Univerzita Karlova v Praze
1. Lékařská fakulta

Specializace ve zdravotnictví
Zdravotnická technika a informatika

Bc. Martin Zavadil

Informační systém pro řízení klinických studií ve fakultní nemocnici

Clinical trial management information system for university hospital

Diplomová práce

Vedoucí práce: Ing. Jiří Haase, MBA

Praha 2010

Prohlášení:

Prohlašuji, že jsem závěrečnou práci zpracoval samostatně a že jsem uvedl všechny použité informační zdroje. Současně dávám svolení k tomu, aby tato závěrečná práce byla archivována v Ústavu vědeckých informací 1. lékařské fakulty Univerzity Karlovy v Praze a zde užívána ke studijním účelům. Za předpokladu, že každý, kdo tuto práci použije pro svou přednáškovou nebo publikační aktivitu, se zavazuje, že bude tento zdroj informací řádně citovat.

Souhlasím se zpřístupněním elektronické verze mé práce v Digitálním repozitáři Univerzity Karlovy v Praze (http://repozitar.cuni.cz). Práce je zpřístupněna pouze v rámci Univerzity Karlovy v Praze

Souhlasím – Nesouhlasím *

V Praze, 28.5.2010

Martin Zavadil

* Nehodící se škrtněte

Identifikační záznam:

ZAVADIL, Martin. *Informační systém pro řízení klinických studií ve fakultní nemocnici. [Clinical trial management information system for university hospital]*. Praha, 2010. 66 s., 2 příl. Diplomová práce (Mgr.), Univerzita Karlova v Praze, 1. lékařská fakulta, Vedoucí práce Haase, Jiří.

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Jiřímu Haasemu, MBA za jeho pomoc při vytváření této práce. Také bych rád poděkoval MUDr. Martinu Holcátovi, MBA náměstkovi pro léčebnou a preventivní péči Fakultní nemocnice v Motole bez jehož vstřícnosti by tato práce nemohla vzniknout.

Abstrakt

Tato diplomová práce se zabývá problematikou klinického hodnocení léčiv a jejich řízením. Klinické hodnocení léčiv je nejpodstatnější částí životního cyklu léku, které mají za cíl splnit požadavky pro registraci a běžné užívání v terapeutické praxi. Cílem této práce je návrh informačního systému pro správu a řízení klinických studií ve Fakultní nemocnici v Motole za využití Yourdonovy moderní strukturované analýzy. Jeho vypracování podle požadavků zadavatele tak, aby umožnil uživatelsky přívětivé webové rozhraní, s důrazem na usnadnění administrativních úkonů a přehlednosti dat.

Abstract

This paper is focused on questions about clinical trials of pharmaceuticals and it's management. Clinical trials of pharmaceuticals is essential part of drug's life cycle, which is aiming to meet registration requirments and to be useful for therapeutical practice. The primary goal of this paper is designing of the information system for management and control of clinical trials in University Hospital Motol using the Yourdon modern structured analysis. It's build-up according to requirments of submitter so it provides user friendly web interface with emphasis on convenience administration processes and data clarity.

Obsah

1. Klinické hodnocení léčiv

Klinickým hodnocením humánních léčiv se rozumí jejich systematické testování, prováděné na léčených osobách nebo zdravých dobrovolnících za účelem prokázání nebo ověření účinků léčiv, zjištění jejich nežádoucích účinků či určení farmakokinetických a farmakodynamických parametrů. Tato definice klinického hodnocení či klinické studie znamená, že léky, které jsou registrovány a standardně používány, musely projít několika – nejméně však třemi – fázemi klinického hodnocení, během kterých je testována účinnost nového léčiva, jeho nežádoucí účinky a další vlastnosti.

V minulosti se v jednotlivých zemích provádělo testování nových léků za různých podmínek, které byly mnohdy dost rozdílné. Výsledky hodnocení tak nebyly srovnatelné a dostatečně validní. Zároveň mohlo docházet k poškození pacientů či zdravých dobrovolníků, kteří se testování účastnili. V 80. letech 20. století byly tedy definovány obecné podmínky provádění klinických hodnocení, známých pod souhrnným pojmem Správná klinická praxe (Good Clinical Practice, GCP). Jedná se o soubor mezinárodně uznávaných etických a vědeckých požadavků, které musejí být dodržovány při navrhování provádění a dokumentování klinického hodnocení s účastí lidských subjektů a při zpracování zpráv a hlášení o těchto hodnoceních. GCP obsahuje standardní postupy pro přípravu, provádění a zpracování výsledků studie tak, aby byla zajištěna věrohodnost dat a ochrana práv, integrity a důvěrnosti údajů o subjektech. Při provádění klinického hodnocení je nezbytné GCP principy dodržovat i proto, že jsou v České republice zakomponovány v legislativě (prováděcí vyhláška Ministerstva zdravotnictví ČR a Ministerstva zemědělství ČR č. 472/2000 Sb.). Každé klinické hodnocení musí mít schválení Státního ústavu pro kontrolu léčiv (SÚKL) a Etických komisí (multicentrické a lokální EK). Pokud nejsou tato schválení doložena, není možné klinické hodnocení provádět.

Každé klinické hodnocení má určitý design: otevřené, zaslepené (pozorovatel nebo subjekty nevědí, do které skupiny jsou zařazeni), dvojitě zaslepené studie (pozorovatel ani subjekty nevědí, do které skupiny jsou zařazeni), kontrolované placebem či používající ke srovnání standardní léčbu.

1.1 Proces klinického hodnocení

Obecně lze říci, že celý proces klinického hodnocení je tradičně dělen na čtyři fáze, přičemž fáze I až III předcházejí případné registraci léčiva. Z hlediska informačního lze studie také rozdělit na studie explorační a studie konfirmační. Explorační studie nemusejí vycházet z testované hypotézy, mohou být vedeny retrospektivně nebo prospektivně, mohou, ale nemusejí srovnávat dvě nebo více skupin. Informace získané z těchto exploračních fází klinického výzkumu pak mohou přispět a často i výrazně napomoci zhodnocení efektu i bezpečnosti nového léčiva. Toto se ukazuje velmi přínosné zejména v posledních letech, tedy v době zavádění nových cílených molekul neboli tzv. biologické léčby nádorů, kdy se zvyšuje potřeba „odhalit“ prediktivní faktory a biologické markery pro účinek takto působících léčiv. Faktem zůstává, že průkazné závěry lze pak učinit až na základě optimalizovaných randomizovaných konfirmačních studií, které jsou také uznávány všemi regulatorními orgány při závěrečném posuzování nového léčiva. Konfirmační studie představují kontrolovaný optimalizovaný systém založený na srovnání různých léčebných postupů, většinou současného standardního a experimentálního ramene léčby.

Pokud bychom se vrátili k tradičnímu dělení, pak celý proces klinického hodnocení dělíme na fázi I až IV. Klinická hodnocení fáze I navazují bezprostředně na preklinické experimenty prováděné na laboratorních zvířatech (na zvířecích modelech). Jedná se tedy o experimenty, ve kterých je nově vyvíjený léčivý přípravek poprvé podán lidským subjektům. V této fázi vývoje léčiv je prioritou hodnocení bezpečnosti, primárním hodnoceným cílem je nejčastěji stanovení tzv. maximální tolerovatelné dávky (MTD), kterou je možné definovat jako takovou dávku přípravku, při níž jsou projevy toxicity ještě akceptovatelné, případně

kontrolovatelné a zvládnutelné ošetřujícím personálem. Takovou dávku přípravku, která již způsobuje nepřípustné a nezvládnutelné projevy toxicity, označujeme jako tzv. dávku limitující toxicitu (DLT).

Na základě exaktně stanovené MTD v projektech fáze I se stanovují optimální dávkovací režimy hodnoceného přípravku pro testovaní v následných fázích klinického hodnocení, zaměřených již především na analýzu účinnosti (fáze II až IV). Pro většinu onkologických studií fáze I platí, že jsou z důvodu charakteru podávaného hodnoceného léčiva prováděny za účasti pacientů, což klade významné nároky na provádění studie nejen z hlediska odborného a organizačního, ale také z hlediska etického. Počet zařazených subjektů hodnocení do experimentů fáze I nebývá zpravidla vyšší než 12 až 20. Randomizace se v experimentech

fáze I používá pouze v případě nezávažných diagnóz jako např. při vývoji přípravků pro hypertenzi, běžné infekce apod., například u onkologických studií etické zásady neumožňují tento princip použít.

Cílem klinického hodnocení fáze I není pouze výpočet MTD, ale rovněž zjištění spektra nežádoucích účinků, které hodnocený přípravek může vyvolat, tedy popis tzv. bezpečnostního profilu přípravku.

Součástí klinických hodnocení fáze I bývají často také farmakokinetické, tzv. PK studie. Cílem farmakokinetických analýz je získat podrobnější informace o osudu léčivých přípravků v organismu, předmětem zájmu jsou údaje o vstřebávání přípravku (absorpce), dále o jeho rozdělování (distribuci) v organismu, metabolických přeměnách, interakcích s ostatními látkami a vylučování.

V experimentech fáze II je cílem zhodnocení účinnosti. Výsledky těchto experimentů jsou významným rozhodovacím bodem žadatele o registraci léčivého přípravku pro jeho testování v dalších fázích. Každý krok testování je značně finančně a organizačně náročný, a je proto nutné testovat jen přípravky se skutečně velmi nadějnými vlastnostmi. Hlavním cílem této fáze testování je tedy neukončit ji

v případě slibné účinnosti, a naopak ukončit v případě průkazu účinnosti nedostatečné.

K tomu, abychom byli schopni vyhodnotit účinnost nového léčiva již v této fázi klinického hodnocení, používáme při plánování protokolu klinické studie tzv. surrogate endpoints, neboli zástupné, náhradní cílové parametry. Zhodnocením těchto parametrů bychom měli být schopni predikovat klinický přínos nového léčiva, přičemž musíme mít stále na paměti, že závěrečné hodnocení celkového přínosu daného léčiva či nové léčebné kombinace musí vycházet z přísně naplánované konfirmační studie fáze III.

Nejčastěji používaným parametrem ve smyslu „surrogate endpoint“ je tzv. response rate (RR). Response rate je možno definovat jako procentuální dosažení kompletních nebo parciálních odpovědí na podávanou léčbu, a to v předem definovaných časových intervalech. Obecně lze říci, že response rate je považován za vhodný parametr například při objektivizaci protinádorové aktivity, neboť spontánní regrese nádoru je výjimečná a zmenšení nebo vymizení nádoru pak lze jednoznačně přičíst léčebnému efektu podávané léčby.

Fáze II a III klinického hodnocení jsou nejčastější v podmínkách pro která vzniká tento informační systém.

2. Informační systémy

Informační systém je soubor lidí, technických prostředků a metod (programů), zabezpečujících sběr, přenos, zpracování, uchování dat, za účelem prezentace informací pro potřeby uživatelů činných v systémech řízení. U informačního systému (IS), stejně jako u jiných systémů, rozlišujeme strukturu a chování. Struktura, tj. rozmístění prvků, se zabývá prací s informacemi. Chování, tedy vlastní informační proces, zahrnuje jednotlivé operace s informacemi. To znamená získávání (sběr) informací, přenos informací od zdroje k místu zpracování, evidování na místě zpracování, ukládání informací pro jejich budoucí použití, třídění a posuzování kvality vlastností informací, vyhledávání doplňkových

informací, jejich výběr, analýzu a syntézu, využívání informací, což je naplněním vlastního cíle práce s informacemi.

2.1 Automatizované informační systémy

Moderním typem informačního systému je automatizovaný informační systém, který v hlavních nebo ve vybraných informačních operacích (vazbách) používá informační techniku a technologii. Potenciální informace mají v takových informačních systémech podobu dat a jsou uloženy v bázích dat. V porovnání s neautomatizovanými informačními systémy mají automatizované informační systémy tyto přednosti:

- Mnohem rozsáhlejší texty, obrazy a záznamy znaků lze uchovávat na podstatně menší ploše

- Informace lze vyvolávat (získat) na obrazovku, či vytisknout na tiskárně v podstatně kratším čase a technicky je možné je přenášet na geograficky neomezené vzdálenosti na Zemi, a to i družicovým přenosem.

- Umožňují přístup z více hledisek k poznatkům uloženým v počítačových bázích dat. V porovnání se způsobem ukládání a vyhledávání v tradičních dokumentech (knihách, časopisech) přináší ukládání znalostí ve znalostních (kognitivních, expertních) bazích dat automatizovaných informačních systémů nové možnosti reprezentace znalostí, než je tomu v knihách, časopisech, odborných dokumentech.

2.2 Rozdělení systémů

Obecně systémy dělíme na tvrdé a měkké. Tvrdé systémy jsou spojovány s jedním specifickým problémem, naopak v měkkých systémech vystupuje celá řada faktorů, které jsou obecnější. Podle toho, zda nastává interakce s okolím dělíme systémy na uzavřené a otevřené. Dále pak systémy dělíme na deterministické a stochastické, což značí jednoznačné nebo statistické chování.

