávat MySQL.

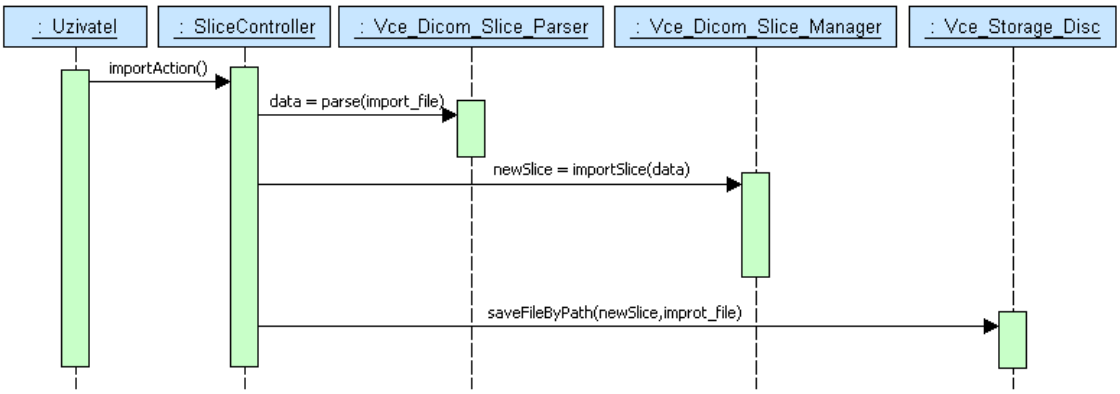
Zend framework implementuje řízení příchozího požadavku pomocí architektury MVC. Nabízí taktéž řadu tříd pro práci s databázemi (implementuje mj. architektonické vzory Table data gateway). Díky těmto třídám lze implementovat aplikaci nezávislou na konkrétní databázi.

V implementace architektura MVC v tomto frameworku mapuje požadavky na metody kontrolérů. Odeslané formuláře a proměnné jsou dostupné přes bázovou třídu kontroléru (Zend_Controller_Action).

Výstup může být jak v jazyce HTML (pro webové rozhraní), tak i v XML (pro zpracování aplikacemi). Přepínání mezi těmito dvěma vrstvami zajišťuje komponenta View modelu MVC.

Úplná programová dokumentace serverové části se nachází v příloze 3.

