VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

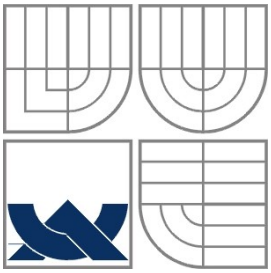
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IS PRO EVIDENCI DĚTÍ PRO ŠKOLY V AFRICE

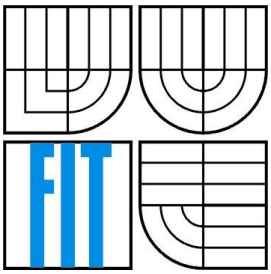
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE DAVID MIKULKA
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IS PRO EVIDENCI DĚTÍ PRO ŠKOLY V AFRICE

IS FOR SCHOOL REGISTRATION IN AFRICA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DAVID MIKULKA
AUTHOR

VEDOUCÍ PRÁCE — ING. TOMÁŠ KAŠPÁREK
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Centrum výpočetní techniky Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Mikulka David**
Obor: Informační technologie
Téma: **IS pro evidenci dětí pro školy v Africe**
Kategorie: Databáze

Pokyny:

1. navrhněte IS s webovým rozhraním pro evidenci dětí pro školy vedené charitativními organizacemi v Africe.
2. Diskutujte potřebné funkce a rysy návrhu
3. Proveďte implementaci systému pomocí PHP a MySQL.

Literatura:

- Dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1. a 2.

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

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

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

Vedoucí: **Kašpárek Tomáš, Ing.**, CVT FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

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

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato bakalářská práce obsahuje popis návrhu a implementace informačního systému pro evidenci dětí pro projekt Adopce na dálku®. Informační systém eviduje informace o dětech, kterým je z výše uvedeného projektu placeno školné na zdejších školách. O každém dítěti jsou vedeny informace osobní a rodinné, zdravotní karta dítěte, informace o jeho sponzorovi, a hlavně informace týkající se školy, kterou navštěvuje, jež jsou zaměřeny především na vedení údajů o placení školného. Přístup k aplikaci je hierarchický podle pravomocí jednotlivých uživatelů autentizovaných heslem. Systém umožňuje vytvářet nové záznamy o dětech a jejich editaci, také dokáže generovat tisknutelné seznamy dětí podle různých kritérií. Aplikace je vytvořena pro efektivní a přehlednou správu výše uvedených informací.

Klíčová slova

Informační systém, dítě, relační databáze, charita, entitní množina, entita, atribut, index, primární klíč, cizí klíč, integrita databáze, konzistence databáze, databázová tabulka, PHP, cookies, session, formulář, MySQL, XHTML, CSS, JavaScript, synchronizace

Citace

David Mikulka: IS pro evidenci dětí pro školy v Africe, bakalářská práce, Brno,
FIT VUT v Brně, 2007

Abstract

This document contains description of a design and implementation of an information system for school registration for project Adopce na dálku®. The information system audits the information about children whom school fees are paid form the project mentioned above. The information personal, about child's family, higienic condition, about sponsor and mainly the information about school, which are ranged especially to audit the information about school fees payment, are stored. Access to this application is hierarchic in compliance with rights of an individual users autentificated by password. The system allows to create new child's records and its editing, it can generate lists for printing in accordance with various criteria, too. The system is created for effective and synoptical management of information presented above.

Key Words

Information system, child, relational database, charity, database entity, attribute, index, primary key, foreign key, integrity, database integrity, database consistence, database table, PHP, cookies, session, form, MySQL, XHTML, CSS, JavaScript, synchronisation

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Tomáše Kašpárka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem při vypracování čerpal.

.....
David Mikulka
květen 2007

Poděkování

Děkuji Ing. Tomáši Kašpárkovi za vedení a pomoc při psaní této bakalářské práce.

Obsah

1 Úvod

Informační systém je modelem konkrétního systému reálného světa. Zpracovává data a informace reálného světa, tedy tvoří tzv. konceptuální model. Slouží k uchovávání a zpracovávání informací a dat. Tento systém získává informace prostřednictvím jeho uživatelů, kteří je do systému vkládají, upravují a provádějí s nimi různé operace. Z tohoto hlediska je informační systém definován jako tzv. otevřený systém, protože je zpětnovazebně propojen se svým uživatelem. Hlavní úlohou informačního systému je co nejvíce zautomatizovat práci se zdroji a tedy zefektivnit a usnadnit činnost člověku.

Informační systém pro evidenci dětí pro projekt Adopce na dálku® slouží k uchovávání a správě informací charity, která působí v Africe, konkrétně v Ugandě. Zde je prostřednictvím výše uvedeného projektu financováno školné místním dětem, které studují zdejší školy. Studium těchto dětí je sponzorováno lidmi z vyspělého světa. Prostřednictvím charity jsou dětem poskytovány prostředky pro studium. Charita tyto prostředky spravuje a zaznamenává informace o tom, jak je s nimi nakládáno a za jakým účelem jsou použity. Konkrétně se jedná o placení školného a o placení zdravotních potřeb a léků. Proto je nutné vést záznamy o dětech samotných, jejich sponzorech a samozřejmě o tom, na jaké účely a v jaké míře byly prostředky použity na školné a léky. Pro účely charity je také nutné s těmito údaji pracovat. Činnost charity je rozprostřena na velkém území a rozdělena na oblasti. Je ale nutné uchovávat informace centralizovaně.

V následujících kapitolách této práce je tedy popsán celý informační systém jak z hlediska jeho návrhu, tak samotné implementace podle konkrétního zadání charity.

V kapitole č. 2 jsou popsány přesné požadavky charity na informační systém, které je potřeba striktně dodržet, aby byl systém plně odpovídající účelům jeho vytvoření a aby co nejvíce ulehčil a zefektivnil práci charitativním pracovníkům.

V další kapitole je podrobně popsán celkový návrh a implementace databáze informačního systému. Jedná se zde o řešení a návrh konceptuálního modelu relační databáze a jeho převedení na reálné databázové tabulky v závislosti na pravidlech použitého vývojového prostředí a na typu databázového serveru. Každý logický celek celého procesu je popsán a uvozen svou podkapitolou.

Kapitola s názvem Webová aplikace obsahuje popis webové aplikace v závislosti na databázi a na požadavcích, které jsou definovány pro činnost systému.

Závěrem je celkově zhodnocen výsledek, čili vytvořený informační systém z hlediska kvality, která je definována požadavky na informační systém a z hlediska jeho uživatelského rozhraní.

Na úplném konci tohoto dokumentu je popsána literatura, ze které bylo čerpáno a také je zde uveden seznam příloh, které jsou součástí dokumentu.

2 Požadavky charity

2.1 Informace o charitě

Organizace, pro kterou je tento informační systém vytvářen, má název Uganda czech development trust (dále jen UCDT) Kampala, což je koordinační centrum projektu Adopce na dálku® Arcidiecézní charity Praha (dále jen ACHP) v Ugandě, která je členem sdružení Charita Česká republika. ACHP je zakladatelem projektu Adopce na dálku®, ve kterém prostřednictvím dárců z České republiky umožňuje vybraným dětem získat vzdělání přímo v jejich přirozeném kulturním prostředí a tím i dalšími projekty zároveň podporuje rozvoj celé komunity.

UCDT v současné době registruje v Ugandě přibližně 3500 dětí. Těmto dětem je placeno školné podle toho, jaký typ školy studuje a podle věku dítěte, zaznamenávají se údaje o zdravotním stavu, platí se za ně lékařské ošetření a léky, dávají se jim učebnice, uniformy a jednou za rok dar. UCDT pracuje na širokém území a její činnost je rozdělena na oblasti. V každé oblasti působí tým pracovníků, který se stará o děti, které v té konkrétní oblasti školy navštěvují. UCDT je koordinována z Prahy, kam se v pravidelných časových úsecích zasílá celá kartotéka, aby byly synchronizovány údaje v Africe a v Praze.

2.2 Databáze

2.2.1 Současný stav

V současné době je databáze složena z klasické papírové kartotéky. Nevýhody tohoto řešení jsou zřejmé. Ať už bereme v úvahu manipulaci s jednotlivými kartami, při které může lehce dojít ke ztrátě některých jejích součástí, dále také např. složitá editace údajů, která je možná jedině přepisem původních a přidáním nových dat. Navíc je to zcela nevhodné řešení pro synchronizaci dat mezi Ugandou a nadřízeným střediskem v Praze, které se uskutečňuje zasíláním celé kartotéky jednou za určitý čas.

Jediný náznak převodu papírové kartotéky do elektronické podoby je psaní záznamů do programu MS Excel. Tento způsob však také nijak neulehčuje práci s databází. Můžeme to označit jako pouhý přepis kartotéky do elektronické podoby.

2.2.2 Karta dítěte

O každém dítěti je uchovávána celá řada informací, týkající se jak dítěte samotného, tak údajů o jeho rodině, o jeho sponzorech, o zdravotním stavu a samozřejmě o škole, kterou navštěvuje. Všechny tyto údaje tvoří dohromady tzv. kartu dítěte. Veškeré tyto údaje je třeba v čase měnit, proto je potřeba aby byly editovatelné. Karta zaznamenává tyto informace:

Osobní údaje:
- jméno a příjmení
- pohlaví
- datum narození
- náboženství
- oblíbený předmět
- oblíbené aktivity
- domácí povinnosti
- životní cíle
- spěšné potřeby
- oblíbené jídlo
- popis dítěte
- fotografie
- s kým dítě žije
- dary

Charita:
- číslo dítěte
- status
- soc. pracovník
- kód oblasti

Sponzor:
- jméno a příjmení
- společnost
- číslo sponzora

Rodina:
- otec a matka
  - jméno a příjmení
  - zaměstnání
  - zdravotní stav
- sourozenci
  - jméno a příjmení
  - věk
  - pohlaví
- dům
- půda
- pěstované rostliny
- chovaný dobytek
- komentář k rodině

Zdravotní karta:
- datum ošetření
- diagnóza
- léčba
- výdaje za ošetření
- výdaje za léky
- poznámky

Škola:
- typ školy
- jméno školy
- třída
- vysvědčení
- zaplacené školné
- knihy

Téměř všechny údaje, samozřejmě kromě jmen a několika dalších popisů, jsou takového charakteru, že se dají standardizovat, tedy vytvořit jakousi množinu hodnot, kterých mohou nabývat.

2.2.3 Operace s údaji

V databázi je často potřebné vyhledávat informace o dětech podle různých kritérií. Tato kritéria jsou: jméno, číslo dítěte, jméno sponzora, číslo sponzora, aj. Takto vyhledané informace jsou organizovány jako seřazené seznamy.

Další hojně využívanou operací je tisk seznamů. Tyto seznamy se filtrují podle těchto kritérií: číslo oblasti, sociální pracovník, škola, typ školy, děti využívající [1]SAF, seznamy dětí bez rodičů (sirotků), seznamy k placení školného s částkami podle škol, seznamy dětí, jejichž školné je již zaplaceno či nezaplaceno v každém termínu, tisk podle data návštěvy lékaře.

Samozřejmě je třeba do systému informace vkládat, aby bylo vůbec možné s nimi provádět výše uvedené operace. S tímto ruku v ruce jsou spojeny i operace úpravy údajů.

2.3 Informační systém

Informační systém je členěný na několik částí. První část slouží k různým operacím nad záznamy o samotných dětech – tisknutí seznamů podle různých kritérií a vyhledávání dětí podle všelijakých klíčů. Tyto operace mohou provádět všichni sociální pracovníci, kteří budou se systémem pracovat.

Další část řeší veškeré operace týkající se placení školného, resp. záznamů o jejich placení. Tuto část systému může ovládat pouze uživatel, kterého můžeme označit jako účetního. Ten může samozřejmě kromě těchto operací používat i první, výše zmíněnou část systému.

Záznamy, které se vytvářejí o návštěvě dítěte u lékaře, má na starosti osoba, kterou můžeme označit jako zdravotníka. Ten bude mít jako jediný v kompetenci práci s tímto druhem záznamů.

Veškeré výše uvedené operace a části systému můžou ovládat pouze osoby, kterým se říká administrátoři. Ti mají neomezený přístup ke všem informacím a operacím. Zároveň právě tyto osoby přidělují všem uživatelům tzv. uživatelská práva, která definují, do které části systému bude mít který uživatel právo zasahovat a nahlížet.

1 SAF = speciální fond, používaný pro financování speciálních výdajů

Cílem, který je hlavním důvodem tvorby celé aplikace je, aby byla uživateli co nejvíce usnadněna práce, tedy aby byly všechny operace co nejvíce automatizovány a aby byl uživateli dopřán co největší luxus při práci. Toho lze dosáhnout klasickými prvky, kdy např. namísto ručního vypisování nejrůznějších informací, které se velmi často opakují, bude použito výběrových políček, atd.

Informační systém bude fungovat přímo v Ugandě, kde bude mít také hardwarové a softwarové prostředky pro provoz. Jelikož bude informační systém zpočátku používán ve zkušebním provozu, aby pracovníci UCDT vyzkoušeli, zda zvládnou s IS pracovat a zda informační systém splňuje jejich představu a požadavky, bude, jak už bylo výše zmíněno, používán pouze v Ugandě. S provozem v pražském koordinačním se v dohledné době nepočítá, proto implementace synchronizace databází dvou paralelně běžících IS není nutné prozatím implementovat.

Celý informační systém musí být vytvořen tak, aby server mohl běžet na slabším PC s operačním systémem WindowsXP.

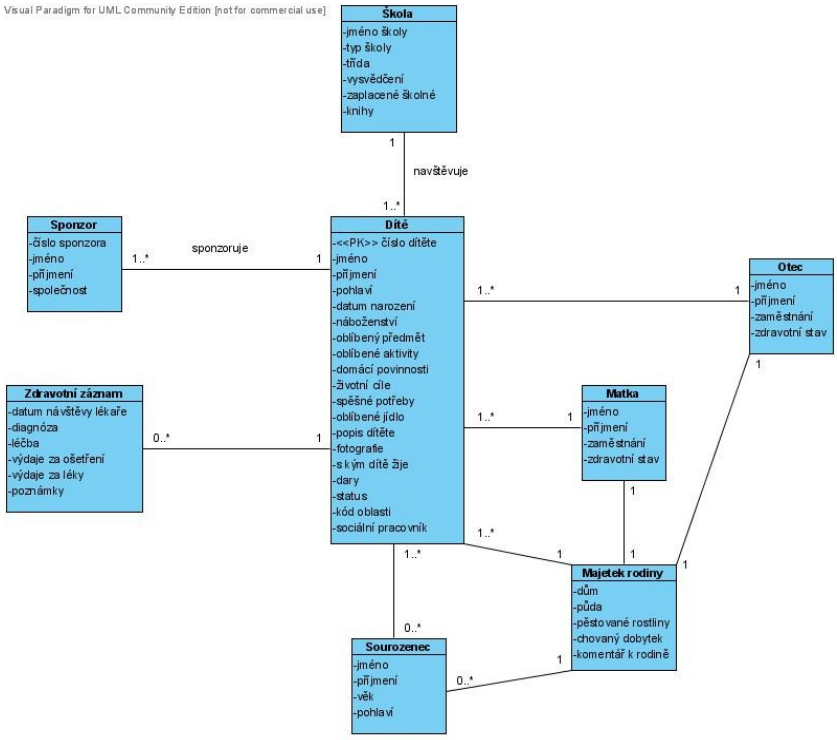
3 Databáze informačního systému

Databáze je registr (papírová agenda, kartotéka) v elektronické podobě, v podobě perzistentních dat. Perzistentní data jsou data trvalá, tedy z hlediska aplikace jsou to data, která jsou uchovávána i tehdy, pokud právě aplikace samotná neběží. V tom je rozdíl od dat, se kterými pracují ostatní nedatabázové aplikace, které získávají data teprve za běhu a po skončení provádění aplikace jsou tato data ztracena.