Statické a dynamické, tzn. lineární nebo diferenciální systémy a podle časových událostí dělíme systémy na spojité a diskrétní.

2.3 Význam a důvody budování efektivního informačního systému

Přínos IS by měl být především v oblasti strategických výhod. Využití IS je cesta, jak se vyrovnat s rostoucí složitostí rozhodovacích procesů. Tyto procesy jsou spojené především s rostoucím počtem skutečností, které je nutné brát v úvahu při rozhodování, dále pak se zkracováním doby potřebné k rozhodnutí, s růstem rizik z opožděného nebo chybného rozhodnutí a také se zrychlením inovací. Z marketingového hlediska umožňuje IS lepší chápání vývoje na trhu a chápání potřeb zákazníků, zlepšení spolupráce se zákazníky, získávání lepších informací o chodu podniku, sledování obchodních trendů a využívání získaných informací pro optimalizaci, a také menší závislost na jednotlivých pracovnících. Klíčovou podmínkou dosažení přínosů IS je včasnost a dostupnost informací pro všechny, kteří jej potřebují ke své práci.

2.3 Projektování informačního systému

Vývoj softwaru (SW) jako základní podmínky používání informačních technologií je komplikovaná a nákladná činnost. Souhrnem metod a zásad vývoje SW, jeho instalací a udržováním v provozu se zabývá softwarové inženýrství. Organizace řízení tvorby a vývoje systému má tyto fáze:

- marketing a specifikace cílů – hledání odpovědí na otázky *proč* a *co*
- specifikace požadavků – stanovují se termíny řešení, zpřesňují ceny, stanovují zdroje, formuluje se přesná odpověď na otázku *co*, částečně na otázku *jak*
- návrh systému – volba hardwaru (HW), volba SW, návrh rozhraní a struktury dat

- kódování (programování) částí
- testování – testování částí (unit tests), integrační testování, testování funkcí, předávací testování
- oživení a předání – instalace HW a základního SW, instalace systému, předávací testy, zkušební provoz
- údržba – odstraňování chyb zjištěných za provozu, vylepšování funkcí (corrective maintenance)
- stažení z provozu

2.4 Architektura systému

Informační systémy mohou mít různou architekturu, jsou aplikovány v různých oblastech a mohou využívat nejrůznější techniky. Rychlý vývoj a velká konkurence přináší stále nové produkty a nová jména. Z hlediska architektury se často uvádějí:

- Monolitní informační systémy, které jsou koncipovány jako jeden celek.
- Federativní informační systémy. Tyto IS jsou budovány jako soubor relativně samostatných systémů úzce spolupracujících prostřednictvím nadřazeného společného aparátu.
- Kooperující systémy jsou volnější verzí federalizovaných IS. Kooperující IS jsou obvykle technicky realizovány jako soubor spolupracujících aplikací bez výrazného společného aparátu.
- Distribuované IS jsou takové, které existují na síti a jejich data i procesy jsou rozprostřeny po síti, nemají tedy jediný server. Variantou distribuovaných IS jsou globální IS rozprostřené na celosvětových sítích. Na lokálních sítích lze distribuované IS navrhovat jako logický monolit, tedy

podobně jako IS nedistribuované. Distribuovanost je vlastnost do jisté míry nezávislá na tom, zda je IS monolitní, federativní či kooperující. Velké IS jsou často distribuované a kooperující.

2.5 Ochrana informací a informačního systému

Bezpečný informační systém lze definovat jako systém, který chrání informace během jejich vstupu, zpracování, uložení, přenosu a výstupu proti ztrátě dostupnosti, integrity a důvěrnosti a po jejich likvidaci proti ztrátě důvěrnosti.

Z hlediska způsobu ohrožení informačního systému rozlišujeme dva druhy. Ohrožení úmyslné a ohrožení nedbalostní. Do ohrožení úmyslného patří především tzv. počítačové pirátství a ohrožení počítačovými viry. V rámci trestné činnosti na automatizovaných informačních systémech hovoříme o počítačové kriminalitě. Nedbalostní ohrožení může být způsobené lidským faktorem (např. chybami operátorů), chybnými vstupními daty, chybami programového vybavení, selháním HW, prostředím (např. výpadek proudu, přírodní katastrofa). Při hodnocení úniku nebo zneužití informací se ukazuje, že nejslabším článkem v celém systému ochrany je lidský faktor. Nejrizikovějším faktorem úniku informací se jeví vlastní zaměstnanci. Odhaduje se, že 80-90% případů porušení ochrany informací je způsobeno právě jimi.

Komplex ochrany informačního systému by měl zahrnovat bezpečnost osob, bezpečnost majetku, bezpečnost informací. Ochranou informačního systému rozumíme komplex organizačních (administrativních i režimových), technických, programových a sociálně-personálních opatření s cílem minimalizace možných ztrát informací v daném systému vznikajících a obíhajících.

2.6 Bezpečnostní management a jeho úloha při ochraně informací

Bezpečnostní management je zaměřený na komplexní ochranu vlastnictví, a to fyzického i nehmotných statků, včetně osob. Hlavní činnost bezpečnostního manažera je vytváření strategie informační bezpečnosti organizace. Z hlediska

ochrany jednotlivých objektů (prvků systému) lze rozlišovat následující druhy opatření:

- organizační (organizační řád, pracovní řád, technologické postupy)
- administrativní (předpisy, pokyny, směrnice apod.)
- režimové (pokyny pro činnost v krizových situacích)
- právní (preventivní, represivní)
- fyzické (předpisy stavební, elektrotechnické, vzduchotechnické a další)
- technická (pro HW i SW)
- personální (pro výběr a výchovu personálu) a sociálně-psychologické

V zájmu bezpečnostního managementu je důsledná aplikace všech dostupných bezpečnostních opatření. Současným uplatněním a zohledněním bezpečnostních principů, základních rizikových oblastí a dostupných bezpečnostních opatření lze dosáhnout vysoké míry bezpečnosti neboli minimalizace nebo úplného odstranění rizik pro chráněné subjekty.

3. Metody a použitý software

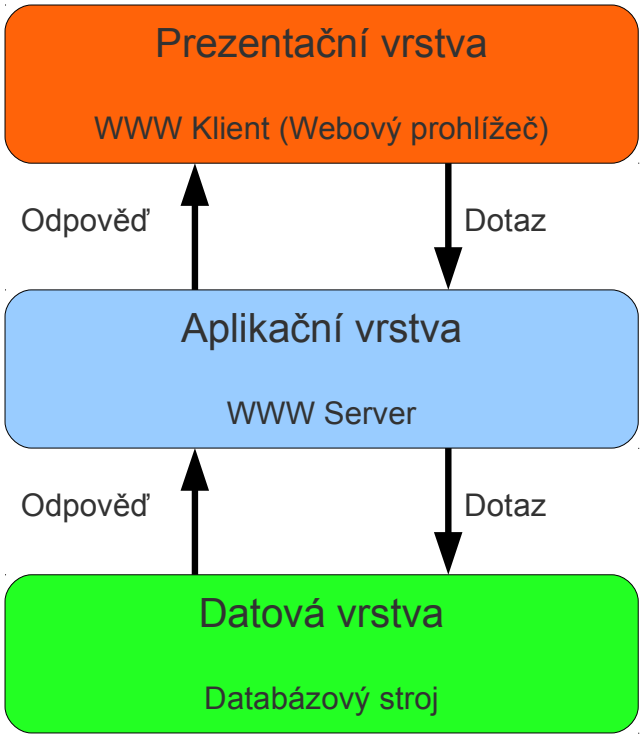
Při vypracování své diplomové práce jsem využil základní metody běžně používané při vývoji informačních technologií. Jako základ mi posloužila třívrstvá architektura, která je nejznámějším případem vícevrstvé architektury a na jejímž principu je provozováno mnoho webových aplikací.

Všechny softwarové prostředky, které jsem využil pro vypracování práce jsou šířeny pod některou z Open Source licencí. Základem byl operační systém Linux s verzí jádra 2.6.31, konkrétně distribuce Ubuntu ve verzi 9.10 desktop a 10.04 desktop.

Ubuntu je linuxová distribuce pro pracovní stanice, založená na Debian GNU/Linux. Je sponzorována společností Canonical Ltd (vlastněnou Markem Shuttleworthem) a název distribuce je odvozen z jihoafrického pojmu *Ubuntu* znamenajícího přibližně „lidskost ostatním“. Na rozdíl od Debianu pravidelně zveřejňuje nové verze každých 6 měsíců s podporou na dalších 18 měsíců; tímto způsobem se Ubuntu snaží poskytnout aktualizovaný a stabilní operační systém pro běžného uživatele s použitím svobodného softwaru.

3.1 Třívrstvá architektura

Většina informačních systémů je vyvíjena na základě třívrstvé architektury složené, jak již z názvu vyplývá, ze tří vrstev, které mezi sebou komunikují. Vrstvy datové, která je představena objektovým nebo relačním databázovým serverem, vrstvy aplikační, což jsou různé aplikace nebo aplikační servery a nakonec vrstvy prezentační, která data nějakým způsobem zobrazuje, to zajišťuje internetový prohlížeč.



Obr. 1 Třívrstvá architektura

3.1.1 Datová vrstva

Nejnižší vrstvou je vrstva datová, která zajišťuje napojení na systém řízení báze dat a také základní operace nad daty, jako jsou ukládání a výběry. K realizaci datové vrstvy jsem zvolil relačně databázový systém s otevřeným zdrojovým kódem PostgreSQL 8.4.3.

PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než patnáct let aktivního vývoje a má vynikající pověst pro svou spolehlivost a bezpečnost. Běží nativně na všech rozšířených operačních systémech včetně Linuxu, UNIXů (AIX, BSD, HP-UX, SGI-IRIX, Mac OS X, Solaris, Tru64) a Windows. Stoprocentně splňuje podmínky ACID, plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury. Obsahuje většinu SQL92 a SQL99 datových typů, např. INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP. Vývojáři PostgreSQL se snaží o respektování a implementaci standardu ANSI SQL. K systému existuje kvalitní volně dostupná dokumentace

včetně českých překladů FAQ a FAQ pro operační systémy firmy Microsoft. Výkonnostně nezaostává za srovnatelnými komerčními systémy a v častokrát je i předčí.

PostgreSQL je šířen pod PostgreSQL licencí (The PostgreSQL Licence [6]) , která je jednou z nejliberálnějších open source licencí. Tato licence umožňuje neomezené bezplatné používání, modifikaci a distribuci PostgreSQL a to ať pro komerční nebo nekomerční využití. PostgreSQL můžete šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně.

3.1.2 Aplikační vrstva

Aplikační neboli funkční vrstva je prostředníkem mezi prezentační vrstvou a vrstvou datovou. Provádí transformaci dat mezi jejich požadavky. Pro aplikační vrstvu jsem zvolil jazyk PHP ve verzi 5.3.2.

PHP (rekurzivní zkratka pro *PHP: Hypertext Preprocessor*, původní název ovšem byl *Personal Home Page tools*) je široce použitelný Open Source víceúčelový skriptovací jazyk, který speciálně určený pro vývoj webových aplikací a může být dobře zakompnován do HTML kódu. Syntaxe jazyka je inspirována několika programovacími jazyky a je relativně jednoduše naučitelná. Hlavním cílem PHP je upožnit webovým vývojářům psát dynamicky generované webové stránky poměrně snadným způsobem a za přiměřeně krátkou dobu.

PHP skripty jsou spouštěny na straně serveru a v prezentační vrstvě je tak zobrazen pouze výsledek jejich činnosti. Tato filozofie je velice užitečná a umožňuje serveru provést více operací a výsledek odeslat prohlížeči jako obyčejnou HTML stránku. Také umožňuje měnit výsledný obsah stránky na základě vstupních proměnných, což já osobně považuji za největší výhodu PHP vedle otevřené licence.

3.1.3 Prezentační vrstva

Vrstva prezentační zajišťuje odeslání požadavku uživatele a prezentaci jeho výsledku. Jako prezentační vrstvu jsem vybral kombinaci kaskádových stylů, a jazyka HTML.

Pro publikování informací v globální distribuci je zapotřebí univerzálně srozumitelný jazyk. Určitý druh mateřského jazyka, kterému by mohly všechny počítače porozumět. Publikační jazyk v prostředí World Wide Web je HTML.

HTML dává autorům internetových stránek prostředky k publikování online dokumentů s nadpisy, textem, seznamy, fotografiemi a mnoha dalšími. Umožňuje získávat online informace prostřednictvím hypertextových odkazů. Nabízí možnost návrhu formulářů pro řízení transakcí se vzdálenými službami, pro použití při vyhledávání informací, vytváření rezervací nebo objednávání produktů. V neposlední řadě lze vkládat různé interaktivní tabulky, video a audio soubory a další aplikaci přímo do dokumentu.

