

Informační systém neziskové organizace Oblastní charita Břeclav

Information system NGO Regional charity Breclav

Bc. Pavel Čech

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Pavel ČECH**

Osobní číslo: **A09748**

Studijní program: **N 3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Téma práce: **Informační systém neziskové organizace Oblastní charita Břeclav.**

Zásady pro vypracování:

1. Analyzujte problematiku informačních systémů neziskových organizací a vypracujte literární rešerši na dané téma.
2. Formou projektu navrhnete informační systém pro zvolenou organizaci.
3. Realizujte navržený informační systém a provedte jeho otestování.
4. Provedte diskusi nad řešením a implementací projektu.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. WELLING, Luke; THOMSON , Laura. MySQL : průvodce základy databázového systému. Vyd. 1. Brno : CP Books, 2005. 255 s. ISBN 8025106713.
2. GROFF, James R.; WEINBERG, Paul N. SQL : kompletní průvodce. Vyd. 1. Brno : CP Books, 2005. 936 s. ISBN 80-251-0369-2.
3. FILIPPO , Di Pisa. Beginning Java? and Flex : migrating Java, Spring, Hibernate, and Maven developers to Adobe Flex. New York : Apress, 2009. 425 s.
4. SANDERS, William B. ActionScript 3.0 Programming : Overview, Getting Started, and Examples of New Concepts. 1st ed. Sebastopol : O'Reilly, 2007. 74 s.
5. GASSNER, David. Flash Builder 4 and Flex 4. Indianapolis : Wiley Publishing, 2010. 1056 s.
6. LOTT, Joey; SCHALL, Darron; PETERS , Keith. ActionScript 3.0 : cookbook . Sebastopol : O'Reilly, 2006. 556 s.
7. SANDERS, William B.; CUMARANATUNGE, Chandima. ActionScript 3.0 : design patterns. Cambridge : O'Reilly, 2007. 509 s.
8. Adobe Creative Team. Adobe Flash CS4 Professional. Vyd. 1. . Brno : Computer Press, 2009. 389 s.

Vedoucí diplomové práce:

doc. Mgr. Roman Jašek, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této práce je vytvořit informační systém pro potřeby neziskové organizace Oblastní charita Břeclav. Hlavní důraz při vývoji byl kladen na přiblížení se specifickým požadavkům a možnostem uživatele, který bude s aplikací pracovat. Pro analýzu požadavků a celého systému byly použity metody softwarového inženýrství. Vývoj pak probíhal pomocí nástroje Adobe Flash Builder na platformě Adobe Flex. Výsledný informační systém lze označit jako tzv. desktopovou aplikaci, která pro svůj běh využívá prostředí Adobe AIR, jehož hlavní výhodou je nezávislost na operačním systému. Lokální běh aplikace umožňuje také maximální využití hardwarového výkonu počítače a zároveň odpadá nutnost zabezpečeného přenosu dat, které by bylo potřeba zajistit v případě webové aplikace.

Klíčová slova: Nezisková organizace, informační systém, Adobe, Flex, AIR, Flash Builder

ABSTRACT

The aim of this work is to create an information system for the use of the non-profit organization called Regional Charity Břeclav. The main emphasis during the developing was placed on the approximation to specific requirements and user's abilities who will work with the application. For the requirements analysis as well as the analysis of the system itself the software engineering methods were used. The development was carried out with the help of Adobe Flash Builder based on the Adobe Flex Platform. The resulting information system can be described as a desktop application. This application uses the Adobe AIR environment for its run and the main advantage is the operating system independence. This desktop application enables the maximum utilization of computer hardware performance. Moreover, it eliminates the need for secure data transmission which would be necessary to provide in case of web application.

Keywords: non-profit organization, information system, Adobe, Flex, AIR, Flash Builder

Především bych chtěl poděkovat za vedení této práce svému vedoucímu, panu doc. Mgr. Romanu Jaškovi, Ph.D. a také pracovníci Oblastní charity Břeclav, paní Haně Jagošové, za její ochotu při spolupráci.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 ANALÝZA NEZISKOVÝCH ORGANIZACÍ.....	11
1.1 CHARITA ČR	11
1.1.1 Oblastní charita Břeclav	11
1.1.2 Vnitřní činnost.....	13
1.1.3 Veřejná činnost organizace	13
1.2 ČLOVĚK V TÍSNI.....	14
1.2.1 Vnitřní činnost.....	14
1.2.2 Veřejná činnost.....	14
1.2.3 Používaný software	15
1.3 ADRA.....	16
2 SOUČASNÝ INFORMAČNÍ SYSTÉM.....	18
2.1 VÝHODY.....	18
2.2 NEVÝHODY	18
2.3 ZÁKON	18
3 POUŽITÉ NÁSTROJE.....	19
3.1 FLEX.....	19
3.1.1 Jazyk MXML	19
3.1.2 Jazyk ActionScript	20
3.1.3 Adobe Flash Player	20
3.1.4 Adobe AIR	21
3.1.5 Adobe Flash Builder 4.....	21
3.1.6 Adobe Flex SDK	23
3.2 SQLITE	24
3.3 MYSQL WORKBENCH.....	24
3.4 ENTERPRISE ARCHITECT	24
II PRAKTICKÁ ČÁST	25
4 SOFTWAREOVÁ ANALÝZA	26
4.1 POŽADAVKY NA SYSTÉM	26
4.1.1 Nefunkční požadavky.....	26
4.1.2 Funkční požadavky	27
4.2 DIAGRAMY PŘÍPADŮ UŽITÍ.....	28
4.3 DIAGRAM TŘÍD	29
4.4 SEKVENČNÍ DIAGRAM.....	31
4.5 DIAGRAM AKTIVIT.....	33
5 UKLÁDÁNÍ DAT	35
5.1 UKLÁDÁNÍ DO SOUBORU	35
5.2 DATABÁZE	36
6 VZHLED APLIKACE.....	38

6.1	ÚVODNÍ OBRAZOVKA	39
6.2	OBRAZOVKA PRO VEDENÍ OSOB.....	40
6.2.1	Přidání osoby.....	41
6.2.2	Vymazání osoby.....	42
6.2.3	Upravení osoby	44
6.2.4	Vyhledání osoby.....	45
6.3	OBRAZOVKA PRO VEDENÍ DIPLOMŮ.....	47
6.3.1	Vytvoření diplomu	47
6.3.2	Vyhledání diplomu.....	48
7	PROGRAMOVÁ LOGIKA.....	49
7.1	INICIALIZACE APLIKACE	50
7.2	PŘEPÍNÁNÍ ZÁKLADNÍCH OBRAZOVEK	50
7.3	ZOBRAZENÍ FORMULÁŘE	51
7.4	ÚLOŽIŠTĚ DAT APLIKACE	52
7.5	PRÁCE S DATABÁZÍ.....	52
7.5.1	Transakce	53
7.5.2	Vložení dat do databáze	53
7.5.3	Editace dat v databázi.....	54
7.5.4	Vyhledání identifikátoru záznamu	54
7.5.5	Vyhledání dat	54
7.5.6	Zobrazení vyhledaných informací.....	55
7.6	PRÁCE SE SOUBORY	55
7.6.1	Zálohování dat.....	56
7.7	ROBUSTNOST SYSTÉMU	56
7.7.1	Kontrola vstupních formulářů.....	56
7.7.2	Chyby při práci s databází.....	57
7.7.3	Chyby při práci se soubory.....	58
7.8	VIZUALIZACE DAT	58
8	TESTOVÁNÍ APLIKACE	59
8.1	TESTOVÁNÍ UŽIVATELSKÝCH VSTUPŮ	59
8.2	TESTOVÁNÍ CHYB V SOBORECH.....	60
8.3	TESTOVÁNÍ INTEGRITY DATABÁZE	60
9	BEZPEČNOST APLIKACE	61
10	NASAZENÍ APLIKACE	63
	ZÁVĚR	64
	ZÁVĚR V ANGLIČTINĚ.....	65
	SEZNAM POUŽITÉ LITERATURY.....	66
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	68
	SEZNAM OBRÁZKŮ	69
	SEZNAM TABULEK.....	70
	SEZNAM KÓDŮ	71
	SEZNAM PŘÍLOH.....	72

ÚVOD

Cílem této práce je vytvořit informační systém pro potřeby Oblastní charity Břeclav. Hlavní důraz při vývoji bude kladen na specifické požadavky, které bude mít uživatel systému. Je totiž nutné vytvořit aplikaci, která bude svým uživatelským rozhraním co nejvíce přístupná myšlení, potřebám a možnostem konkrétního uživatele.

Práce se skládá ze čtyř základních bodů – analýza stávající situace v oblasti informačních systémů pro neziskové organizace a popis použitých nástrojů (Kapitoly 1, 2 a 3), analýza požadavků na systém a návrh konkrétního řešení (Kapitoly 4 a 5), realizace navrženého řešení (Kapitoly 6 a 7) a následná diskuze nad výslednou aplikací (Kapitoly 8, 9 a 10).

Analýza současného systému si klade za cíl pochopit fungování organizace jako celku, popsání zaběhlých řešení spojených s provozem a navrhnout možné vylepšení. Pro potřeby analýzy je nezbytná spolupráce s pracovníky organizace. Protože organizace spolupracuje s lidmi, o kterých si uchovává soukromé informace, je nutné seznámit se také s platnými zákony o ochraně osobních údajů. Pro širší pohled na fungování neziskových organizací bude použito informací nejen od Oblastní charity Břeclav, ale také od neziskových organizací ADRA a Člověk v tísni.

Návrh systému je nejdůležitější částí celé práce. V předchozím kroku je nutné správně popsat existující nedostatky a přesně formulovat jejich řešení. Na základě těchto informací bude následně prováděna realizace. Při návrhu je vhodné použít metody softwarového inženýrství, jako jsou – modely funkčních a nefunkčních požadavků, modely případu užití, návrhová schémata databází a návrh testů pro výslednou kontrolu.

Třetím bodem je realizace navržené aplikace, postup vývoje, popis implementace logiky a vzhledu. Aplikace bude vytvářena ve vývojovém prostředí Flash Builder 4 od společnosti Adobe. Toto prostředí je voleno záměrně, kvůli předchozím zkušenostem.

Závěrečnou částí práce je provést diskuzi nad výsledky, které výsledná aplikace přinesla. Budou popsány provedené testy a jejich výsledky, možné chyby a jejich řešení. Popsány budou také vzniklé výhody a nevýhody, spolu s návrhem možného dalšího rozšíření aplikace.

I. TEORETICKÁ ČÁST

1 ANALÝZA NEZISKOVÝCH ORGANIZACÍ

Pro správný informační systém je velmi důležité dobře vystihnout a popsat, co se od systému očekává. Je nutné vytvořit komunikační kanál mezi uživatelem systému a jeho tvůrcem. V praxi se převážně uživatel a tvůrce vůbec nepotkají, ale jejich prostředníkem jsou právě nástroje analýzy. Hlavním problémem je správně pochopit, co uživatel od systému očekává, tyto očekávání vhodně rozdělit podle důležitosti a případně doplnit o speciální požadavky, o kterých nemusí mít uživatel povědomí.

Primární snahou analýzy je zjistit potřeby a nároky na vyvíjený software. Při zjišťování těchto potřeb je nezbytné pochopit fungování a členění celé organizace. Z tohoto důvodu je v následující kapitole nejvíce rozebírána organizace Charita ČR, pro jejíž Oblastní charitu Břeclav je informační systém vyvíjen. Zároveň jsou zde také uvedeny informace o používání informačních systémů organizacemi ADRA a Člověk v tísni.

1.1 Charita ČR

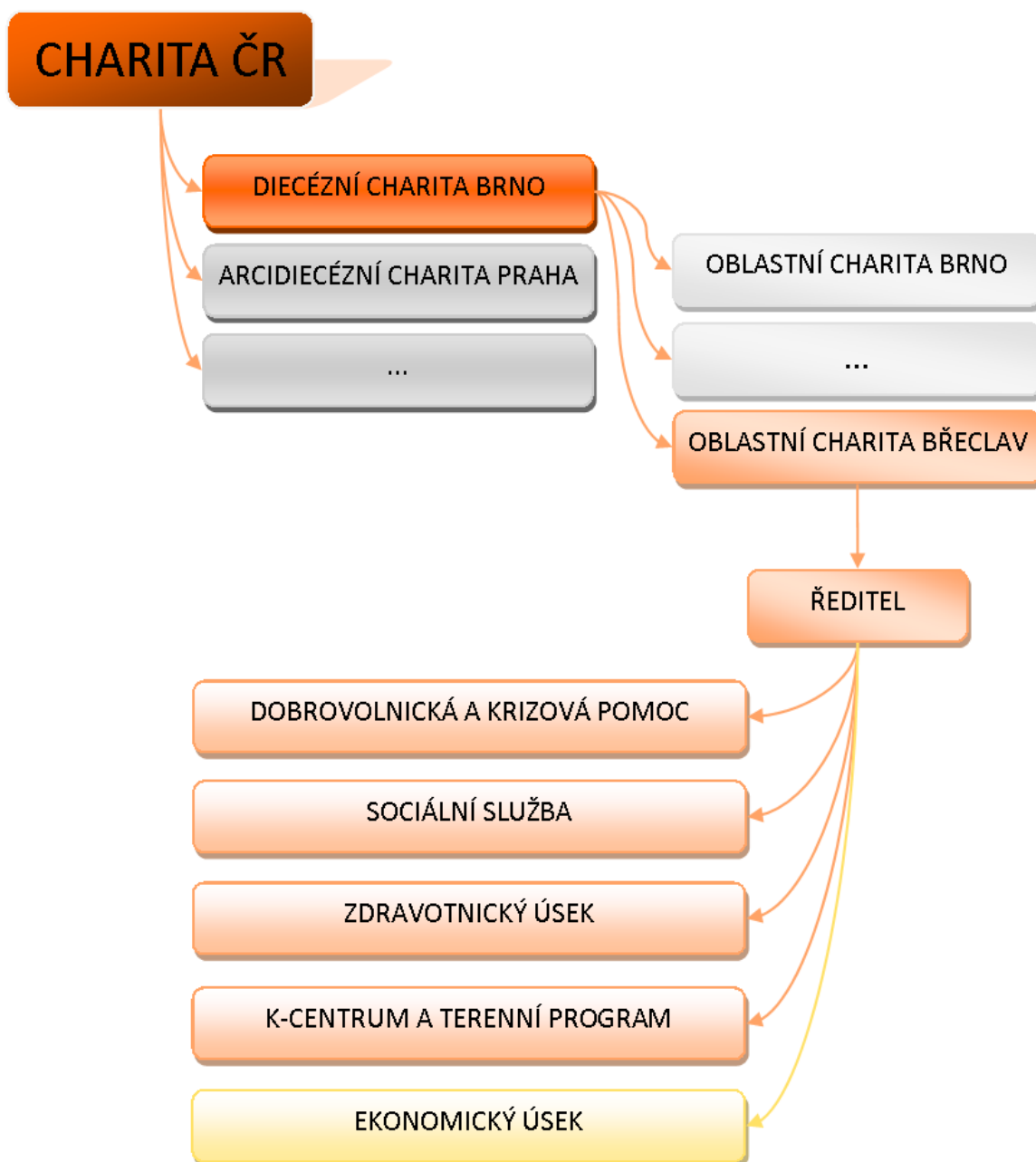
Charita ČR je součástí římskokatolické církve. Tomu také odpovídá struktura územního členění. Každá z arcidiecézní a diecézní charity je členem Charity ČR a jedná se o nejvyšší územní členění. Dalším článkem jsou oblastní a městské charity. Ty jsou podřízeny dle územního členění příslušné arcidiecézní nebo diecézní charitě.

Tento systém má tedy tři úrovně, kde každá organizační jednotka má jiné priority podle potřeb v daném území. V mé práci se dále zaměřuji pouze na potřeby Oblastní charity Břeclav. Dá se ovšem předpokládat, že výsledný software by mohl být používán i jinými oblastními charitami, protože základní systém členění je podobný, pouze s rozdílným důrazem na jednotlivé složky.