5.7.1 Import souboru DICOM

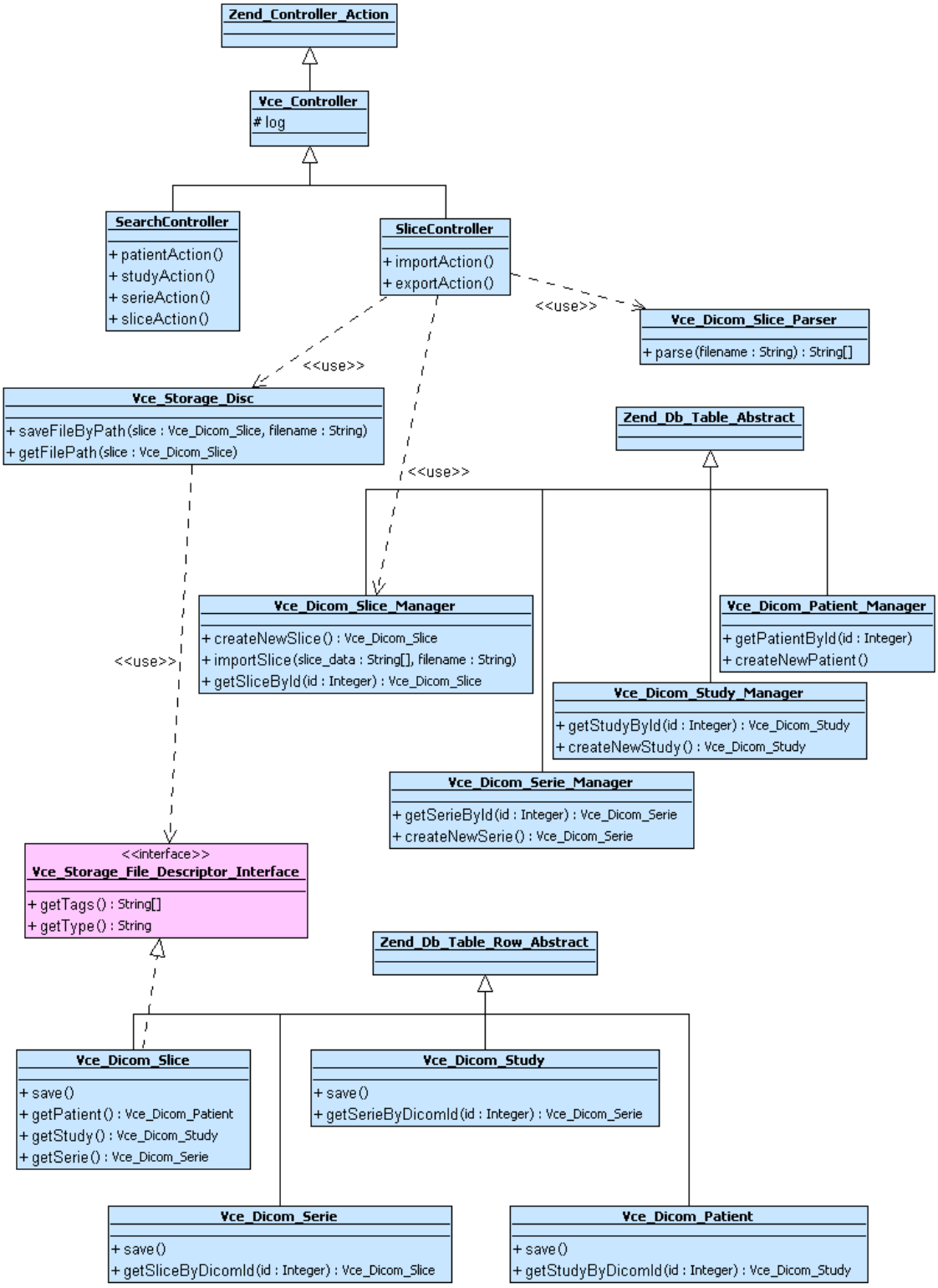


Obrázek 26 Diagram sekvence – Import souboru DICOM do serverové databáze

Uživatel nebo databázový klient odešle soubor DICOM metodou POST na server. SliceController nejprve získá ze souboru potřebná data (jméno pacienta, id pacienta, …). Pokud se v databázi nenachází příslušný pacient, tak je vytvořen (podobně pro study, sérii a řez). Nakonec se soubor DICOM uloží do úložiště. Místo, kam se soubor do filesystému uloží, určuje k němu příslušný objekt Vce_Dicom_Slice, který slouží jako přístupový deskriptor.

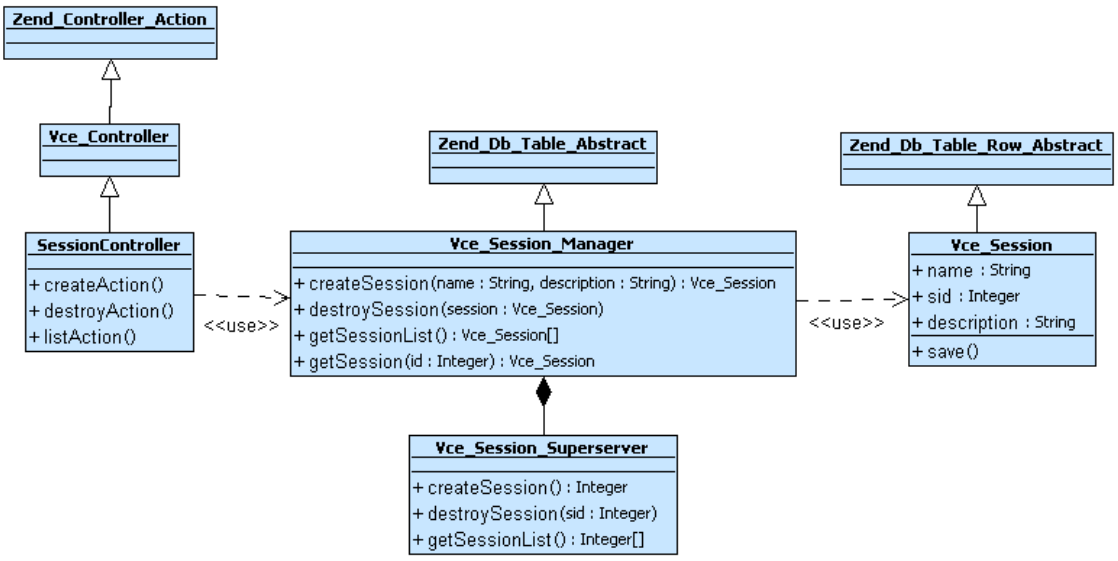
5.7.2 Diagram návrhových tříd – správa souborů DICOM