HTML bylo původně vyvinuto Timem Berners-Lee pro CERN a popularizováno po uvedení webového prohlížeče Mosaic, který byl produktem NCSA. V průběhu devadesátých let se dynamicky rozvíjel v přímé souvislosti s rozvojem internetu. Během této doby byl HTML rozšířen o mnoho funkcionalit.

HTML 4.01, které jsem použil pro prezentační vrstvu svého systému, je revize verze 4.0, která jako první obsahovala mechanismy pro vylepšení formulářů, odkazů, rámů, tabulek a kaskádových stylů.

3.2 Provozní prostředí

Pro provoz systému jsem určil webový server, který podporuje skriptovací jazyk PHP a kódování UTF-8 , ve kterém jsou napsány veškeré zdrojové kódy i uložená data. Vybral jsem si server Apache 2.2

Apache server je jeden z nejúspěšnějších webových serverů. Jeho úspěch je založen především na tom, že je zdarma a přitom je relativně výkonný, přenositelný

na různé platformy (na rozdíl od MS IIS) a především obsahuje velké množství modulů, které silně rozšiřují jeho užitnost.

Apache server obsahuje jádro, které je ve své podstatě triviální. Kolem jádra jsou napojeny moduly zajišťující splnění požadavků volaných z vyšších vrstev. Moduly lze připojit při kompilaci jako statické anebo je zavádět při spuštění Apache serveru (takzvané DSO moduly. Výhodou statických modulů je především rychlost, naopak nevýhodou je nemožnost připojení jiných modulů vzhledem k monolitickému zkompilování serveru. DSO moduly jsou pomalejší, změní-li se však verze některého z nich stačí překompilovat a zavést pouze tento modul a není třeba kompilovat celý server.

Vývoj Apache serveru započal v roce 1993 v National Center for Supercomputing Aplications (NCSA) na Universty of Illinois pod původním názvem NCSA HTTPd. Záhy však vývojový tým opustil hlavní programátor, čímž došlo ke zpomalení vývoje až v roce 1998 k definitivnímu zastavení celého projektu. Tento server se však mezitím stačil značně rozšířit a správci webových serverů k němu začali dodávat vlastní úpravy. V dalším vývoji sehráli hlavní úlohu Brian Behlendorf a Cliff Skolnick tím, že založili e-mailovou konferenci a začali koordinovat distribuci uživatelských úprav. Již v roce 1995 vydali první veřejnou verzi s označením 0.6.2. Následovalo založení Apache group která je dnes základním kamenem vývojářského týmu a přepsaní celého kódu.

Od roku 1996 je Apache nejpopulárnější server na internetu a ačkoliv dosáhl svého maxima okolo roku 2005 stále si drží první pozici. To byl také jeden z důvodů proč jsem zvolil právě tento server. Apache server je také úzce spjatý s operačním systémem linux a jazykem PHP.

Vývojář	*Počet*	*Procento*
Apache	112 747 166	54,55%
Microsoft	50 572 540	24,47%
Google	14 592 133	7,06%
nginx	12 673 962	6,13%
lighttpd	1 657 584	0,80%
ostatní	20 669 109	6,99%
Celkem	212 912 494	100,00%

Tab 1, - podíl na trhu serverů na všech doménách (březen 2010) zdroj http://news.netcraft.com/

3.3 Programy použité pro návrh a implementaci systému

Pro vývoj informačního systému jsem použil pouze programy vydávané pod některou z licencích registrovaných u Open Source Initiative.

3.3.1 Gedit

Jalo základní nástroj pro psaní zdrojových kódů mi posloužil textový editor gedit ve verzi 2.30, což je oficiální textový editor pracovního prostředí GNOME. Jak uvádějí jeho tvůrci je tento editor zaměřený především na jednoduchost a snadnost používání i když je to poměrně mocný a široce použitelný nástroj. Podporuje zvýrazňování syntaxe, práci se záložkami, pluginy s dalšími funkcemi pro pohodlnější používání. Gedit je šířen pod licencí GNU General Public Licence.

3.3.2 pgDesigner 1.2.17

PgDesigner je aplikace pro navrhování PostgreSQL databází ve verzích 7.x a 8.x a poskytuje uživateli relativně rozsáhlý soubor možností. V první řadě je to kompletní editor datového modelu. Podporuje všechny PostgreSQL verze a dovoluje automatickou aktualizaci vazeb mezi tabulkami. Také obsahuje průvodce pro tvorbu pohledů a generátor reportů se statistikou. V neposlední řadě umožňuje tisk diagramů, export do SQL a řízení projektu dle tabulky diagramu.

3.3.3 Bluefish editor 1.0.7

Tento editor je užitečný nástroj určený především pro vývojáře webových stránek a má pro ně mnoho výhod. Já jsem ho bohužel příliš nevyužil, protože mi nevyhovoval jeho způsob zvýrazňování syntaxe.

3.3.4 Klienti – webové prohlížeče

Klienti slouží k prohlížení internetových stránek. Tyto aplikace dokáží komunikovat s HTTP serverem, zpracovávají kód, jako je například formát HTML, a podle stanovených standardů jej zformátují a zobrazí pak webovou stránku. Každý prohlížeč se řídí podle částečně odlišných standardů, některé části HTML kódu interpretuje jeden prohlížeč jinak než druhý a stránka se pak zobrazí v každém prohlížeči jiným způsobem. Proto jsem musel navrhnout a zformátovat kód HTML tak, aby většina uživatelů používajících různé klienty mohla přehledně se systémem pracovat. Při vývoji systému jsem používal pro zobrazení internetových stránek tři nejpoužívanější prohlížeče a testoval, zda se rozvržení stránky a jejích komponent přibližně podobá. Těmito třemi klienty jsou Microsoft Internet Explorer, Mozilla Firefox a Opera.

4 Analýza a návrh informačního systému

Při analýze systému jsem byl postaven před otázku jakým způsobem budu systém vytvářet. Zda se přikloním k objektovému přístupu a nebo zda dám přednost strukturovanému pojetí. Nakonec jsem zvolil strukturovanou analýzu. Tato analýza, typická pro oblast vývoje softwarových produktů, je užitečná zejména pro vývoj informačních systémů u kterých je častější změna funkčních celků a méně častá změna dat.

Bylo nezbytné zjistit jak funguje proces řízení klinických studií. V současné době je situace poněkud nepřehledná. V nemocnici neexistuje osoba, která by měla přehled o tom kolik a jakých typů studií je v běhu ani archiv kde by se dali vyhledat komplexní informace o studiích již ukončených. Tyto a další úkoly řeší společně útvar náměstka pro léčebnou a preventivní péči, právní odbor a jednotlivá pracoviště, kde klinická hodnocení probíhají. Z toho důvodu vznikla potřeba vytvořit systém který by přehledným způsobem zobrazoval stav klinických hodnocení napříč všemi pracovišti. Také byla vznesena otázka zda a jakým způsobem by bylo možné zobrazovat a zpracovávat základní ekonomická data týkající se studií. Především šlo o to aby vedení nemocnice mělo v případě potřeby přehledy celkových příjmů a nákladů na klinická hodnocení.

4.1 Strukturovaná analýza

Strukturovaná analýza je jednou z prvních etap životního cyklu každého softwaru. Na analýzu ovšem obvykle nestačí jen nástroje. Člověk, který systém vyvíjí, musí mít také know-how. Musí tedy vědět nejen co má použít (jaké nástroje), ale také jak to má použít (jakou zvolit metodiku, příp kombinace metodik). Ve strukturované analýze se metodika definuje jako postup ověřený zkušenostmi nebo formálně, který použitím modelovacích nástrojů vytváří ucelený systém konzistentní z vnitřku i z vnějšku. Znamená to tedy, že výsledné hodnoty (grafy, diagramy, popisy, definice) jednotlivých nástrojů budou navzájem vyvážené.

4.2 Postup při analýze a návrhu systému

Životní cyklus softwaru při použití Yourdonovy metodiky (Yourdon Modern Structured Analysis, [5]) je poněkud odlišný od jiných typů cyklů , které by se dali charakterizovat jako posloupnost analýza, návrh, implementace, testování a provoz softwaru.

Analytik nejprve soustředí své úsilí na vytvoření esenciálního modelu systému, , který nezávisí na technických ani implementačních omezeních a slouží jako neměnný, dlouhodobě stabilní základ. Z něj se pak vytváří model implementační.

YMSA je metodika typu „zdola-nahoru" - postupuje tedy od nejnižších vrstev, slučuje nejmenší procesy do větších celků až k hlavním procesům na systémovém DFD a kontextovém diagramu.

4.2.1 Esenciální model

Zde je třeba vyvarovat se implementačních závislostí: zbytečných pamětí, kontrole chyb a validace dat a procesů nebo redundancí všech typů. V esenciálním modelu pracuje každý proces nekonečně rychle. Každý datový tok představuje komunikační cestu s nulovým dopravním zpožděním.

model okolí systému - Z názvu jde poznat, že se jedná o model, který si spíše všímá pohledu na systém z vnějšku. Zaměřuje se hlavně na jeho nejbližší okolí - rozhraní systému.

model chování systému - Popisuje vnitřní uspořádání systému, zkoumá interní (řídicí) události, které nejsou z vnějšku vidět.

To jsou dvě části, ze kterých se esenciální model skládá. Každý má své postupy, jak splnit popsanou charakteristiku. Obecný způsob, jak postupovat při vytváření modelů systému podle metodiky YMSA je následující:

1. vytvoření modelu okolí systému
2. vytvoření prvotního modelu chování systému

3. dokončení esenciálního modelu
4. vytvoření implementačního modelu

4.3 Model okolí

Při vytváření modelu okolí systému používáme těchto tří prostředků - popis účelu systému, kontextový diagram datových toků a seznam vnějších událostí.

V popisu účelu stručně a výstižně formulujeme hlavní cíle, kterých by mělo být dosaženo po nasazení navrhovaného systému - účel, kvůli kterému se systém vyvíjel. Samotný jazyk popisu by mohl být méně technický, protože jde o nástroj sloužící především pro management firmy, pro kterou se systém vyvíjí.

Kontextový diagram zobrazuje toky mezi externím světem a systémem. Nadto, kontextový diagram by neměl obsahovat implementačně závislé prvky. Ty se vypouštějí a uvažuje se prozatím ideální model bez technických omezení.

Jestliže tvoří analytik pomocí metodiky YMSA seznam událostí, obvykle vychází z kontextového diagramu a zkoumá, jaké činnosti v systému je ten který terminátor oprávněn vykonávat. Alternativou je použití ERD.

Jakmile je model okolí hotov, je vhodné jej podrobit kontrole, kterou označujeme jako vyvažování a která se provádí po ukončení každé fáze a na závěr jednotlivých ucelených částí i celé analýzy. Zajišťuje se tak konzistence, robustnost a tím pádem i bezpečnost systému.

Značně výhodné je také vytvořit i další modely systému, které přímo nejsou součástí modelu okolí, a to zvláště datový slovník popisující základní datové elementy, ERD tvořící základ datového pohledu na systém. Poněkud se tak ztíží počáteční vyvažování systému, ale zároveň si analytik usnadní budoucí práci na dalších modelech.

4.3.1 Účel systému

Jak jsem již zmínil výše, účelem systému má být přehledné zobrazení informací o klinických studiích probíhajících v nemocnici, o postupu v procesu

schvalování, o jejich stavu, o podrobnějších informacích o nákladech a výnosech a přehled subjektů vystupujících v klinickém hodnocení. Tedy přehled zadavatelů, zkoušejících a pracovišť, na kterých studie probíhají. Systém byl pracovně nazván CTIS (Clinical Trial Information systém).

4.3.2 Seznam vnějších událostí

Pro události uskutečňované v systému budou uvažovány tyto uživatelské role:

Uživatel – je přihlášený uživatel, který má právo zobrazovat, přidávat, upravovat a mazat údaje v systému týkající se studií. Nemůže spravovat uživatele a údaje, které se studií přímo netýkají.