Data uložená v databázi jsou integrovaná, což můžeme chápat jako data sjednocená s úplným nebo částečným odstraněním redundance, data sdílená, což umožňuje přístup více uživatelů. Co se týká bezpečnosti dat, tak je možné snadněji omezit přístup k datům omezením práv uživatelů, a lze také snadněji zajistit integritu (správnost z hlediska splnění omezení) dat. Jedná se o tzv. integritní omezení, která reflektují reálný svět.

3.1 Návrh databáze

3.1.1 Konceptuální schéma databáze



obrázek 1: Základní konceptuální model databáze – ER diagram

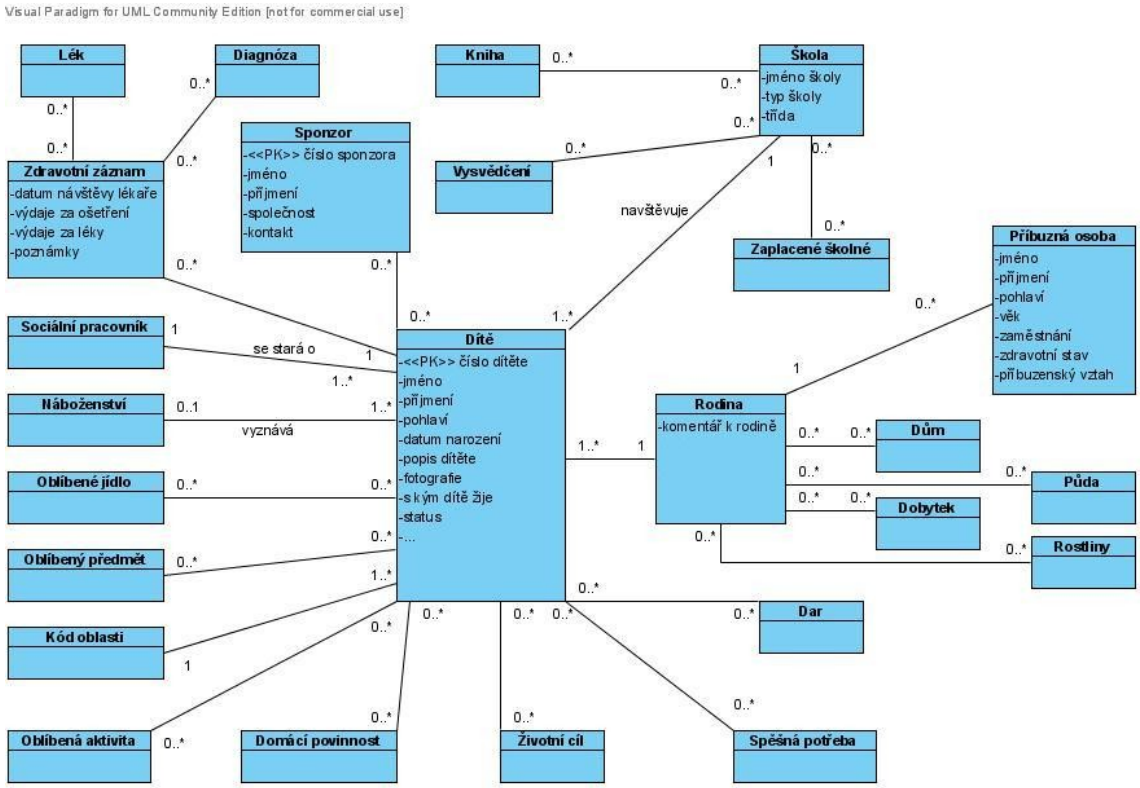
Návrh databáze informačního systému se skládá z několika etap. První etapou je vytvoření tzv. konceptuálního modelu. Tento model je vytvořen na základě znalostí získaných při specifikaci systému, což je analýza systému na základě konkrétních požadavků charity. Z těchto požadavků můžeme navrhnout základní konceptuální model. Jeho objekty mohou být identické s kartou dítěte, která je popsána v kapitole 2.2.1. Model takové databáze tedy vypadá tak, jak popisuje výše uvedený obrázek 1.

Tento návrh je velmi nedokonalý. Obsahuje mnoho opakujících se informací. Naopak také ale tento model nemůže z hlediska návrhu zahrnout všechny požadované informace.

Databáze je správně navržena, pokud splňuje několik základních vlastností. První vlastností je, že databáze je integrovaná, tedy zbavená (částečně nebo úplně, samozřejmě čím více tím lépe) nadbytečných údajů a duplicit. Jak je patrné z obrázku 1, je jasně vidět, že entitní množina(dále jen EM) **Otec** a **Matka** mají definovány stejné atributy. Proto je výhodné, aby tyto dvě EM byly spojeny do jedné, přičemž aby byla zachována informace o tom, zda se jedná o matku či otce, zajistíme přidáním atributu *pohlaví*, pomocí kterého budeme moci zjistit zda se jedná o ženu (matku) nebo muže (otce). Provedeme-li tuto úpravu, musíme si ihned všimnout, že nově vytvořená EM, např. s pracovním názvem **Rodič** má opět téměř stejné atributy jako EM **Sourozenec**. Proto je nasnadě spojit ještě tyhle dvě EM. Tím vznikne EM, kterou pojmenujeme **Příbuzná osoba** a bude mít atributy *jméno*, *příjmení*, *pohlaví*, které jsou pro všechny rodinné příslušníky stejné. Dále atribut *věk*, který je převzatý z původní EM **Sourozenec** a který se může hodit i pro ostatní rodinné příslušníky, a dále *zaměstnání* a *zdravotní stav*, které jsou na tom stejně. Dále abychom byli schopni určit zda jde o matku či otce či sourozence, musíme zavést další atribut *příbuzenský vztah*, který tohle bude definovat. Nyní však musíme také zajistit vztah mezi dítětem a jeho příbuznými. K tomu bude sloužit další nová EM **Rodina**, která ponese právě informace o rodinných vztazích. Každá rodina bude mít tedy jednoznačný identifikátor, na který se budou odkazovat všichni rodinní příslušníci - jak sponzorované dítě, tak jeho rodiče a sourozenci. Zároveň můžeme tuto EM využít pro evidenci rodinného majetku, který je momentálně reprezentován EM **Majetek rodiny**. Takže všechny atributy z EM **Majetek rodiny** převedeme do EM **Rodina**. Těmito úpravami jsme také odstranili další z problémů návrhu databáze, který se nazývá integrita dat. Z tohoto hlediska jsou vždy sporné návrhy, ve kterých se objevují kruhová propojení mezi EM, tak jak tomu bylo v původním návrhu právě ve vztazích mezi EM **Dítě**, **Otec**, **Matka**, **Majetek rodiny**. Pokud se tam totiž takovéto kruhové vazby objevují, dochází zde potom také k nejednoznačnosti mezi vztahy a také k redundanci některých údajů.

Nyní máme vyřešeny zásadní problémy, které obsahoval původní návrh databáze, a tak můžeme přejít k dalším úpravám. Tyhle úpravy se týkají toho, že většina údajů, které o dítěti samotném

potřebujeme ukládat, jsou takového typu, že se dají standardizovat, a proto je lepší vytvořit pro tyto údaje samostatné EM. Potom bude u každého dítěte vždy jenom odkaz na konkrétní entitu z dané EM. Entita je objekt reálného světa, který splňuje všechny atributy dané EM. Z tohoto důvodu vytvoříme nové EM namísto atributů *náboženství*, *oblíbený předmět*, *oblíbené aktivity*, *domácí povinnosti*, *životní cíle*, *spěšné potřeby*, *oblíbené jídlo*, *dar*, *sociální pracovník* a *kód oblasti* z EM **Dítě**, dále to samé provedeme s atributy *vysvědčení*, *zaplacené školné* a *knihy* z EM **Škola**, taktéž atributy *dům*, *půda*, *rostliny* a *dobytek* z EM **Rodina**, a konečně i pro *diagnóza* a *léčba* z EM **Zdravotní záznam**. Všechny tyhle údaje se v záznamech dětí mnohokrát opakují. Např. velký počet dětí může onemocnět určitou nemocí, proto je vždy při vytváření zdravotního záznamu vhodné tuto diagnózu vybrat z připraveného seznamu, což je určitě uživatelsky příjemné a navíc to zajišťuje konzistenci dat.

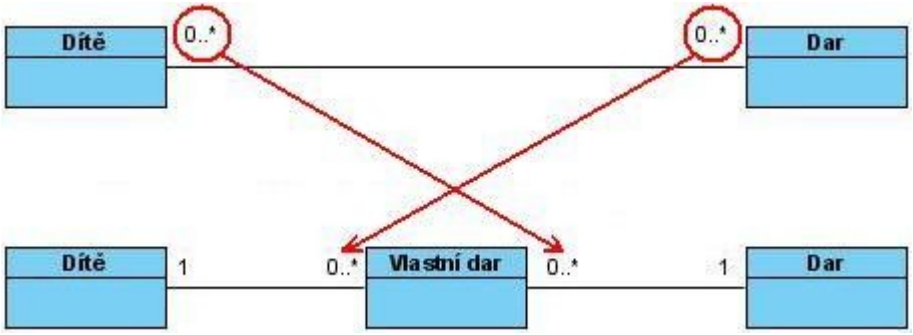


obrázek 2 – Vylepšený konceptuální model – ER diagram

Následuje ošetření vztahů mezi vytvořenými EM. U každého vztahu je třeba určit jak kardinalitu, která určuje maximální počet vztahů daného typu, ve kterých může participovat jedna entita, a také členství, které určuje naopak minimální počet vztahů daného typu. Označení se zapisuje takto: X..Y, kde X značí členství a Y kardinalitu. Dále v textu se budeme zabývat pouze kardinalitou, která je

pro návrh velmi důležitá. Pokud kardinalita je více než 1 a není přesně dáno, jakého čísla nabývá, označuje se buďto hvězdičkou (*) nebo např. velkými písmeny N a M. Tedy tam, kde je předpoklad, že informací téhož typu bude třeba zaznamenat více v rámci jednoho záznamu, jedná se o vztah s kardinalitou N:M, která říká, že až N záznamů v jedné EM může obsahovat až M záznamů ze druhé EM. Oproti tomu kardinalita typu 1:N značí, že právě jeden záznam z jedné EM může obsahovat až N záznamů ze druhé EM. Např. vztah mezi EM **Náboženství** a **Dítě** má kardinalitu 1:N, protože každé dítě vyznává maximálně jedno náboženství, kdežto k jednomu konkrétnímu náboženství se může hlásit více dětí. Naopak vztahem N:M opatříme EM **Diagnóza** a **Zdravotní záznam**, protože jeden zdravotní záznam může obsahovat až M diagnóz a také jeden záznam odpovídající jedné diagnóze může mít až N dětí. Podle těchto pravidel tedy definujeme všechny vztahy mezi EM. Všechny zmíněné úpravy jsou patrné na obrázku 2.

Vylepšený model databáze obsahuje ještě celou řadu nevyřešených skutečností. První a základní z nich je kardinalita vztahů M:N. Tyto vztahy jsou při návrhu logického modelu, který popisuje právě tabulky relační databáze, neřešitelné. Proto musíme tyto vazby odstranit. K tomu se využívá tzv. vazební entitní množina(dále jen VEM), která nahrazuje přímou vazbu M:N. Takže je-li kupř. vztah mezi EM **Dítě** a **Dar** 0..M (na obrázcích je N či M nahrazeno hvězdičkou) na straně dítěte a na straně daru je vztah definován jako 0..N, vytvoříme tedy VEM **Vlastní dar**. Ta bude mít na straně dítěte kardinalitu 1 a na straně VEM bude mít hodnotu, která byla původně na straně EM **Dar**, a to 0..N. Naopak při vytvoření vztahu mezi darem a VEM bude na straně daru kardinalita 1 a na straně VEM bude původní kardinalita ze strany dítěte, čili 0..M. Princip této úpravy je také pro správné pochopení akce znázorněný na obrázku 3.



obrázek 3 – Ukázka odstranění vztahu s kardinalitou N:M

V další části návrhu se budeme zabývat vytvářením atributů všech EM postupně, a také revizí vztahů mezi jednotlivými EM, protože možností návrhu je samozřejmě více a my potřebujeme najít nejoptimálnější pro náš informační systém, aby v konečné podobě byla databáze schopna zaznamenat všechna propojení jednotlivých záznamů a všechny jejich souvislosti nejenom tak, jak je požadováno zadavatelem, ale také tak, aby byla databáze co nejvíce připravena na změny v průběhu procesu údržby softwaru. Pokud by totiž byly potřebné změny ve schématu databáze větší než kosmetické, vedlo by to zcela jistě ke ztrátě integrity a konzistence dat v databázi, což je nepřípustné.

Začněme tedy od nejjednodušších EM. Jsou to EM, které nesou informace, které nejsou příliš významné. Jedná se o EM **Oblíbené jídlo**, **Oblíbený předmět**, **Oblíbená aktivita**, **Domácí povinnost**, **Životní cíl**, **Spěšná potřeba** a **Náboženství**. Jsou to opravdu málo významné a spíše doplňující informace, se kterými se pracuje opravdu velmi málo, navíc nejsou nikterak závislé, ba ani od nich nejsou odvislé žádné jiné údaje v ostatních EM. Atributy těchto EM jsou vždy pouze dva. Prvním je tzv. primární klíč (dále jen PK), který je základním prostředkem adresace entit z dané EM. PK tedy jednoznačně identifikuje každou entitu, proto musí být zajištěno jeho nutné integritní omezení (dále jen IO), že musí být tzv. unikátním klíčem (dále jen UK) a nesmí za žádných okolností nabývat nedefinované hodnoty NULL. UK je takový, že pokud vezmeme všechny entity dané EM, můžeme některou z domén entity považovat za UK tehdy a jen a pouze tehdy, kdy je zaručeno, že tato doména této entity nenabývá za žádných okolností stejné hodnoty jako tatáž doména kterékoliv jiné entity z dané EM. Abychom tedy zajistili toto IO, je vhodné použít pomocný atribut, jehož hodnoty pro každou entitu budou takové, aby bylo zajištěno, že každá entita bude díky tomuto atributu vždy jednoznačně rozlišitelná od všech ostatních entit téhož typu. K tomu se používá číselný atribut, který se inkrementuje s každou novou entitou. Druhým a posledním atributem těchto EM je název té které aktivity, povinnosti, potřeby, cíle, předmětu a jídla. Všechny tyto EM mají navíc vztah M:N k EM **Dítě** (kromě EM **Náboženství**, ta má vztah 1:M), tedy musíme vytvořit VEM ke každé z nich. Když máme ošetřeny jejich vztahy, definujeme hned i atributy těchto VEM. Je to jednoduché. VEM musí obsahovat PK EM, které spojují. Pokud si nejsme jisti, zda tyto dva atributy jsou schopny splnit IO pro PK, můžeme samozřejmě i zde definovat stejný atribut jako v EM samotných, který bude zároveň PK.