Diagram ukazuje statickou strukturu tříd části systému implementující správu souborů DICOM. Kvůli rozsáhlosti diagramu jsou některé atributy a pomocné třídy vynechány.



Obrázek 27 Diagram návrhových tříd

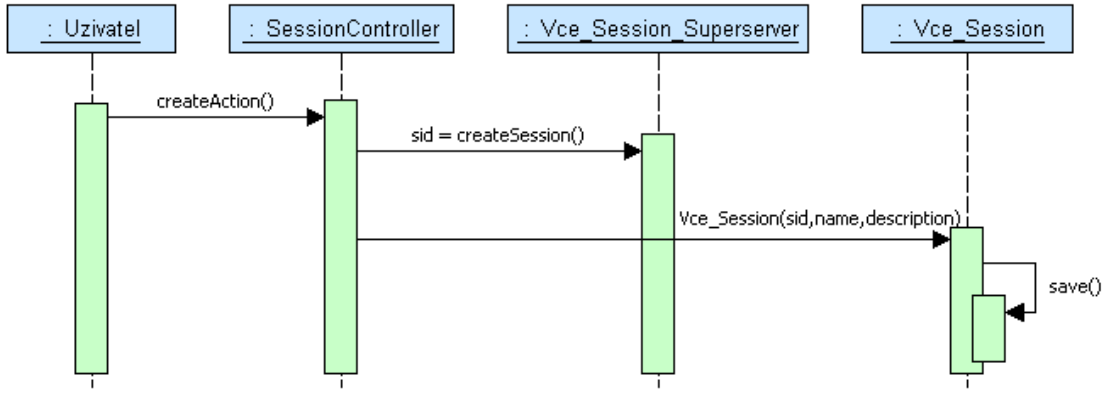
5.7.3 Diagram návrhových tříd – správa VCE session



Obrázek 28 Diagram návrhových tříd – správa VCE session

Třída Vce_Session_Superserver realizuje komunikaci s unixovým démonem VCE superserver pomocí unixových socketů (popis protokolu v kapitole 5.5.2). Hodnota sid session jednoznačně identifikuje.