1.1.1 Oblastní charita Břeclav

Oblastní charita Břeclav se dělí do pěti částí (Obrázek 1), které jsou podřízeny jednomu řediteli. Z hlediska vykonávaných činností můžeme jednotky rozdělit na ty, které jsou určeny pro veřejnost a ty, které jsou nutné pro vnitřní činnost organizace. Pro službu veřejnosti je určena jednotka dobrovolnické a krizové pomoci (má za cíl pomáhat při nenadálých událostech jako jsou povodně, hromadné nehody a požáry), dále pak jednotka sociálních služeb (návštěvy nemocných, nákupy, pečovatelská služba), jednotka zdravotnického úseku (zdravotní sestry vykonávají ošetření předepsaná lékařem) a

jednotka K-centra a terénního programu (pomáhá lidem s drogovou závislostí). Pro vnitřní činnost organizace je pak určena jednotka ekonomického úseku, která má za úkol vytvářet zázemí pro činnost celé organizace (vede účetnictví, výplaty a evidenci majetku). Pro navrhování informačního systému je nutné pochopit fungování jednotlivých částí organizace, jejich potřeby a stávající systém.



Obrázek 1: Členění Charity ČR

1.1.2 Vnitřní činnost

Oblastní charita Břeclav je nezisková organizace, která musí vést své účetnictví, informace o majetku a jiná data pro svůj vlastní chod. Tuto činnost má na starosti ekonomický úsek a využívá k tomu informační systém Vema. Tento software byl vybrán na základě výběrového řízení, mimo jiné také díky nejlevnější ceně. Podle informací zaměstnanců je tento program funkční, bezproblémový a jeho obsluhu se snadno a rychle naučili. Z těchto důvodů tedy nemá smysl snažit se předělávat něco, co funguje a zaměstnanci jsou s tím spokojeni.

1.1.3 Veřejná činnost organizace

Většina činností Oblastní charity Břeclav je přirozeně směřována k potřebným lidem, tedy k široké veřejnosti. Na obrázku (Obrázek 1) jsou úseky poskytující veřejné služby vyznačeny světle oranžovou barvou. Každý z těchto úseků vykonává tyto aktivity:

- Dobrovolnická a krizová pomoc
 - Farní akce
 - Tříkrálová sbírka
 - Jednorázové akce
 - Krizové akce
- Sociální služby
 - Sociální projekty
- Zdravotnický úsek
 - Zdravotnické projekty
- K-centrum a terénní program
 - Podpora drogově závislých

Výše popsané aktivity, lze rozdělit do dvou skupin. První skupina obsahuje taková akce, které nelze vykonávat bez pomoci dobrovolníků, kdy dobrovolníkem může být kdokoliv z široké veřejnosti. Druhá skupina pak dělá takové činnosti, které naopak potřebují kvalitní zaměstnance, případně proškolený personál. Pro správné fungování první skupiny je nezbytná spolupráce zaměstnanců s dobrovolníky, které lze dělit do těchto skupin:

- Pomocní dobrovolníci
 - jednorázová spolupráce
- Projektoví dobrovolníci
 - spolupráce na vybraném projektu
- Dobrovolníci farních charit
 - spolupráce na úrovni farních celků
- Team Morava
 - databáze lidí ochotných k různému druhu pomoci

1.2 Člověk v tísní

Organizační struktura obecně prospěšné společnosti Člověk v tísní je velmi odlišná od výše popsané struktury organizace Charita ČR. Přirozeně společný je ale systém, kdy některá oddělení slouží veřejnosti a jiná mají za úkol zabezpečovat vlastní fungování společnosti.

1.2.1 Vnitřní činnost

Chod společnosti zabezpečují následující vnitřní oddělení:

- Finanční oddělení
- Právní oddělení
- Mediální oddělení
- Provozní oddělení

Činnost těchto oddělení je průřezová celou společností. Jejich cílem je plnit požadavky oddělení, která se orientují na poskytování pomoci veřejnosti. Systém funguje tak, že vnitřní oddělení pracují na zakázkách pro vnější oddělení.

1.2.2 Veřejná činnost

Veřejná pomoc lidem se dá rozdělit do čtyři hlavních oddělení:

- Humanitární a rozvojová pomoc
- Podpora lidských práv
- Programy sociální integrace
- Informační a vzdělávací programy

- Filmový festival Jeden svět
- Jeden svět na školách
- Varianty
- Migrace

První dvě výše popsaná oddělení se dále člení podle zemí, ve kterých je jejich činnost vykonávána (např. Podpora lidských práv se dělí na sekci v Barmě, v Kubě, v Bělorusku a další). Druhá dvě oddělení vykonávají svoji činnost na území ČR. První tři oddělení jsou konečné organizační jednotky, zatímco oddělení Informační a vzdělávací programy se ještě dále dělí na čtyři pododdělení, jak ukazuje seznam.

1.2.3 Používaný software

Obecně prospěšná společnost Člověk v tísni používá pro potřeby svého chodu informační systém Navision – Microsoft Business Solutions. Jedná se o program ze skupiny ERP softwarů (Enterprise Resource Planning = Plánování podnikových zdrojů). Aplikace shromažďuje, poskytuje a analyzuje veškeré důležité informace týkající se organizace. Primárně je systém Navision použit pro vedení účetnictví, sledování ekonomiky organizace a nedávno byl také nově zaimplementován modul pro správu kontaktů.

Informační systém Navision je provozován ve dvou variantách - s trvalým a s občasným připojením k internetu. Každá mise, kterou zaštiťuje organizace Člověk v tísni, má vytvořenou vlastní databázi informací, k níž se připojuje. Všechny takto vytvořené databáze jsou pak podmnožinou centrální databáze, která je spravována vedením organizace v Praze.

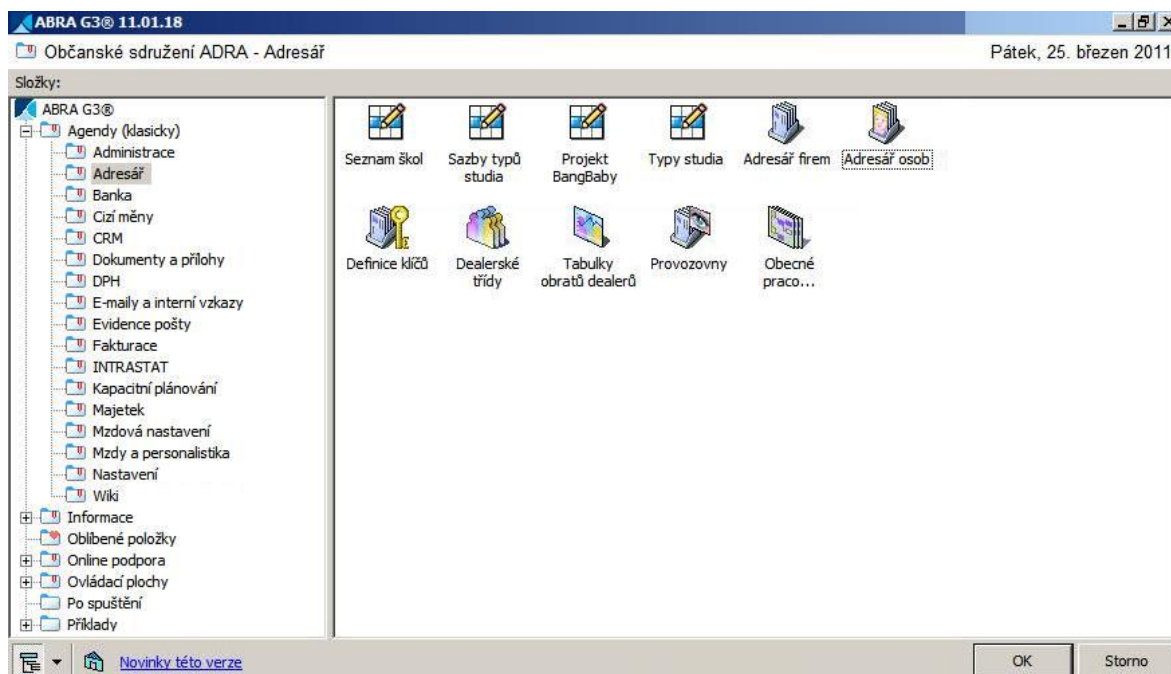
Pokud se jedná o databázi, která je v oblasti s trvalým připojením k internetu, pak se provádí aktualizace upravovaných dat průběžně při běhu aplikace. V případě, kdy je aplikace bez stálého připojení, dochází k lokálnímu ukládání dat a synchronizace je provedena, až je připojení dostupné. Verze bez připojení je používána v Afganistánu, Angole, Etiopii, Arménii, na Haiti a na Srí Lance. Naopak verze s dobrým připojením jsou používány v Iráku, Gruzii a Pákistánu.

V roce 2011 probíhá implementace softwaru ELO Digital Office, který slouží ke správě dokumentů. Cílem zavádění této novinky je zjednodušení práce s množstvím papírových informací, jejichž potřebná archivace znesnadňuje jejich vyhledávání a práci s nimi.

Každé oddělení používá mimo společných, centrálních informačních systémů i své vlastní nástroje, které jsou přizpůsobeny jejich činnosti. Jako příklad lze uvést sekci filmového festivalu „Jeden svět“, která používá informační systém DATAKAL.

1.3 ADRA

Poslední popisovanou organizací je občanské sdružení ADRA. Pro tuto organizaci je nejdůležitější ekonomické oddělení, které využívá software ABRA (Obrázek 2). Pomocí této aplikace jsou ukládány informace o vedení majetku, pokladen a dárců. Jinak řečen, tento systém slouží k vedení veškerých účetních operací a je tak nejvíce využívaným nástrojem celého ekonomického oddělení.



Obrázek 2: Informační systém ABRA používaný v organizaci ADRA

Druhým používaným softwarovým nástrojem je program Intuo (Obrázek 3), který využívají oddělení zahraničních projektů a oddělení fundraisingu a PR. Pomocí tohoto programu jsou uchovávány informace o spolupracujících a partnerských organizacích, o zájemcích o spolupráci, dobrovolnících, zaměstnancích a jejich dovolených. Dále se zde ukládají informace o domácích a zahraničních projektech organizace – kdo na projektu pracuje, rozpis využívání finančních zdrojů a různé dokumenty související s daným projektem.

The screenshot displays the 'Intuo - Projekty' application window. The main area shows a table of projects with columns for country, project name, ID, status, and financial data. Below the table, the details for the selected project '1.Porodnice pro Itibo' are shown, including its manager (Knaibl), category (INT-Rozvojový), and various administrative fields like start/end dates and status.

Země	Název projektu	01 Číslo projektu	Stav	01 ADRA ČR (CZK)	02 Celkem (CZK)	08 Nepřímé náklady ADRA ČR (CZK)
Keňa	1.Porodnice pro Itibo	07-CZ-INT-003	Ukončen	2499868	2959868	
Keňa	2.Stomatologie pro Itibo	07-CZ-INT-013	Ukončen	1047000	1047000	
Keňa	3.Výstavba lůžkové části pr...	09-CZ-INT-018	Ukončen	2663032	2663032	
Česká republika	365+ aneb mince denně	09-CZ-CZ-005	Probíhá			
Bosna a Hercegovina	Adopce v balíku	03-CZ-INT-001	Ukončen			
USA	ADRA pomáhá hurikánovým...	05-CZ-INT-013	Ukončen	3018000		
Indonésie	ADRA pro Sumatru	05-CZ-INT-001	Ukončen	52000000	97907000	
Česká republika	ADRA Úl	09-CZ-CZ-014	-----			
Česká republika	Aktivity Adráků	09-CZ-CZ-016	Probíhá			
Bangladéš	BangBaby – Sponzorování d...	99-CZ-INT-001	Probíhá			
Zambie	Bezpečné mateřství	08-CZ-INT-009	Zamítнутý	3999000	3999000	238785
Česká republika	Budování kapacit NNO v ČR...	08-CZ-VZD-006	Ukončen	170120	16903500	
Srí Lanka	Bydlení a vodní zdroje pro o...	10-CZ-INT-006	Probíhá		0	11985
Česká republika	Cena Michala Velíška	09-CZ-CZ-015	-----			
Srbsko	Centrum pro oběti domácích...	06-CZ-INT-010	Ukončen	7307560	7307560	272764
Moldavsko	Cesta k předškolnímu inkluz...	10-CZ-INT-024	Probíhá	6974368	6974368	109330
Indonésie	Čas na školu v Sulawesii, In...	10-CZ-INT-013	Zamítнутý			
Česká republika	Česko proti chudobě	05-CZ-VZD-001	Ukončen			
Česká republika	ČpCh-Cesty k napliňování R...	08-CZ-VZD-003	Ukončen	36000	2149000	
Česká republika	ČpCh-Cesty k napliňování ro...	07-CZ-VZD-002	Ukončen			
Česká republika	ČpCh-Ještě máme šanci přis...	10-CZ-VZD-05	Probíhá	36000	741622	
Česká republika	ČpCh-Ještě máme šanci přis...	09-CZ-INT-004	Schválený			
Ukrajina	Dětem z dětského domova...	09-CZ-INT-004	-----		115600	
Rumunsko	Distribuce sadbových bram...	07-CZ-INT-017	Ukončen	142637	142637	
Ukrajina	Dobrovolníci pro dětský do...	11-CZ-INT-002	Probíhá		120000	

1.Porodnice pro Itibo

Manažer: Knaibl Kategorie: INT-Rozvojový Složka projektu: 07-CZ-INT-003

Obecné Parametry Organizace Osoby Aktivita Přístup Soubory Úkoly Časová osa Dary

Název projektu: 1.Porodnice pro Itibo Kategorie: INT-Rozvojový

Primární org.: ADRA o.s. Typ fakturace: Fakturovat mimo CeMF

Země: Keňa Důležitost: Střední

Desk officer: Knaibl Tomáš Aktivní: Stav: Ukončen

Asistent desk officer: Zahájení pl.: 1. 6. 2006 Zahájení sk.: 1. 6. 2006

Šablona: Ukončení pl.: Ukončení sk.: 30. 4. 2007

Poznámka:
Od roku 2006 ADRA financuje (především z veřejných sbírek), rekonstrukci zdravotnického střediska na východě Keni, v obci Itibo. K rekonstruované a vybavené ambulanci (ordinace a využití střediska jako vzdělávacího zařízení pro místní a české lékaře a zdravotníky.

Obrázek 3: Informační systém Intuo používaný v organizaci ADRA

2 SOUČASNÝ INFORMAČNÍ SYSTÉM

Jak již bylo zmíněno (Kapitola 1.1.2), informační systém pro potřeby chodu organizace je funkční a proto není předmětem této práce.

Ostatní zaměstnanci ale již nemají možnost pracovat s informacemi pomocí moderních informačních systémů a využívají převážně papírovou podobu informačního systému, případně si snaží práci usnadnit digitalizací ve formě jednoduché tabulky v osobním počítači.

2.1 Výhody

Současný stav má výhodu v tom, že neklade prakticky žádné nároky na uživatele v oblasti zkušenosti s prací na počítači. Pracovat s papírovou formou informačního systému je velmi jednoduché a především pro starší pracovníky se tak jedná o vhodnou formu správy dat.

2.2 Nevýhody

Vyhledávání v současném systému je velmi složité. Není také možné vést nějaké pomocné statistiky nebo zobrazovat data podle různých kritérií. Dále tento systém klade vysoké časové nároky na zaměstnance.

2.3 Zákon

Ať už je informační systém v jakékoliv podobě, je nutné dodržovat platné zákony. Ukládané informace o spolupracujících dobrovolnících mají charakter osobních údajů, protože z nich lze zjistit identita osoby. Při jejich spravování je tedy nutné postupovat dle zákona č. 101/2000 Sb., který upravuje nakládání s těmito údaji.