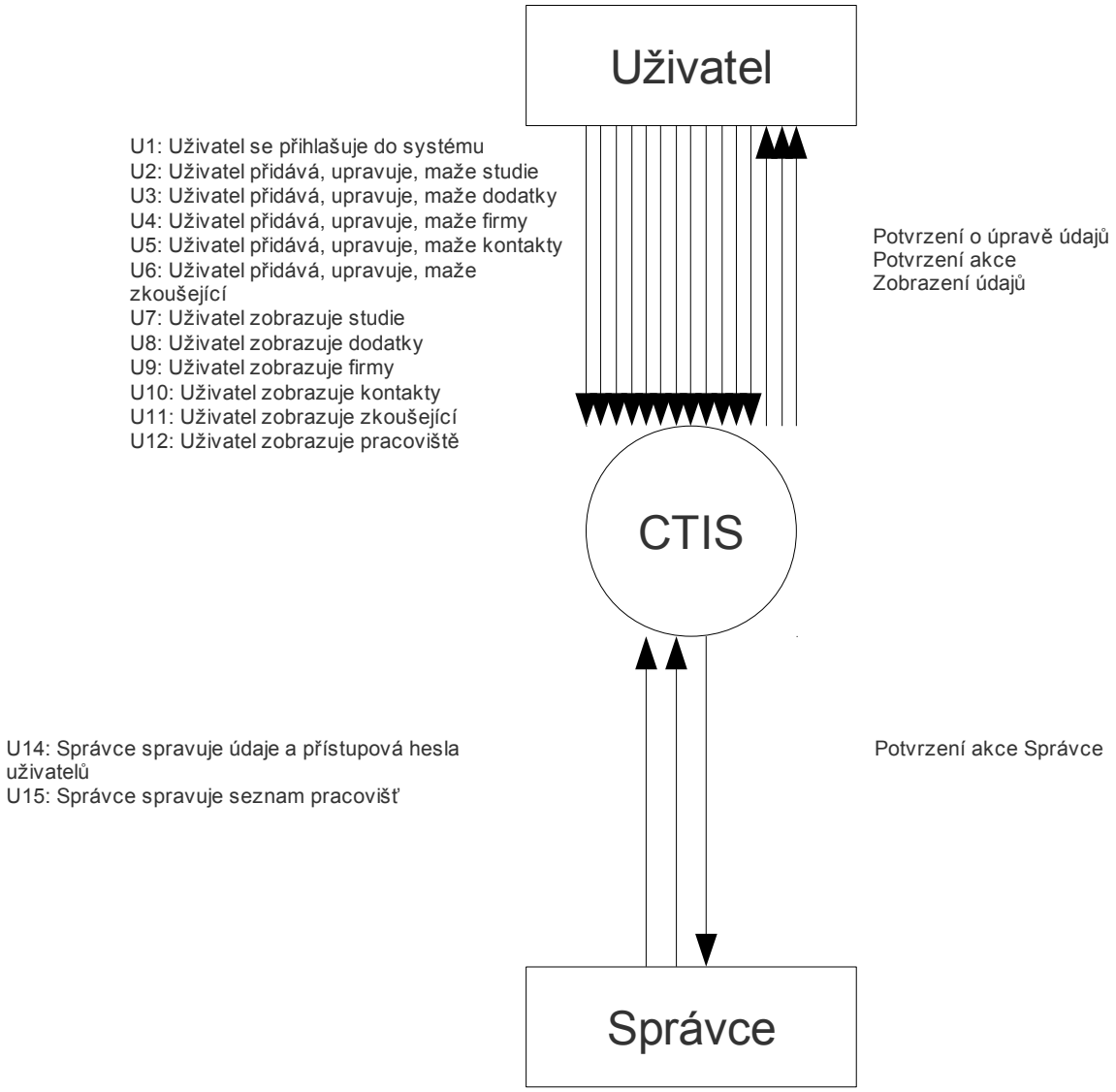
Správce – je uživatel přihlášený do systému s právem administrace. Má stejná práva jako Uživatel navíc však má právo vkládat, měnit a mazat Uživatele a spravovat všechny údaje v systému

Pro definované uživatele existují následující události:

- U1: Uživatel se přihlašuje do systému
- U2: Uživatel přidává, upravuje, maže studie
- U3: Uživatel přidává, upravuje, maže dodatky
- U4: Uživatel přidává, upravuje, maže firmy
- U5: Uživatel přidává, upravuje, maže kontakty
- U6: Uživatel přidává, upravuje, maže zkoušející
- U7: Uživatel zobrazuje studie
- U8: Uživatel zobrazuje dodatky
- U9: Uživatel zobrazuje firmy
- U10: Uživatel zobrazuje kontakty
- U11: Uživatel zobrazuje zkoušející

- U12: Uživatel zobrazuje pracoviště
- U13: Správce má stejná práva jako Uživatel
- U14: Správce spravuje údaje a přístupová hesla uživatelů
- U15: Správce spravuje seznam pracovišť

4.3.3 Kontextový diagram datových toků



Obr. 2 kontextový diagram datových toků systému CTIS

4.4 Model chování systému

I když model chování popisuje vnitřní strukturu systému, vychází jednoznačně z modelu okolí a přebírá od něj dosud vytvořené materiály (kontextový diagram, seznam událostí a popis účelu systému).

Protože YMSA je metodikou, která ve svém postupu užívá techniku typu „zdola-nahoru", využije se nejdříve seznamu událostí, a to tímto způsobem:

- Pro každou událost se vytvoří samostatný proces na nejnižší úrovni DFD a patřičně podle názvu události se proces pojmenuje.
- I když několik odezev v systému bude stimulováno jedinou událostí, pro každou takovou odezvu bude vytvořen na DFD samostatný proces.
- Naopak, pokud jedna odezva v systému pochází od vícero událostí, je sjednocena pod jeden proces. Vstupní a výstupní data toků tohoto procesu jsou pak identická pro různé události.
- Pokud se ovšem skutečně jedná o navzájem odlišné odezvy a pouze stejně pojmenované, je nutné pojmenovat každou odezvu jedinečně.
- Kde je nezbytné předávat data mezi procesy pracujícími v časovém rozpětí, doplní se esenciální paměti.
- DFD se podle potřeb doplní o řídicí procesy. Vytváří se CDFD a STD .
- Takto vzniklý diagram se označuje jako prvotní systémový DFD - diagram nulté úrovně. Zároveň s ním se vytváří i ERD a datový slovník.

Následně se systémový DFD vyvažuje vůči kontextovému diagramu a seznamu událostí z modelu okolí. V této fázi je vytvořen prvotní model chovaní systému popisující procesy pouze jako logickou strukturu - bez implementačních aspektů. Podle obecného postupu teď zbývá dokončit esenciální model a vytvořit tak základ pro implementační model.

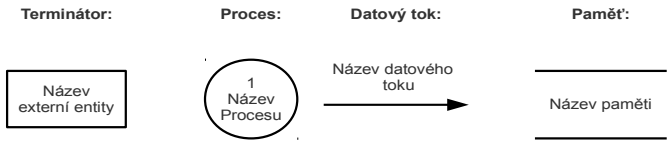
4.4.1 Víceúrovňový diagram datových toků

Víceúrovňový diagram datových toků (DFD – Data Flow Diagram) je jeden z nejpoužívanějších modelovacích nástrojů strukturované analýzy, který poskytuje funkčně orientovaný pohled na systém. Slouží tedy k modelování funkcionality systému. DFD zobrazuje systém jako síť procesů. Tyto procesy plní určité funkce a pomocí datových toků si mezi sebou předávají data. [7]

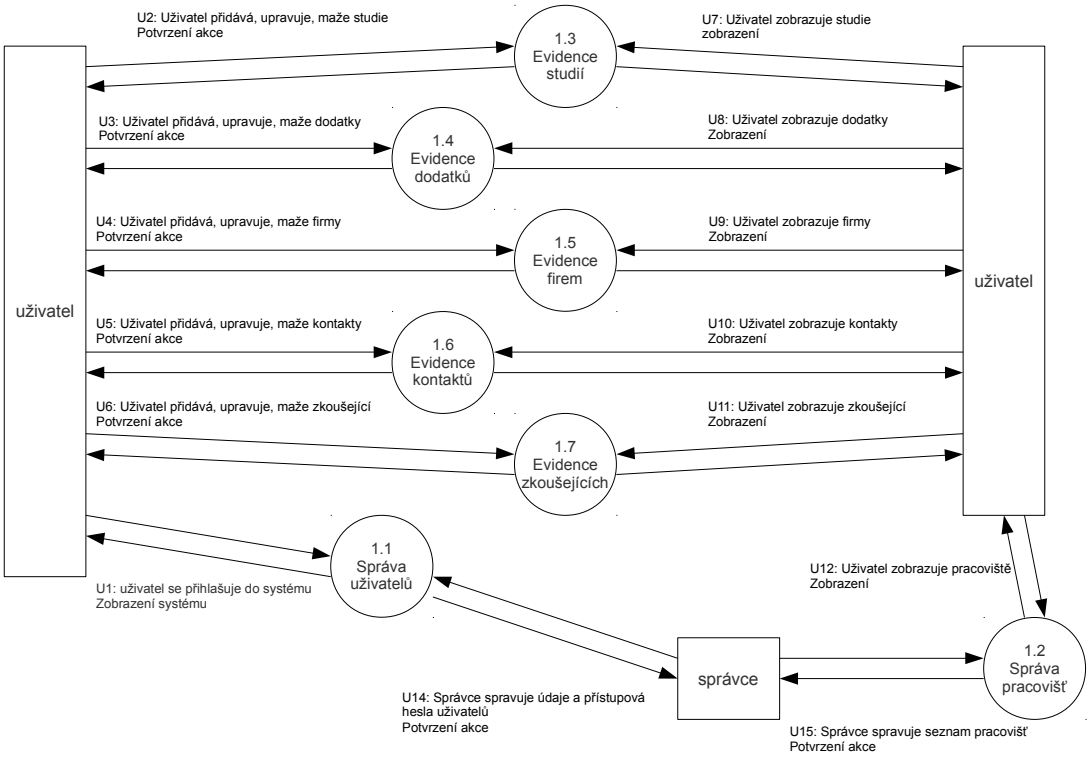
Diagram datových toků obsahuje následující čtyři komponenty [7]:

- Terminátory – reprezentují externí entity, které patří do okolí systému a se kterými systém komunikuje. Všechny informace, které systém přijímá nebo vysílá jsou vysílány, respektive přijímány terminátory. Terminátory jsou většinou osoby nebo skupiny osob, ale mohou to být i jiné systémy, se kterými systém komunikuje. Analytik ani systém nemůže změnit obsah terminátorů nebo způsob jakým pracují.
- Procesy – jsou jediné části systému, které převádějí vstupy na výstupy. Každý proces by měl být jednoznačně identifikován a vhodně pojmenován, buďto jedním slovem, jednoduchou větou nebo frází. Jméno vyjadřuje, co daný proces dělá. Ke každému procesu na DFD existuje buď minispecifikace, nebo je dekomponován na nižší úroveň DFD, kde jsou znázorněny jeho subprocesy.
- Datové toky – popisují pohyb informačních paketů nebo fyzických materiálů mezi jednotlivými částmi systému. Datové toky jsou pojmenovány podle toho jaká data přenášejí. Některé datové toky nemusí být pojmenovány, pokud je zřejmé jaká data přenášejí. Typicky to jsou datové toky směřující z nebo do paměti. Šipka znázorňuje směr toku dat. Řídící toky, které neobsahují žádná data se zakreslují přerušovanou čárou.

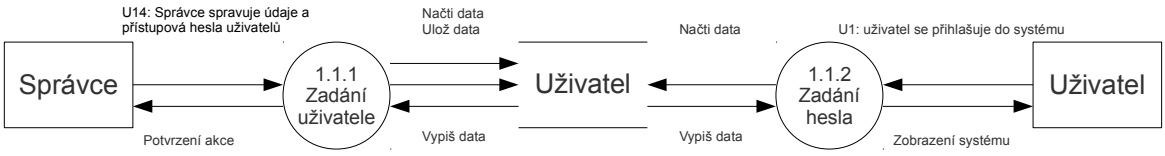
- Paměti – jsou pasivní prvky systému, kde se data ukládají pro pozdější zpracování. Povolené operace jednoho datového toku nad pamětí jsou buď nedestruktivní čtení, zápis, modifikace nebo mazání.