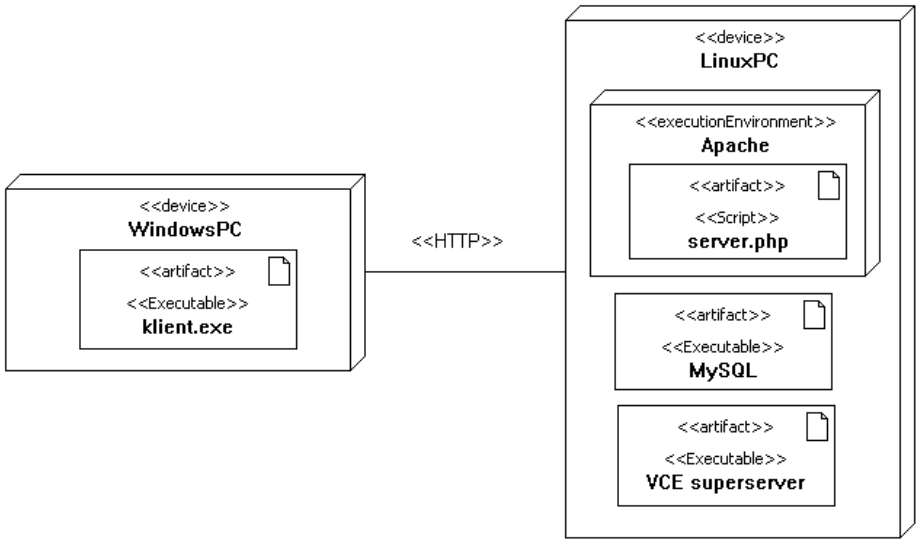
5.7.4 Vytvoření nové VCE session



Obrázek 29 Diagram sekvence – Vytvoření nové VCE session

Uživatel (přes webové rozhraní) nebo klientská aplikace kolaborativního prostředí si vyžádá spuštění nové session. SessionController spustí novou session prostřednictvím instance třídy Vce_Session_Superserver. Získaný identifikátor společně s popisem a jménem session uloží do databáze prostřednictvím instance třídy Vce_Session (vytvořena třídou Vce_Session_Manager).

5.8 Diagram nasazení



Obrázek 30 Diagram nasazení systému

Diagram nasazení zobrazuje klienta běžícího na klientském počítači (klient je však napsán multiplatformě), který komunikuje pomocí protokolu HTTP se serverem. Server vykonává požadované činnosti. Přitom využívá služeb databáze MySQL (export a import souborů DICOM) a VCE superserveru (správa VCE session).

5.9 Testovaní implementace

Hotový systém prošel několika testy funkcionality. Tyto testy by měly ověřit správnost implementace algoritmů a zjistit chování systému v různých situacích, které lze očekávat při jejím běžném provozu. Dosažené časové výkony zjištěné těmito testy jsou pouze orientační. Jejich výsledek závisí na mnoha faktorech (hardware počítače, aktuální zatížení systému, rychlost sítě a dalších).

5.9.1 Import souborů

Tato skupina testů má zjistit správnost a efektivitu implementace importování souborů.

5.9.1.1 Import z lokálního adresáře

Podmínky testu

Do prázdné klientské databáze se budou importovat soubory, které jsou umístěny v jednom adresáři. Adresář obsahuje pouze soubory ve formátu DICOM. Soubory jsou různé. Počet snímků v adresáři je 44. Očekává se správnost importu (správné zařazení) – zjišťuje se jeho rychlost.

Výsledek testu

Klient importoval všech 44 souborů. Všechny soubory byly správně zařazeny. Doba trvání importu byla průměrně 13 vteřin – tj. asi 3 soubory za vteřinu (testováno 3x).

5.9.1.2 Import ze serverové databáze

Podmínky testu

Do prázdné klientské databáze se budou importovat soubory ze serverové databáze. Průměrné zpoždění sítě je 6 ms, průměrná rychlost přenosu 490 KB/s (stahování). Počet importovaných snímků je 11 – celková velikost snímků je 5809 kB. Očekává se správnost importu (správnost zařazení) – zjišťuje se jeho rychlost.

Výsledek testu

Klient importoval všech 11 souborů. Všechny soubory byly správně zařazeny. Doba trvání importu byla průměrně 16 vteřin (testováno 3x). Celkový čas přibližně odpovídá době stahování souborů (5809/490 = 12 s) sečtené s dobou importu 11 souborů (z předchozího testu: (13*11)/44 = 3 s).

5.9.1.3 Import do serverové databáze

Podmínky testu

Do serverové databáze se budou importovat soubory z klientské databáze. Průměrné zpoždění sítě je 6 ms, průměrná rychlost přenosu 490 KB/s (odesílání). Počet importovaných snímků je 11 – celková velikost snímků je 5809kB. Očekává se správnost importu (zařazení souborů do databáze) – zjišťuje se jeho rychlost.

Výsledek testu

Server importoval všech 11 souborů. Všechny soubory byly správně zařazeny. Doba trvání importu byla průměrně 14 vteřin (testováno 3x). Přibližně to odpovídá době odesílání souborů (5809/490 = 12 s) + doba importu 11 souborů ().

5.9.1.4 Import s duplicitami

Podmínky testu

Do klientské databáze se budou importovat soubory, které jsou umístěny v jednom adresáři. Adresář obsahuje pouze soubory ve formátu DICOM. Počet snímků v adresáři je 44, z toho 20 souborů je stejných (24 jedinečných). Očekává se správnost importu (ohlášení duplicit).

Výsledek testu

Klient správně importoval 24 souborů. Zbylých 20 označil jako duplicitní. Doba trvání importu byla průměrně 6 vteřin (testováno 3x).