Další EM množina, kterou budeme definovat má název **Sponzor**. Požadavky na informace, které potřebujeme o sponzorech vědět jsou definované přímo charitou. Jedná se o *číslo sponzora*, které je generováno charitou. Toto číslo má každý sponzor jedinečné a slouží pro jeho identifikaci v informačním systému. Tento atribut tedy splňuje IO pro PK, protože samozřejmě žádný sponzor nesmí mít nedefinované číslo, tedy nesmí být NULL. Proto můžeme bez dalšího otálení označit *číslo sponzora* jako PK této EM. Dalšími atributy jsou klasické informace, tedy jméno a příjmení sponzora

nebo název společnosti, pokud se jedná o právnickou osobu. To máme tedy atributy *jméno*, *příjmení*, *společnost*. Poslední atribut bude definovat kontakt na tuto osobu a bude mít tedy název *kontakt*. Jako v minulých případech i zde musíme řešit vztah mezi touto EM a EM **Dítě**. Protože i zde platí vztah M:N, musíme zde již známým způsobem definovat VEM.

O každé dítě se stará právě jeden sociální pracovník a zároveň jeden sociální pracovník má na starosti více dětí. Z toho lze odvodit vztah mezi **Dítětem** a **Sociálním pracovníkem**, který bude 1:N. A jaké informace je třeba zaznamenávat? Protože se jedná o osobu, určitě potřebujeme znát jeho *jméno*, *příjmení*, *věk*, *pohlaví*, popř. také *národnost*, protože tito pracovníci nemusí být místní. Aby bylo možné v případě potřeby každého pracovníka kontaktovat, budeme uvádět i *vesnici*, ve které žije a také jeho *telefonní číslo*.

Jak již bylo uvedeno v kapitole 2.1, působení charity je rozděleno do několika oblastí. V každé oblasti je několik kanceláří, ve kterých pracují sociální pracovníci. Můžeme říct, že každý sociální pracovník má své pole působnosti určeno oblastí a kanceláří, ve které pracuje. Podle toho, ve které lokalitě dítě žije, je mu přidělen soc. pracovník. Proto EM **Kód oblasti** obsahuje právě *kód*, který je pro každou oblast pevně daný, a také *kancelář*, kterých může být, jak už bylo uvedeno, v jedné oblasti více. Kanceláře jsou většinou pojmenovány podle vesnic, ve kterých jsou zřízeny. Z několika minulých vět je patrné, že je třeba vytvořit vazbu mezi EM **Kód oblasti** a **Sociální pracovník**, která bude typu 1:N. V tomto případě nemá význam definovat vztah mezi touto EM a EM **Dítě**, protože by vzniklo tzv. kruhové spojení EM **Dítě**, **Sociální pracovník** a **Kód oblasti**, což může vést k porušení integrity, a navíc by to bylo zbytečné. Důvod tohoto tvrzení je jasný. Pokud bychom chtěli zjistit, ve které oblasti určité dítě žije, dohledáme to přes sociálního pracovníka.

Pojďme teď dokončit návrh EM, které dohromady dávají tzv. zdravotní kartu dítěte, která obsahuje veškeré informace o návštěvách dítěte u lékaře. Tedy typicky jaké jsou zjištěny diagnózy, léky a jejich cena, datum návštěvy lékaře, celkové výdaje za konkrétní ošetření, kdo je jejich plátcem, popř. další poznámky. Jak je patrné z vylepšeného ER diagramu na obrázku 2, může se stát, že jedno ošetření může zahrnovat několik diagnóz, stejně jako několik různých druhů aplikovaných medikamentů. Musíme tedy opět vytvořit VEM pro tyto vztahy M:N.

Co se týká EM **Lék**, která obsahuje seznam léků, a každý záznam konkrétně tedy *název léku* a jeho *doporučené dávkování* a také *dávku léku*. Jako PK zde definujeme pomocný atribut. Takto definovaný PK je, jak je možné si všimnout, velmi často využíván. Aby nebyla tato operace pořád dokola opakována, budeme dále uvažovat, že PK je definován právě takto, pokud nebude uvedeno jinak.

Diagnóza je další EM, která je součástí lékařské zprávy. Žádné další informace než *název diagnózy* nejsou požadovány.

Všechny ostatní informace jsou obsaženy v EM **Zdravotní záznam**. Ten obsahuje *datum návštěvy lékaře*, *výdaje za ošetření*, *výdaje za léky* a *poznámky*. Také by mohlo být vhodné znát informaci o tom, kdo platil výše uvedené výdaje za ošetření. Dalo by se předpokládat, že když nějaké dítě někdo sponzoruje, tak právě on platí i tyhle výdaje. Proto také vytvoříme vztah 1:N mezi **Zdravotním záznamem** a **Sponzorem**.

V tomto okamžiku jsme tedy schopni zaznamenat velké množství informací o každé návštěvě dítěte u lékaře. Kdy se návštěva konala, jaká diagnóza(y) byla(y) zjištěny, jaké byly aplikovány léky, kolik bylo za ošetření zaplaceno celkem, kolik z toho stály léky, a také kým bylo vše zaplaceno.

Nyní se budeme podrobněji zabývat zaznamenáváním údajů o školách, které dítě studuje a studovalo. Historie školní docházky není zanedbatelná. Souvisí to s informacemi, kolik finančních prostředků bylo již investováno do vzdělání každého dítěte. Kromě těchto údajů je přinejmenším vhodné také shromažďovat oficiální doklady o studiu, tedy vysvědčení.

Začneme u EM **Kniha**. Tato EM slouží pro zachování informací o tom, jestli dítě vlastní nějaké knihy, resp. učebnice. Určitě je třeba vědět o jakou knihu se jedná, tedy jaký je její *název*, kdo ji napsal - *autor*, *rok vydání*, *cena*, a kód *ISBN*, což je alfanumerický kód určený pro jednoznačnou identifikaci knižních vydání. I když se ve většině případů jedná o učebnice, nikterak to nesouvisí se školou, kterou dítě studuje. Proto vazba mezi touto EM a EM **Škola** bude zrušena a namísto toho se bude vztahovat k EM **Dítě**. Musíme si uvědomit ještě jednu věc. Každé dítě si totiž může pořídit několik knih, tedy vztah mezi EM **Dítě** a **Kniha** je M:N. Proto bude třeba vytvořit i VEM pro ošetření této vazby. Nyní to tedy bude vypadat tak, že EM **Kniha** bude obsahovat seznam knih, které kdy byly pořízeny některým dítětem. Informace, které dítě si kterou knihu půjčilo, se bude nacházet právě ve VEM, kterou si pojmenujeme např. **Vlastní knihu**. Také je vhodné, aby se vědělo, kdo konkrétnímu dítěti financoval pořízení knihy. Toto sice není požadováno charitou, ale může se to hodit někdy v budoucnu. Proto tedy musíme vytvořit novou vazbu mezi **Sponzorem** a EM, která nese informace o tom, které dítě si kterou knihu pořídilo, což odpovídá EM **Vlastní knihu**, nikoliv **Kniha**. Vytvořená vazba je typu 1:N.

Než se pustíme do návrhu dalších EM týkajících se školních záznamů, je třeba jasně definovat požadavky a výchozí situaci. Začneme tím, že musíme mít vytvořený seznam všech škol. Každá škola má svůj *název* a také musí být uvedeno o jaký *typ školy* se jedná. Také vytvoříme vztah mezi EM **Škola** a **Kód oblasti**, protože musí být také dáno, ve které oblasti se která škola nachází, a také proto, že za komunikaci se školami v určité oblasti je zodpovědný určitý sociální pracovník z určité kanceláře v té oblasti. Typ tohoto spojení je 1:N.

V tomto okamžiku máme definováno, jak má vypadat seznam škol, a tak můžeme vyřešit spojení dítěte a škol, které dítě navštěvuje. Protože dítě může navštěvovat více škol a také více tříd v každé

škole, musíme vytvořit VEM, která generuje potřebné spojení dítěte a školy, kterou studuje. Nazvěme si tuto novou EM **Navštěvuje školu**. Tato EM bude obsahovat záznamy o tom, které dítě navštěvuje kterou školu. Každá škola má několik ročníků (tříd). Proto zde definujeme tomu odpovídající atribut *třída*. Jsme tak schopni říct, že to a to dítě navštěvuje tu a tu školu v tom a tom ročníku. Dále potřebujeme k těmto informacím přidat jaká je *výše školného*, která se mimochodem mění v závislosti na oblasti, ve které se škola nachází, a také období, ve kterém dítě tuto školu studuje, tedy jakýsi *školní rok*. Konečně jsme tedy schopni bezezbytku vyhledat a zaznamenat, které dítě v určitém školním roce studovalo jakou školu ve které třídě, a jaké má být tedy zaplaceno školné. Vytvoříme-li spojení typu 1:N k EM **Dítě** a **Navštěvuje školu**, můžeme vytvářet až N záznamů pro každé dítě, čímž zároveň vyřešíme i historii těchto záznamů.

Jedním z důvodů, proč vlastně vytvářet informační systém pro charitu je, aby bylo zjednodušeno a vůbec umožněno zaznamenávat údaje o zaplacení školného studujících dětí. Již máme vytvořen prostor pro to, abychom byli schopni zaznamenat informace o studiu každého dítěte. Teď je k tomu potřeba definovat EM s názvem **Zaplacené školné**, která bude zaznamenávat právě údaje o zaplacení školného. Z informací získaných z charity vyplývá, že školné se platí několikrát do roka, přibližně 2x až 3x podle typu školy. Musíme tedy počítat s tím, že pro jeden ročník studia budeme zaznamenávat třeba až tři údaje o zaplacení školného a sice každý tento údaj by měl znát *datum zaplacení školného*, a určitě *částku*, která byla zaplacena. Připojíme-li tuto EM pomocí vztahu 1:M k EM **Navštěvuje školu**, můžeme přesně zjistit o které dítě a kterou školu se jedná. Navíc vytvořením vazby 1:N k EM **Sponzor** můžeme konkrétně říct, kým bylo školné zaplaceno.

Abychom měli školní záznamy kompletní, vytvoříme EM **Vysvědčení**. Už z názvu plyne i její funkce, a sice zaznamenávat školní výsledky. Jak známe z našich poměrů, vysvědčení se rozdává několikrát během jednoho školního roku. Proto tato EM bude vztažena k EM **Navštěvuje školu** s kardinalitou 1:N, která zajistí více záznamů vysvědčení pro jeden školní rok. Jediným atributem této EM je *vysvědčení*.

Než se dáme do asi nesložitější části návrhu databáze, definujeme dvě menší EM **Dar** a **SAF**. První z jmenovaných zaznamenává údaje o tom, jaký dar dítě dostalo. Není třeba vytvářet mnoho atributů, stačí pouze *název daru* a *cena*. Protože každé dítě dostává dárek přibližně jednou za rok bude to tedy vazba typu M:N, a proto musíme vytvořit i patřičnou VEM. Stejně jako u **Knihy**, **Zaplaceného školného** a **Zdravotního záznamu** je výhodné vědět, kdo výdaje za dar zaplatil. Tak už známým způsobem vytvoříme vazbu 1:N mezi **Sponzorem** a **Darem**.

SAF je speciální fond charity, který slouží k financování speciálních požadavků. Děti, které žádají o pomoc z tohoto fondu musejí být registrované a pokud už byly pro ten konkrétní *účel* prostředky poskytnuty, musíme zaznamenat o jakou *částku* se jedná. Vztah EM **SAF** k EM **Dítě** je 1:N.

Poslední nevyřešenou oblastí tedy zůstává rodina dítěte a majetek. Hned na úvod je třeba ujasnit, co vlastně si máme představit pod pojmem rodina v Africe. Vyjdeme z jakéhosi prototypu evropské rodiny, která je tvořena matkou, otcem a jejich potomky. Taková rodina se v Africe vyskytuje opravdu pouze sporadicky. Více se vyskytující jen situace, kdy je rodina neúplná, tedy chybí jeden z rodičů (např. otec je nezvěstný, mrtvý nebo není vůbec znám), dále pak že chybí oba rodiče, čili děti jsou sirotci. V mnoha případech se také vyskytují situace, kdy dítě žije v komunitě, kde se o něj stará úplně někdo jiný.

Z tohoto pohledu není návrh uvedený na obrázku 2 příliš povedený. Abychom byli s to zaznamenat veškeré vazby mezi dětmi a jejich blízkými osobami, bude lepší, když vytvoříme obecnou EM **Osoba**, která bude mít atributy definující údaje, které jsou společné jak pro dítě, tak i pro jemu blízké osoby, ať už se jedná o otce, matku, sourozence, babičku, dědečka, aj. Jedná se tedy o *jméno*, *příjmení* a *pohlaví*. Nezbytné je, zejména u dítěte, znát datum narození. I zde ale musíme být citliví na tuto problematiku. Zde je situace taková, že ne vždy se ví přesné datum narození ať už dítěte nebo jakékoliv jiné osoby. Velmi často je znám pouze rok narození. Proto musíme definovat zvlášť *rok narození*, *měsíc narození* a *den narození*. Atributy *zaměstnání* a *zdravotní stav* jsou zde hlavně pro příbuzné osoby dítěte, což slouží k řešení tzv. sociální důležitosti dětí. Tato EM tedy tvoří tzv. základní stavební kámen pro identifikaci osob.

Když z EM **Dítě**, kterou již máme vytvořenou, odebereme atributy, které jsou shodné s těmi z EM **Osoba**, zůstanou nám pouze atributy *popis dítěte*, *fotografie*, *s kým dítě žije*, *status* a *číslo dítěte*. Atribut *status* nese informaci o tom, zda je dítě momentálně již někým sponzorováno, zda na svého sponzora teprve čeká nebo naopak už bylo jeho sponzorování ukončeno a dítě bylo z projektu vyloučeno. *Číslo dítěte* slouží pro jednoznačnou identifikaci dítěte v projektu. K těmto atributům doplníme ještě *datum registrace*, což je datum, kdy bylo dítě zaregistrováno do projektu a *datum počátku adopce*, které udává kdy si dítě vybral sponzor. Vztah mezi **Dítětem** a **Osobou** je 0..1:1, což lze pojmenovat tak, že každé dítě je také osoba a má v této EM zaznamenány své údaje, ale naopak ne každá osoba je dítě.