3 POUŽITÉ NÁSTROJE

Hlavním vývojovým nástrojem pro vytváření informačního systému bude vývojová platforma Flex od společnosti Adobe. Ta poskytne základní oporu při tvorbě grafického uživatelského rozhraní a při vývoji logiky informačního systému. Tento vývojový nástroj jsem zvolil proto, že jsem s ním již pracoval a je to nástroj určený pro vytváření aplikací se zajímavým grafickým zobrazením. Pro ukládání dat bude použita databáze v prostředí MySQL. Formát XML pak poslouží jako nosič dat mezi databází a grafickým uživatelským rozhraním.

3.1 Flex

Framework Flex je bezplatným prostředím pro tvorbu graficky bohatých aplikací od společnosti Adobe. Výhodou tohoto prostředí je otevřenost zdrojového kódu. Díky tomu je možné prohlížet si vytvořené zdrojové kódy a modifikovat si existující části kódu podle vlastní potřeby každého programátora. Pro získání zdrojových kódů lze např. použít stránky společnosti Adobe.(1)

Vývoj aplikací je možné provádět v bezplatném prostředí Adobe Flex SDK nebo využít výkonného placeného programátorského prostředí Adobe Flash Builder. Vytvářené aplikace lze rozdělit podle způsobu provozování na webové (běží v Adobe Flash Player) a desktopové (běží v Adobe AIR).

Pro vytváření aplikací slouží programovací jazyky MXML a ActionScript. Každý z jazyků je primárně určen k jinému cíli, ale jejich použití se může překrývat.

3.1.1 Jazyk MXML

Jedná se o jazyk, který je implicitně určen pro tvorbu grafického uživatelského rozhraní aplikace. Může být ale také použit pro vytváření některých částí logiky programu. Např. můžeme vytvořit komponentu, která není v aplikaci vidět a slouží k zajištění přístupu aplikace ke vzdáleným datům nebo přístupu do databáze.

Základem pro jazyk MXML je, jak již z názvu vyplývá, jazyk XML. Jedná se tedy o značkovací jazyk vytvořený firmou Macromedia. První písmeno může podle některých názorů ukazovat právě na společnost Macromedia nebo na slovo „Magic“.(2)

3.1.2 Jazyk ActionScript

Je objektově orientovaný programovací jazyk, který je v prostředí Flex primárně určen pro vytváření logiky aplikace, obdobně však jako u jazyka MXML, lze jazyk ActionScript využít pro vytváření vzhledu aplikace. Tento jazyk vychází ze standartu ECMAScript.

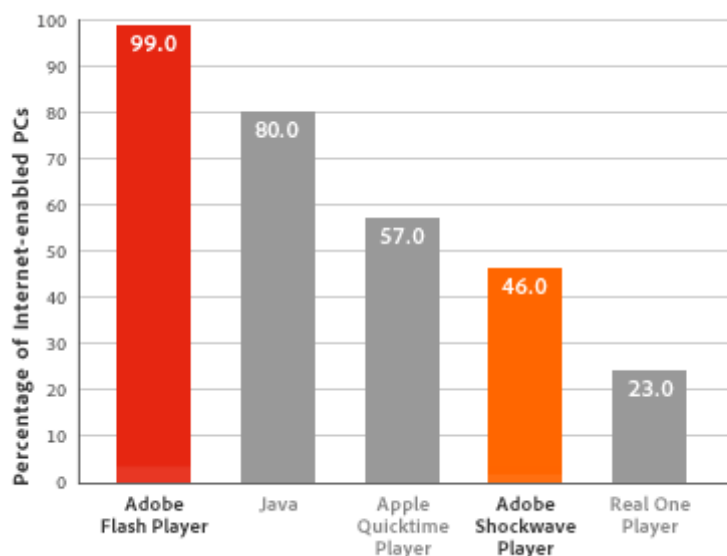
Zdrojový kód je pomocí kompilátoru převáděn do binárního kódu, který je následně vložen do výsledného SWF souboru. Takto vytvořený soubor je pak zobrazován a prováděn pomocí aplikace ActionScript Virtual Machine, která je součástí programů Adobe Flash Player a Adobe AIR.

3.1.3 Adobe Flash Player

Jedná se o běhové prostředí pro aplikace, které jsou spouštěny ve webovém prohlížeči. Tento doplněk webového prohlížeče zaručuje, že vytvořená aplikace se bude zobrazovat na všech hlavních prohlížečích stejně. Prohlížeče mohou obsah někdy zobrazovat různě a nastává tak problém s optimalizací vzhledu pro každý z hlavních prohlížečů. Vytvořená aplikace je tak zcela nezávislá na tom, jaký prohlížeč použije uživatel k jejímu zobrazení. Jako překážka se může zdát nutnost nainstalování doplňku Adobe Flash Player do používaného prohlížeče. Tato nevýhoda je ale velmi zmenšena obecnou rozšířeností tohoto doplňku pro různé prohlížeče. Společnost Adobe uvádí, že 99% počítačů, které jsou připojeny k internetu, mají tento doplněk nainstalovaný.



Obrázek 4: Ikona programu Adobe Flash Player. (4)



Obrázek 5: Rozšířenost softwarových nástrojů na počítačích připojených k internetu. (3)

3.1.4 Adobe AIR

Tento program je obdobou programu Adobe Flash Player. Jedná se o běhové prostředí, které slouží k zobrazování lokálních aplikací vytvořených na platformě Flex. Výhodou tohoto softwaru je jeho nezávislost na operačním systému. Tato aplikace tak zaručuje stejný vzhled a funkcionalitu na různých platformách a tím dělá vytvořenou aplikaci multiplatformní. Zajímavá je také podpora jiných zařízení, než jsou pouze počítače. Výsledným efektem je, že jedna aplikace může běžet na počítači, stejně tak jako na mobilním telefonu.



Obrázek 6: Ikona programu Adobe AIR. (5)

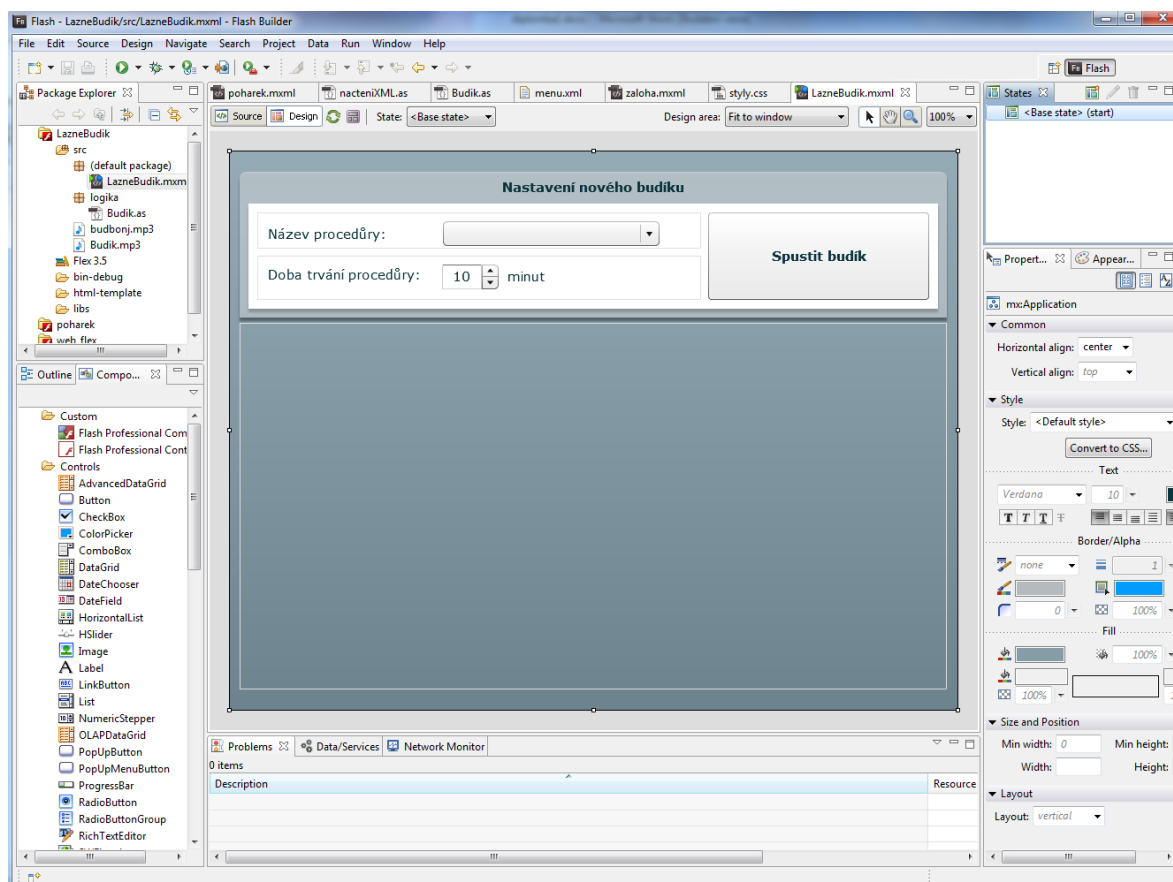
3.1.5 Adobe Flash Builder 4

Jedná se o placený softwarový nástroj, jehož cílem je usnadnit vývoj aplikací na platformě Flex. Aplikace Flash Builder poskytuje vývojáři kvalitní podporu při tvorbě aplikací – např. nabízí dokončování psaných příkazů a automatické vytváření standartních kódů.

Vývojové prostředí (Obrázek 7) se skládá ze dvou základních částí – grafické prostředí pro návrh vzhledu (režim *Design*) a prostředí pro psaní zdrojového kódu (režim *Source*).

Vytváření vzhledu aplikace je velmi jednoduché právě díky podpoře grafického prostředí, ve kterém stačí jednotlivé komponenty výsledné aplikace naskládat na zobrazovanou plochu. Vkládání těchto komponent je také možné pomocí textového prostředí, ale jedná se o méně efektivní způsob práce. Implementace samotné logiky aplikace je pak prováděna výhradně v prostředí pro psaní zdrojového kódu.

Prostředí zobrazuje vývojáři přehled o struktuře vytvářených projektů pomocí oblasti *Package Explorer*, dále pak zobrazuje schéma vytvořeného zdrojového kódu v oblasti *Outline* – a to buď z pohledu jazyka MXML nebo jazyka ActionScript a samozřejmě je oblast pro vypisování chyb a varování. Tyto zobrazované oblasti jsou stejné pro režimy zobrazení *Design* a *Source*. V režimu *Design* je navíc zobrazována záložka se seznamem existujících komponent a oblast obsahující vlastnosti vybrané komponenty z grafického uživatelského rozhraní.



Obrázek 7: Vývojové prostředí Flash Builder 4

Pro správné fungování vývojového prostředí požaduje společnost Adobe následující minimální požadavky: (6)

- Procesor 2 GHz nebo rychlejší
- Microsoft® Windows® XP s aktualizací ServicePack 3, Windows Vista® Ultimate nebo Enterprise (32 nebo 64bitová verze spuštěná v 32bitovém režimu), Windows Server® 2008 (32bitová verze) nebo Windows 7 (32 nebo 64bitová verze spuštěná v 32bitovém režimu)
- 1 GB paměti RAM (doporučují se 2 GB)
- 1 GB volného místa na pevném disku
- Prostředí Java™ VirtualMachine (32bitová verze): IBM® JRE 1.5, Sun™ JRE 1.5, IBM JRE 1.6 nebo Sun JRE 1.6
- Monitor s rozlišením 1 024 x 768 (doporučuje se 1 280 x 800) s grafickou kartou s hloubkou barev 16 bitů
- Jednotka DVD-ROM
- Eclipse 3.4.2 oder 3.5 (pro instalaci Plug-ins)



Obrázek 8: Ikona programu Adobe Flash Builder. (7)

3.1.6 Adobe Flex SDK

Tento nástroj je bezplatnou alternativou k produktu Adobe Flash Builder 4. Obsahuje knihovny existujících komponent a kompilátor Flex. Samotný vývoj aplikace je pak prováděn např. pomocí textového editoru. Tato varianta neumožňuje vytvářet grafické rozhraní aplikace pomocí vizuálního návrhu.



Obrázek 9: Ikona vývojového prostředí Flex SDK. (8)

3.2 SQLite

Pro ukládání dat byl použit databázový systém SQLite. Původní představa použití databáze MySQL, nad kterou by běželo rozhraní v podobě PHP skriptů, bylo vyhodnoceno jako zbytečně složité. Bylo by totiž nutné provádět dotazy v prostředí Flex, dále dle typů dotazů volat správné PHP skripty a následně přistoupit k datům v databázi. Po získání dat by se musela data opět zpracovat v prostředí PHP a poslat zpět pro zobrazení do Flex aplikace. Tento celý proces je do jisté míry tím nesmyslnější, pokud aplikace běží pouze lokálně a není tedy nutné mít databázi umístěnou na síti internet. Po zvážení všech těchto okolností se jeví kombinace MySQL a PHP jako neefektivní. Po zjištění, že programové prostředí Flex Builder má přímo implementovanou knihovnu SQLite pro správu dat, využití tohoto řešení zvítězilo.

Knihovna SQLite vytváří datový soubor s příponou db, ve kterém jsou obsaženy všechny databázové tabulky, které jsou součástí jedné databáze. Takto vytvořený soubor je nezávislý na souborovém systému operačního programu, což spolu se stejnou platformní nezávislostí běhového prostředí Adobe AIR vytváří multiplatformní aplikaci.

3.3 MySQL Workbench

Jedná se o modelovací nástroj pro MySQL, který umožňuje modelovat databáze, vytvářet SQL dotazy a spravovat databáze. Pro potřeby této práce byla využita modelovací část tohoto programu, pomocí níž byl vytvořen návrh databázové struktury, která slouží pro ukládání dat.

3.4 Enterprise Architect

Tento program je využíván při softwarové analýze vytvářeného programu. Pro grafické znázornění zachycených požadavků na systém používá jazyk UML. Pomocí tohoto jazyka byly vytvořeny diagramy zachycující požadavky na systém, diagramy případů užití, diagramy tříd, sekvenční diagramy a diagramy aktivit.

II. PRAKTICKÁ ČÁST

4 SOFTWAREVÁ ANALÝZA

Pomocí nástrojů softwarového inženýrství byla provedena analýza požadavků na systém. Požadavky byly sbírány rozhovorem s pracovníci Oblastní charity Břeclav a pozorováním její práce se současnou podobou uložených informací. Na základě těchto získaných informací byly vytvořeny formální požadavky na systém a dohodnuta obecná představa o funkčnosti výsledné aplikace.

Ze získaných informací se dále vytvářely diagramy případů užití, které popisují práci uživatele se systémem. V těchto diagramech jsou zachyceny všechny funkční vlastnosti systému, které uživatel požaduje.

Diagramy případu užití pak sloužily jako základ pro vytváření tříd, jejichž implementaci se vytvářel cílový program. Diagramy tříd byly do velké míry nahrazeny možností vývojové technologie Flex, díky níž je možno vytvářet komponenty, které mohou definovat jak svůj vzhled, tak také vlastní funkčnost.

Další nástroje softwarového inženýrství, jako jsou aktivitní a sekvenční diagramy, byly použity pouze částečně pro grafické znázornění některých uživatelských aktivit vykonávaných s budoucí aplikací.

Celá softwarová analýza je rozsáhlý systém grafů, popisující celý vytvářený systém. Různé části tohoto systému se často opakují, jen s minimálními rozdíly a jejich zobrazení by neúměrně zvětšilo rozsah této práce, proto jsou zde zobrazeny grafy pouze pro část *Vedení databáze osob* navrhovaného systému.

4.1 Požadavky na systém

Požadavky, které má aplikace plnit, můžeme rozdělit do dvou skupin. První skupinu tvoří tzv. funkční požadavky. Jak již název napovídá, jedná se o požadavky na funkčnost aplikace. Patří sem tedy popis činností, které je možné pomocí aplikace dělat. Druhá skupina se pak nazývá nefunkční požadavky, kam patří vše, co nesouvisí přímo s během aplikace a její činností.