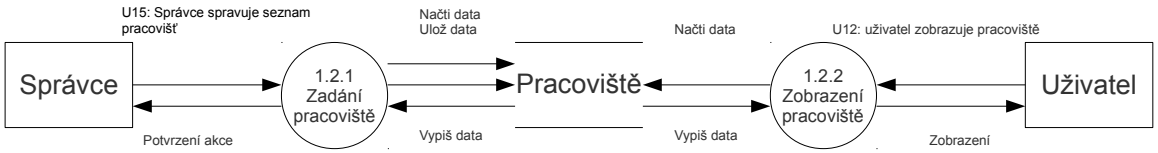
Obr. 3 Prvky DFD podle Yourdonovi notace [5]



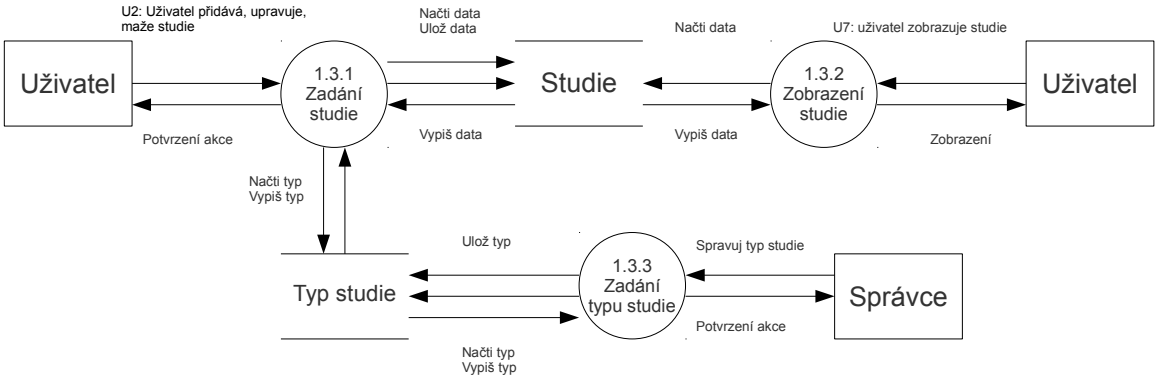
Obr. 4 DFD 1. úrovně systému CTIS



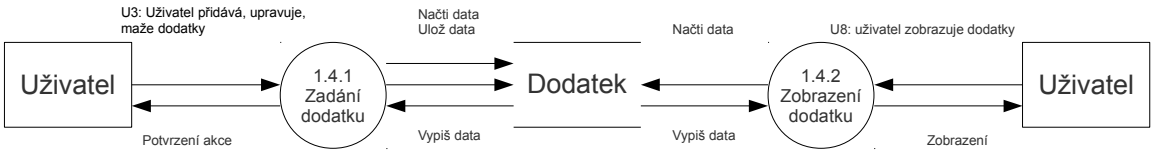
Obr. 5 DFD 2. úrovně – Správa uživatelů



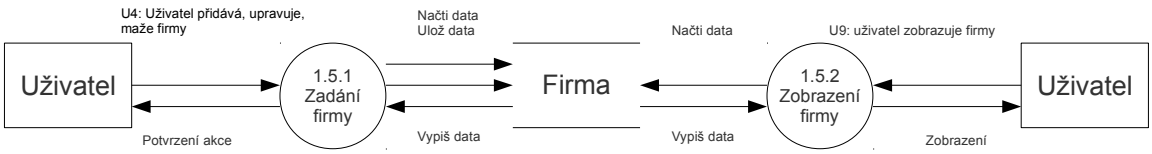
Obr. 6 DFD 2. úrovně – Správa pracovišť



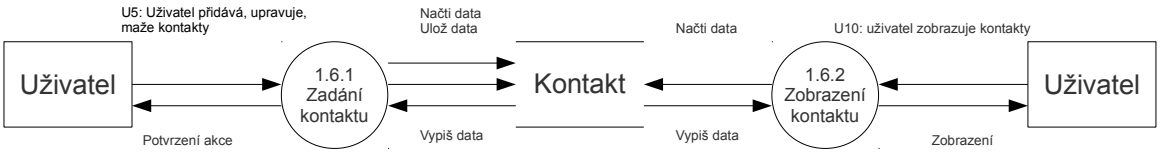
Obr. 7 DFD 2. úrovně – Evidence studií



Obr. 8 DFD 2. úrovně – Evidence dodatků



Obr. 9 DFD 2. úrovně – Evidence firem



Obr. 10 DFD 2. úrovně – Evidence kontaktů

Obr. 11 DFD 2. úrovně – Evidence zkoušejících

4.4.2 Entitně relační diagram

Entity-Relationship Diagram používá datově orientovaný pohled na systém. Jeho úkolem je prezentovat statickou formu systému. Je do jisté míry podobný relační databázi. A protože řada informačních systémů je implementována právě pomocí relační databází, používá se v praxi pro modelování datové struktury systému právě ERD. Základními prvky v ERD jsou entitní a vztahové množiny, dále entity a vztahy samotné a jejich atributy.

Entita - objekt v ERD, který reprezentuje nějaký objekt z reálného světa. Je to objekt, o němž stojí za to uchovávat si informace. Jedná se o konkrétní objekty, v případě systému CTIS se jedná například o zkoušejícího nebo studii.

Entitní množina - skupina entit stejného typu. Každá entita v dané entitní množině má stejný počet totožných atributů.

Vztah - relace, vazba, souvislost mezi entitami, o níž uchováváme informace. Vztah nelze odvodit či vypočítat, musí se jednoduše pamatovat.

Vztahová množina - Jestliže existuje mezi dvěma entitními množinami nějaký vztah, vzniká pak tzv. vztahová (relační) množina. U ní se pak tvůrce IS musí zaměřit na tzv. aritu jednotlivých relací.

Atribut - neboli datový element je nějaká důležitá vlastnost, kterou má každá entita nebo vztah a kterou potřebujeme uchovat a/nebo pracovat s ní. Může to být například telefonní číslo kontaktu, evidenční číslo studie nebo název pracoviště, na kterém studie probíhá. Entita nebo vztah mají alespoň jeden ale zpravidla více atributů. Skupina atributů, které jednoznačně určují entitu v dané entitní množině, se nazývá primární klíč. Další atributy, které by mohly být primárním klíčem, ale nebyly vybrány, se označují jako alternativní klíče.

V některých literaturách ovšem dochází k terminologickému zaměňování pojmů. Nejčastěji dochází k záměnám pojmů *typ entity* a *entita* nebo *typ vztahu* a *vztah*.

Já zde budu používat pojmy entitní (vztahové) množiny a entity (vztahu). Každá relace mezi entitními a/nebo vztahovými množinami musí mít nějakou aritu (kardinalitu, násobnost). Arita udává, kolik objektů se dané relace účastní. Arita může být různého typu a může být různě graficky znázorněna.

Může se místo nich, ale většinou společně s nimi, použít značení číslicemi a písmeny v poměru. Vyskytují se tyto typy povinných (nepovinných) vztahů 1:1. .N (1:0. .N), dále pak 1:1 (1:0,1) a konečně vztah M: N, který v praxi nelze efektivně implementovat jinak než pomocí vztahové množiny. První entitní (určující) množina má se vztahovou (závislou) množinou relaci o aritě 1:N a druhá entitní (určující) množina má s ní relaci o aritě 1:M.

Mimo nepovinné relace, kdy může být jedna entita ve vztahu i s nulovým počtem entit ze druhé množiny, existuje ještě možnostpočítat s nulovým počtem i na straně první entitní množiny. Tak vznikají relace 0 , 1 : 1 . . N nebo 0 , 1 : 0 , 1 atd. Těmto relacím, kdy je nula i na levé straně poměru, se říká neidentifikační. Pro grafické odlišení od identifikačních relací se používá čárkované značení. U identifikačních vazeb je primární klíč určující množiny součástí primárního klíče

závislé množiny. Co se týče neidentifikačních relací, primární klíč určující množiny je sice součást entity (vztahu) závislé množiny, ale už pouze jako jeden z atributů - není prvkem tvořícím primární klíč závislé množiny.

4.4.3 Popis entit systému CTIS

Název: **uzivatel**

Typ: Kernel

Objektem typu uzivatel je každá osoba přihlášená do systému.

Název: **studie**

Typ: Kernel

Objektem typu studie je každá studie definovaná svým typem, mající svého zkoušejícího a prováděná svojí firmou.

Název: **firma**

Typ: Kernel

Objektem typu firma je každá společnost, které bude možno přiřadit neb je přiřazená jedna nebo více studií.

Název: **kontakty**

Typ: Kernel

Objektem typu kontakty je osoba, které můžeme přiřadit kontaktní informaci a firmu a která má na starost jemu přidělenou klinickou studii na straně zadavatele (firmy).

Název: **typ_studie**

Typ: Kernel

Objektem typu typ_studie skupina typů klinického hodnocení do které bylo, je nebo bude možno přiřadit nějakou studii.

Název: **dodatky**

Typ: Kernel

Objektem typu dodatky jsou změny ve studiích, které podléhají samostatnému schvalovacímu řízení v nemocnici.

Název: **zkousejici**

Typ: Kernel

Objektem typu zkousejici je každá osoba, která provádí studii. Jeden zkoušející může provádět více studií avšak každá studie má pouze jednoho zkoušejícího.

Název: **klinika**

Typ: Kernel

Objektem typu klinika je pracoviště nemocnice na kterém probíhá klinické hodnocení.

Název: **stav**

Typ: Kernel

Objektem typu stav je umístění studie v jednotlivých bodech procesu schvalování. Jedná se o pracoviště, která mají v kompetenci akceptovat nebo připomínkovat studii před schválením.

Pozn. Název této entity je poněkud zavádějící ale vychází z praxe, kdy se studie v průběhu schvalování nacházejí v určitém stavu, tedy na konkrétním pracovišti, které je aktuálně posuzuje.

Název: **situace**

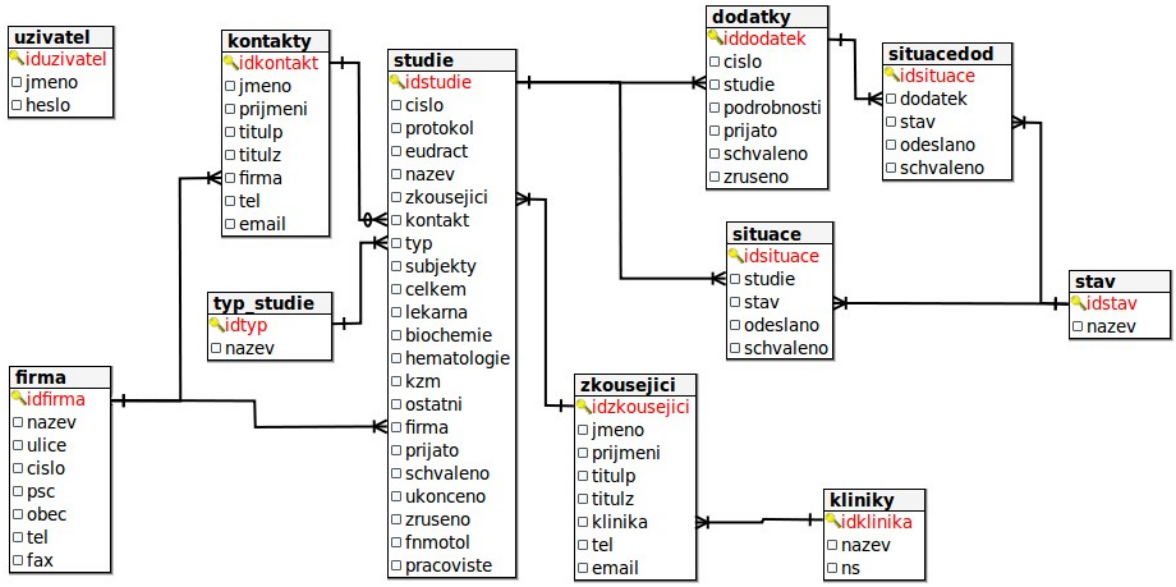
Typ: Associative

Objektem typu situace je reprezentace vazby mezi studií a jejím stavem, která určuje časové značky vstupu a výstupu z jednotlivých stavů.

Název: **situacedod**

Typ: Associative

Objektem typu situacedod je reprezentace vazby mezi dodatkem a jeho stavem, která určuje časové značky vstupu a výstupu z jednotlivých stavů.



Obr. 12 Entitně relační model systému CTIS

5 Implementace informačního systému

Systém CTIS je klient-server aplikací, která klade relativně malé nároky na hardware a software uživatelské stanice. Pro jeho provoz stačí běžný počítač s přístupem do lokální sítě a webový prohlížeč.

Architektura klient-server vychází z toho, že v drtivé většině případů je možné požadavek na data blíže specifikovat a žádat tak pouze data, která klienta skutečně zajímají. Databázový server v takovém případě nepředstavuje pouze úložiště dat, ale aktivní článek, který na základě požadavků klienta vybere vhodnou výslednou množinu dat, případně ji předzpracuje a předá k dalšímu zpracování, obvykle pouze ke zobrazení, klientovi. Značnou výhodou je v takovém případě zejména ušetření v přenosu dat a podstatně menší nároky na výkonnost klientů. Podstatnou část zpracování probíhá na straně serveru. Díky centralizovanému charakteru zpracování nabízí databázové platformy s architekturou klient-server také celou řadu dalších možností, jako je transakční zpracování, konkurenční přístup, vyspělejší zabezpečení apod. Snadnější je mnohdy také správa klient-server aplikace. Například změna logiky může znamenat pouze jeden zásah na straně databázového serveru a nikoli na všech klientech.

Vývoj databázové aplikace využívající architektury klient-server je však náročnější, nicméně vynaložené investice se vrátí díky nižším nárokům na výsledné provozní prostředí.

Architektura klient-server nemusí být vždy výhodná – například v rámci aplikací charakteru jednouživatelského jednoduchého účetnictví si vývojáři plně vystačí i s tzv. souborovými databázemi. Obdobně jako v případě všech ostatních oblastech informačních technologií i zde totiž záleží především na konkrétních požadavcích.

5.1 Návrh uživatelského rozhraní

Stěžejním bodem uživatelského rozhraní systému je bezproblémová navigace. Často se také klade důraz na jednoduchost, srozumitelnost, uživatelskou přívětivost a logické seskupení částí tak, aby ovládání bylo co nejvíce intuitivní. Při vývoji by se nemělo opomenout zpřístupnění uživatelského rozhraní pro zrakově či sluchově postižené nebo pro uživatele se zhoršenou motorikou horních končetin. Vzhledem k primárnímu určení mého systému do pracovního prostředí, kde se výskyt takových uživatelů nepředpokládá, není tato podmínka prioritou.