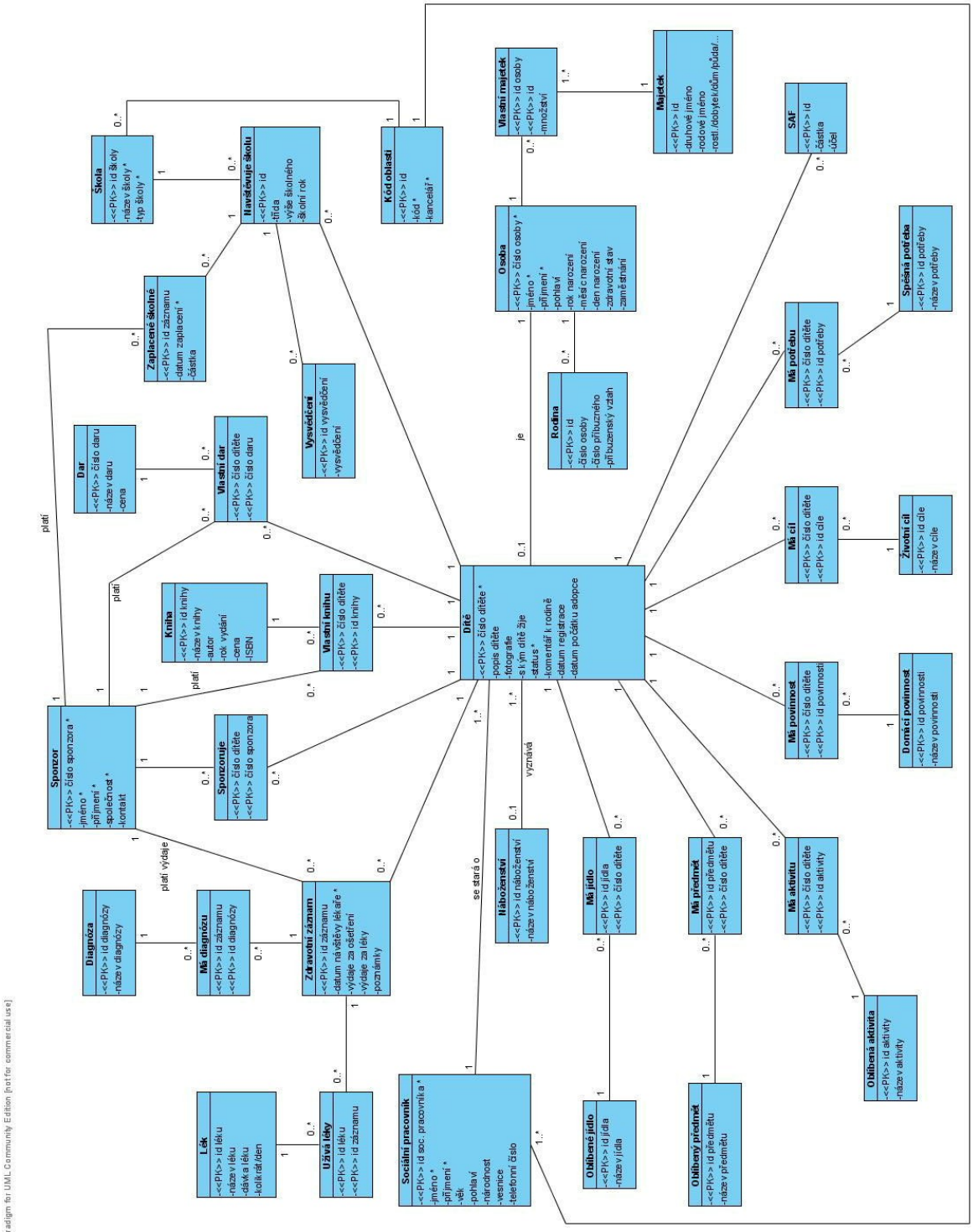
Otázku rodinných příslušníků nám pomůže vyřešit EM **Rodina**, která však bude mít jinou funkci než má ve vylepšeném schématu z obrázku2. Vycházíme z již definované EM **Osoba**. Ta popisuje všechny informace o osobách jako takových, ať už je to dítě samotné, nebo jeho otec, matka, sourozenec či nějaký jiný příbuzný, ale nedokážeme z ní vyčíst, o jaký příbuzenský poměr se mezi jednotlivými osobami jedná. Proto vytvoříme vztah mezi EM **Rodina** a **Osoba**, který bude typu 1:M,

a definujeme atributy pro **Rodinu** takové, že musíme vytvořit atribut *číslo osoby*, který bude ukazovat na určitou osobu z příslušné EM, *číslo příbuzného*, které ukazuje do stejné EM ale na jinou osobu, která je příbuzná osobě, na kterou ukazuje první atribut. Nakonec musíme být také schopni přesně určit, o jaký vztah se tedy jedná pomocí atributu *příbuzenský vztah*. Typy příbuzenských vztahů můžeme definovat jakékoliv. Když si představíme základní tři typy, tedy otec, matka, sourozenec, jsme schopni zahrnout celou řadu dalších vztahů od těchto odvozených, tedy tetu, strýce, snachu, švagra, dědečka, babičku, atd., protože vždycky se jedná o něčího otce, matku nebo sourozence. To je právě v tomto systému velmi výhodné, protože děti jsou často vychovávány vzdálenějším příbuzným nebo jinou osobou než rodiči.

Teď už vyřešíme pouze majetkové záležitosti. Z požadavků charity vyplývá, že majetek se počítá na osobu, z čehož nám ihned je jasné, že veškeré údaje o majetku se budou vztahovat k EM **Osoba**. Z modelu databáze z obrázku 2 vyčteme, že se mají ukládat informace o tom, jestli osoba vlastní dům, půdu, rostliny nebo dobytek. Pokud to vezmeme z obecného hlediska, tak každá rostlina i zvíře má rodové a druhové jméno, čímž je jednoznačně určena. Z tohoto důvodu tedy můžeme sloučit EM **Rostliny** a **Dobytek** do jedné EM, kterou pojmenujeme obecně **Majetek**. Tedy k atributům *rodové jméno* a *druhové jméno* přidáme ještě další atribut, který bude značit, zda se jedná o rostlinu či dobytek. Protože každý může pěstovat více druhů rostlin a taky chovat více druhů dobytka, jedná se tedy vzhledem k EM **Osoba** o vztah M:N. Proto musíme vytvořit VEM **Vlastní majetek**, která ponese záznamy o tom, jaké druhy rostlin či dobytka vlastní která osoba. K těmto záznamům je tedy vhodné doplnit informaci o *množství*.

Takto obecně navržená EM, jako je právě **Majetek** vyvolává otázku, zda by nebyla schopna nést údaje i o domu či o půdě. Odpověď je velmi jednoduchá. Všeobecně lze říct, že atributy *druhové* a *rodové jméno* jsou vlastně dva od sebe odlišné názvy, z nichž jeden řadí zaznamenávanou věc z obecného hlediska a ten druhý název toto obecné zařazení blíže upřesňuje. Mohlo by to tedy ve výsledku fungovat třeba právě pro půdu, kde jeden název bude uvádět, že se jedná o půdu, konkrétněji např. o pastvinu. Navíc atribut *množství* v EM **Vlastní majetek** je schopen nést informace o tom, jakou výměru půda má, tedy s tím, že je předem uvedena míra, ve které se údaje zadávají. Nyní jsme tedy dokázali, že s EM **Majetek** a **Vlastní majetek** si vystačíme pro evidenci veškerého majetku každé osoby.

Tímto lze považovat návrh databáze za hotový a zcela odpovídající jak požadavkům charity, tak i z hlediska dodržení pravidel správného návrhu databáze jako takového. Jako úplně poslední věc návrhu je určení tzv. indexů, což jsou atributy, o kterých víme, že budou velmi často používány jako vyhledávací klíče v databázových dotazech. Označíme-li je jako index, bude potom databáze připravena na to že se s nimi bude pracovat častěji a provede tak patřičné optimalizace, aby byly dotazy prováděny rychleji.



obrázek 4 – Úplné konceptuální schéma databáze

3.1.2 Schéma systémové databáze

Při provozu IS je generováno mnoho údajů, které je nutné pro další funkci a usnadnění ukládat. Je potřeba hned na začátku říci, že tato data nemají nic společného s informacemi o dětech. Můžeme je nazvat např. provozními informacemi. Tyto údaje mají, stejně jako údaje o dětech, předem daný formát, a protože je jich během používání IS nashromážděna celá řada, je výhodné pro jejich shromažďování využít výhod databáze. Ještě než se pustíme do návrhu systémové databáze, je třeba předeslat, že tato databáze není žádnými vazbami spojena s databází z kapitoly 3.1.1.

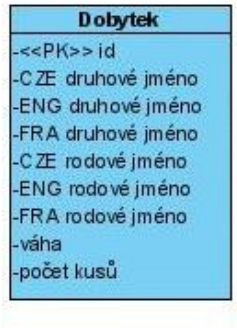
Práva pro přístup do IS musí splňovat kritéria která jsou nutná pro každý systém, který je chráněný heslem. Uživatelů IS bude velké množství, a heslo stejně tak jako přihlašovací jméno uživatele musí být pro každého jedinečné. Proto je nutné přesně určit pravidla, podle kterých se budou hesla a přihlašovací jména generovat. Pro ukládání, editaci a vlastně veškerou práci s těmito údaji bude vytvořena speciální systémová tabulka, kterou definuje EM **Práva** (viz. obrázek 7), která všechny tyto záznamy bude shromažďovat.

Jak už vyplývá z požadavků zadavatele, není sice nutné připravit IS na synchronizace mezi Ugandou a Prahou, nicméně můžeme, alespoň v teoretické rovině, nastínit, jak se bude databáze upravovat při implementaci synchronizace. Abychom byli schopni synchronizaci, jejíž popis najdete v kapitole 5 této práce, implementovat, je nutné přidat ke každé EM atribut *mtime*, který nese informaci o tom, kdy byl každý záznam naposledy editován. Další atribut, který se bude hodit má název *stime*, který zase udává datum a čas, kdy byl konkrétní záznam naposled synchronizován. Další atribut který je třeba přidat bude mít název *ctime* a bude udávat datum a čas, kdy byl záznam v databázi vytvořen. To nám pomůže pro snazší určení záznamů, které se mají synchronizovat a které ne, stejně tak jako pro kontrolu dalších vlastností záznamu jako takového. Tedy kromě těchto dvou atributů, které přidáme do každé EM v návrhu databáze na obrázku 4, ještě vytvoříme EM **Synchronizace**. Tato EM bude sloužit pouze k zaznamenávání informací o tom, kdy se konala synchronizace, který uživatel ji prováděl, jestli byla dokončena, jelikož se to nemusí vždy podařit, a kolik záznamů bylo celkem synchronizováno, protože se určitě ne vždy bude provádět upload celé databáze. Konečnou podobu této EM lze vidět na obrázku 7.

3.1.3 Úprava databáze pro vícejazyčný systém

Výchozím jazykem IS je anglický jazyk, protože je univerzální a celosvětově uznávaný jazyk, kterým se domluví převážná většina lidí, a protože je používán pro komunikaci mezi pracovníky UCDT a domorodými obyvateli v Ugandě. Není však zaručeno, že budou s IS pracovat pouze osoby, které ovládají angličtinu na vysoké úrovni. Je tedy nasnadě, aby aplikace podporovala více jazyků. Jedním z nich je jazyk český, jelikož mnoho sociálních pracovníků je z České republiky. Dále se v budoucnu počítá z rozšířením IS i do jiných států, kde se tato charitativní činnost rozjíždí a kde již také funguje. Všechny tyto důvody vedou k potřebě, aby byla aplikace schopna komunikovat ve více jazycích, čili připravit aplikaci pro co nejsnadnější definování nového jazyka aplikace.

Kromě aplikace samotné musí být samozřejmě na změny jazyka schopna reagovat i databáze. Jistě ne všechna data jsou závislá na zvoleném jazyku. Mezi ně patří jména dětí a všech osob, jejichž záznamy jsou zaznamenávány, stejně tak veškeré číselné hodnoty datem narození dítěte počínaje a pomocnými identifikátory konče. Naopak údaje podléhající jazyku, který je právě aplikací používán, jsou např. údaje o pohlaví osob, jména zvířat a názvy rostlin, názvy darů, oblíbených předmětů, jídel, apod.

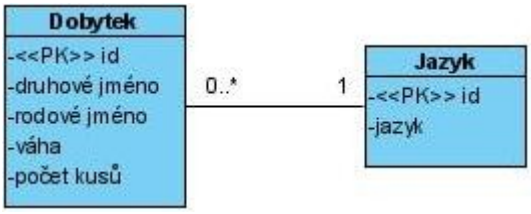


obrázek 5 – EM s více-jazyčnou podporou

Proto musíme databázi těmto skutečnostem přizpůsobit. Jedním z řešení je vytvoření (pro všechny textové atributy, kterých se to týká) jejich duplicity s tím, že v jejich názvu bude uveden jazyk, ve kterém jsou zapsány hodnoty těchto atributů. Potom už jen stačí v závislosti na užívaném jazyku vybrat ten atribut, který mu odpovídá. Toto řešení ale není příliš vhodné. Důvodů je hned několik. Chceme-li pro aplikaci definovat nový jazyk, musíme měnit datovou strukturu všech EM v databázi. Můžeme tedy toto řešení označit za neflexibilní a za nečisté z hlediska datového návrhu. Navíc při větším počtu jazyků vzniknou obrovské EM. Názorná ukázka je na obrázku 5 (jedná se pouze o ukázku, obsah ani schéma nijak nesouvisí s žádnou z databázových tabulek z návrhu IS).

Jiné řešení je výhodnější. Hlavním rozdílem oproti minulému řešení je vytvoření nové EM **Jazyk**, která uchovává informace o jazycích, které aplikace podporuje. Dále pro demonstraci využijeme a upravíme EM zobrazenou na obrázku 5. Odstraníme duplicitní atributy a pro identifikaci jazyka vytvoříme vazbu mezi EM **Jazyk** a **Dobytek** s kardinalitou 1:N. Můžeme se na to dívat tak, že nyní je každá entita identifikovatelná atributem *id* a také jazykem, ke kterému se vztahuje. Převedeme-li si EM **Dobytek** na tabulku, znamená to, že každá entita, nesoucí název určitého druhu dobytka, představuje právě jeden řádek této tabulky. Pokud tedy bude jazyků více, bude více stejných řádků se

stejným druhem dobytka, přičemž tyto řádky se liší pouze identifikátorem do tabulky jazyků a samotným názvem. Ovšem i toto řešení má své nevýhody. Základní z nich je, že pokud EM obsahuje i jiné než textové atributy, tak (v našem příkladu jsou to *váha* a *počet kusů*), dochází k duplikaci těchto dat do všech řádků tabulky, což samozřejmě souvisí s velkým nárůstem velikosti databáze. Navíc změny těchto atributů by se vždy musely projevit ve všech záznamech které obsahují stejný druh dobytka patřícího jednomu člověku, čímž hrozí vznik nekonzistence, který by bylo třeba hlídat na úrovni aplikační logiky, což není moc bezpečné. Nástin tohoto řešení nalezneme na obrázku 6.



obrázek 6 - Řešení vícejazyčnosti vhodné pro náš IS

Pro náš IS je tohle řešení zcela vyhovující. Do všech jazyků se budou překládat pouze údaje, které jsou v systému používány jako informace, které si uživatel vybírá ze seznamu nabízených možností. Ostatní informace, které uchovávají důležitá data o dětech, osobách, zaplaceném školném, zdravotních záznamech, atd. zůstanou nepřekládána.