4.1.1 Nefunkční požadavky

Informační systém poběží na počítači pracovnice Oblastní charity Břeclav a pro svůj běh nepotřebuje připojení k síti internet, ani žádné jiné lokální síti. Již z této úvodní informace lze vytvořit některé nefunkční požadavky. Aplikace nemusí být připojena na internet, což

vytváří možnost plného využití výpočetního výkonu počítače. Pokud by totiž aplikace běžela v prostředí internetu, kladlo by to jisté omezení v podobě rychlostních limitů přenášovaných dat a samotného zobrazování v prohlížeči.

Za druhý požadavek můžeme považovat informaci, že informační systém bude jednouživatelský – bude s ním pracovat jedna konkrétní pracovnice. I v případě rozšíření systému na více poboček, bude na pobočce vždy pouze jedna pracovnice, která bude se systémem pracovat.

Na počítači, kde má aplikace běžet, je používán operační systém Windows XP. Vzhledem k jisté zastaralosti toho systému se dá očekávat v příštích letech přechod na nový operační systém. Z tohoto důvodu je lepší vytvářenou aplikaci navrhovat jako multiplatformní. Zde lze využít výhody běhového prostředí Adobe AIR, které je nezávislé na operačním systému.

4.1.2 Funkční požadavky

Jsou rozděleny do pěti základních skupin podle toho, s jakými informacemi provádějí určité operace. Aplikace má ukládat data o osobách, akcích, věcech k propůjčení potřebným (např. vysoušeče a spací pytle) a vytvářet pozvánky na akce a poděkování lidem za účast na akcích.

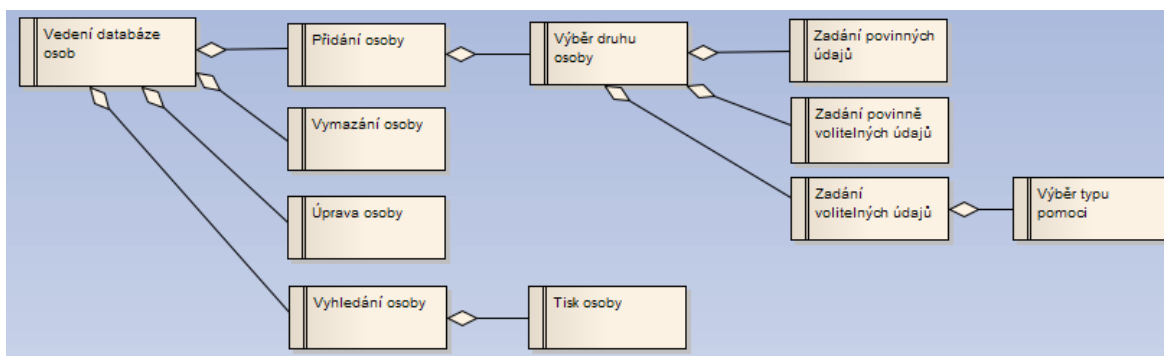
U všech skupin jsou stejné základní požadavky. Jedná se o správu dat, tedy o vytváření nových záznamů a vyhledávání, upravování a mazání již existujících. Mimo správu dat byl také vznesen požadavek na tisk pozvánek a poděkování. Samotná funkce jednotlivých aktivit je pak závislá na typu a množství dat, s nimiž pracuje.

Hlavní dění organizace je soustředěno kolem akcí, které pořádá a pro tyto akce potřebuje dobrovolníky. Společnou vlastností dobrovolníků a akcí je typ pomoci, který je na jedné straně dobrovolníky poskytován a na straně druhé akcí vyžadován. Po konzultaci s pracovníci Oblastní charity Břeclav byly typy pomoci rozděleny do těchto kategorií:

- Manuální a likvidační práce
- Logistická pomoc
- Psychosociální pomoc
- Humanitární pomoc
- Duchovní pomoc
- Krizová pomoc

- Základní zdravotnická pomoc

Pro ilustraci zjištěných požadavků slouží následující obrázek (Obrázek 10), který zachycuje funkční požadavky na vedení databáze osob. Je zřejmé, že nejsložitější akcí je přidání nového záznamu do databáze. Ukládané informace jsou rozděleny do tří kategorií – povinné údaje, povinně volitelné a volitelné. U povinně volitelných je nutné zadat alespoň jeden z možných údajů. V případě volitelných údajů je pak možnost vybrat z předem definovaných typů pomoci, které umí osoba poskytnout.

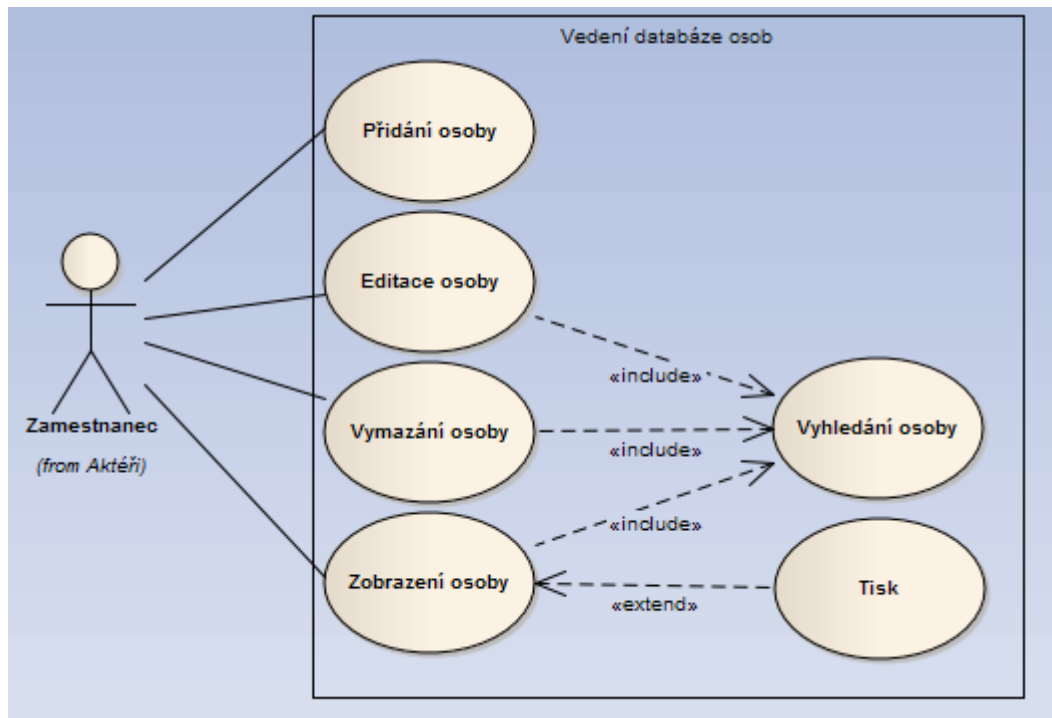


Obrázek 10: Funkční požadavky na vedení databáze osob

4.2 Diagramy případů užití

Popisy jednotlivých požadavků na systém umožnily vznik diagramů popisujících případy užití. V těchto diagramech je znázorněno, jak uživatel vykonává jednotlivé činnosti, které spadají do určité skupiny. Protože diagramy případů užití vychází z požadavků, tak jsou tyto diagramy také rozděleny pěti skupin.

Na obrázku (Obrázek 11) jsou znázorněny případy užití popisující vedení databáze osob. Každý případ užití obsahuje detailní popis toho, co uživatel a systém dělají. Dále jsou zde znázorněny dvě vazby. Vazba typu *include* znázorňuje, že případ užití obsahuje jiný případ užití. V tomto případě *Editace osoby*, *Vymazání osoby* i *Zobrazení osoby* obsahuje *Vyhledání osoby*. Naopak vazba typu *extend* rozšiřuje funkcionalitu případu užití *Zobrazení osoby*, kdy přidává případ užití s názvem *Tisk*. Zatímco vazba *include* je vykonána vždy, vazba *extend* se nemusí vykonat.



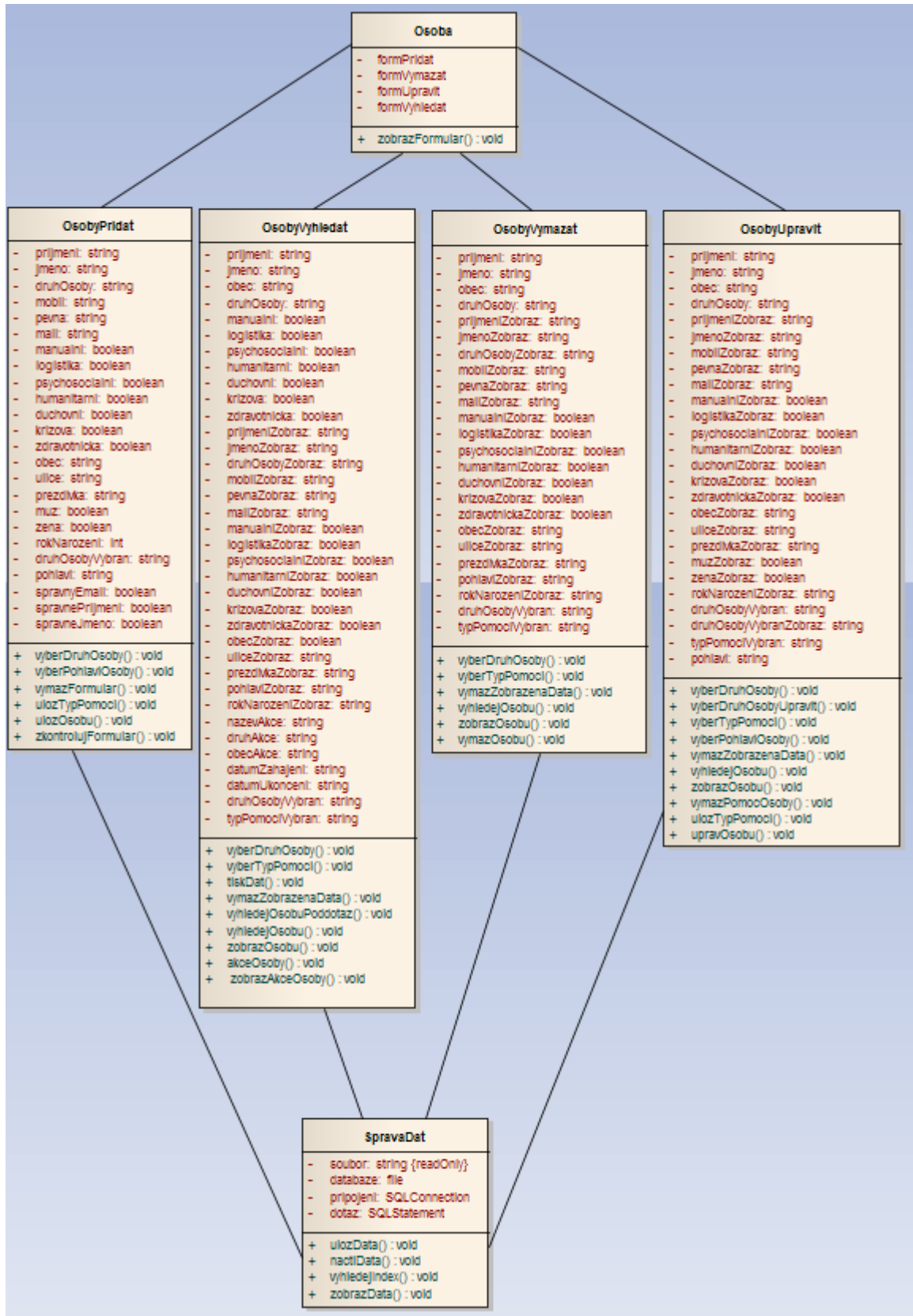
Obrázek 11: Případy užití pro vedení databáze osob

4.3 Diagram tříd

Úkolem diagramů tříd je více přiblížit návrhovou fázi samotnému programování. V těchto diagramech jsou tedy zobrazeny návrhy jednotlivých tříd spolu s jejich atributy a operacemi.

Hlavním úkolem třídy je vytvořit vzor, ze kterého se pak odvozují jednotlivé instance, které vykonávají požadovanou funkcionalitu. Při vytváření tříd ale bylo využito možností vývojového prostředí Adobe Flex, které umožňuje použít při programování dva jazyky. V jazyku MXML je vytvářeno grafické uživatelské rozhraní pomocí komponent, které v sobě obsahují i obslužné funkce potřebné pro práci s informacemi od uživatele. Samotná třída má pak funkci výběru požadované komponenty a její zobrazení uživateli.

Ilustraci diagramu tříd pro práci s osobou poskytuje následující obrázek (Obrázek 12). Můžeme na něm vidět třídu *Osoba* a třídu *SpravaDat* a dále pak komponenty *OsobyPridat*, *OsobyVyhledat*, *OsobyVymazat* a *OsobyUpravit*. Z diagramu jsou patrné vztahy, které se promítnou také do vývoje. Třída *SpravaDat* je centrální třídou celého systému a poskytuje funkce pro práci s databází. Třída *Osoba* je pak jakýmsi rozcestníkem, pro zobrazení správné komponenty. Každá komponenta pak slouží k zajištění požadované funkcionality.



Obrázek 12: Diagram tříd pro část aplikace týkající se správy osob.

4.4 Sekvenční diagram

Pomocí sekvenčních diagramů je popisován časový průběh vybrané akce. Jako ukázka byl vybrán průběh přidání nové osoby do databáze.

Na diagramu (Obrázek 13) je jeden uživatel (*Zaměstnanec*), dvě třídy (*Osoba* a *SpravaDat*) a jedna komponenta (*OsobyPridat*). Pokud uživatel chce přidat do databáze informace o novém člověku, použije tlačítko, kterým vyvolá funkci *zobrazFormular*. Uživateli se zobrazí formulář pro zadání údajů o nové osobě, který po vyplnění uživatel odešle kliknutím na tlačítko pro uložení dat.

Před samotným uložením je nutné převést informace do textové podoby – to se týká druhu osoby a pohlaví osoby. Tyto informace jsou získány ve formuláři pomocí netextových komponent, které jsou popsány níže v textu (Kapitola 6). Pro převedení do textové podoby jsou volány funkce *vyberDruhOsoby* a *vyberPohlaviOsoby*.

Funkce *ulozOsobu* provede uložení informací do databáze. Dále je nutné do databáze uložit také typ pomoci, kterou umí osoba poskytnout. K tomu slouží funkce *ulozTypPomoci*. Protože může jeden člověk poskytovat více typů pomoci, je nutné zjistit, jaký typ pomoci člověk umí poskytnout a následně jej uložit. K tomu slouží operátor *alt*, který vykoná volané funkce, pokud je splněna definovaná podmínka. V diagramu jsou naznačeny pouze dvě podmínky, protože ostatní varianty by byly úplně stejné, až na spouštěcí podmínku. Vždy se jedná o stejnou posloupnost kroků, která uloží požadovaný typ pomoci k vybrané osobě.