5.9.1.5 Import souborů do neprázdné databáze

Podmínky testu

Do klientské databáze obsahující 20 souborů se budou importovat soubory, které jsou umístěny ve více adresářích. Adresář obsahuje kromě souborů ve formátu DICOM i další 2 soubory s příponou „txt“. Celkový počet snímků v adresářích je 44. Všech 20 snímků v databázi je obsaženo i v importovaných souborech. Očekává se správnost importu (ohlášení duplicit).

Výsledek testu

Klient našel 44 souborů DICOM a správně importoval 24. Zbylých dvacet označil jako duplicitní. Doba trvání importu byla průměrně 6 vteřin (testováno 3x).

5.9.2 Obnova poškozené databáze (klient)

Tento test ověřuje správnost a efektivitu implementace obnovy klientské databáze.

Podmínky testu

Klientská databáze obsahuje 44 souborů. Soubor databáze je záměrně poškozen. Testuje se rychlost obnovení a funkčnost nové databáze.

Výsledek testu

Klient obnovil databázi v průměru za 13 vteřin. Vytvořil zálohu původní databáze a správně importoval všech 44 souborů z původní poškozené databáze. Nová databáze je zcela funkční.

5.9.3 Vyhledávání

Tento test ověřuje správnost a efektivitu implementace vyhledávání snímků v klientské i serverové databázi.

Podmínky testu

Databáze obsahuje nějaké záznamy. Provádí se vyhledávání podle nejrůznějších kritérií tak, aby vždy byla nalezena alespoň jedna položka. Dále jsou zadány podmínky, které nesplňuje žádná položka. Očekává se správné vyhledání záznamů – testuje se rychlost.

Výsledek testu

Systém správně vyhledává podle všech kritérií. Při zadání dotazu bez vyplněných kritérií klient (i webové rozhraní) hlásí příliš obecný dotaz. Při zadání nesplnitelných podmínek systém ohlásí prázdný výsledek dotazu. Při tomto počtu souborů v databázi je rychlost vyhledávání neměřitelná (pouze při vyhledávání záznamů na serveru se projevuje zpoždění způsobené přenosem dotazu a výsledku přes síť).

5.9.4 Výpadek spojení se serverem

Tento test ověřuje chování klienta při výpadku spojení se serverem.

Podmínky testu

Při komunikaci klienta se serveru dojde k výpadku spojení. Klient musí regulérně ohlásit výpadek v adekvátním čase a nabídnout způsob řešení.

Výsledek testu

Při importu do klientské databáze z databáze serverové byl simulován výpadek sítě odpojením počítače od počítačové sítě. Klient po asi 6 vteřinách ohlásil, že není schopen dokončit požadovanou operaci a korektně nabídl import již stažených souborů.

5.10 Použité nástroje

Při návrhu a implementaci byly použity následující nástroje:

- Eclipse SDK s pluginy PDT (vývojové prostředí pro jazyk PHP), C++, Topcased (vývojové prostředí pro jazyk UML) a SVN (http://www.eclipse.org/)
- Kompilátor MinGW verze 3.5.2 (http://www.mingw.org/)
- DialogBlocks 4.24 – vývojové prostředí pro wxWidgets (http://www.anthemion.co.uk/dialogblocks/)
- TortoiseSVN – systém pro správu verzí (http://tortoisesvn.tigris.org/)
- phpMyAdmin – správa serverové databáze MySQL (http://www.phpmyadmin.net/)
- sqlite3 – konzolový správce databáze SQLite (http://www.sqlite.org/)
- doxygen – generátor programové dokumentace pro jazyk C++ (www.doxygen.org/)
- phpDocumentor – generátor programové dokumentace pro jazyk PHP (http://www.phpdoc.org/)

6 Výsledky

Systém implementovaný podle analýzy zadání umožňuje práci s databází obrazových medicínských souborů DICOM.

Systém se skládá z klientů a serveru, který slouží jako centrální datové úložiště. Každý klient má svou vlastní lokální databázi, se kterou je možné pracovat offline.

Klientská část systému nabízí funkce pro import souborů do své databáze. Import zahrnuje rozpoznání duplicit v importovaných souborech. Klient dokáže do své databáze naimportovat průměrně 3 soubory za vteřinu (rychlost ověřená při testovaní implementace), což je rychlost dostatečná pro běžné použití na klientských stanicích.