Výše uvedené problémy, související s tímto řešením vícejazyčnosti, můžeme zanedbat, neboť všechny EM, kterých se tato operace týká, obsahují pouze atributy, které budou různé právě v závislosti na jazyce. Proto nevzniká žádná duplikace dat.

Abychom tedy připravili databázi pro více jazyků, stačí vytvořit novou EM **Jazyk**, která bude mít dva atributy. První má název *zkratka* a jedná se o mezinárodně uznávanou zkratku, která je tvořena třemi písmeny velké abecedy (CZE, ENG, atd.) Druhým atributem je celý *název* jazyka (Čeština, English). Jak tato EM vypadá lze vidět na obrázku 7.

obrázek 7 – Systémové tabulky IS

3.2 Parametry databáze

Databáze IS je implementována v jazyce MySQL. Pro definici sloupců všech tabulek jsou využívány datové typy takové, které dokáží v plné míře obsáhnout každou hodnotu, které může daná doména nabýt. Společně s tím však dbáme na to, aby celá databáze zabírala co nejméně místa v paměti počítače.

Jenom pro orientaci uvedeme několik údajů právě o paměťové náročnosti databáze. Musíme však předem upozornit, že tato čísla je potřeba brát s rezervou a spíše pouze jako orientační.

Databázové tabulky mají velikost v průměru přibližně 210B, plus jedna tabulka je rozsáhlejší, která má velikost asi 64,5KB, kterou ovlivnil atribut, který může obsahovat dlouhý text o několika stovkách znaků.

Pokud tedy vezmeme v úvahu, že každá tabulka obsahuje právě jeden záznam, jehož hodnoty plně využívají paměťových možností každého datového typu každé domény, bude databáze na paměťovém médiu zabírat paměť o přibližné velikosti 70,7KB. Pokud bude každá tabulka obsahovat záznamy se stejnými paměťovými nároky, ale o počtu záznamů 3500, bude celková velikost databáze asi 245MB.

Jak už ale bylo jednou řečeno, tyto údaje je třeba brát s rezervou. Musíme počítat s tím, že některé sloupce záznamů např. nebudou definovány, atp. Na druhou stranu je ale třeba si uvědomit, že ačkoli nyní počítáme s číslem 3500, které téměř odpovídá počtu dětí registrovaných v projektu Adopce na dálku v Ugandě, ne každá tabulka obsahuje právě jeden záznam pro každé dítě. Mnoho z nich má těch záznamů mnohem více. V průměru tedy můžeme říct, že každá tabulka má asi 5 záznamů, které patří jednou dítěti.

Vezmeme-li v potaz všechny tyto argumenty, můžeme (ale zase jenom přibližně) definovat procentuální vytíženost záznamu. Za stoprocentní vytíženost tedy označíme případ, kdy záznam nabývá takových hodnot, že plně využije paměťových možností, které mu byly dány jeho datovým typem. V průměru však tato vytíženost nepřesáhne 70% paměťových nároků.

Pokud bereme v úvahu v minulých dvou odstavcích definované průměrné hodnoty, můžeme vypočítat orientační velikost databáze pro x dětí. Víme-li, že mají-li všechny tabulky o jednom záznamu velikost 70,7KB, a pokud záznamy zabírají pouze 70% kapacity, dostaneme asi 50KB využitého prostoru. Pokud dále uvažujeme 3500 dětí, o nichž se vedou záznamy a že v každé tabulce je 5 záznamů ke každému dítěti, tak se v konečném součtu dostaneme na číslo 845MB paměťového prostoru, který databáze zabírá.

4 Webová aplikace

4.1 Použité prostředky a jazyky

4.1.1 PHP

Hypertext Preprocessor (PHP) je skriptovací programovací jazyk, který se používá především k tvorbě dynamických webových stránek. Jeho výhodou je jednoduché začlenění do struktury HTML kódu.

Skripty jazyka PHP jsou prováděny na straně serveru, na kterém webová aplikace běží. Klientovi, tedy internetovému prohlížeči je již zasílán výsledek skriptů. Velkou výhodou a také důvodem, proč je tento jazyk oblíbený a velmi rozšířený, je jeho syntaxe, která je sloučena z několika programovacích jazyků, jako jsou Perl, jazyk C, Java. PHP je jazyk beztypový, což znamená, že při deklaraci proměnných se nezadává její typ. Ten je odvozen automaticky při její definici. S tímto ruku v ruce souvisí také možnost kdykoliv změnit typ proměnné,což se děje také automaticky, když přiřadíme proměnné novou hodnotu jiného typu. V PHP ve verzi 5 je již také více rozvinuta objektová část tohoto jazyka, která je velmi podobná jazyku Java a umožňuje programátorům použít pro práci objektově orientovaná programovací paradigmata, která zlepšují úroveň abstrakce vyvíjených aplikací.

PHP je platformě nezávislý, což vedlo k jeho rychlému rozšíření a také k jeho vysoké oblibě použití. Sám o sobě obsahuje širokou škálu knihoven, které jeho využití dále rozšiřují. V kombinaci s databázovými aplikacemi (hlavně MySQL) poskytuje PHP vynikající prostředky pro tvorbu dynamických webových aplikací s podporou databáze.

Tento informační systém byl programován v jazyce PHP 5.2.3 a je provozován na serveru Apache-1.3.37, na Unixové platformě.

4.1.2 MySQL

MySQL je multiplatformní databázový systém. Z komunikačního hlediska je MySQL dialektem jazyka SQL, což jej spolu s možností provozu na všech platformách a také s volně šiřitelnou licencí řadí mezi nejoblíbenější a nejpoužívanější databázové systémy.

MySQL v kombinaci s jazykem PHP a serverem Apache je nyní, troufám si tvrdit, nejpoužívanější softwarový balíček pro provoz webových aplikací s podporou databáze.

Náš IS využívá databázového systému MySQL 4.0.27, a pro vývoj bylo také využito služeb aplikace phpMyAdmin 2.5.5-pl1.

4.1.3 JavaScript

JavaScript je programovací jazyk používaný při tvorbě internetových stránek. Jeho výhodou je, stejně jako u PHP, jednoduché začlenění do HTML kódu.

JavaScript je multiplatformní programovací jazyk. Oproti jazyku PHP se jedná o jazyk klientský, čili jeho vyhodnocení je prováděno na straně klienta, tedy na straně webového prohlížeče. To je výhodné zejména proto, že některé operace je možné provádět přímo na straně klienta bez komunikace se serverem, kterému tímto ubíráme zátěž.

Skripty psané v tomto jazyce jsou interpretovány, tedy není nutno provádět kompilaci zdrojových kódů.

JavaScript je jazykem objektovým. Využívá objektů internetového prohlížeče a objektů zabudovaných v jazyce. Díky nim je možné přistupovat ke všem prvkům dokumentu a provádět s nimi různé operace.

Syntaxe jazyka je velmi podobná jazyku Java, což je jeden z důvodů, proč se tento skriptovací jazyk jmenuje JavaScript.

4.2 Aplikace

4.2.1 Webová aplikace – celkový pohled

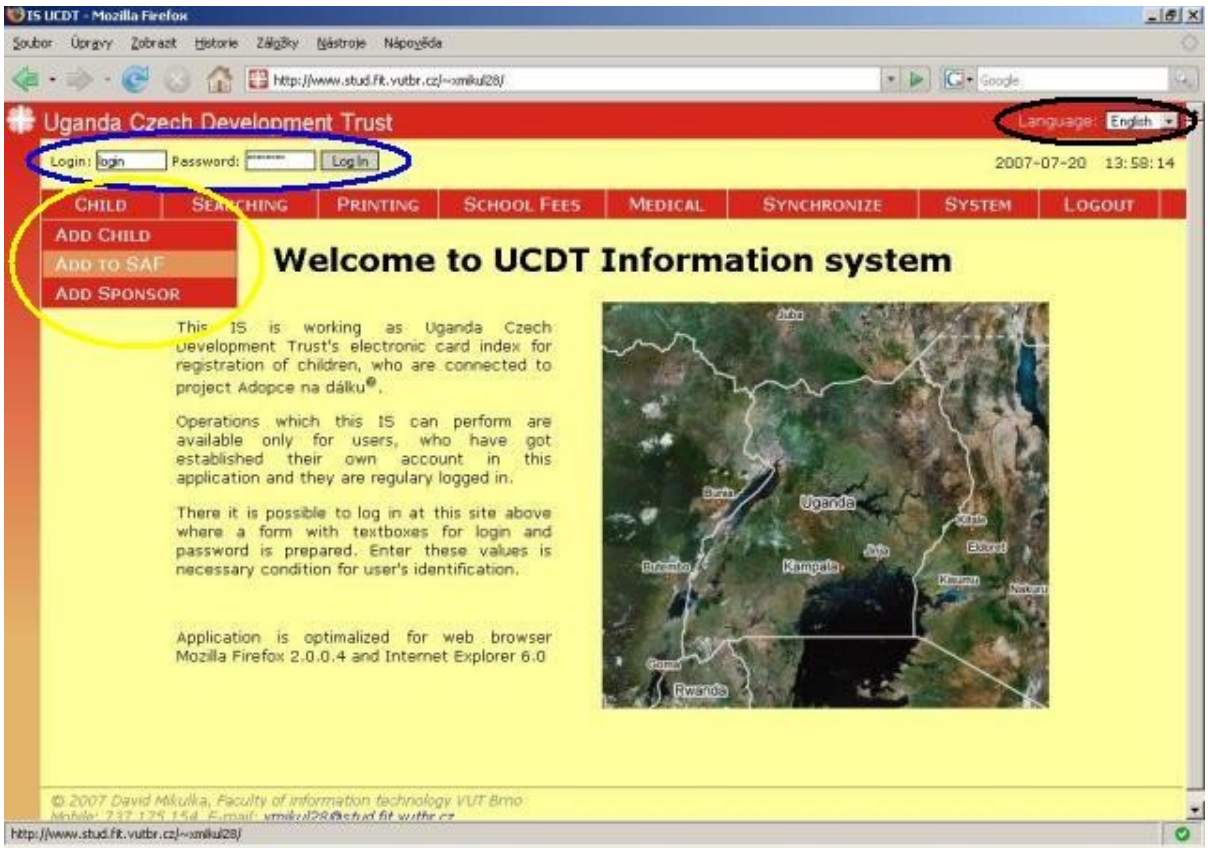
Informační systém je realizován jako klasická webová aplikace, která je umístěna na webovém serveru a uživatel se k ní připojuje prostřednictvím internetového prohlížeče. Tento IS je optimalizován pro prohlížeče Mozilla Firefox 2.0.0.4 a Internet Explorer 6.0, čili pro bezchybnou funkčnost je třeba použít právě těchto prohlížečů, ve verzích výše uvedených nebo novějších.

Celkové grafické uspořádání prvků IS je navrženo tak, aby bylo přehledné a jednoduché, aby se v něm uživatel lehce orientoval, aby neobsahovalo žádné nadbytečné prvky, které by uživatele rozptylovaly a odváděly jeho pozornost. Počet ovládacích prvků je minimalizován, aby uživatel, pro přístup k jakékoliv operaci, byl nucen učinit minimum akcí (pohyb myši, stisk tlačítka myši, atd.).

Jako první úkon, který je při vstupu do IS uživatel nucen provést, je přihlášení. Bez řádného přihlášení není možné provádět žádnou z operací, které IS nabízí. Formulář pro zadávání přihlašovacího jména a hesla je vyznačen modrým oválem na obrázku č. 8. Zde uživatel zadává svůj login a heslo. Oba tyto údaje jsou jedinečné. Pokud uživatel zadá nesprávné heslo či přihl. jméno, je zobrazen dialog, kde je tato skutečnost uživateli oznámena a přihlášení se neprovede. V opačném případě je přihlašovací formulář nahrazen základními údaji o přihlášeném uživateli v pořadí: přihl. jméno, jméno a příjmení, oprávnění, vesnice, kde vesnice je název vesnice, ve které sociální pracovník žije.

IS zná čtyři typy oprávnění:
- Administrátor – může provádět veškeré operace, které IS nabízí.
- Účetní – vyhledávání a tisk seznamů, placení školného
- Lékař – vyhledávání a tisk seznamů, lékařské záznamy
- Soc. pracovník – pouze vyhledávání a tisk seznamů

Aplikace má vícejazyčnou podporu. Uživatel si tedy může pro pohodlnou práci s aplikací vybrat jazyk, který je mu nejbližší. Výchozím jazykem je angličtina, dále IS nabízí češtinu. Jazyk je možné si zvolit v pravé horní části aplikace, viz. obrázek č. 8, černá elipsa.



Obrázek č.8 - Webová aplikace - celkový pohled

Žlutou elipsou je označena ukázka chování základní lišty, jejímž prostřednictvím se uživatel dostane k potřebným operacím. Nabízí-li základní nabídka další seznam konkrétních operací, je tento seznam zobrazen automaticky při přejetí kurzorem myši přes základní nabídku, tak jak je to znázorněno na obrázku č.8. Uživatel si dále kliknutím vybere konkrétní operaci, kterou chce provést. Ne všechny základní položky menu však submenu obsahují.

Kliknutím na požadovanou operaci se na pracovní ploše zobrazí příslušný formulář. Pracovní plochou označujeme veškerý prostor pod lištou s menu.

Celou aplikaci je možné ovládat počítačovou myší, což je pro uživatele nejjednodušší způsob komunikace. Také celkové ovládání je intuitivní, čili každý uživatel, který umí minimálně na základní uživatelské úrovni pracovat s počítačem, je schopen IS bez problémů ovládat.

Jednotlivými položkami hlavního menu a operacemi, které poskytují se budeme zabývat v následujících kapitolách.