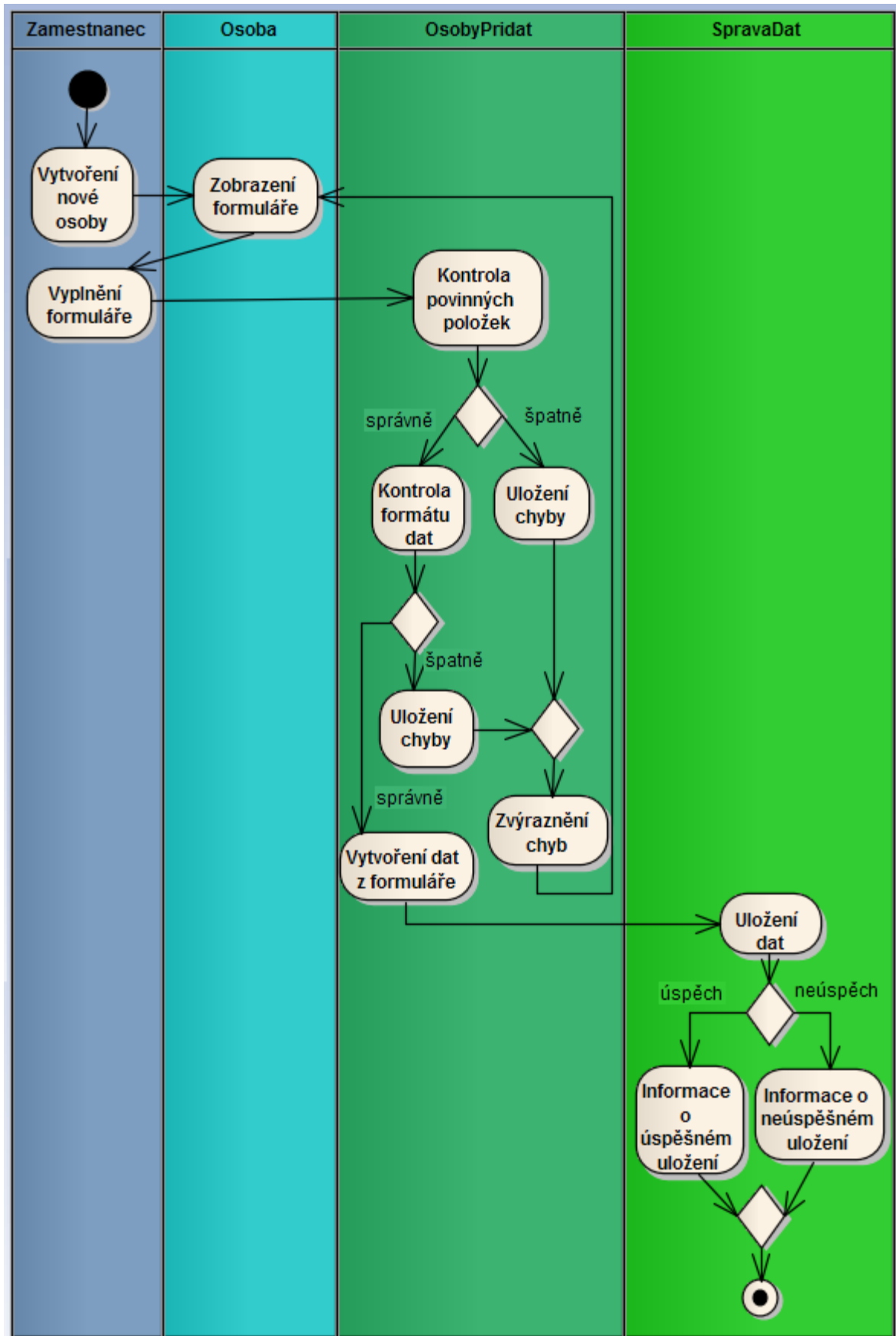
4.5 Diagram aktivit

Cílem tohoto diagramu je znázornit tzv. řídicí tok (sekvenci vykonávaných kroků) procesu.

Diagram aktivit na obrázku (Obrázek 14) popisuje proces případu užití, při kterém uživatel ukládá novou osobu do databáze. Proces je spuštěn uživatelem, který klikne na příslušné tlačítko v grafickém rozhraní. Systém na tento požadavek reaguje zobrazením formuláře, který následně uživatel vyplní a odešle ke zpracování. Z diagramu můžeme vidět, že popsané činnosti se zatím odehrávají pouze mezi uživatelem *Zamestnanec* a třídou *Osoba*.

Komponenta *OsobyPridat* provede kontrolu vstupních dat na povinné položky a na správný formát dat a v případě nalezení chyby informuje uživatele zvýrazněním chybných položek ve formuláři.

Pokud jsou data ve správném formátu, provede třída *SpravaDat* uložení dat. O výsledku zápisu dat do databáze informuje uživatele hlášením. Po zobrazení hlášení je diagram ve svém koncovém bodu a proces případu užití končí.



Obrázek 14: Diagram aktivit procesu přidání osoby

5 UKLÁDÁNÍ DAT

Data vytvořená uživatelem aplikace je potřeba ukládat, jinak by aplikace ztrácela smysl. Pro ukládání dat byla využita databáze v prostředí SQLite, ale také ukládání do souborů. Obě varianty ukládání dat jsou prováděny zápisem na místní disk, což může způsobit potíže při nedostatečném oprávnění uživatele systému. Tyto omezení jsou různá a závisí na typu operačního systému.

Ukládání do souborů vzešlo z požadavků softwarové analýzy, při které byl zjištěn požadavek na vytváření pozvánek na pořádané akce a také na vytváření ocenění pro dobrovolníky, kteří se účastnili nějaké akce. Pro ukládání těchto pozvánek a ocenění poskytuje vývojová platforma Flex jednoduší uložení ve formě obecného objektu, který je možno uložit do samostatného souboru.

Všechny informace ukládané při práci s aplikací jsou umístěny do složky *Charita*, ve které jsou dále vytvořeny podsložky *Databaze*, *Podekovani* a *Pozvanky*. Celá tato datová struktura je pak umístěna do složky dokumentů operačního systému, který na počítači běží.

5.1 Ukládání do souboru

Uložení vytvořené pozvánky nebo diplomu je prováděno do souboru, protože každá pozvánka i diplom se skládají z několika prvků a každý z těchto prvků má větší počet vlastností, které specifikují jeho zobrazení na vytvářeném dokumentu.

Při použití databáze by bylo nutné vytvořit několik tabulek, které by byly schopny tuto datovou strukturu uložit. Vzhledem k lokálnímu běhu aplikace mohlo být využito jednoduší řešení, které spočívá v uložení pozvánky nebo diplomu jako datového objektu do samostatného souboru.

Oproti databázi je zde ale také jedna nevýhoda a tou je vyhledávání již vytvořených souborů. Pro správné nalezení je nutné vytvářet jedinečné názvy souborů, aby nedocházelo k přepisování. Zde je nutné jisté propojení s databází, aby se dalo určit, který soubor patří k jaké osobě nebo akci. Pozvánky jsou tedy ukládány pod název odpovídající identifikátoru akce v databázi. U diplomů je situace složitější, protože jeden člověk může mít více diplomů, nelze použít identifikátor osoby a stejně tak za jednu akci může mít diplom více lidí. Jako vhodný a jednoznačný název je tak vybráno spojení obou identifikátorů. Jména vytvářených souborů jsou ještě doplněna o text „Pozvanka“ nebo „Podekovani“.

5.2 Databáze

Vytvořené úložiště dat se skládá ze čtyř základních tabulek – jedná se o tabulky *Akce*, *Osoby*, *Veci* a *Vypujcitel*. Tyto tabulky představují základní kostru databáze a nesou primární informace vytvářené uživatelem. Mezi těmito tabulkami ale také existuje množství vztahů, které nesou sekundární informace a je nezbytné je také ukládat.

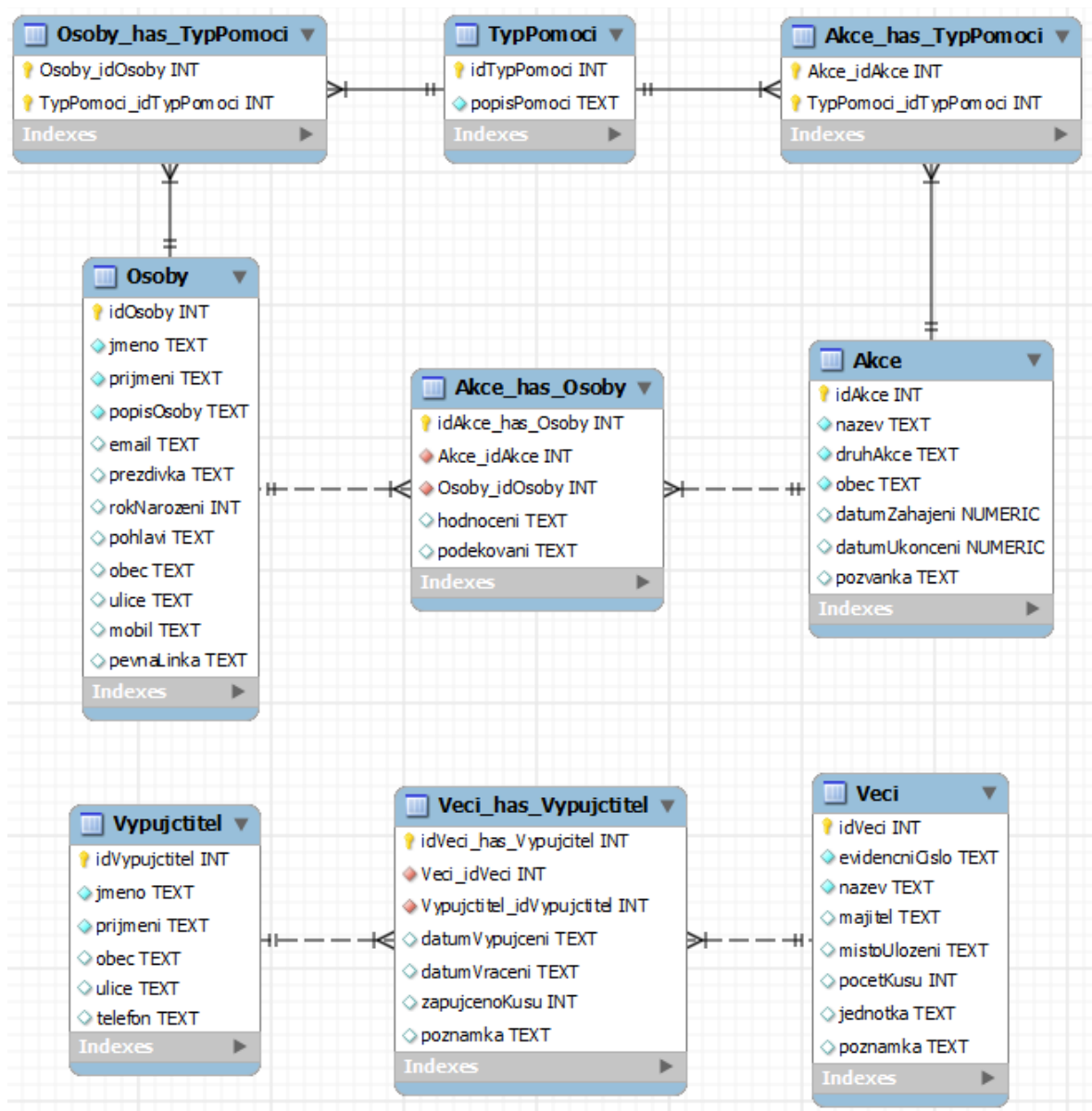
První problematikou, kterou je potřeba v databázi zachytit, je skutečnost, že lidé, kteří spolupracují s Oblastní charitou Břeclav, se účastní akcí pořádaných touto organizací. Protože jeden člověk se může účastnit více akcí, ale zároveň jedné akce se může účastnit více lidí, vzniká mezi těmito entitami vztah typu M:N. Tento vztah je nutné uložit do vlastní tabulky a vzniká tak tabulka *Akce_has_Osoby*, která mimo samotného vzájemného přiřazení osob k akcím ještě ukládá informaci o tom, jak daný člověk na vybrané akci pracoval a také, jestli mu bylo vytvořeno poděkování. Tyto informace se týkají vždy jak osoby, tak i akce a pro každou vazbu mohou být jiné, proto je nutné jejich uložení do této tabulky.

Další tabulky vznikají tím, že osoby i akce potřebují ukládat informace o typech pomoci. Ty byly popsány ve fázi softwarové analýzy (Kapitola 4.1.2) a byla pro ně vytvořena tabulka *TypPomoci*. Z pozorování a pomáhání při povodních v obci Chrastava bylo zjištěno, že při povodních, což je jeden z typu akce, je nutná především manuální a psychosociální pomoc. A to z důvodů rozsáhlých škod a velkého počtu zasažených obyvatel. Oproti tomu např. při hromadné autonehodě je potřeba spíše logistická a základní zdravotnická pomoc.

Mezi osobami a typy pomoci a mezi akcemi a typy pomoci je tedy potřeba vytvořit tabulky pro zachycení vzájemných vztahů. Jedná se o tabulky *Osoby_has_TypPomoci* a *Akce_has_TypPomoci*.

Poslední část databáze tvoří tabulky potřebné pro ukládání informací o výpůjčkách. Člověk, který je postižen nenadálou událostí, si může vypůjčit jednu nebo více věcí pro svou potřebu. Mezi tímto člověkem a majetkem vzniká tedy vztah typu M:N, který je zachycen v tabulce *Veci_has_Vypujcitel*. Dalšími atributy tohoto vztahu je poznámka, která slouží k zapsání např. poškození při výpůjčce, informace o množství, které bylo zapůjčeno a také datum vypůjčení a datum vrácení.

Při vytváření databáze bylo nutné přihlídnout k omezením, která existují v prostředí SQLite. To umožňuje ukládat pouze datové typy integer, real, text a blob. V návrhu tudíž nebylo možno použít datové typy *enum*, *date* a *boolean*. Systém tedy používá datové typy integer a text. Celá databáze je pak vytvořena v programu MySQL Workbench a její E-R diagram je znázorněn na následujícím obrázku (Obrázek 15).



Obrázek 15: E-R diagram databáze

6 VZHLED APLIKACE

Pro vytvoření grafického uživatelského rozhraní aplikace se kladl důraz na požadavky uživatele systému. Proto nebylo zvoleno standardní menu, které obsahuje seznam položek, ze kterých si uživatel může vybrat, ale devět základních tlačítek, které má uživatel stále na ploše aplikace. Těmito tlačítky se uživatel přepíná mezi vedením databáze osob, akcí, věcí, výpůjček, vypůjčitelů, nasazením osob na akce, vytvářením pozvánek a poděkování a zálohováním.

Všechny sekce mají dále společná tlačítka pro práci s daty. Jedná se o tlačítko pro přidání nového záznamu a pro vymazání a úpravu existujícího záznamu. Sekce hlavních tabulek mají navíc také tlačítko pro vyhledání, které umožňuje vyhledat požadovanou položku a následně ji vytisknout. Jednotlivá tlačítka pak zobrazují formuláře pro vykonání požadovaného úkolu. Výjimkou je sekce zálohy, která obsahuje tlačítka pro vytvoření zálohy a pro obnovení dat ze zálohy.

Dalším společným rysem všech částí aplikace je barevné rozlišení a použití efektů při zobrazení. Barevné zvýraznění odpovídá jemnějšímu odstínu barvy tlačítka, které tuto skupinu zobrazuje. Pro zobrazení byl použit efekt *Zoom*, který pomocí svých čtyř atributů nastavuje výchozí šířku a výšku a cílovou šířku a výšku. Jedná se o procentuální velikost zobrazovaného objektu udávanou v rozmezí hodnot 0 až 1.

Obrazovky pro vedení databáze osob, akcí, věcí a vypůjčitelů jsou z hlediska vzhledu stejné a proto je zde na ukázkou popsána pouze obrazovka pro vedení osob (Kapitola 6.2).

Druhým typem obrazovek jsou ty, při kterých dochází spojením dvou informací z různých tabulek k vytvoření nové vazby. Do této skupiny patří ostatní části aplikace a jako ilustrace zde bude popsána část pro správu diplomů.

Při vytváření vzhledu byly použity následující komponenty, které slouží pro komunikaci mezi uživatelem a systémem:

- *TextInput* (textový řádek)
- *PopUpMenuButton* (rolovací menu)
- *CheckBox* (slouží pro zaškrtnutí položky)
- *RadioButton* (je určen pro výběr pohlaví)
- *NumericStepper* (počítadlo pro zadání roku narození)
- *DateField* (kalendář pro zadání data)

- *ColorPicker* (výběr barvy)
- *Image* (vlození obrázku)
- *TextArea* (víceřádkové textové pole)

6.1 Úvodní obrazovka

Základním prvkem při tvorbě uživatelského rozhraní je komponenta *Canvas*, která ohraničuje určitou skupinu prvků. Tento prvek uživatel přímo nevidí a plní tak pouze pomocnou úlohu při správném zobrazování ostatních prvků. Je totiž například jednodušší skrýt skupinu prvků obsaženou v komponentě *Canvas*, než ukryvat každý prvek zvlášť.