Jelikož systém není určen široké veřejnosti rozhodl jsem se nevyvíjet aktivitu v grafické úpravě systému ale soustředit se na to aby byl co nejvíce funkční a uživatelsky přátelský i v pouhém textovém režimu bez použití stylů. To také usnadňuje aby se systém správně zobrazoval ve většině dnes běžně používaných prohlížečích. Nakonec grafickou úpravu za pomoci CSS stylů lze dopracovat později.

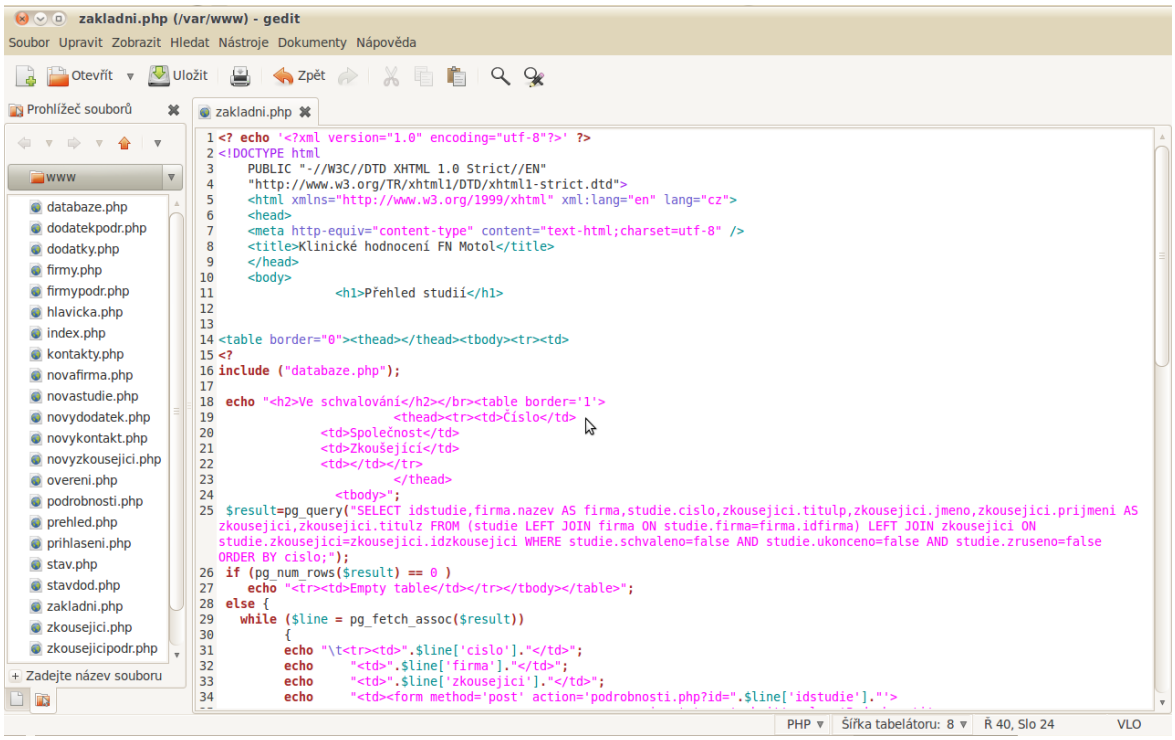
5.2 Postup při implementaci

Při vytváření systému byly použity technologie a nástroje popsané v kapitole 3. V první řadě bylo potřeba vytvořit prázdnou databázi. Vzhledem k relativně propracované analytické části práce se v této fázi nevyskytl žádný problém. Pro vytvoření databáze jako takové jsem využil nástroj Psql což je interaktivní terminál pro přístup k databázovému systému PostgreSQL v prostředí příkazové řádky.

Pro samotné vytvoření tabulek v databázi jsem však již používal grafické rozhraní phpPgAdmin ve verzi 4.2.2 což je webová aplikace napsaná v PHP, která slouží ke správě databázového serveru PostgreSQL. Práce s tímto nástrojem je snadná a intuitivní a případné chyby nebo nepřesnosti se pomocí ní upravují velice jednoduše.

Po vytvoření databáze přišlo na řadu psaní zdrojových kódů webového rozhraní se všemi požadovanými funkcemi. Pro psaní kódu jsem zprvu používal program Bluefish Editor, který je zmíněn v kapitole 3.3.3 nicméně verze kterou

jsem měl k dispozici mi způsobovala menší problémy s orientací v textu neboť mi odmítal zvýrazňovat syntaxi php kódu zatímco HTML zvýrazňoval bez problémů. Z tohoto důvodu jsem záhy přešel na editor Gedit u kterého jsem vytrval až do konce. Ačkoliv je tento editor ve své podstatě minimalistický oproti konkurenci, všechny základní mnou požadované funkce splňoval na výbornou.



Obr. 13 Textový editor Gedit

V průběhu psaní zdrojových kódů webového rozhraní probíhala první fáze testování, kdy jsem odstraňoval relativně velké množství syntaktických a logických chyb v kódu, které často zabraňovali úspěšnému zobrazení stránek v prohlížeči. V druhé fázi bylo na řadě testování hotových částí systému nad testovacími daty vloženými do databáze. Ani tato fáze se neobešla bez problémů. Nejčastěji byla na vině nesprávná syntaxe sql příkazů vkládaných do kódu php. Závěrečná fáze testování je naplánována na květen a červen v ostrém provozu ve Fakultní nemocnici v Motole tak, aby v červenci a srpnu mohlo proběhnout dolaďování systému pro potřeby praktického nasazení.

5.3 Instalace systému

K nainstalování systému je potřeba mít administrátorský přístup k serveru podporujícímu jazyk PHP a databázi PostgreSQL. Do hlavní složky se nahrají všechny soubory z adresáře www, který je na přiloženém CD a který obsahuje všechny zdorojové kódy systému. Pro úspěšné připojení k databázi je potřeba změnit údaje v souboru databaze.php, tak aby odpovídali vytvořené databázi. Tento soubor je ve všech php souborech volán příkazem include, pokud je potřeba připojení k databázi.

Po vytvoření prázdné databáze PostgrSQL a nastavení přístupových opravnění je možné vytvořit samotnou databázi a to velice jednoduše. Stačí spustit skript studie.sql uložený na přiloženém CD v adresáři db. Data ze souboru studie.sql jsou přílohou č. 1 této práce.

5.4 Popis systému

Výsledný informační systém CTIS je webovým portálem určeným pro spravování klinických studií ve zdravotnickém zařízení. Jeho základní součásti jsou:

- Seznam studií a jejich základní údaje.
- Seznam zkoušejících a jejich základní údaje.
- Seznam kontaktů spojených s jednotlivými studiemi a společnostmi.
- Seznam společností, které jsou zadavateli klinických hodnocení.
- Seznam dodatků měnících schvalované studie nebo studie, které v nemocnici aktuálně probíhají.

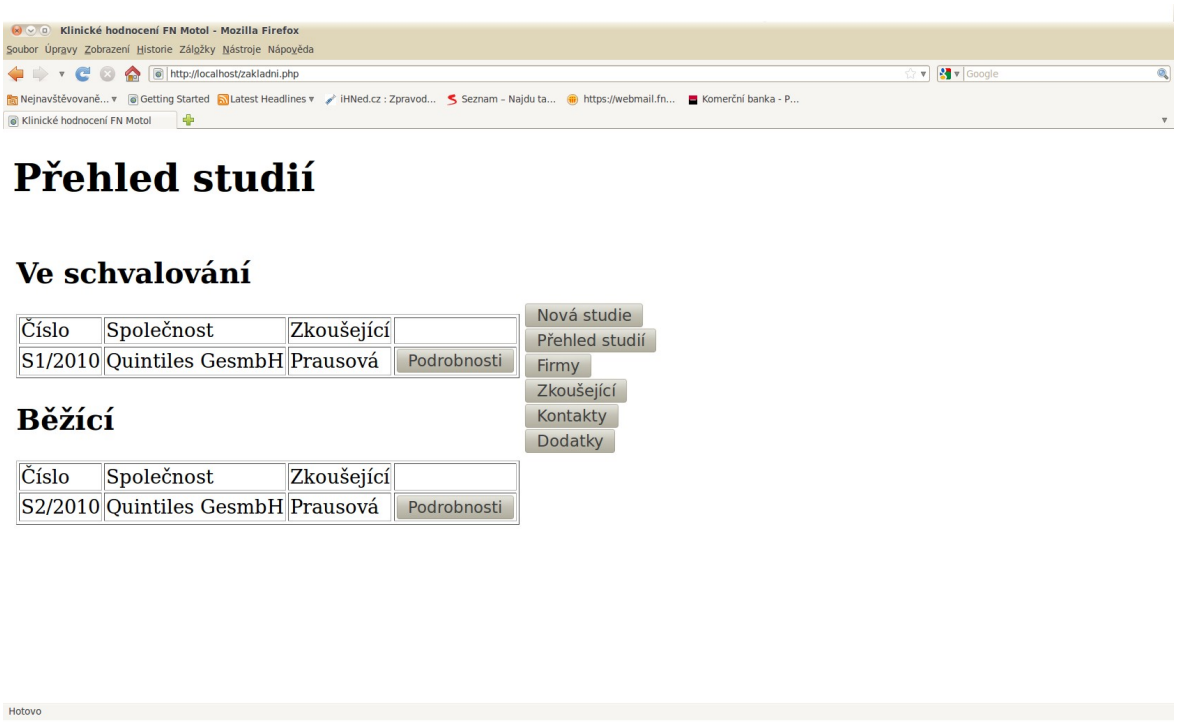
Jako základní funkce požadovaná ze strany nemocnice, bylo přehledný způsob sledování stavu studie v průběhu jejího schvalování. V ideálním případě tak aby bylo možno zpětně vysledovat, které kroky v tomto procesu zabírají nejvíce času.

5.4.1 Popis pracovního prostředí

Při vstupu na webové rozhraní se nejdříve objeví stránka požadující přihlášení do systému. Bez tohoto kroku není možné postoupit na hlavní stránku a zobrazit jakékoliv údaje. Z důvodu bezpečnosti není možné ve webovém rozhraní přidávat, měnit nebo mazat uživatele. Toto může provést pouze administrátor databáze.

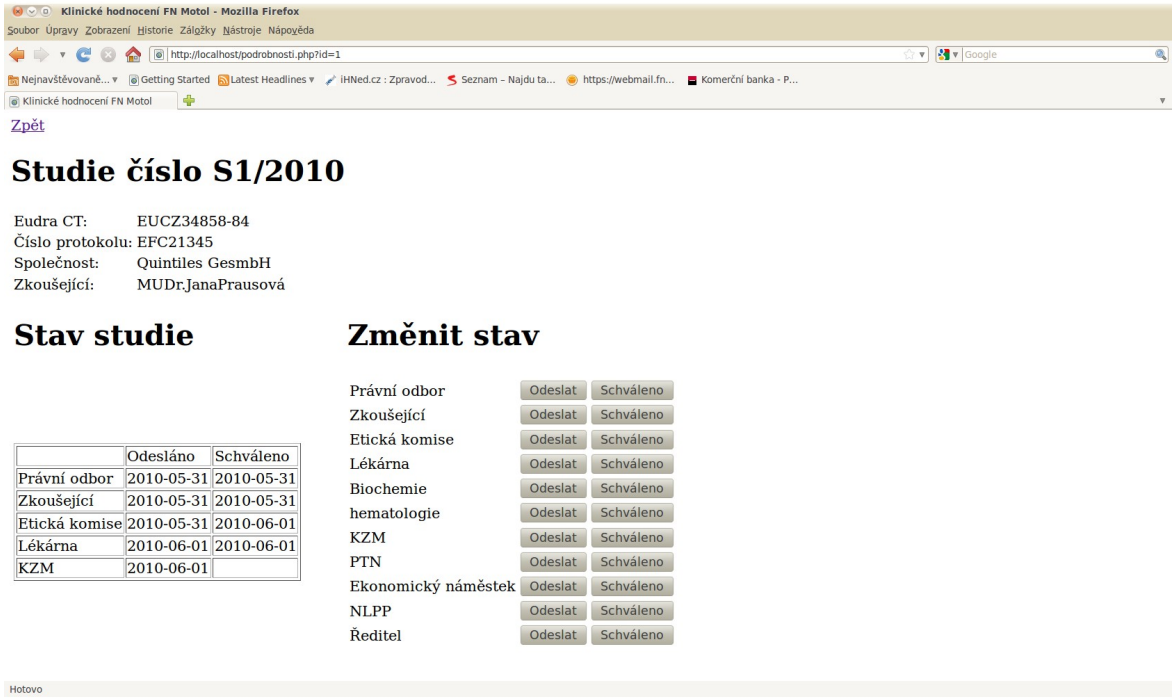
Hlavní strana zobrazuje přehled studií. Tedy dvě tabulky zobrazující seznam studií které jsou v procesu schvalování a tabulku se seznamem studií, které jsou schváleny a aktuálně probíhají. V každém řádku tabulky je uvedeno číslo studie, název společnosti, která je zadavatelem studie, příjmení zkoušejícího a tlačítko podrobnosti. Tlačítko podrobnosti zobrazí kromě dalších informací o studii také její aktuální stav. Všechny studie se mohou nacházet celkem ve čtyřech situacích ve vztahu k nemocnici. V první řadě je to situace „Přijato“. Studie je přijata automaticky při zadání nové studie a je zobrazena na úvodní stránce v tabulce „Ve schvalování“. Pokud studie projde úspěšně schvalovacím procesem a je schválena ředitelem přechází do situace „Schváleno“ a je nadále zobrazována na úvodní stránce v tabulce „Běžící“. Dále ještě může být studie v situaci „Ukončeno“ nebo „Zrušeno“. V prvním případě se jedná o studie které úspěšně proběhly a byly ukončeny podle plánovaného harmonogramu. V druhém případě jde o studie, které byli z nějakého důvodu zrušeny před plánovaným ukončením nebo v průběhu schvalování. Studie v situaci „Ukončeno“ a „Zrušeno“ se na hlavní straně nezobrazují.