4.2.2 Dítě

Menu Dítě nabízí tři operace. Operace s názvem **Přidat sponzora** provází uživatele procesem přidávání nových sponzorů do databáze informačního systému. Veškeré informace, které se o sponzorech registrují, jsou jejich jméno a příjmení, nebo název společnosti, pokud se jedná právnickou osobu, kontakt na sponzora a jeho identifikační číslo, které se používá pro jeho jednoznačnou identifikaci v rámci charity. Aplikace nabízí uživateli vygenerovat toto číslo automaticky, pokud uživatel zadává nového sponzora, nebo jej může zadat ručně. Pokud však zadavatel zadá číslo, které odpovídá některému již zaregistrovanému sponzorovi, nebude tento sponzor do databáze vložen, stejně tak pokud se budou všechny ostatní údaje přesně shodovat s některým jiným sponzorem.

Další podmenu **Přidat do SAF** poskytuje uživateli možnost přihlásit dítě do fondu SAF. Vstupní formulář je velmi jednoduchý. Obsahuje pouze tři položky, kde první je Charita's No. dítěte, které identifikuje dítě, které do SAF přihlašujeme, dále částka, o kterou je žádáno a účel, na který mají být peníze použity. Všechny tři údaje jsou povinné.

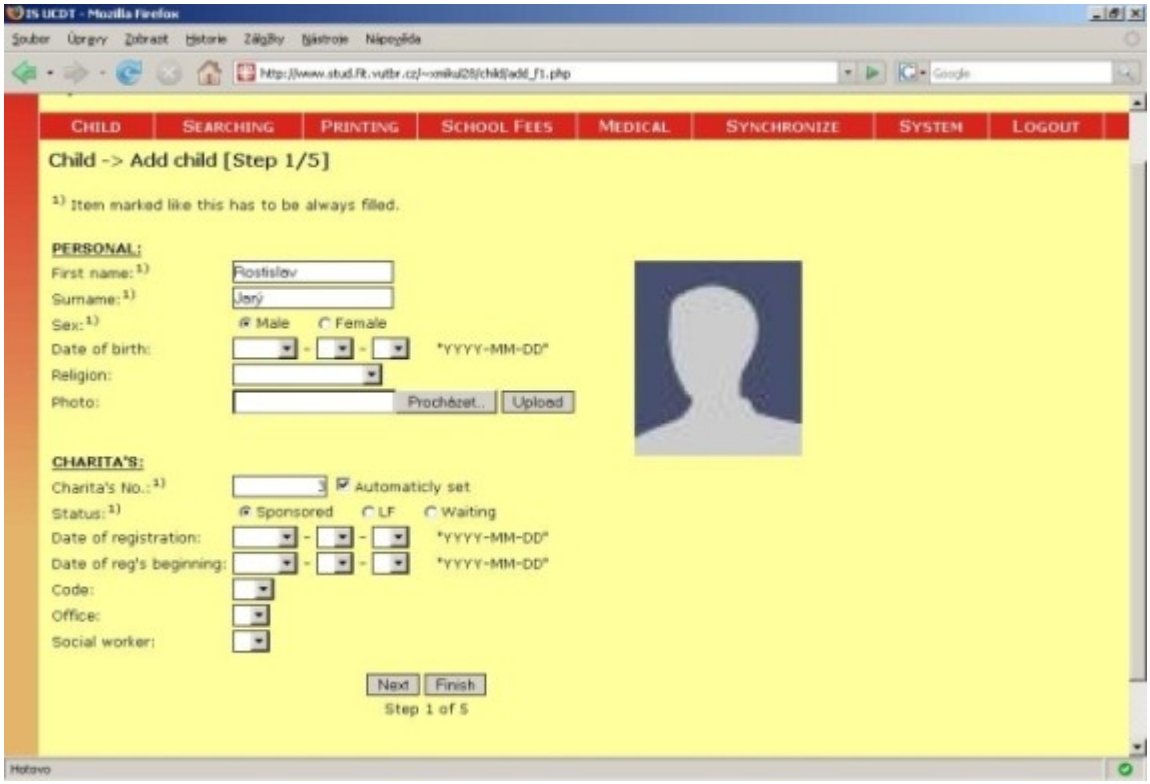
Poslední položkou je **Přidat dítě**. Jedná se o proces registrace nového dítěte do programu a tím i do databáze IS. Je to nejsložitější proces v celém IS, proto si jej nyní popíšeme podrobněji.

Operace přidávání dítěte do databáze můžeme označit jako vytváření karty dítěte, která obsahuje velké množství údajů (viz. kapitola 2.2.2).Tyto údaje lze rozdělit na několik logických celků. Proto je proces registrace dítěte rozdělen na pět částí, z nichž každá registruje informace logicky souvislé, aby uživatel neztratil při vytváření karty dítěte přehled. V průběhu procesu je uživateli umožněno se pohybovat z jedné části do druhé pomocí tlačítek **Dále** a **Zpět**, přičemž údaje zadané v části, ze které se pokouší odejít, zůstanou po návratu na tuto část zobrazeny, takže není třeba se obávat ztráty jakýchkoliv již zadaných údajů. Každá část také obsahuje tlačítko **Dokončit**. To je zde z důvodu, že ne vždy jsou při vytváření karty známy všechny informace o dítěti, nebo je třeba registrovat jenom základní údaje, které je možné zadat již na prvních záložkách. Každá záložka má také v horní části stručný popis, co na té záložce nalezneme, o jaké údaje se jedná, popř. poznámky, které značí omezení některých položek, aby byl uživatel se všemi výjimkami seznámen.

První záložka obsahuje formulář s nejdůležitějšími informacemi. Kromě jména, příjmení a pohlaví dítěte je možné uvést také datum narození. Není to však povinná položka, protože ne vždy je datum narození známo. Zadávání data v celém IS je stejné jako zde. Prvním důvodem je skutečnost, že nemusí být známo přesné datum narození, ale může být známo částečné, třeba pouze rok narození. Dalším důvodem je nutnost zadávat datum tak, aby bylo správné, tedy aby se zabránilo nevalidnímu

zápisu data, jako je např. 31. února, ale naopak když je přestupný rok, tak aby bylo nabídnuto i 29. února. Dalším zajímavým prvkem je pole pro výběr souboru, který má být vložen jako fotografie dítěte. Zde platí omezení, že soubor musí odpovídat MIME typu *image/jpeg*. Pokud se podaří nahrátí souboru, bude fotografie, kterou jsme nahráli zobrazena namísto siluety na obrázku v pravo.

Charita's No. dítěte, což je ustálené pojmenování registračního čísla dítěte pro identifikaci v IS je možné zadat ručně, což se doporučuje použít v případě, že registrujeme dítě, které je již jednou registrováno ve staré papírové kartotéce a pouze jej přepisujeme do elektronické podoby. Pokud registrujeme nové dítě, je lepší využít automatického nastavení tohoto čísla, čehož dosáhneme zaškrtnutím příslušného políčka. Následně je do kolonky doplněno nové Charita's No. Je třeba však poznamenat, že tahle funkce funguje v prohlížeči Internet Explorer 6.0 až když je tlačítkem myši kliknuto kdekoliv jinde na stránku (nejspíš nesprávná funkce metody JavaScriptovské metody onChange na prvek checkbox). Další zajímavostí je zadávání kódu oblasti, kanceláře a jméno sociálního pracovníka, který se bude o dítě starat. Je nutné zadávat tyto údaje postupně. Není totiž možné zadat kancelář či jméno sociálního pracovníka aniž bychom věděli, do které oblasti je dítě zařazeno. Proto jsou zpočátku všechny položky prázdné, teprve postupným výběrem se naplňují další výběrová tlačítka. Tento postup je zvolen v mnoha dalších částech IS a slouží jako ochrana konzistence dat, přičemž uživateli je tím poskytnuto pohodlí při zadávání údajů. Celá první záložka je zobrazena na obrázku č.9:



Obrázek č.9 - Přidat dítě - krok 1

Druhá záložka slouží k zadávání doplňujících informací o dítěti. Uživatel má u každé kategorie možnost výběru z hodnot, které jsou v IS uloženy k tomuto účelu. Teprve po stisku tlačítka **Přidat** je vybraná položka zaregistrována. Položka, která je při ukládání vybrána ve výběrovém tlačítku se nebere v potaz! Pokud se snaží uživatel registrovat položku, kterou již jednou registroval, bude mu to připomenuto v oznamovacím okně. Toto platí také v celém IS.

Třetí záložka slouží k registraci údajů o rodině dítěte – o otci, matce a sourozencích. Pokud nevyplníme žádnou položku v kategorii Otec a Matka, znamená to, že dítě otce či matku nemá. Pokud chceme zaregistrovat sourozence, musíme jejich iniciály zadat do připravených políček a stejně jako na minulé záložce i tady platí, že registrování jsou pouze sourozenci, kteří jsou uvedení ve spodním seznamu.

Majetek a škola, kterou dítě navštěvuje, to jsou dvě hlavní domény, které se registrují na záložce číslo čtyři. U majetku funguje způsob registrace stejný jako byl uveden v předchozím textu a školu může uživatel vybírat taky postupně od výběru kódu oblasti, ve které se má škola nacházet až po vyplnění třídy a výše školného, jejichž položky se zobrazí pouze a jenom jsou-li vyplněny všechny předchozí hodnoty.

Posledním krokem registrace dítěte je seznam sponzorů, kteří poskytují registrovanému dítěti finanční prostředky. Protože jedno dítě může sponzorovat více osob, je zde připraven seznam.

V každém kroku registrace je možné využít tlačítka **Dokončit**, čímž se proces registrace ukončí. Je třeba ale poznamenat, že za řádný konec registrace se považuje až toto tlačítko na poslední páté záložce. Ty předchozí jsou pro předčasné dokončení, z čehož vyplývá, že když např. uživatel vyplní informace v poslední páté záložce, vrátí se zpět na záložku třetí a stiskne tam tlačítko **Dokončit**, budou uloženy jen údaje ze záložek č. 1, 2 a 3, údaje ze záložky 4 a 5 budou ignorovány.

4.2.3 Vyhledávání a Tisk

Sekce Vyhledávání a Tisk jsou si velmi podobné. Liší se pouze výběrem položek, ze kterých je možné sestavit vyhledávací filtry, proto je budeme popisovat společně.

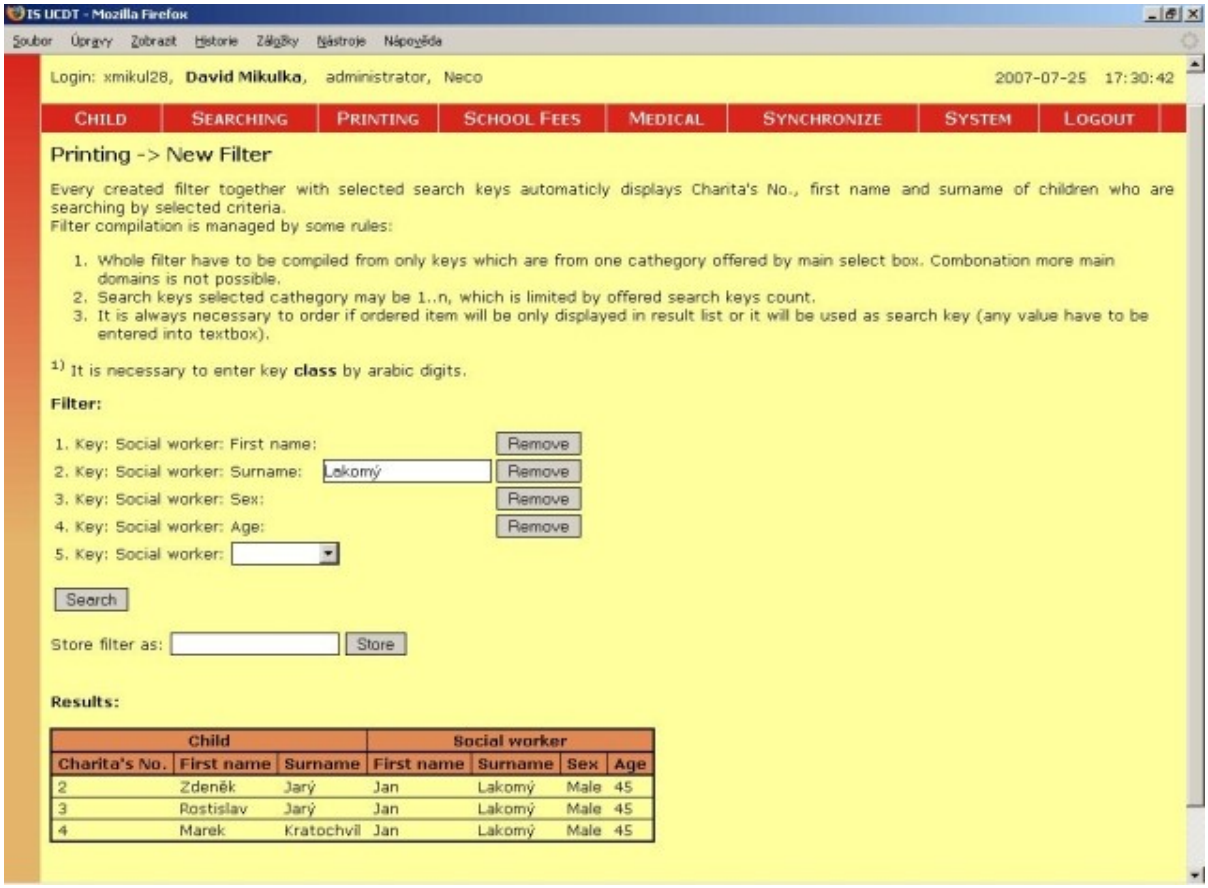
Obě záložky nabízejí submenu, které obsahuje položku **Použít filtr** a **Nový filtr**. První jmenovaná slouží pro rychlejší vyhledávání za použití již vytvořených a uložených filtrů. Uživateli je vždy zobrazen seznam jím uložených filtrů, které může použít. Uživatel si vybere jeden z nich, zadá vyhledávací hodnoty, stiskne tlačítko pro vyhledávání a je mu předložen výsledek.

Záložka **Nový filtr** slouží k definování nových filtrů. Jako stěžejní prvek je zde na začátku hlavní výběrové políčko, ze kterého si uživatel vybírá hodnotu, podle které chce vyhledávat. Jakmile je základní hodnota vybrána, je zobrazeno buď další nabídka rozšiřující základní výběr nebo, pokud vybraná hodnota již žádné rozšiřující klíče neposkytuje, je zobrazeno výběrové políčko, které se ptá,

zda bude vybraná položka sloužit pouze jako informace, která má být v konečném seznamu zobrazena, nebo zda má být použita jako vyhledávací klíč. Tímto výběrem se považuje jeden vyhledávací klíč za definovaný a je tak přidán do seznamu. Pokud byla položka zvolena jako vyhledávací klíč, bude k ní doplněno textové, popř. výběrové pole, samozřejmě i tato položka bude zobrazena ve výsledném seznamu. Protože můžou vždy nastat okolnosti, kdy bude nutné smazat již definovaný vyhledávací klíč, je u každého z nich tlačítko **Odebrat**, které příslušný klíč odebere.