Po spuštění aplikace se uživateli zobrazí úvodní obrazovka (Obrázek 16), která je rozdělena do dvou částí. První část představuje statické menu, které je uživateli k dispozici po celou dobu běhu aplikace a druhá je vyhrazena pro práci uživatele. Úvodní obrazovka dále obsahuje informační řádek, tzv. *status*, ve kterém je uživatel informován v jaké sekci programu se právě nachází a logo neziskové organizace.

Menu se skládá z tlačítek, která jsou vytvořena pomocí komponent *Button*. Tyto komponenty jsou doplněny o obrázky, které popisují funkci daného tlačítka. Pro správné zobrazení a zarovnání tlačítek je použita komponenta *VBox*. Jedná se o jakousi speciální variantu komponenty *Canvas*, kdy prvky v ní obsažené jsou vertikálně zarovnány pod sebe. Nastavením atributu výšky tlačítka pomocí procent vede k dosažení plného využití prostoru, kdy každé tlačítko zabírá stejnou plochu.

Plocha pro práci uživatele se systémem obsahuje deset komponent typu *Canvas*, z nichž devět odpovídá tlačítkům v menu a poslední slouží k zobrazení loga organizace. Na úvodní obrazovce je zobrazena pouze oblast obsahující logo, ostatní oblasti jsou skryté a zobrazí se po kliknutí na příslušné tlačítko. Zobrazení je provedeno změnou atributu *visible*, příslušné komponenty, z hodnoty *false* na hodnotu *true*. Je zde také použit efekt průhlednosti, kdy je oblast s logem zobrazena stále, a ostatní oblasti jsou zobrazovány nad toto logo s nastavením atributu průhlednosti na 90%.



Obrázek 16: Úvodní obrazovka vytvořené aplikace

6.2 Obrazovka pro vedení osob

Po kliknutí uživatele na tlačítko odkazující na osoby, se zobrazí nová plocha, která nabízí čtyři základní tlačítka pro správu databáze osob (Obrázek 17). Tato tlačítka jsou opět pro danou obrazovku statická a uživatel je má stále k dispozici.



Obrázek 17: Tlačítka pro správu databáze osob

Ke správnému zobrazení tlačítek je použita komponenta *HBox*, která zajišťuje horizontální zarovnání, které spolu s procentuálním nastavením šířky tlačítka opět vede k maximálnímu využití prostoru a velikosti tlačítek.

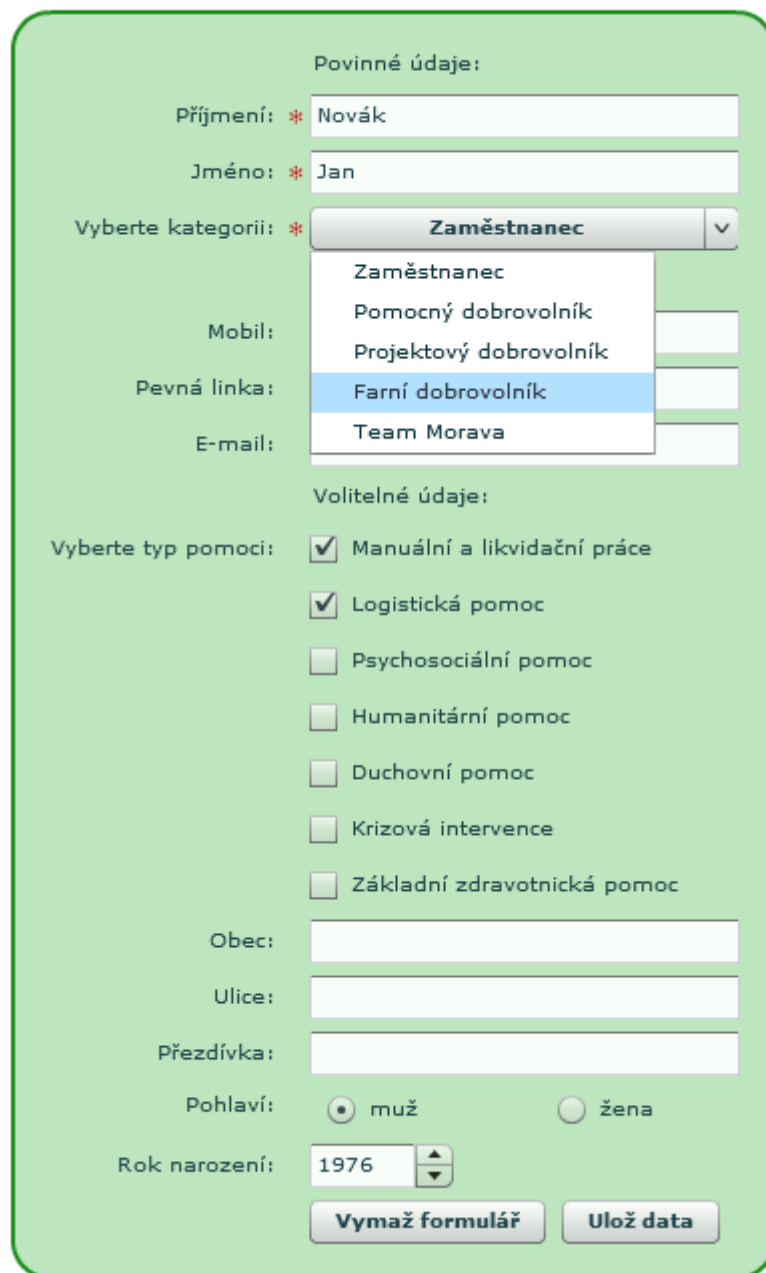
6.2.1 Přidání osoby

Pro přidání nové osoby do databáze je uživateli zobrazen formulář, který obsahuje všechny možné položky, které lze uložit. Pro formulář poskytuje jazyk MXML speciální komponentu *Form*, jejíž jednotlivé položky se označují jako *FormItem*. Jejich zajímavou vlastností je vlastnost *required*, které pokud je přiřazena hodnota *true*, zajistí zobrazení červené hvězdičky u položky formuláře a napovídá tak uživateli, že se jedná o povinnou položku. Do výše popsaných položek jsou vloženy příslušné komponenty pro interakci s uživatelem a je tak vytvořen vstupní formulář.

Komponenta *Form* zajišťuje graficky ujednocený vzhled jednotlivých položek, kdy popisky a samotné komponenty pro interakci s uživatelem jsou jak horizontálně, tak i vertikálně zarovnány. Pomocí atributů této komponenty lze také dosáhnout zaoblení rohů, nastavení barvy rámečku a barvy pozadí.

Obsluhu celého formuláře zajišťuje dvojice tlačítek, z nichž první slouží pro vymazání hodnot zobrazených ve formuláři a druhé pro uložení vyplněných informací do databáze. Výsledný formulář pro přidání nové osoby je zobrazen na obrázku (Obrázek 18).

Přidání nové osoby



Povinné údaje:

Příjmení: * Novák

Jméno: * Jan

Vyberte kategorii: * Zaměstnanec

- Zaměstnanec
- Pomocný dobrovolník
- Projektový dobrovolník
- Farní dobrovolník
- Team Morava

Mobil:

Pevná linka:

E-mail:

Volitelné údaje:

Vyberte typ pomoci:

- Manuální a likvidační práce
- Logistická pomoc
- Psychosociální pomoc
- Humanitární pomoc
- Duchovní pomoc
- Krizová intervence
- Základní zdravotnická pomoc

Obec:

Ulice:

Přezdívka:

Pohlaví: muž žena

Rok narození: 1976

Vymaž formulář **Ulož data**

Obrázek 18: Formulář pro přidání nové osoby

6.2.2 Vymazání osoby

Posloupnost událostí, které je potřeba vykonat při vymazání osoby z databáze, je složitější než při uložení osoby do databáze. Proto je tento proces rozdělen do tří částí, tedy do tří obrazovek, které se postupně zobrazí uživateli.

Pokud uživatel chce vymazat nějakou dříve uloženou osobu z databáze, pak je nejprve nutné tuto osobu vyhledat. Na obrázku (Obrázek 19) je vidět, že vyhledávání je možné provádět podle příjmení, jména, obce a kategorie.

Vymazání osoby

1. krok - vyhledání osoby

Příjmení:

Jméno:

Obec:

Kategorie: Vše v

Vyhledej osobu

Obrázek 19: Formulář pro vyhledání osoby při vymazání

Pokud existují nějaké záznamy, které odpovídají zadaným hodnotám, jsou zobrazeny do tabulky, ze které si uživatel může vybrat konkrétní osobu. Je totiž možné, že při vyhledání bude nalezeno více osob a je tedy na uživateli, aby následně vybral tu, kterou měl na mysli. Popsanou tabulku můžeme vidět na následujícím obrázku (Obrázek 20).

Vymazání osoby

2. krok - kliknutím na řádek vyberte osobu, kterou chcete vymazat

Jméno	Příjmení	Kategorie	Přezdívka	Obec
Libor	Maňák	Pomocný dobrovolník		
Hana	Machálková	Pomocný dobrovolník		
Pavel	Čech	Zaměstnanec		

Zpět

Obrázek 20: Tabulka zobrazující vybrané osoby při vymazání

Poté, co uživatel klikne na vybraný řádek tabulky, zobrazí se mu všechny uložené informace o vybrané osobě. Data by mohla být vymazána okamžitě po kliknutí na příslušný řádek, ale z důvodu bezpečnosti a ujištění se uživatele, že chce vymazat právě tuto osobu, je mu zobrazen poslední formulář (Obrázek 21), na jehož konci je tlačítko pro vymazání osoby. Uživatel může použít také tlačítko zpět pro opětovné zobrazení tabulky a

to např. z toho důvodu, že po zobrazení všech informací zjistil, že tuto osobu vymazat nechce.

Vymazání osoby

3.krok - vymazání osoby

Příjmení:	Čech
Jméno:	Pavel
Kategorie:	Zaměstnanec
Mobil:	606769192
Pevná linka:	
E-mail:	
Typ pomoci:	Manuální a likvidační práce Logistická pomoc Psychosociální pomoc Humanitární pomoc Duchovní pomoc Krizová intervence
Obec:	
Ulice:	
Přezdívka:	
Pohlaví:	muž
Datum narození:	1985

Obrázek 21: Přehled informací o vymazávané osobě

6.2.3 Upravení osoby

Další potřebnou aktivitou při správě databáze je možnost uživatele měnit již vytvořené záznamy. K tomu slouží skupina obrazovek, s jejichž pomocí uživatel může provádět potřebné změny.

Postup provedení úpravy údajů je ze dvou třetin stejný jako postup při vymazání údajů. První dvě obrazovky jsou tak totožné, pouze jinak barevně zvýrazněné. Rozdíl nastává až v poslední obrazovce, která na rozdíl od obrazovky pro vymazání poskytuje uživateli možnost měnit zobrazené hodnoty a následným stisknutím zobrazeného tlačítka uložit

provedené změny. Na obrázku (Obrázek 22) je vidět formulář pro provedení změn. Muselo zde být přidáno tlačítko typu *CheckBox*, jehož zatrhnutím uživatel zobrazí menu pro změnu kategorie osoby.

Upravení osoby

3. krok - změnu údajů

Příjmení:

Jméno:

Kategorie: Zaměstnanec
 Změnit

Mobil:

Pevná linka:

E-mail:

Typ pomoci: Manuální a likvidační práce
 Logistická pomoc
 Psychosociální pomoc
 Humanitární pomoc
 Duchovní pomoc
 Krizová intervence
 Základní zdravotnická pomoc

Obec:

Ulice:

Přezdívk:

Pohlaví: muž žena

Rok narození:

Obrázek 22: Formulář pro změnu údajů osoby

6.2.4 Vyhledání osoby

Tato část práce s databází osob slouží k zobrazení informací a k jejich možnému vytištění. Vyhledávání osob je zde rozšířeno o možnost vyhledání podle typů pomoci, které hledaná

osoba umí poskytnout. Formulář pro vyhledání osoby pro potřeby zobrazení informací je vidět na následujícím obrázku (Obrázek 23).

Vyhledání osoby

1. krok - vyhledání osoby

Příjmení:

Jméno:

Obec:

Kategorie: ▾

Vyberte typ pomoci:

- Manuální a likvidační práce
- Logistická pomoc
- Psychosociální pomoc
- Humanitární pomoc
- Duchovní pomoc
- Krizová intervence
- Základní zdravotnická pomoc

Obrázek 23: Formulář pro vyhledání osoby při zobrazení a tisku

Následně zobrazená tabulka osob, které odpovídají parametrům zadaných do formuláře, je stejná jako v případě vymazání nebo upravení osoby. Změna je až u zobrazení osoby, kde je tlačítko pro vymazání nahrazeno tlačítkem pro vyvolání tiskového menu. Dále je také přidána další tabulka, která zobrazuje akce, kterých se vybraná osoba v minulosti zúčastnila. Tato tabulka je doplněna o funkci pro zobrazení podrobností o vybrané akci, která je volána při kliknutí na řádek tabulky. Všechny popsané informace zobrazuje následující obrázek (Obrázek 24).

Vyhledání osoby

3. krok - zobrazení údajů

Příjmení: Čech
 Jméno: Pavel
 Kategorie: Team Morava
 Mobil: 606769192
 Pevná linka:
 E-mail: pavel.cech@centrum.cz
 Typ pomoci: Manuální a likvidační práce
 Logistická pomoc

Duchovní pomoc

Základní zdravotnická pomoc

Obec: Ladná
 Ulice: Mlýnská
 Přezdívka: dvera
 Pohlaví: muž
 Datum narození: 1985

Akce, kterých se účastnila vybraná osoba - kliknutím zobrazte podrobnosti.

Název	Druh akce	Obec
Povodně - Liberecko	Krizové akce	Chrastava

Název akce: Povodně - Liberecko
 Druh akce: Krizové akce
 Obec: Chrastava
 Datum zahájení: 08.04.2010
 Datum ukončení: 10.05.2010
 Hodnocení:

Obrázek 24: Zobrazení informací při vyhledání osoby

6.3 Obrazovka pro vedení diplomů

Při vytvoření diplomu je podobný postup jako při jeho upravení a z těchto důvodů je zde popsán pouze proces vytvoření nového diplomu. Obdobně je tomu u vyhledání a vymazání diplomu a stejně tak je zde tedy popsáno pouze vyhledání.

6.3.1 Vytvoření diplomu

Při vytvoření nového diplomu musí uživatel nejprve vyhledat akci, ze které má být diplom udělen. Poté je potřeba vyhledat ještě člověka, pro kterého je diplom určen. Vyhledání těchto informací zajišťuje trojice obrazovek, která se logikou neliší od výše popsaných postupů vyhledávání (Kapitola 6.2.4).

Nejprve uživatel zadá parametry pro vyhledání akce, následně vybere akci z tabulky a zobrazí se mu detailní informace o akci. Zároveň je mu zobrazena druhá tabulka, která obsahuje seznam osob, které se vybrané akce zúčastnily. Uživatel kliknutím na příslušný řádek vybere osobu a zobrazí se mu podrobné informace o této osobě společně s oknem obsahujícím hodnocení osoby a tlačítkem pro vytvoření diplomu.

Obrazovka pro vytvoření diplomu (Obrázek 25) se pak skládá ze dvou částí, kdy první část slouží pro nastavování vlastností vybraného objektu a druhá pak představuje reálné zobrazení vytvářeného diplomu.