Na hlavní straně je také umístěno tlačítkové menu, které umožňuje další navigaci v systému.



Obr. 14 Hlavní strana

Po stisknutí tlačítka podrobnosti na konci řádku u tabulek se studiemi se zobrazí jejich další údaje. V případě, že je studie ve schvalovacím procesu se také zobrazí tabulka s přehledem v jaké fázi schvalovacího procesu se kdy studie nacházela a nachází a soustava tlačítek umožňující tyto fáze měnit. Tento postup byl zvolen z důvodu častých dotazů společností na aktuální stav schvalování studie a odhadu doby zbývající do jejího schválení.

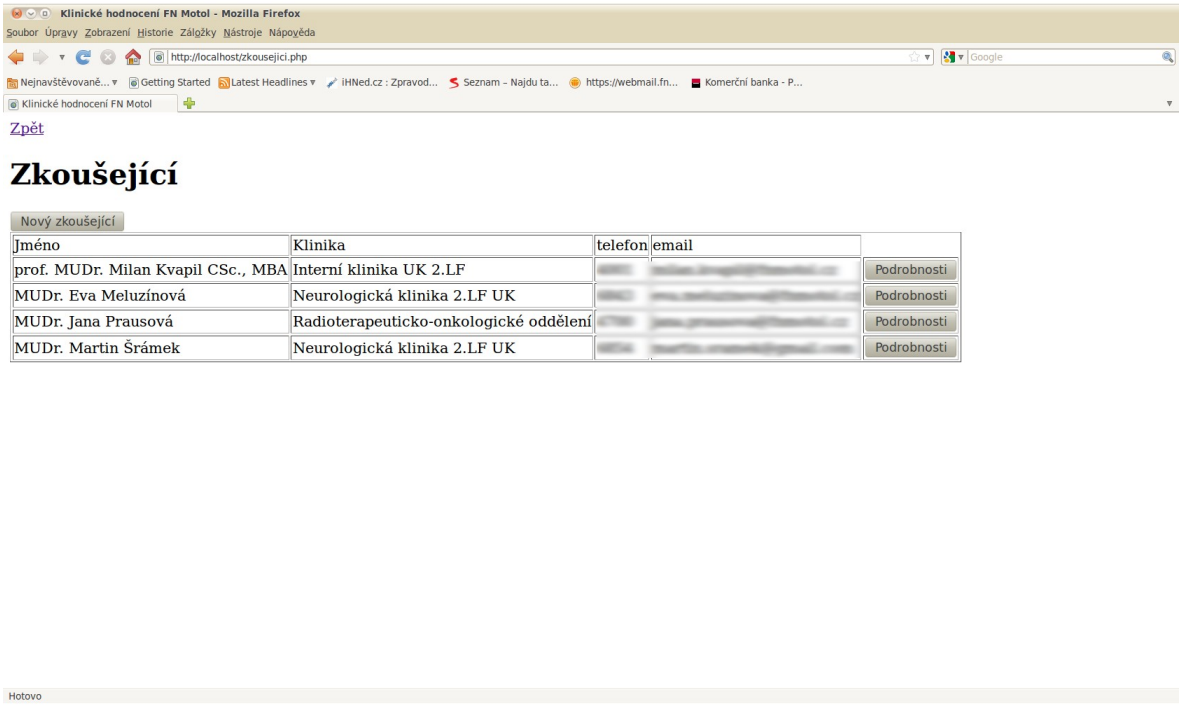


Obr. 15 Podrobnosti studie ve schvalování

Na úrovni databáze je toto sledování zajištěno tabulkou *situace*, ve které každý záznam reprezentuje stavy (mezikroky v procesu schvalování – pracoviště), přiřazené ke studii a jejich vstupní a výstupní časový otisk. Hlavním důvodem tohoto řešení bylo umožnit v budoucnu statistické zpracování časových údajů pro jednotlivé mezikroky schvalovacího procesu a lépe tak odhadovat čas potřebný pro schválení studií.

Na stejném principu funguje i sledování stavu dodatků ke studiím. Pro ně ovšem slouží zvláštní tabulka v databázi *situacedod*.

Na dalších stranách jsou pak umístěny seznamy firem, kontaktů a zkoušejících umožňující vložení nového záznamu. Vložení nové studie je možné kliknutím na tlačítko Nová studie na hlavní straně. Systém zatím neumožňuje tyto záznamy měnit. Tato funkcionalita bude dodána v průběhu závěrečného dolaďování v červenci a srpnu.



Obr. 16 Strana se seznamem zkoušejících

5.4.2 Vkládání dat

Vkládání dat je jednou ze základních funkcí systému a v této kapitole bych rád tuto funkci blíže popsal. CTIS má k dispozici formuláře pro vkládání těchto dat:

- Vložení nové studie
- Vložení nového dodatku ke studii
- Vložení nového zkoušejícího
- Vložení nové firmy
- Vložení nového kontaktu

Při zpracování dat z formulářů je na místě opatrnost. Počítal jsem s tím, že uživatel se dříve či později pokusí do formuláře vložit neplatná data. Navíc, někdy může být formulář terčem útoku a nedostatečné ověření může způsobit bezpečnostní riziko v aplikaci. Proto jsem se rozhodl vstupní data ověřovat.

Protože data z formuláře jsou k dispozici jakožto proměnné ve skriptu, může nejjednodušší ověřování spočívat v kontrole těchto proměnných. Takže jsem vytvořil skript který ověří platnost podle charakteru vstupních dat. Příklad skriptu pro ověření emailu je zde:

```
<?
  function JeEmail ($cislo)
  {
    return ereg("^.+@.+\..+$",$cislo);
  }

  if (empty ($_POST))
  {
?>
<form method="post" action="itself.php">
  E-mail: <input name="email">
    <input type="Submit" name="odesli">
</form>
<?
  }
  else
  {
    if (!JeEmail($_POST["email"])) echo "Zadejte email ve
správném tvaru";
    else echo "Zpracovávám ". $_POST["email"];
  }
?>
```

Toto ověřování však mělo zásadní problém. Uživateli jsem sice nepovolil použít neplatná vstupní data, ale místo všeho jiného mu skript jen poslal upozornění. Vhodnější by však bylo při chybě zobrazit opět původní formulář. Což mi celý skript trošku zamotalo, protože jsem musel testovat jednak to, zda již byl formulář odeslán, a jednak to, zda byl odeslán se správnými daty. Nová verze proto vypadala takto:

```
<?
  function JeEmail ($cislo)
  {
    return ereg("^.+@.+\..+$",$cislo);
  }
  $BudemeZobrazovat=true;
  if (!empty($_POST)) // odeslano bude probihat kontrola
  {
    if (!JeEmail($_POST["email"]))
```

```
    {
      // kontrolou jsme neprošli
      echo "Zadejte email ve správném tvaru";
    }
    else
    {
      // kontrolou jsme prošli
      $BudemeZobrazovat=false;
      echo "Zpracovávám ". $_POST["email"];
    }
  }
if ($BudemeZobrazovat):?>
  <form method="post" action="itself.php">
    E-mail: <input name="email">
      <input type="Submit" name="odesli">
  </form>
<?endif;?>
```

K celému problému jsem přistoupil pesimisticky. Předpokládal jsem, že ve většině případů budeme muset formulář zobrazit. Takže jsem si na to vytvořil logickou proměnnou $BudemeZobrazovat. Tato proměnná je ze začátku nastavena na TRUE, tedy že formulář zobrazovat budeme.

Následuje podmínka testující, zda se již formulář odeslal. Pokud ho skript zatím neodesílal, tělo podmínky se neprovede a formulář se zobrazí. Jestliže se formulář ale již odeslal, dostává se skript dovnitř, do těla podmínky, kde má vnořenou podmínku. Pokud, a pouze pokud projde kontrolou, nastaví proměnnou $ZobrazitFormular na FALSE a může se formulář zpracovat.

Jestliže ale testem správnosti formulář neprojde, zobrazí se upozornění ("Zadejte email ve správném tvaru") a opustí obě podmínky. Proměnná $BudemeZobrazovat zůstala nasatavena na TRUE, takže se po upozornění formulář zobrazí, což je přesně to, co jsem po skriptu požadoval.

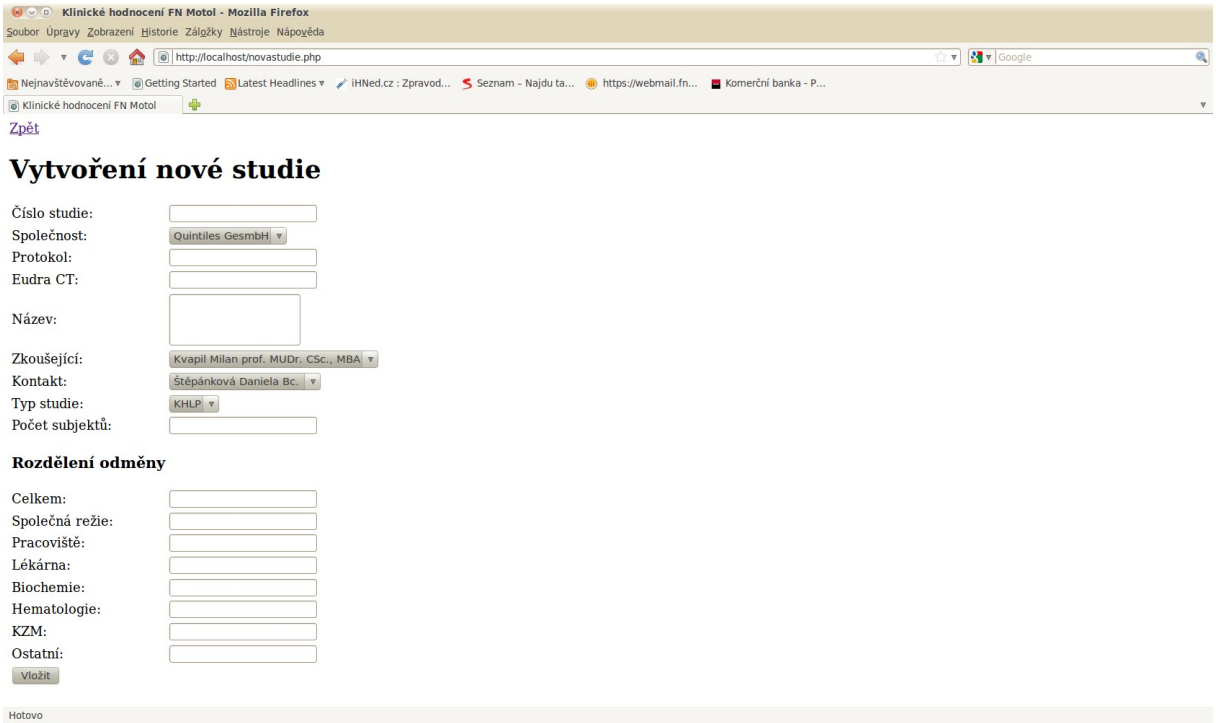
Na celém přístupu je výhodné hlavně to, že samotná definice formuláře je až na konci skriptu a neplete se do jeho zpracování.

Tento skript však ještě nebyl zcela ideální. Když se mají údaje opravovat, chybí tam předvyplněné ty původní. Uvědomil jsem si, že takové řešení by bylo pro uživatele značně nepřátelské protože kdyby formulář obsahoval deset polí a chyba by byla jen v jednom, je jasné, že by musel vyplnit všech deset polí znovu. Jednoduchou úpravou skriptu jsem ale dosáhl toho, že když jej uživatel vyplňuje již poněkolikáté, tak tam pokaždé předchozí hodnoty zůstanou:

```
    if                  ($BudemeZobrazovat):?>
  <form        method="post"        action="itself.php">
    E-mail:       <input       name="email"       value="<?echo
$_POST["email"]?>">
     <input        type="Submit"        name="odesli">
  </form>
<?endif;?>
```

Když se má formulář zpracovávat po několikáté, tak jsou již prvky pole $_POST k dispozici a můžou se tedy použít jejich hodnoty jako výchozí hodnoty prvků formuláře.