Jakmile je dokončen výběr vyhledávacích klíčů, je možné stiskem tlačítka **Hledat** zobrazit seznam vyhledaných hodnot podle nastaveného filtru. Pokud bude toto tlačítko stisknuto aniž by byl vyplněn každý ze definovaných klíčů, nebude vyhledáno nic a uživatel bude upozorněn na tuto skutečnost. Pokud bude všechno v pořádku a přesto nebude pod vyhledávacím filtrem zobrazen seznam, znamená to, že nebyla nalezena žádná položka odpovídající zadaným klíčům. Naopak seznam s nalezenými hodnotami bude vypadat jako na obrázku č. 10.

Pro uložení vytvořeného filtru musí uživatel zadat do příslušného textového pole název filtru. Pokud uživatel zadá název, který je stejný jako v minulosti jím definovaný jiný filtr, bude původní filtr smazán. Filtr je uložen na serveru pod zvoleným jménem s příponou *.ftr*. Uživatel tuto příponu zadávat nemusí, bude doplněna automaticky.



Obrázek č. 10 - Tisk - Nový filtr

V předchozích odstavcích popsaný způsob definování vyhledávacích filtrů byl zvolen a použit jako optimální. Je zde sice patrné omezení volnosti ve výběru hodnot, které si smí uživatel zvolit jako klíče, nicméně v další etapě života tohoto softwaru je možné položky upravit podle dalších konkrétních požadavků zadavatele.

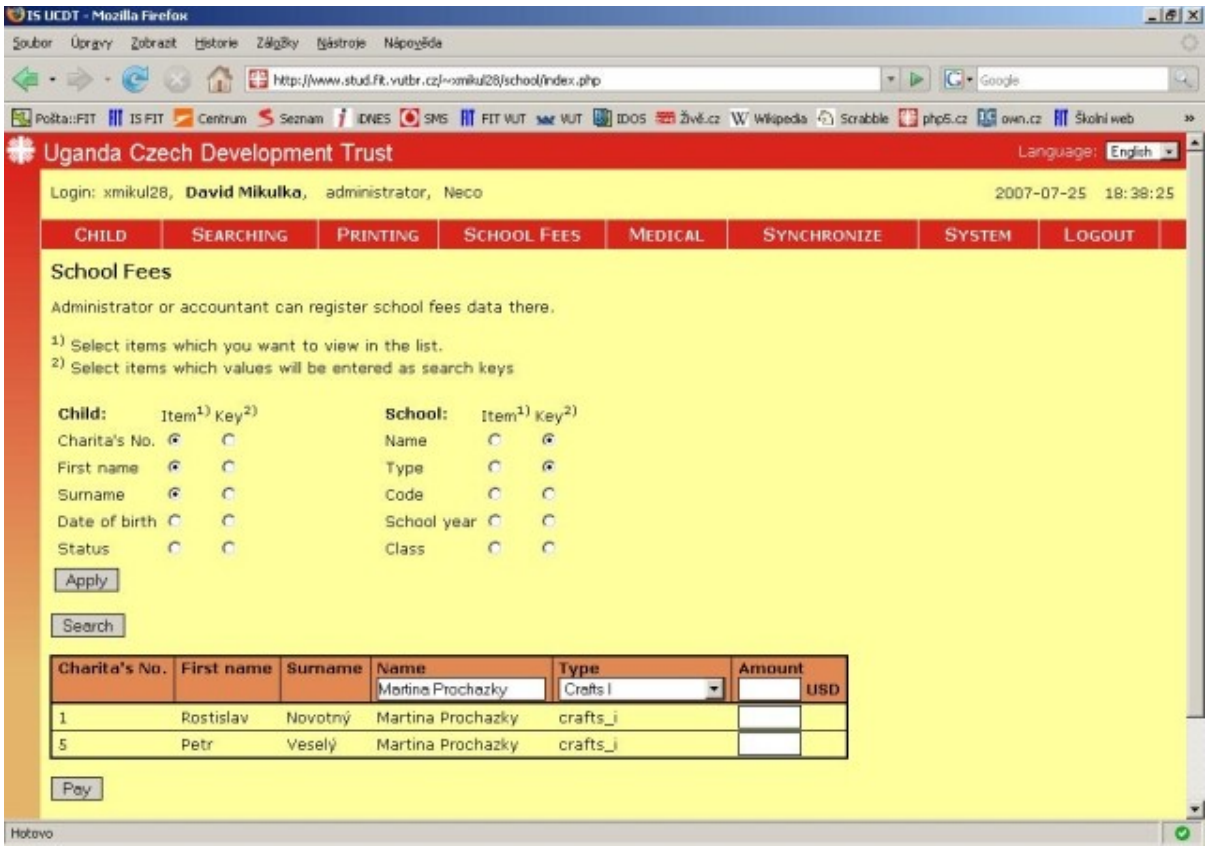
Cestu, která nabízí uživateli veškeré údaje a ten si z nich vybírá není v tomto případě možné se ubírat. Důvod je jednoduchý a základní. Vnitřní uložení dat v databázi je uživateli skryto a v žádném případě neodpovídá úhlu pohledu uživatele. Mohla by tak nastat situace, kdy uživatel zvolí za vyhledávací klíče několik hodnot, které spolu vůbec nesouvisí a tudíž podle nich není možné vyhledávat, protože je složitá kontrola na úrovni aplikace, která musí rozpoznat, zda lze data spojit do souvislostí či nikoliv a případně to uživateli oznámit, stejně tak když taková situace nastane, uživatel může s těží hledat správnou sestavu klíčů, aby se mu podařilo požadovaný filtr sestavit.

4.2.4 Placení školného

Placení školného je další velmi významnou a pilotní operací, kterou IS zajišťuje. Tuto operaci může provádět pouze uživatel typu účetní a administrátor.

První činnost, kterou musí uživatel udělat, je zvolit si, stejně jako u vyhledávání, kritéria, podle kterých se budou dětí, jimž má být školné zaplaceno, vyhledávat. Princip výběru je však jiný než v sekcích Vyhledávání a Tisk. Při vytváření tohoto filtru není nutné vybírat z široké škály položek. Stačí pouze základní iniciály dětí, které jsou tu proto, aby bylo možné vyhledat i malé množství dětí, např. jedno konkrétní dítě. Dále pak potřebujeme znát název a typ školy, jíž bude školné zaplaceno, školní rok, za který se platí, třídu, kterou děti navštěvují a také kód oblasti, ve které se škola nachází. Není totiž vyloučeno, že se některá škola určitého typu a jména může vyskytovat ve více oblastech.

Když máme vybrána všechna kritéria, stisknutí tlačítka **Použít** se nám zobrazí záhlaví tabulky, ve které budou zobrazeny výsledky hledání. Toto záhlaví odpovídá vybraným klíčům a tam, kde byla některá položka zvolena jako vyhledávací klíč, je také vstupní pole pro zadání hodnoty klíče. Po zadání hodnot klíčů a stisknutí tlačítka **Hledat** se zobrazí tabulka s vyhledanými hodnotami. Navíc se u každého řádku tabulky, jako poslední sloupec, zobrazí vstupní pole, do kterého se může zadat částka, která má být zaplacena. Jako základní částka se však bere ta, která bude vyplněna v záhlaví tabulky. Pokud budou položky všech řádků nevyplněny a v záhlaví bude např. částka 1000 USD, bude pro všechny nalezené položky zaplaceno 1000 USD. Pokud bude v některém vstupním poli jiná hodnota, bude tomu konkrétnímu dítěti zaplaceno tomu odpovídající školné. Tento princip je zde použit kvůli možnosti ovlivnit každou položku, a zároveň aby nebylo třeba zadávat pro velké množství dětí jednu sumu opakovaně.



Obrázek č. 11 - Placení školného

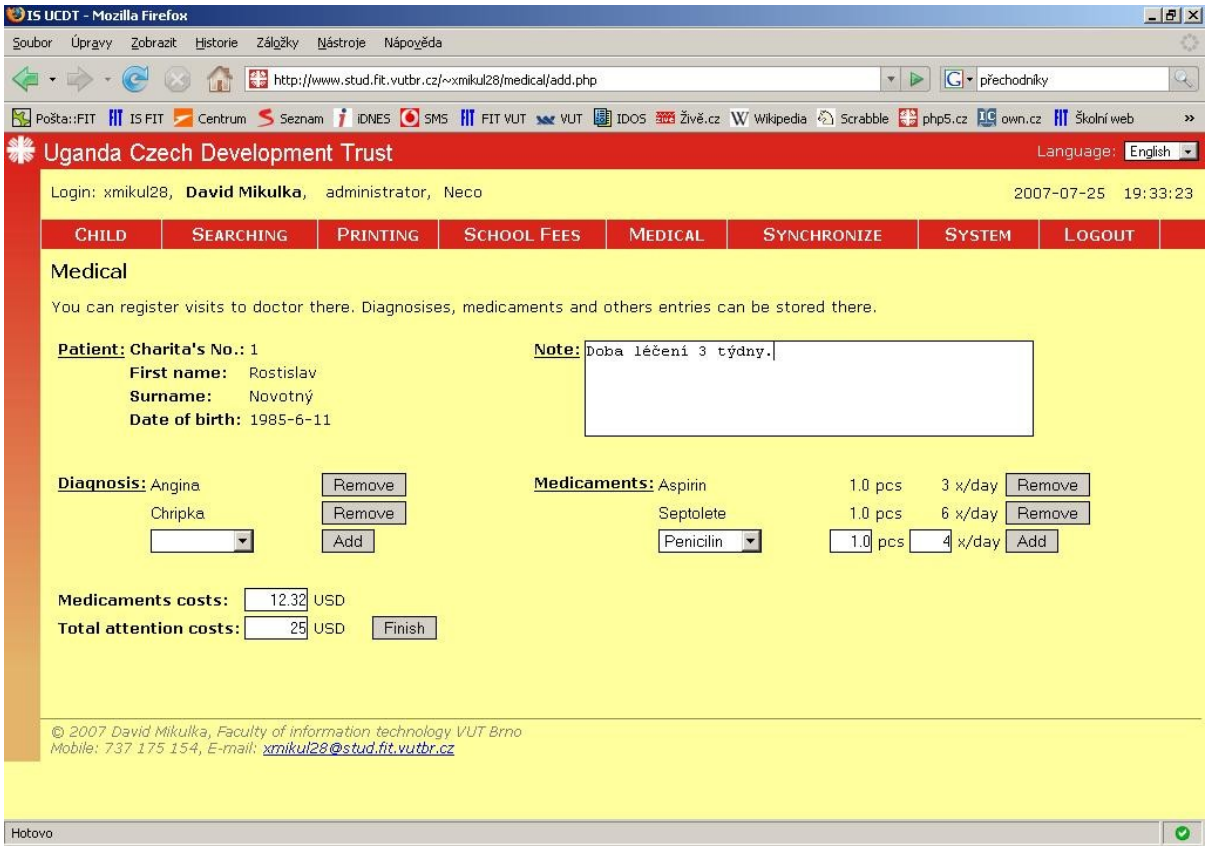
Po úspěšném vyhledávání, kdy byl nalezen aspoň jeden záznam odpovídající zadaným klíčům, je pod tabulkou zobrazeno i tlačítko **Zaplatit**. Pokud bude toto tlačítko stisknuto a v zápatí nebude vyplněna žádná suma, která se má zaplatit, bude zobrazeno tomu odpovídající hlášení a zaplacení se neprovede. V opačném případě je provedeno zaplacení a zrušení celé tabulky, což je provedeno z bezpečnostních důvodů, aby uživatel nemohl stisknout tlačítko **Zaplatit** opakovaně.

Kromě účelného použití jiného typu výběru a definice filtru je tento způsob použit i kvůli demonstraci dalšího způsobu, jak lze filtr koncipovat. Zadavatel tak má možnost vyzkoušet obě varianty a může si tak v budoucnu vybrat jeden způsob, který mu bude více vyhovovat a ten pak bude v další verzi použit pro veškeré filtry. Na obrázku č. 11 je ukázka operace placení školného.

4.2.5 Lékařské záznamy

Uživateli s oprávněním administrátora nebo lékaře přicházejícího do sekce Lékařské záznamy, je zobrazena úvodní stránka, která obsahuje pouze tři textová pole. První z nich slouží k zadání Charita's No. dítěte, kterému je poskytnuto lékařské ošetření, další dvě pro zadání jeho jména a příjmení. Tyto údaje slouží pro vyhledání dítěte. Není třeba použít další a jiná vyhledávací kritéria, protože každé dítě, které je ošetřováno, zná svoje jméno a příjmení a kdyby nastala situace, že by takových dětí bylo více, tak je zobrazeno i jejich identifikační číslo, jež je unikátní.

Po stisku tlačítka **Hledat** je zobrazen seznam dětí jejichž iniciály odpovídají zadaným klíčům. Tam si uživatel vybere konkrétního jedince, který je ošetřován a tlačítkem **Vybrat** u jeho záznamu je aplikaci zadáno, aby přešla na formulář pro zadání záznamu o návštěvě lékaře. Tento formulář je zobrazen na obrázku č. 12.



Obrázek č. 12 - Lékařské záznamy

Na tomto formuláři jsou vlevo nahoře zobrazeny základní údaje o dítěti. Vpravo nahoře je vstupní textové pole, do kterého lékař zapisuje poznámky týkající se léčby pacienta. Ve spodní části se zadávají diagnózy a případně aplikované léky. Postup je již známý a osvědčený. Jedno vylepšení

je patrné u zadávání léků. Každý lék má totiž v databázi uloženo jeho doporučenou dávku užití a taktéž doporučený interval užití. Pokud je tedy některý lék vybrán, jsou tato doporučená čísla uživateli nabídnuta v následujících dvou polích. Pokud však lékař stanoví jiné dávkování a interval, může jednoduše přepsat tyto hodnoty novými, které budou považovány za směrodatné.