Obrázek 25: Obrazovka pro vytvoření nového diplomu

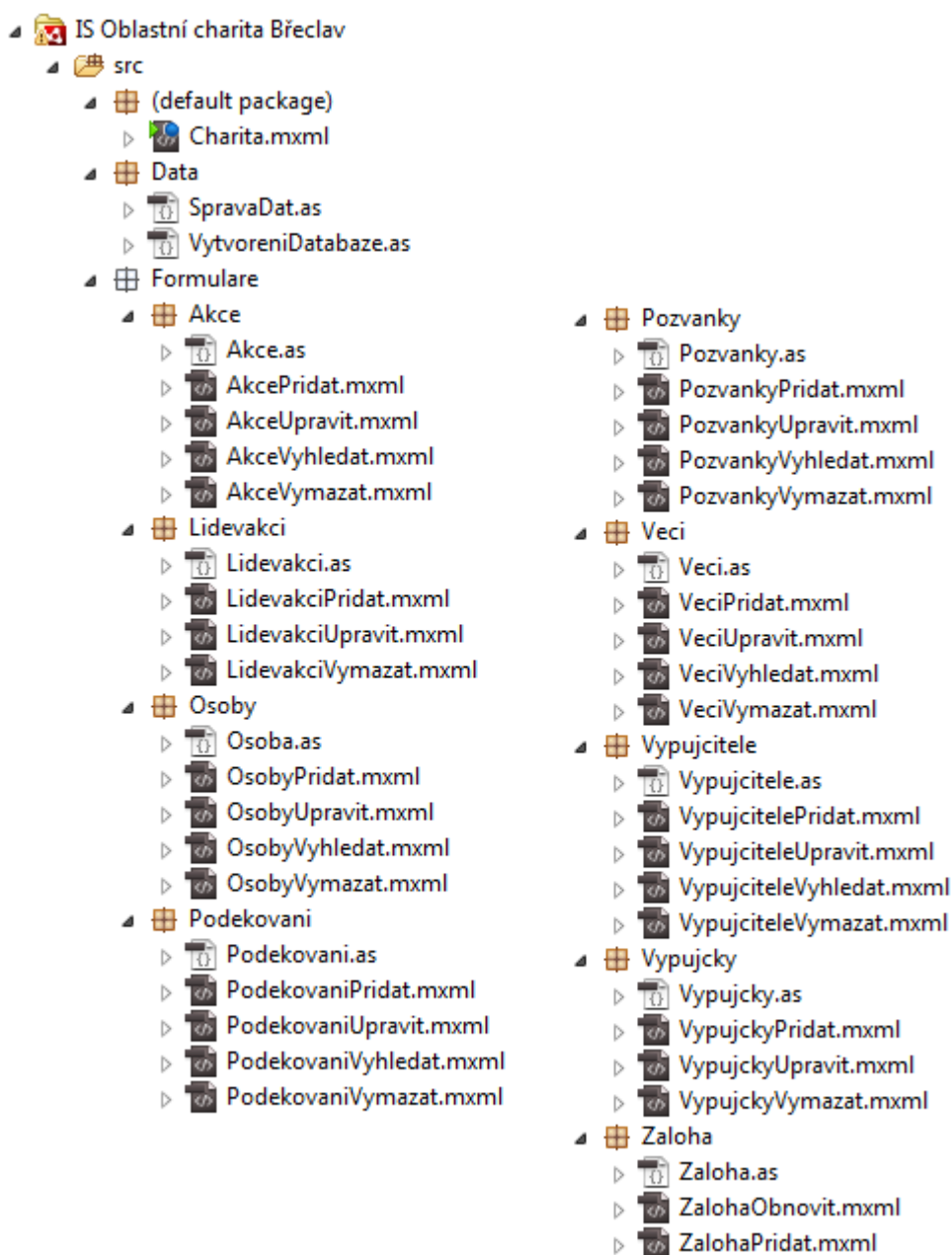
6.3.2 Vyhledání diplomu

Při vyhledání dříve vytvořeného diplomu je potřeba opět nejprve vyhledat akci, ze které byl diplom udělen. Po vybrání akce se uživateli zobrazí seznam osob, které se akce účastnily, a byl jim diplom již vytvořen. Uživatel vybráním osoby zobrazí diplom, který je možné vytisknout pomocí tlačítka pro tisk.

Problém jak u diplomů, tak i u pozvánek se projevil v tom, že jejich reálné zobrazení zabírá na pracovní ploše aplikace příliš mnoho místa (odpovídá velikosti formátu A4). Proto bylo maximální poskytnuté místo pro zobrazení plakátu zmenšeno a pro pohyb po plakátu je potřeba využít posuvník. Jedná se o kompromis, kdy uživatel stále vidí editační menu, ale plakát si musí rozdělit přibližně na dvě části a po nich se posouvat.

7 PROGRAMOVÁ LOGIKA

Tato kapitola popisuje implementaci logiky vytvářené aplikace a pro její lepší pochopení je nejprve popsána základní kostra celého programu. Hlavní zdrojový kód celé aplikace je obsažen v souboru *Charita.mxml*, pro správu databáze byl vytvořen balíček *Data*, který obsahuje třídy *SpravaDat.as* a *VytvoreniDatabaze.as*. Všechny formuláře sloužící pro komunikaci uživatele se člení do balíčků pro správu akcí, osob, nasazení osob, poděkování, pozvánek, věcí, vypůjčitelů, vypůjček a záloh. Každý z těchto balíčků obsahuje třídu a potřebné formuláře. Popsanou strukturu ilustruje následující obrázek (Obrázek 26).



Obrázek 26: Organizace zdrojových kódů

7.1 Inicializace aplikace

Při spuštění aplikace je potřeba vykonat několik základních údajů, které zajistí správné zobrazení aplikace uživateli, ale také vytvoření potřebných datových struktur, nutných pro běh a práci s daty. K tomu slouží funkce *init*, která je volána pomocí atributu *applicationComplete* komponenty *WindowedApplication*, v souboru *Charita.mxml*.

Funkce *init* nejprve provede zvětšení okna aplikace na maximální možnou velikost, danou rozlišením obrazovky. K tomu je potřeba získat referenci na zobrazované okno a následně volat funkci *maximize*, jak ukazuje následující příklad zdrojového kódu (Kód 1).

```
var hlavniOkno:NativeWindow = this.stage.nativeWindow;

hlavniOkno.maximize();
```

Kód 1: Maximalizace obrazovky aplikace

Další akcí po spuštění aplikace je nastavení hodnoty statusového řádku. Prostředí Flex používá k přístupu k tomuto grafickému elementu jednoduchou globální proměnnou s názvem *status*. Stačí tedy do této proměnné přiřadit textový řetězec a ten je vypsán na obrazovku. V průběhu běhu aplikace se obsah statusového řádku mění v souvislosti na tom, jak se uživatel přepíná mezi jednotlivými částmi aplikace.

Poslední částí ve funkci *init*, je vytvoření databáze pro ukládání dat. Zde je vytvořena instance třídy *VytvoreniDatabaze*, pomocí jejíchž funkcí jsou vytvořeny nové tabulky. Při vytváření každé tabulky je použit SQL dotaz, který obsahuje klauzuli *IF NOT EXISTS*, která zaručuje vytvoření tabulky, pouze pokud ještě neexistuje. Bez použití této klauzule by při každém spuštění aplikace došlo k přemazání uložených dat v databázi.

7.2 Přepínání základních obrazovek

Jak již bylo popsáno v části popisující vzhled aplikace (Kapitola 6.1), je uživateli zobrazováno deset různých základních obrazovek. Přepínání mezi těmito obrazovkami zajišťuje funkce *vyberHlavniMenu*, která má jeden parametr typu *int*, a ten určuje, která obrazovka bude zobrazena. Pokud tedy uživatel klikne na tlačítko *Osoby*, funkce zobrazí základní obrazovku pro práci s osobami. Ta obsahuje část pro zobrazení menu pro vedení databáze osob (Obrázek 17) a část pro zobrazování formulářů.

Protože je potřeba zajistit vždy stejné výchozí zobrazení každé z deseti základních ploch, je volána funkce *removeAllChildren*, která zajistí odstranění všech dynamicky přidaných formulářů k této základní obrazovce a ta tedy obsahuje pouze původní statické prvky. Tento postup je nutný z toho důvodu, že přepnutí mezi základními obrazovkami je uživateli umožněno neustále a pokud by nedošlo vždy k odstranění přidaných formulářů, ty by se pak uživateli zobrazily místo výchozí prázdné obrazovky.

7.3 Zobrazení formuláře

Veškerá činnost uživatele spojená s databází je vytvořena na základě komunikace pomocí formulářů, proto jich má každá z deseti základních částí aplikace několik. Jedná se o komponenty, které byly vytvořeny speciálně dle specifikace požadavků. Například při vedení databáze osob, existují formuláře pro přidání nové osoby, vymazání, upravení a vyhledání, tak jak jsou k vidění ve fázi softwarové analýzy (Obrázek 12).

Každá z deseti hlavních obrazovek má vlastní funkci, která dále obsluhuje správné zobrazení dalšího obsahu. V této funkci je vždy vytvořena instance třídy, která odpovídá vybrané obrazovce a pomocí volání funkce *zobrazFormular* této třídy je pak zobrazen požadovaný formulář na určené místo.

Protože uživatel opět může volně přecházet mezi například přidáním nové osoby do upravení osoby, je opět nutné zajistit výchozí zobrazení těchto částí. Proto je opět volána také funkce *removeAllChildren*, která tuto podmínku zajistí.

Následující kód ukazuje popsany postup, kdy při kliknutí uživatele na kterékoliv z tlačítek pro správu osob je uživateli zobrazen příslušný formulář. O jaký formulář se jedná, označuje vstupní parametr *index*.

```
protected function vyberFormularLide(index:int):void{  
  
    var osoba:Osoba = new Osoba;  
  
    lideUvod.removeAllChildren();  
  
    osoba.zobrazFormular(lideUvod, index);  
  
}
```

Kód 2: Funkce pro zobrazení požadovaného formuláře

7.4 Úložiště dat aplikace

Jak již bylo dříve popsáno v textu (Kapitola 5), pro ukládání dat byla použita databáze a ukládání dat do souborů. Z pohledu implementace se ale vždy jedná o ukládání do souboru, protože i databáze je pouze datový soubor na disku. Jediný rozdíl je v přístupu k těmto datům.

Ukládání dat na lokální disk sebou nese jistá omezení, která se liší dle typu operačního systému. Protože je ale vývojová platforma Flex multiplatformní, poskytuje programovací jazyk ActionScript možnost přístupu do základních adresářů různých platforem pomocí těchto funkcí:

- `applicationStorageDirectory` – vytvoří unikátní dočasnou složku při instalaci
- `applicationDirectory` – složka, kam byla aplikace nainstalována (pouze pro čtení)
- `desktopDirectory` – složka uživatelské plochy
- `documentsDirectory` – složka dokumentů uživatele
- `userDirectory` – složka uživatele (9)

Pro ukládání do složky dokumentů tak bylo využito funkce `applicationDirectory`. Pro vytvoření navržené adresářové struktury posloužili funkce `resolvePath`, která jako argument přijímá popisující adresářovou strukturu a funkce `createDirectory`, která vytvoří adresářovou strukturu zadanou jako argument předchozí funkce. Postup vytvoření databázového souboru je popsán v následující ukázce kódu (Kód 3). Podobným způsobem jsou pak vytvářeny všechny ostatní potřebné soubory, rozdíl je pouze v názvu a umístění.

```
var database:File =  
File.documentsDirectory.resolvePath("Charita/Databaze/database.db");  
  
database.parent.createDirectory();
```

Kód 3: Vytvoření adresářové struktury pro uložení databáze

7.5 Práce s databází

Pro komunikaci s databází byla vytvořena třída `SpravaDat`, která obsahuje funkce `zacatekTransakce`, `konecTransakce`, `chybaTransakce`, `ulozData`, `upravData`, `nactiData`, `vyhledejIndex` a `zobrazData`. Třída dále obsahuje proměnné pro databázový soubor (objekt

třídy *File*), navázání připojení (objekt třídy *SQLConnection*), provedení SQL dotazu (objekt třídy *SQLStatement*) a získání výsledku (objekt třídy *SQLResult*).

Funkce, které slouží pro vykonávání *SQL* dotazů, používají návratové hodnoty, které informují o úspěšném nebo neúspěšném vykonání dotazu. Pokud dojde k chybě, je vrácena hodnota *-1*, případně *null* v závislosti na typu volané funkce.

7.5.1 Transakce

Při běhu aplikace dochází často k práci s více tabulkami najednou, což při neočekávané chybě může vést k pouze částečnému provedení požadovaných změn. Z tohoto důvodu bylo nutné použít transakce, které slouží k tomu, aby se buď vykonaly všechny požadované změny, nebo žádná.

Jazyk Actionscript podporuje práci s transakcemi pomocí volání funkcí *begin*, *commit* a *rollback*. První funkce označuje začátek transakce, druhá pak potvrzuje všechny vykonané dotazy od prvního volání funkce *begin*, poslední pak naopak všechny dotazy zruší. Volání těchto tříd je implementováno třídě *SpravaDat* pomocí funkcí *zacatekTransakce*, *konecTransakce* a *chybaTransakce*.

První funkce nejprve otevře spojení s databázovým souborem pomocí funkce *open* třídy *SQLConnection*, následně toto spojení přiřadí do vlastnosti *sqlConnection* třídy *SQLStatement*. Druhé dvě třídy pak mimo potvrzení nebo zrušení transakce také uzavírají spojení voláním funkce *close*.

7.5.2 Vložení dat do databáze

Funkce pro uložení dat do databáze (Kód 4), je používána pro dotazy typu *INSERT*. Má jeden parametr, který obsahuje SQL dotaz, jenž se má vykonat. Ten je vložen do vlastnosti *text* třídy *SQLStatement*, vykonán příkazem *execute* a následný výsledek dotazu je získán metodou *getResult*. Z něj je pak funkcí *lastInsertRowID* získán index nového záznamu, který byl vytvořen databází.

Pokud při vykonání příkazu došlo k chybě, je tato chyba zachycena v sekci *try* a následně ošetřena sekcí *catch*, kde funkce vrací hodnotu *-1*. Pokud vše proběhlo bez problémů, navrácí funkce hodnotu indexu nového záznamu v databázi.

```
public function ulozData(text:String):int{

    dotaz.text = text;                //uložení textu SQL dotazu

    try{

        dotaz.execute();              //vykonání dotazu

    }catch (error:SQLException){      //ošetření chyb

        return -1;                    //návratová hodnota -1 značí chybu

    }

    vysledek = dotaz.getResult();     //získání výsledku dotazu

    return vysledek.lastInsertRowID;  //získání id přidaného řádku

}
```

Kód 4: Funkce pro ukládání dat do databáze

7.5.3 Editace dat v databázi

Pro editaci dat slouží dotazy *UPDATE* a *DELETE*. Protože společným výsledkem provedení těchto SQL dotazů je počet řádků, které byly dotazem změněny, obsluhuje je jedna funkce – *upravData*. Její tělo má oproti předcházející funkci jediný rozdíl a to ve významu návratové hodnoty, kdy úspěšný průběh funkce vrací počet řádků, které byly ovlivněny vykonáním dotazu. K tomu slouží funkce *rowsAffected* třídy *SQLResult*.

7.5.4 Vyhledání identifikátoru záznamu

V případě uložení nějaké vazby do tabulek je potřeba zjistit identifikátory jednotlivých stran, jichž se vzájemná vazba týká. K tomu byla vytvořena funkce *vyhledejIndex*, která má dva vstupní atributy. Mimo atributu pro SQL dotaz, který je stejný jako u předchozích funkcí, má navíc atribut určující název indexu, který se má vyhledat. Postup provedení dotazu je stejný jako v předchozích případech. Pro získání identifikátoru je pak použita vlastnost *data*, která obsahuje pole objektů, které odpovídá výsledku provedení dotazu.

7.5.5 Vyhledání dat

Pro vyhledání požadovaných dat slouží funkce *nactiData*. Její významný rozdíl oproti dříve popsáním funkcím je ten, že její návratovou hodnotou je instance třídy *SQLResult* a vrací tak celý výsledek provedení dotazu, aniž by jej sama nějak zpracovávala.

7.5.6 Zobrazení vyhledaných informací

Poslední funkcí třídy *SpravaDat* je jednoduchá funkce pro zobrazení vyhledaných informací. Funkce *zobrazData* očekává při svém volání dva parametry – data, která se mají zobrazit a identifikátor komponenty *DataGrid*, která má data zobrazit.

7.6 Práce se soubory

Pro čtení a zápis do souborů, které jsou využity pro ukládání pozvánek a diplomů, je potřeba zvolit jiný způsob než tomu bylo v případě databázového souboru. Princip vychází z toho, že pro zapisovaná nebo čtená data je potřeba vytvořit objekt třídy *fileStream*. Tento objekt pak používá funkce *open* (pro otevření souboru), *writeObject* (pro zápis objektu) a *close* (pro zavření souboru).