Vyhledávání v naimportovaných datech je velice rychlé díky použití databázových indexů (správa těchto indexů se ovšem negativně projevuje při importu). U vyhledaných záznamů systém umožňuje přidání popisu.

Klientská část umožňuje exportovat vyhledaná data do vybraného adresáře. Export může být proveden na všech úrovních hierarchie pacient – studie – serie.

Sérii vyhledanou v lokální databázi lze odeslat do databáze umístěné na serveru. Doba importu serie na server je nejvíce ovlivněna rychlostí sítě – doba zpracování na straně serveru je oproti době přenosu souborů po síti téměř zanedbatelná.

Klient se serverovou částí komunikuje pomocí protokolu HTTP a umožňuje vyhledávat záznamy o vyšetřeních na serveru. Série vyhledané v serverové databázi je schopen stáhnout a naimportovat do své lokální databáze.

Pokud dojde k poškození lokální databáze (havárie disku) je klientská aplikace schopna vytvořit databázi novou a importovat do ní soubory ze staré poškozené databáze. Rychlost této operace závisí zejména na velikosti původní databáze.

Serverová část systému obsluhuje požadavky klientů a navíc spolupracuje s kolaborativním systémem VCE. Umožňuje vytvářet nová VCE session a později je i rušit. Obnova serverové databáze je zajištěna pravidelným zálohováním na úrovni souborového systému.

Pro práci bez klienta server poskytuje webové rozhraní, které ovšem umožňuje pouze vyhledávání a import souborů (příloha 2).

7 Závěr

Výsledná implementace systému pokryla svým rozsahem požadavky na databázový archív obrazových medicínských dat pracující v síťovém prostředí. Systém implementuje následující funkce:

- import a export souborů DICOM do klientské i serverové databáze
- vyhledávání v klientské i serverové databázi podle požadovaných kritérií

Klientská část systému navíc spolupracuje s kolaborativním systémem VCE (plní funkci dialogu pro výběr zdroje snímků pro kolaborativní prostředí). Díky použití vestavěné databáze nezatěžuje klientská část systému hostitelský počítač a takto ušetřený výkon může být využit například kolaborativní aplikací při vizualizaci sdíleného modelu.

Podle zkušeností s provozem této implementace by se mohlo přistoupit ke změně indexování v databázích. Rychlost systému by se dala dále zvyšovat optimalizací implementovaných algoritmů (např. procházení adresářů při importu), popř. jejich úplnou výměnou za výkonnější.

Návrh a implementace by se mohla doplnit o přístupová práva k informacím v databázi (může se jednat o citlivá osobní data) nebo dokonce o šifrování dat v samotných databázích. S tím souvisí i způsob komunikace mezi klientskou a serverovou částí. Současná implementace komunikuje pomocí protokolu HTTP, který není zabezpečený (data jsou posílána v otevřené podobě). Zabránění odposlechu (prozrazení přístupových hesel, osobních dat, …) by se dalo dosáhnout například použitím protokolu HTTPS (protokol HTTP, který je zabezpečen na nižší protokolové vrstvě protokolem SSL). Protokol HTTPS by navíc umožnil serveru identifikovat klienta (uživatele) na základě jedinečného klientského certifikátu.

Literatura

[1] National Electrical Manufacturers Association**,** Digital Imaging and Communications in Medicine (DICOM), 1. března 2008. Dokument dostupný na URL ftp://medical.nema.org/medical/dicom/2008/08_01pu.pdf (květen 2008).

[2] Wikipedia, Endianness, 25. března 2008. Dokument dostupný na URL http://en.wikipedia.org/wiki/Endianness (květen 2008).

[3] Zendulka J.: Informační a databázové systémy: Úvod, základní pojmy databázových technologií. Vysoké učení technické, Fakulta informačních technologií, 2005

[3] Wikipedia, Počítačová síť, 25. března 2008. Dokument dostupný na URL http://cs.wikipedia.org/wiki/Počítačová síť (květen 2008).

[4] RFC 793, Transmission Control Protocol, září 1981. Dokument dostupný na URL http://www.ietf.org/rfc/rfc793.txt (květen 2008).