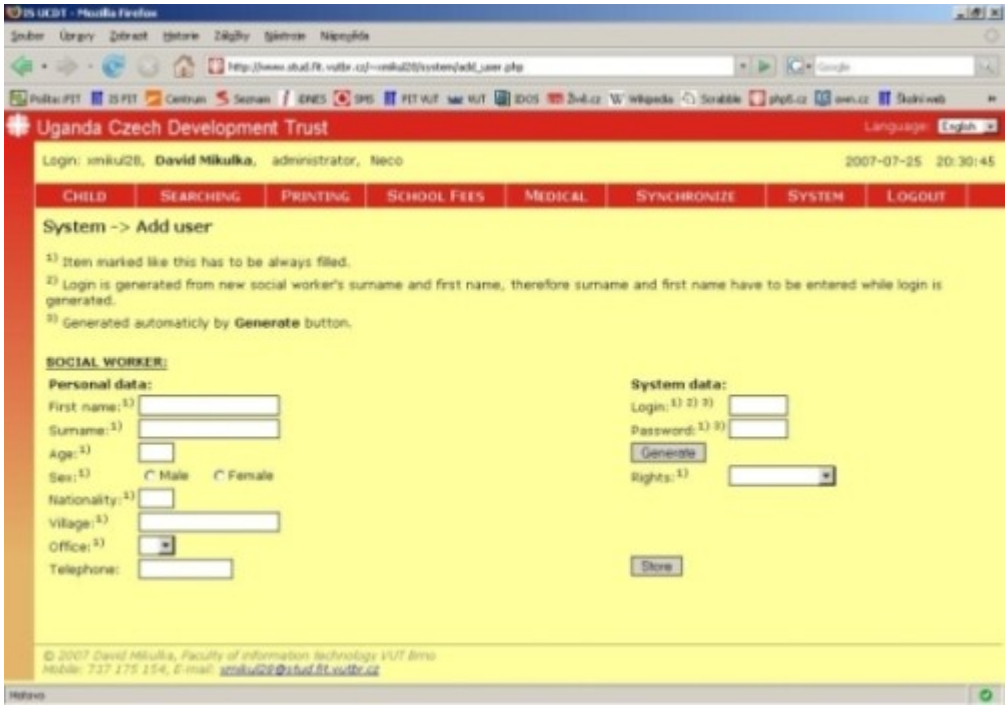
Nakonec uživatel vyplní částku, kterou je nutné uhradit za léky – tato položka je zobrazena pouze pokud jsou nějaké léky lékařem zadány, a celkovou částku za ošetření. Tlačítkem **Dokončit** lékař stvrzuje výše uvedené informace a tím ukončuje zadávání informací o ošetření.

I když lékař nikde neudává datum ošetření, je podle něj možné v sekci Tisk vyhledávat zdravotní záznamy. Datum ošetření je totiž zaznamenáváno automaticky podle data, kdy byl záznam zadán do IS.

4.2.6 Systém

Při přejetí přes menu Systém je uživateli nabídnuto submenu obsahující položky **Přidat záznam** a **Přidat uživatele**. První sekce slouží pro ukládání databázových záznamů, které jsou zobrazovány ve výběrových položkách v celém IS, tzv. číselnících. Některé záznamy musí být zaznamenány pro jednu položku ve všech jazycích aplikace, čemuž odpovídá také počet vstupních polí. Zadávání takových záznamů je jednoduché a intuitivní.

Druhá sekce slouží k přidávání sociálních pracovníků do databáze IS. Formulář zadávání je zobrazen na obrázku č. 13:



Obrázek č. 13 - Přidat Uživatele

Na levé straně se zadávají osobní údaje nového sociálního pracovníka. Na straně pravé jsou potom generovány přihlašovací údaje a vybírají se práva, která bude nový uživatel mít. Stiskem tlačítka **Generovat** se vyplní vstupní pole přihlašovacího jména a hesla.

Přihlašovací jméno je generováno z příjmení uživatele. První písmeno přihlašovacího jména je vždycky x následované prvními pěti písmeny příjmení transformovanými na písmena malé abecedy bez diakritických znamének. Pokud je příjmení menší než pět písmen, je tato pětice doplněna scházejícím počtem písmenek ze jména uživatele. Za touto šesticí následují dvě číslice arabských číslic. Tato dvojice čísel je zde pro případ, že nastane situace, kdy několik lidí má stejnou první šestici písmen. Potom je toto číslo pořadí, které udává, kolikátý uživatel s touto posloupností znaků je v databázi uložen. Tím se stává přihlašovací jméno unikátním pro každého uživatele.

Heslo je generováno jednoduchým algoritmem, který generuje posloupnost 8 znaků, která může obsahovat, stejně jako login, pouze číslice a znaky malé abecedy. Algoritmus je založen na tom, že chceme, aby průměrný počet číslic byl 2 znaky z 8, což je pravděpodobnost 1:4. Takže nejdříve generujeme náhodné číslo od 0 do 500. Pokud je číslo menší jak 100, generujeme znovu náhodné číslo od v intervalu od 48 do 57, což jsou dekadické hodnoty číslic. Pokud prvně generované číslo bylo větší jak 100, generujeme druhé náhodné číslo v rozsahu od 97 do 122, což odpovídá dekadickým hodnotám písmen malé abecedy. Tento úkon provedeme v cyklu 8x, tedy pro každý znak hesla.

Nakonec stiskem tlačítka **Uložit** uložíme nového sociálního pracovníka do databáze a tímto okamžikem má nový uživatel přístup i k operacím IS podle svého oprávnění.

Položku **Odhlásit** není třeba dlouho popisovat, slouží k odhlášení uživatele ze systému. Poslední položka hlavního menu má název **Synchronizace**. Ta je přichystána pro proces synchronizace, který je popsán v následující kapitole.

5 Synchronizace

V této kapitole se budeme zabývat synchronizací databází dvou našich paralelně běžících informačních systémů. Ještě než se tak stane, je třeba říci, že následující text je třeba brát pouze v rovině teoretické, nikoliv jako konkrétní popis implementace. To proto, že samotní zadavatelé nemají dosud konkrétní představu o tom, která data budou synchronizace podléhat a jakým způsobem se má celá synchronizace, z hlediska konkrétních informací, provádět. Dalším důvodem je také fakt, že zadavatel prozatím s využitím synchronizace nepočítá, jelikož bude prozatím v provozu pouze jeden IS, a to v Ugandě.

Důležitou součástí IS je potřeba vzájemné aktualizace údajů, se kterými pracuje centrum v Praze, s daty, které naopak shromažďují pracovníci v Ugandě.

Otázka synchronizace databáze je problém rozsáhlý, proto je mu věnována celá kapitola. Jedná se o velmi složitý proces, který s sebou nese mnohá úskalí, avšak musí být nastaven tak, aby v procesu synchronizace nedocházelo k žádným chybám. Je nutné najít optimální řešení, které bude brát na zřetel všechny komplikace a problémy týkající se synchronizace. Všechny tyto problémy a jejich řešení si postupně rozebereme.

První věc, kterou je třeba si uvědomit je, že proces synchronizace znamená přenos velkého množství dat, podle velikosti databáze a podle doby uplynulé od poslední synchronizace, se může jednat až o několik stovek megabytů. Je nutné zvolit takový způsob přenosu, aby kromě samotných dat, bylo přenášeno co nejméně režijních informací.

Zvolili jsme proto řešení, které pracuje takto: Budeme procházet všechny záznamy všech tabulek zdrojové databáze (ta, která se bere jako aktuální). Každý záznam má v sobě informaci o tom, kdy byl naposled editován (atribut *mtime*) a synchronizován *(stime)*. Pokud je datum a čas poslední editace záznamu historicky mladší než datum poslední synchronizace, je jasné, že byl tento záznam od poslední synchronizace aktualizován. To je signál, že je potřeba jej aktualizovat v cílové databázi. Proto vezmeme data obsažená v tomto záznamu a vytvoříme s nimi textový řetězec coby databázový dotaz (UPDATE nebo INSERT). Tento řetězec pak cílový systém přímo použije jako databázový dotaz. Jak ale poznáme, který z příkazů máme použít? Příkaz UPDATE použijeme v případě, můžeme-li s určitostí říct, že je tento záznam v cílové databázi již uložen, INSERT v opačném případě. Jediná možnost jak to zjistit je opět porovnat datum poslední synchronizace s datem vytvoření záznamu. Pokud byl záznam vytvořen dříve než byla provedena poslední synchronizace, tak bude použito příkazu UPDATE, jinak INSERT, který záznam nově vytvoří.

Jakmile budeme mít vygenerovány všechny dotazy, uložíme je do souboru (pokud bude informací hodně tak je rozdělíme do více souborů) a pošleme je cílovému IS. Ten postupně projde soubor a vykoná příkazy v něm uložené.

Samotný přenos souborů je také komplikovaný. Jedním důvodem je pomalé připojení k internetu, které nás nabádá k co největší úspoře v přenosu dat. Dalším problémem s tímto související, avšak fundamentální, je dostupnost elektrické energie. V Ugandě totiž funguje elektřina pouze několik hodin denně.

Spojením dvou výše uvedených nároků nám vykrystalizují další požadavky na synchronizaci. Tu musí být tedy možné kdykoliv pozastavit a znovu spustit, protože se může stát, a je předpoklad, že se také často stane, že v průběhu synchronizace, jejíž objem dat bude velký, se díky nízké rychlosti připojení natáhne čas nutný k jejímu dokončení přes dobu, kdy bude spuštěna elektrická energie.

V souvislosti s tím je také třeba právě tyto krajní situace ošetřovat. Musíme neustále kontrolovat, zda jsou cílový a zdrojový IS pořád ve spojení, aby v případě, kdy se jeden odpojí, ten druhý stále nečekal na další aktualizace, apod. Kromě toho, pokud dojde k pozastavení či přerušení synchronizace, musíme také zaznamenávat a přesně definovat, které záznamy se nám podařilo aktualizovat a které bude potřeba poslat znovu, atd.

Z výše uvedeného popisu vyplývá, že synchronizace jako taková má velmi striktní pravidla. Vždy je určen zdroj a cíl. Tím je jasně stanoveno, které záznamy se berou jako aktuální a které se mají podle nich obnovovat. Je zřejmé, že pokud dojde k situacím, kdy konkrétní záznam v cílové databázi je novější, ale přesto bude přepsán tím, který je ve zdrojovém IS. Uživatel tedy nemá žádné nástroje na to, aby se případně mohl rozhodnout, zda chce cílový záznam opravdu přepsat, i když je novější, nebo ne.

Všechna tato zjištění a kontrola si žádá další nároky na velikost přenášených informací, a také složitější implementaci síťového aplikačního protokolu přenosu, který by, kromě zjišťování, který záznam je novější, musel být schopen interaktivně spolupracovat s uživatel. Vezmeme-li ale opět v potaz krajní situaci, že každý synchronizovaný záznam byl měněn od poslední synchronizace jak ve zdrojovém, tak v cílovém IS, byl by uživatel pokaždé např. dotazován, jak má být se záznamem dále naloženo. To je, vzhledem k tisícům záznamů, krajně nevhodné a navíc časově velmi náročné.

Proto se musejí definovat pevná pravidla, kterých se bude synchronizace řídit již před jejím započetím. Tohle je však již nad rámec vytvářeného IS a případná implementace „inteligentnější“ synchronizace je možná až jako rozšíření aplikace v budoucnu.

6 Závěr

Tvorba informačního systému, čímž se tato bakalářská práce zabývá, je v současné době velmi svižně se rozvíjející oblast informačních technologií. IS jako takový je však spojením několika oborů z informačních technologií, počínaje znalostí programování pomocí strukturovaného programovacího jazyka, přes databázové systémy, práci se sítěmi a tvorbou uživatelského rozhraní, která zahrnuje grafiku a design, konče.

Hlavním důvodem proč ale byla aplikace vytvořena je zlepšení administrativní práce, která se týká placení školného a zaznamenávání jeho historie. Taktéž vedení lékařských záznamů je nemalou a nezanedbatelnou funkcí tohoto IS.

Podle množství a také podle důležitosti některých údajů a operací s nimi je také systém vybaven zabezpečením v podobě přístupu k němu založenému na autentizaci pomocí hesla, kde každý uživatel má jasně definovány operace, které může provádět.

IS můžeme rozdělit na dvě části. Jedna část je jeho vnitřní struktura a implementace a druhá jeho uživatelské rozhraní, které slouží pro komunikaci mezi IS a jeho uživatelem.

První část je pro uživatele zapouzdřena. Je to sice nejdůležitější část IS, která obstarává chod celého IS, nicméně ale není třeba aby uživatel znal její detaily. Implementace IS je také nejsložitějším a nejnáročnějším procesem ve výrobě aplikace tohoto typu. Její tvůrce musí brát v úvahu všechny požadavky zadavatele, nacházet co nejoptimálnější řešení, snažit se, aby byl IS bezpečný, robustní a přitom spotřebovával co nejméně prostředků na svůj provoz. K implementaci této části bylo použito standardních prostředků, hlavně jazyka PHP a databázového jazyka MySQL.

Druhá část systému, tedy uživatelské rozhraní se stará o to, aby byla uživateli při práci s aplikací co nejvíce usnadněna práce a aby se uživatel v aplikaci dobře vyznal. Zde byly využity výhody značkovacího jazyka HTML s podporou kaskádových stylů CSS, spolu s jazykem JavaScript, jejichž kombinací bylo dosaženo výsledného uživatelského rozhraní.

Ačkoli IS splňuje veškeré požadavky na něj kladené, není proces jeho tvorby zdaleka u konce. Pro to, aby mohl být IS plně a bez obav využit v ostrém provozu, je třeba jej otestovat. Teprve během zkušebního provozu budou nalezeny chyby a nedostatky aplikace, které je nutné odstranit.

Teprve poté je možné uvést IS do plného provozu. Avšak ani tímto okamžikem práce na IS nekončí. V době běhu aplikace vyplynou na povrch další situace a skutečnosti, na které musí být schopen IS reagovat, stejně tak detaily uživatelského rozhraní dovést do dokonalosti tak, aby byl uživateli poskytnut co největší komfort při práci s aplikací.

S tím jsou spojená rozšíření či vylepšení aplikace. Jedním z vylepšení je například inteligentnější systém synchronizace databáze, který je popsán v kapitole 5. Dalším rozšířením může být například přidání dalších klíčů, které lze použít pro vyhledávání v databázi.

7 Literatura

[1] Zendulka, Jaroslav, Rudolfová, Ivana. Databázové systémy, Studijní opora, 2006

[2] Hruška, Tomáš, Křivka, Zbyněk. Informační systémy, Pojem informačního systému, data, procesy, transakce, Studijní opora, 2006

[3] Wikimedia Foundation, Inc. Wikipedia – The Fee Encyclopedia. [online]. URL: <http://en.wikipedia.org/>. [cit. 3.12.2006]

[4] Jun, Adam. MySQL manuál. [online]. URL: <http://mm.gene.cz/>. [cit. 15.2.2007]

[5] Kosek, Jiří. Jak psát web. [online]. URL: <http://www.jakpsatweb.cz>. [cit. 5.1.2007]

[6] The PHP Group. PHP manual. [online]. URL: <http://www.php.net>. [cit. 12.1.2007]

[7] Honza, Petr. Jak na web. [online]. URL: <http://www.jaknaweb.com>. [cit. 2.2.2007]

[8] Grimmich, Šimon. Tvorba webu. [online]. URL: <http://www.tvorba-webu.cz>. [cit. 2.2.2007]

8 Přílohy

Příloha 1. URL adresa informačního systému.

Příloha 2. Nosič CD, na kterém jsou zdrojové kódy aplikace spolu s elektronickou podobou této bakalářské práce.

Příloha 1 URL adresa informačního systému

URL: <http://www.stud.fit.vutbr.cz/~xmikul28/>

Pro přístup a odzkoušení IS je třeba znát přihlašovací jméno a heslo. Pro hodnotitele bakalářské práce je zřízen účet administrátora, který má přístup ke všem operacím, s přihlašovacím jménem **xnovak00** a heslem **9iyxm8dc**.