Vstupní formuláře jsem také doplnil o rozevírací seznamy pro pole, do kterých bylo potřeba vložit místo pevných dat cizí klíče z jiných tabulek.



Obr. 17 Formulář pro vložení nové studie

6 Závěr

Závěrem musím konstatovat, že hlavní cíle vytčené při zahájení práce na tomto projektu byly splněny. Systém je plně funkční a je v současné době (květen 2010) testován v ostrém provozu na sekretariátu náměstka pro léčebnou a preventivní péči ve Fakultní nemocnici v Motole.

Na druhou stran však je třeba říci, že systém není zcela podle mých představ. Grafická úprava musela ustoupit splnění časového harmonogramu a byla odložena na později. Také nebyla ve webovém rozhraní dokončena část která by zobrazovala studie podle libovolného výběru parametrů, což sice nebylo v původním plánu ale v průběhu realizace se tato funkce ukázala jako užitečná. Vzhledem k tomu že je základ relativně dobře zpracován, lze tyto nedostatky doplnit bez nesnází později.

Rád bych také poukázal na zajímavý případ svojí předvídavosti. Na začátku analytické práce kdy jsem se rozhodoval jaké atributy použít pro studie, jsem prosazoval přidat určité ekonomické atributy týkající se odměn a jejich rozdělování. Ačkoliv jsme v diskuzi se zástupci nemocnice dospěli k názoru že tato data nebudou pro tento systém zapotřebí nakonec jsem prosadil jejich zakomponování. Dodatečně se ukázalo, že nový systém vedení nemocnice natolik zaujal, že vzneslo otázku zda by se systém nedal rozšířit i o sledování základních ekonomických údajů o studiích a o jejich statistické zpracování. Přesto, že systém toto v současné době nesplňuje není problém vytvořit nástavbu pracující s existujícími ekonomickými atributy v databázi.

Diplomová práce byla neobyčejně zajímavá a využil jsem při jejím vypracovávání široké spektrum znalostí a dovedností, které jsem získal při studiu 1. lékařské fakulty Univerzity Karlovy v Praze.

7 Použitá literatura

[1] KRÁL, J., *Informační systémy: specifikace, realizace, provoz*. Veletiny, Science, 1998.

[2] LANGEFORS, B., *Teoretická analýza informačných systémov*. Bratislava, Alfa,1981.

[3] LÁTAL, I. a kol. *Ochrana informací, dat a počítačových systémů*. Praha: Eurounion, 1996.

[4] RONDEL R. K. (ed.), VARLE S. A. (ed.), WEBB C. F. (ed.) 2000, *Clinical Data Management*, John Wiley&Sons, New York

[5] YOURDON, E., *Modern structured analysis*, Prentice Hall PTR, srpen 1988

[6] YOURDON, E., COAD, P., *Object oriented analysis*, Prentice Hall PTR, říjen 1990

[7] YURDON, E., ARGILA, C., *Case studies in object oriented analysis and design,* Prentice Hall PTR, červenec 1996

[8] The Open Source Initiative: Internetové stránky - seznam open source licencí http://www.opensource.org/licenses/postgresql (květen 2010)

[9] RÁČEK, Jaroslav. *Strukturovaná analýza systémů*. Masarykova univerzita, 2006, ISBN 80-210-4190-0

[10] KUČEROVÁ, Helena., *Projektovaní informačních systémů*, Vyšší odborná škola informaačních služeb, 2007

[11] VALADE, Jason., *PHP & MySQL For Dummies, 2nd Edition*, Wiley publishing, Indianopolis, 2004

[12] ŠARMANOVÁ, Jana. *Informační systémy a datové sklady*. 1. vydání. Ostrava : Vysoká škola báňská - Technická univerzita Ostrava, 2007. 169 s. ISBN 978-80248-1500-8.

[13] VASWANI, Vikram. *How to do everything with PHP & MySQL*. USA : The McGraw-Hill/Osborne, 2005. 401 s. Doi: 10.1036/0071466541.

[14] GILMORE, Jason W.; TREAT, Robert H. *Beginning PHP and PostgreSQL 8 : From novice to Professional*. New York, NY, USA : Apress, 2006. 895 s. ISBN 1-59059-547-5.

Přílohy

Příloha 1 – SQL skript pro vytvoření databáze

```
SET statement_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = off;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET escape_string_warning = off;

SET search_path = public, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = false;

CREATE TABLE dodatky (
    iddodatek integer NOT NULL,
    cislo character varying(20) NOT NULL,
    studie integer NOT NULL,
    podrobnosti text,
    prijato boolean DEFAULT true NOT NULL,
    schvaleno boolean DEFAULT false NOT NULL,
    zruseno boolean DEFAULT false NOT NULL
);




CREATE SEQUENCE dodatky_iddodatek_seq
    START WITH 1
    INCREMENT BY 1
```

```
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER SEQUENCE dodatky_iddodatek_seq OWNED BY
dodatky.iddodatek;

CREATE TABLE firma (
    idfirma integer NOT NULL,
    nazev character varying(40),
    ulice character varying(30),
    cislo character varying(20) NOT NULL,
    psc character varying(10) NOT NULL,
    obec character varying(20) NOT NULL,
    tel integer,
    fax integer
);



CREATE SEQUENCE firma_cislo_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE firma_cislo_seq OWNED BY firma.cislo;
```

```
CREATE SEQUENCE firma_fax_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE firma_fax_seq OWNED BY firma.fax;

CREATE SEQUENCE firma_idfirma_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE firma_idfirma_seq OWNED BY firma.idfirma;

CREATE SEQUENCE firma_obec_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE firma_obec_seq OWNED BY firma.obec;
```

```
CREATE SEQUENCE firma_psc_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE firma_psc_seq OWNED BY firma.psc;

CREATE SEQUENCE firma_tel_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE firma_tel_seq OWNED BY firma.tel;

CREATE TABLE kliniky (
    idklinika integer NOT NULL,
    nazev character varying(50) NOT NULL,
    ns character varying(10) NOT NULL
);



CREATE SEQUENCE kliniky_idklinika_seq
```

```
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE kliniky_idklinika_seq OWNED BY
kliniky.idklinika;

CREATE TABLE kontakty (
    idkontakt integer NOT NULL,
    jmeno character varying(20) NOT NULL,
    prijmeni character varying(30) NOT NULL,
    titulp character varying(20),
    titulz character varying(20),
    firma integer NOT NULL,
    tel character varying(20),
    email character varying(40)
);



CREATE SEQUENCE kontakty_idkontakt_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;
```

```
ALTER SEQUENCE kontakty_idkontakt_seq OWNED BY
kontakty.idkontakt;

CREATE TABLE situace (
    idsituace integer NOT NULL,
    studie integer NOT NULL,
    stav integer NOT NULL,
    odeslano date,
    schvaleno date
);




CREATE TABLE situacedod (
    idsituace integer NOT NULL,
    dodatek integer NOT NULL,
    stav integer NOT NULL,
    odeslano date,
    schvaleno date
);




CREATE SEQUENCE situacedod_idsituace_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;
```

```
ALTER SEQUENCE situacedod_idsituace_seq OWNED BY
situacedod.idsituace;

CREATE TABLE stav (
    idstav integer NOT NULL,
    nazev character varying(40) NOT NULL
);




CREATE SEQUENCE stav_idstav_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;




ALTER SEQUENCE stav_idstav_seq OWNED BY stav.idstav;

CREATE SEQUENCE stavy_idstav_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE stavy_idstav_seq OWNED BY situace.idsituace;
```

```
CREATE TABLE studie (
    idstudie integer NOT NULL,
    cislo character varying(30) NOT NULL,
    protokol character varying(30),
    eudract character varying(30) NOT NULL,
    nazev text,
    zkousejici integer NOT NULL,
    kontakt integer,
    typ integer NOT NULL,
    subjekty integer,
    celkem money,
    lekarna money,
    biochemie money,
    hematologie money,
    kzm money,
    ostatni money,
    firma integer NOT NULL,
    prijato boolean DEFAULT true NOT NULL,
    schvaleno boolean DEFAULT false NOT NULL,
    ukonceno boolean DEFAULT false NOT NULL,
    zruseno boolean DEFAULT false NOT NULL,
    fnmotol money,
    pracoviste money
);



CREATE SEQUENCE studie_idstudie_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;
```

```
ALTER SEQUENCE studie_idstudie_seq OWNED BY studie.idstudie;

CREATE TABLE typ_studie (
    idtyp integer NOT NULL,
    nazev character varying(20) NOT NULL
);



CREATE SEQUENCE typ_studie_idtyp_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE typ_studie_idtyp_seq OWNED BY
typ_studie.idtyp;

CREATE TABLE uzivatel (
    iduzivatel integer NOT NULL,
    jmeno character varying NOT NULL,
    heslo character varying NOT NULL
);
```

```
CREATE SEQUENCE uzivatel_heslo_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE uzivatel_heslo_seq OWNED BY uzivatel.heslo;

CREATE SEQUENCE uzivatel_iduzivatele_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE uzivatel_iduzivatele_seq OWNED BY
uzivatel.iduzivatel;

CREATE SEQUENCE uzivatel_jmeno_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;
```

```
ALTER SEQUENCE uzivatel_jmeno_seq OWNED BY uzivatel.jmeno;

CREATE TABLE zkousejici (
    idzkousejici integer NOT NULL,
    jmeno character varying(20) NOT NULL,
    prijmeni character varying(30) NOT NULL,
    titulp character varying(20),
    titulz character varying(20),
    klinika integer NOT NULL,
    tel character varying(20),
    email character varying(40)
);



CREATE SEQUENCE zkousejici_idzkousejici_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;



ALTER SEQUENCE zkousejici_idzkousejici_seq OWNED BY
zkousejici.idzkousejici;

ALTER TABLE dodatky ALTER COLUMN iddodatek SET DEFAULT
nextval('dodatky_iddodatek_seq'::regclass);

ALTER TABLE firma ALTER COLUMN idfirma SET DEFAULT
nextval('firma_idfirma_seq'::regclass);
```

```
ALTER TABLE kliniky ALTER COLUMN idklinika SET DEFAULT
nextval('kliniky_idklinika_seq'::regclass);

ALTER TABLE kontakty ALTER COLUMN idkontakt SET DEFAULT
nextval('kontakty_idkontakt_seq'::regclass);

ALTER TABLE situace ALTER COLUMN idsituace SET DEFAULT
nextval('stavy_idstav_seq'::regclass);

ALTER TABLE situacedod ALTER COLUMN idsituace SET DEFAULT
nextval('situacedod_idsituace_seq'::regclass);

ALTER TABLE stav ALTER COLUMN idstav SET DEFAULT
nextval('stav_idstav_seq'::regclass);

ALTER TABLE studie ALTER COLUMN idstudie SET DEFAULT
nextval('studie_idstudie_seq'::regclass);

ALTER TABLE typ_studie ALTER COLUMN idtyp SET DEFAULT
nextval('typ_studie_idtyp_seq'::regclass);

ALTER TABLE uzivatel ALTER COLUMN iduzivatel SET DEFAULT
nextval('uzivatel_iduzivatele_seq'::regclass);

ALTER TABLE zkousejici ALTER COLUMN idzkousejici SET DEFAULT
nextval('zkousejici_idzkousejici_seq'::regclass);

ALTER TABLE ONLY dodatky
    ADD CONSTRAINT dodatky_pkey PRIMARY KEY (iddodatek);

ALTER TABLE ONLY firma
    ADD CONSTRAINT firma_pkey PRIMARY KEY (idfirma);

ALTER TABLE ONLY kliniky
    ADD CONSTRAINT kliniky_pkey PRIMARY KEY (idklinika);
```

```
ALTER TABLE ONLY kontakty
    ADD CONSTRAINT kontakty_pkey PRIMARY KEY (idkontakt);

ALTER TABLE ONLY situacedod
    ADD CONSTRAINT situacedod_pkey PRIMARY KEY (idsituace);

ALTER TABLE ONLY stav
    ADD CONSTRAINT stav_pkey PRIMARY KEY (idstav);

ALTER TABLE ONLY situace
    ADD CONSTRAINT stavy_pkey PRIMARY KEY (idsituace);

ALTER TABLE ONLY studie
    ADD CONSTRAINT studie_pkey PRIMARY KEY (idstudie);

ALTER TABLE ONLY typ_studie
    ADD CONSTRAINT typ_studie_pkey PRIMARY KEY (idtyp);

ALTER TABLE ONLY uzivatel
    ADD CONSTRAINT uzivatel_pkey PRIMARY KEY (iduzivatel);

ALTER TABLE ONLY zkousejici
           ADD CONSTRAINT zkousejici_pkey PRIMARY KEY
(idzkousejici);

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;
```

Příloha 2. - CD

Přiložené CD obsahuje diplomovou práci ve formátu pdf, zdrojové kódy vytvořeného informačního systému a sql skript pro vytvoření databáze.