[5] Eddy W. M.: Defenses against TCP SYN flooding attacks, prosinec 2006. Dokument dostupný na URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/ (květen 2008).

[6] RFC 2616, Hypertext Transfer Protocol – HTTP 1.1, červen 1999. Dokument dostupný na URL http://tools.ietf.org/html/rfc2616 (květen 2008).

[7] RFC 2965, HTTP State Management Mechanism, říjen 2000. Dokument dostupný na URL http://www.ietf.org/rfc/rfc2965 (květen 2008).

[8] Kršek P., Španěl M., Sára V., Šiler O., Štancl V., Švub M.: Síťové kolaborativní prostředí pro 3D modelování v medicíně, In: Širokopásmové sítě a jejich aplikace, Olomouc, CZ, UPAL, 2007, s. 74-77, ISBN 978-80-244-16878

[9] Zavoral F.: Návrhové vzory. Karlova Univerzita v Praze, 2008. Dokument dostupný na URL http://ulita.ms.mff.cuni.cz/pub/predn/PRG024/DesignPatterns.ppt (květen 2008)

[10] Wikipedia, Model-view-controller, 1. května 2008. Dokument dostupný na URL http://en.wikipedia.org/wiki/Model-View-Controller (květen 2008).

[11] Fowler M.: Patterns of Enterprise Application Architecture, Addison-Wesley 2003, 533s., ISBN 03–211-2742–0

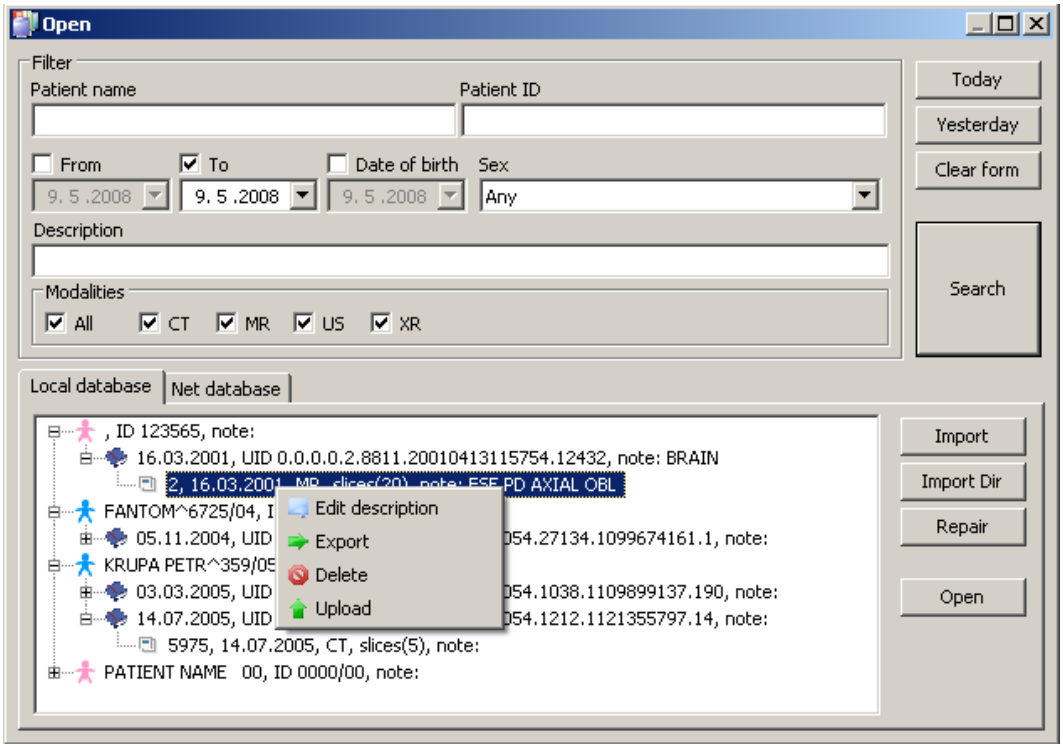
Seznam příloh

Příloha 1: DTD komunikačního protokolu

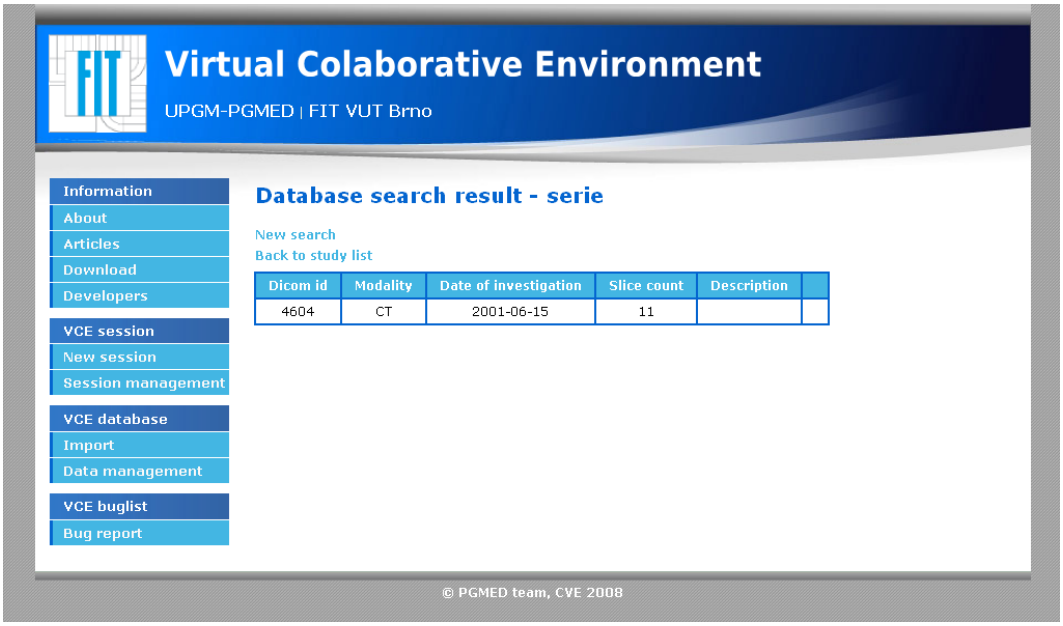
DTD popisuje odpověď serveru při vyhledávání z klienta.

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT results ((patient|study|serie|slice)+|error) >

<!ELEMENT patient (#PCDATA)>
<!ELEMENT study (#PCDATA)>
<!ELEMENT serie (#PCDATA)>
<!ELEMENT slice (#PCDATA)>
<!ELEMENT error (#PCDATA)>

<!ATTLIST patient
      id CDATA #REQUIRED
      dicom_id CDATA ""
      name CDATA ""
      sex CDATA ""
      birthday CDATA ""
      description CDATA ""
>

<!ATTLIST study
      id CDATA #REQUIRED
      dicom_id CDATA ""
      dicom_uid CDATA ""
      date CDATA ""
      description CDATA ""
>

<!ATTLIST serie
      id CDATA #REQUIRED
      dicom_id CDATA ""
      dicom_uid CDATA ""
      modality CDATA ""
      date CDATA ""
      description CDATA ""
>

<!ATTLIST slice
      id CDATA #REQUIRED
      dicom_id CDATA ""
      description CDATA ""
>
```

Příloha 2: Uživatelské rozhraní systému



Klientská část systému (výsledek vyhledávání pacientů s kontextovým menu)



Webové rozhraní serverové části systému

Příloha 3: CD

Obsah CD	
Adresář	Obsah
klient/bin	Zkompilovaná klientská aplikace pro platformu win32
klient/doc	Programová dokumentace klienta
klient/libsrc	Zdrojové soubory použitých knihoven (CppSQLite-3.1, curl-7.18.1, sqlite-3.5.8, tinyxml-2.5.3)
klient/src	Zdrojové soubory klienta
sample_data/	Anonymizované soubory DICOM použité při testování systému
server/application	Zdrojové soubory jádra serveru
server/doc	Programová dokumentace serveru
server/library	Knihovny serverové části (VCE, Zend)
server/public	Kořenový adresář webového rozhraní serveru
server/parser	Zdrojové soubory DICOM parseru
tools/	Obsahuje užitečné utility pro práci s databází SQLite a soubory DICOM
diplomova_prace.pdf	Elektronická forma diplomové práce