Funkce *open* využívá dva argumenty. První ukazuje na soubor, který se má otevřít, druhý pak říká, v jakém režimu se má soubor otevřít. Podle toho, jestli je potřeba data číst nebo zapisovat je volen režim *FileMode.WRITE* nebo *FileMode.READ*.

Stejně jako při práci s databázovým souborem i zde může dojít k chybě při čtení nebo zápisu a je proto vhodné chybové stavy odchytnout a informovat o nich uživatele. Popsaný postup práce se souborem je zobrazen na následujícím zdrojovém kódu (Kód 5).

```
private var fileStream:FileStream = new FileStream();

try{

    fileStream.open(file, FileMode.WRITE);

    fileStream.writeObject(podekovani);

    fileStream.close();

} catch(chyba:IOError){

    Alert.show("Chyba při práci se souborem!", "Chyba souboru");

    fileStream.close();

}
```

Kód 5: Zápis objektu do souboru

7.6.1 Zálohování dat

Samostatnou kapitolou při práci se soubory je proces vytváření datové zálohy a obnovení zálohovaných dat. Při těchto činnostech se totiž pracuje se soubory jako s celky, nikoliv pouze s částmi jejich obsahů a navíc je potřeba spolupráce s uživatelem. Protože zatím co všechny operace se soubory měly doposud dané umístění a název, nyní je potřeba od uživatele zjistit, kam si přeje data zálohovat, případně odkud se mají data pro obnovení načíst.

Při vytvoření zálohy je použita funkce *browseForDirectory*, která vyvolá dialogové okno pro výběr složky. Poté, co uživatel vybere nějakou složku, je zavolána funkce *ulozZalohu*, jako reakce na událost *Event.SELECT*. Tato událost obsahuje odkaz na uživatelem vybranou složku. V této složce je pak vytvořena nová složka, jejíž název odpovídá datu a času vytvoření zálohy. Po vytvoření takové složky je pak volána funkce *copyTo* třídy *File*, která zajistí vytvoření kopie datových souborů.

Postup při obnovování dat ze zálohy je pak velmi podobný jako samotné vytváření zálohy. Nejprve je uživatel vyzván k výběru složky, která obsahuje zálohovaná data, poté jsou soubory zkopírovány do adresáře určeného pro data aplikace a obnovení je tak hotové.

7.7 Robustnost systému

Důležitou částí informačních systémů je jejich zabezpečení proti chybovým stavům. Tyto stavy mohou nastat buď jako chyba uživatele nebo jako chyba vlastního běhu aplikace, která je spojena většinou s chybou při práci se soubory.

7.7.1 Kontrola vstupních formulářů

Formuláře pro vstup dat od uživatele je potřeba testovat, aby se zamezilo vstupu špatného formátu nebo špatného druhu dat. Co se týče této kontroly, je nejvíce testován formulář pro přidání nové soby, protože také obsahuje nejvíce vstupních prvků. Proto bude postup při kontrole dat popsán právě na tomto formuláři, další formuláře pak využívají stejné zabezpečení, jen ne v takovém rozsahu.

Na kontrolu vstupů se můžeme dívat ze dvou pohledů. První nám říká, jestli mají data správný datový typ a druhý, jestli je jejich obsah významově správný. V prvním případě je jistě chybou, pokud například rok narození dobrovolníka obsahuje nějaké písmeno. Uložení takové informace do databáze by vedlo k vyvolání chyby, protože databáze má pro

tuto informaci vyhrazen datový typ *integer*, do kterého písmeno nelze uložit. Pokud bychom chtěli na stejném příkladu vysvětlit i druhý pohled na chybná data, jednalo by se například o pokus uložit rok narození -1972. Tato hodnota jistě lze uložit do datového typu *integer*, ale její význam je jistě chybný a spíše by se mělo jednat o kladnou hodnotu, tedy rok narození 1972.

Protože je databáze navržena tak, že používá pouze datové typy *integer* a *text*, odpovídá tomu i kontrola datových typů v aplikaci. Řešení spočívá v použití vhodné komponenty pro vstupní informace. Pokud je od uživatele očekáván text, je použita komponenta *TextInput*, případně *TextArea* pro víceřádkový vstup. V případě číselné hodnoty je pak použita komponenta *NumericStepper*, která umožňuje navíc nastavit i minimální a maximální možnou hodnotu.

Ke kontrole správného významu vložených informací lze použít tzv. validátory, které jazyk ActionScript nabízí. V aplikaci bylo použito dvou druhů těchto validátorů, první pro kontrolu textu a druhý pro kontrolu e-mailové adresy. Pro ukázkou je popsán druhý případ, který provádí kontrolu nepovolených znaků, špatný formát doménové adresy a další náležitosti nutné pro splnění validního e-mailu. Výhodou této komponenty je také možnost nastavení si vlastních chybových hlášení, která se zobrazí uživateli, což dává možnost vytvořit systém, který je pro uživatele příjemnější.

7.7.2 Chyby při práci s databází

Při práci s databází může nastat neočekávaný problém, který vede buď k nevykonání dotazu, nebo pouze k částečnému vykonání sekvence dotazů. K řešení této problematiky je využito transakcí, jejichž princip již byl popsán (Kapitola 7.5.1).

Pro ukázkou popisu zabezpečení práce s databází je vybrán případ, kdy uživatel chce do systému přidat informace o nové osobě. Většina informací je uložena přímo v tabulce *Osoby*, ale pro uložení poskytované pomoci je použita tabulka *Osoby_has_TypPomoci*, která zachycuje, jaké typy pomoci člověk umí.

Uživatel tedy vyplní formulář a odešle jej ke zpracování. Aplikace nejprve provede zápis informací do tabulky *Osoby* pomocí volání funkce *ulozData*. Výsledkem tohoto prvního dotazu je jednoznační identifikátor, který byl osobou vytvořen databází.

Pokud uživatel vybral jako typ pomoci například duchovní pomoc, je proveden zjišťovací dotaz na získání identifikátoru tohoto druhu pomoci. Po provedení tohoto druhého dotazu

již jsou k dispozici oba potřebné identifikátory a lze je pomocí třetího dotazu vložit do tabulky *Osoby_has_TypPomoci*.

Před voláním prvního dotazu je aktivována transakce, která všechny tři následující dotazy obsahuje. Pokud při některém z dotazů došlo k chybě, vrací funkce obsluhující tento dotaz chybovou hodnotu, která je následně převedena do pomocné proměnné typu *boolean* a tím je indikován stav chyby. Tento chybový stav poté zruší celou transakci a uvede databázi do původní podoby. Pokud chyba nenastala, je transakce potvrzena a data jsou uložena.

7.7.3 Chyby při práci se soubory

Posledním problémem, který může při běhu aplikace nastat, je chyba při práci s pozvánkou nebo diplomem, které jsou ukládány do souborů. Protože jsou při této práci informace ukládány do databáze i do souboru, je potřeba zajistit, aby proběhly obě akce nebo žádná.

Pokud uživatel například vytváří novou pozvánku, je nejprve proveden zápis o této činnosti do databáze a následně je vytvořen nový soubor na disku, kam je pozvánka uložena. Pokud se zápis na disk nezdařil, je nutné odvolat také informace uložené do databáze. Jinak by totiž v databázi byla informace o vytvořeném souboru, ale ten by ve skutečnosti neexistoval. K potvrzení uložených informací je tak opět použita transakce, která začíná při ukládání do databáze a je potvrzena až po bezchybném vytvoření souboru.

7.8 Vizualizace dat

Vizuální zobrazení vyhledávaných dat pomocí tabulky provádí komponenta *dataGrid*, která je pro tuto činnost nejvhodnější. Umí totiž jednoduše zobrazit výsledek databázového dotazu, pokud předem ví, jaké názvy sloupců budou ve výsledku obsaženy. Stačí pouze tyto názvy nastavit pro jednotlivé sloupce do atributu *dataField* a získaná data z databáze pak přiřadit do atributu *dataProvider*.

Protože zobrazení všech údajů získaných z databáze by mohlo vést k nepřehlednosti, jsou zobrazeny pouze základní informace a podrobnější přehled je zobrazen po kliknutí na konkrétní řádek zobrazené tabulky. Této funkcionalitě bylo dosaženo pomocí nastavení posluchače událostí nad zobrazenou tabulkou. Po vyvolání události *ITEM_CLICK* tak dochází k volání funkce, která má zobrazení podrobností na starosti.

8 TESTOVÁNÍ APLIKACE

Testování správného fungování aplikace lze rozdělit na dvě části. První část byla prováděna při vývoji, kdy jednotlivé logické celky kódu byly spouštěny ihned po jejich napsání a tím byla testována jejich funkčnost. Tyto testy prováděly pouze pozitivní testování, tedy testování na očekávané události. Druhá část pak tvoří testování na neočekávané události, které mohou nastat jak chybou uživatele, tak kolizí s operačním systémem.

Aplikace nepoužívá žádné zvláštní hardwarové součásti, a proto není nutné řešit chyby hardwaru. Ty by pravděpodobně vedly jak k pádu aplikace, tak i celého operačního systému.

8.1 Testování uživatelských vstupů

Na vstupu formuláře pro vložení osoby bylo testováno, zda se podaří zadat e-mailovou adresu ve špatném formátu. Seznam testovaných vstupů spolu s reakcí systému ukazuje následující tabulka (**Chyba! Nenalezen zdroj odkazů.Chyba! Nenalezen zdroj odkazů.**).
nto neodhalil chybu v aplikaci a nedovolil zadat špatnou e-mailovou adresu.

Tabulka 1: Zadané vstupy při testování e-mailové adresy

Údaj na vstupu	Reakce systému
@domena.cz	Uživatelské jméno chybí v e-mailové adrese.
jmeno@domena	Chybí údaj o doméně v e-mailové adrese.
jmenodomena.cz	V e-mailové adrese chybí znak ‚@‘.
jmeno% @domena.cz	E-mailová adresa obsahuje nepovolené znaky.
jmeno@.cz	Špatný formát domény v e-mailové adrese.

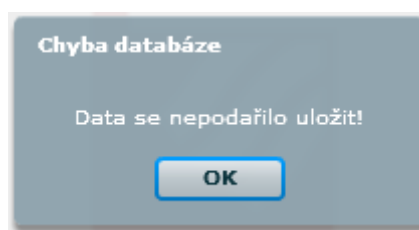
Další test byl prováděn na délku jména a příjmení, kdy byl do vstupního pole zadán pouze jeden. Na tento vstup aplikace reagovala zobrazením informace, že je nutné zadat minimálně dva znaky. Při následném pokusu zadat více než třicet znaků pak aplikace zobrazila informaci, že nelze zadat více než třicet znaků.

Testování číselných vstupů nelze provádět, protože číselné hodnoty jsou zadávány komponentou, která neumožňuje zadat žádný znak. Zároveň je této komponentě také

nastaven omezující rozsah, který zabraňuje zadávání nesmyslných údajů, jako jsou záporné nebo příliš vysoké hodnoty.

8.2 Testování chyb v souborech

Při tomto druhu testů byl uměle vyvolán stav, kdy při práci s aplikací dojde buď k částečnému, nebo úplnému poškození souboru. Částečné poškození bylo simulováno například vymazáním některé z tabulek databáze a úplné vymazání souboru. Aplikace na tento problém reaguje zobrazením chybového hlášení (Obrázek 27), které se svým obsahem liší podle toho, v jaké fázi běhu aplikace k chybě došlo.



Obrázek 27: Chybové hlášení

V případě, kdy dojde k poškození při vypnuté aplikaci, je po zapnutí provedena úvodní posloupnost příkazů, která se pokusí vytvořit nové datové soubory. Toto řešení umožní ale běh aplikace bez dat, která byla v poškozených souborech.

8.3 Testování integrity databáze

Tento druh testování souvisí s předcházejícím testem. Jedná se o testování funkčnosti transakcí. Test byl proveden při přidání nové osoby do databáze, kdy se ukládají informace do dvou tabulek, přičemž druhá z nich byla vymazána. Aplikace reaguje stejně jako v předcházejícím případě, protože se jedná opět o poškození souboru. Důležité pro tento test ale je, že ani část informací o osobě nebyla do tabulky zapsána.

9 BEZPEČNOST APLIKACE

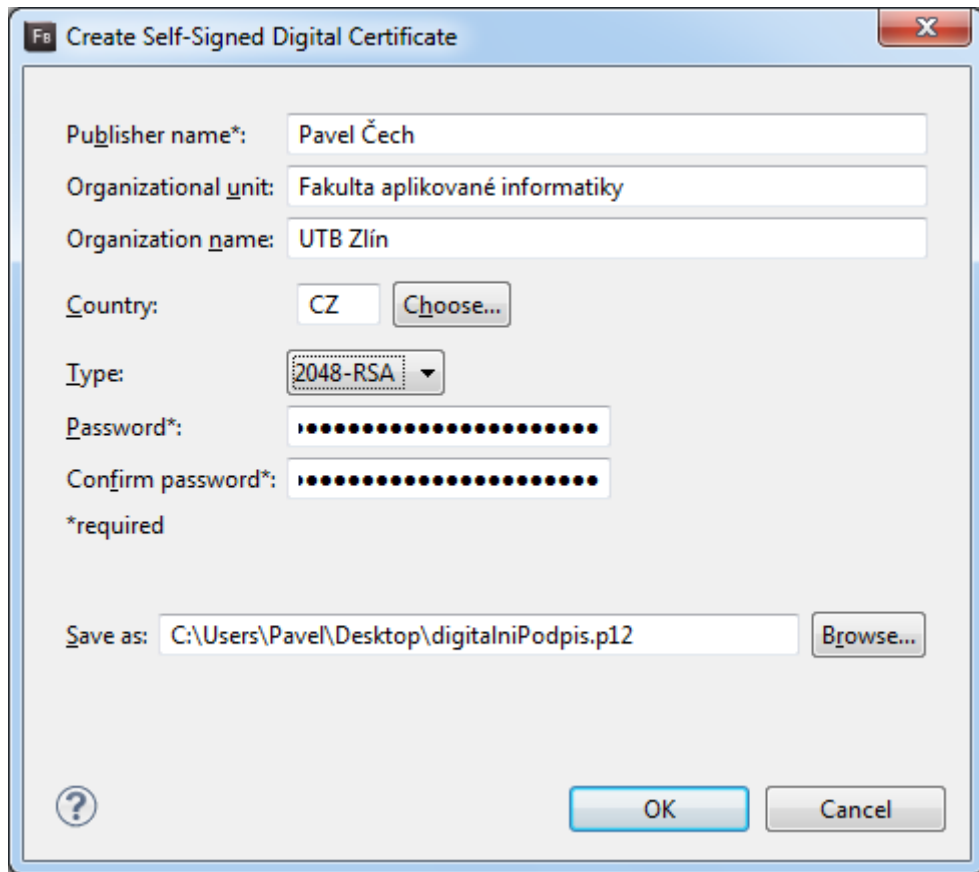
Protože aplikace běží na lokálním počítači a pro svůj chod nevyžaduje připojení k internetu, je otázka bezpečnosti do značné míry odsunuta do pozadí. Nedochozí totiž k přenosu dat mezi datovým uložištěm a aplikací. Navíc počítač, na kterém aplikace běží, je určen pouze jednomu člověku a nehrozí tedy zneužití jinou osobou.

Také ukládané informace nemají charakter velmi citlivých údajů. Jedná se sice o jména, adresy a telefonní čísla a podobně, ale z těchto údajů nehrozí nebezpečí zneužití platebních karet, čísla účtů a jiné závažné problémy. Databáze také ukládá informace pouze o dobrovolnících, nikoliv o těch, kterým byla pomoc poskytnuta. Zde by se totiž jednalo o velmi citlivé údaje, například při poskytnutí pomoci při drogové závislosti. Samotný fakt, že je někdo dobrovolníkem, také není nijak citlivý a jeho prozrazení není závažným problémem.

Z výše popsaných důvodů tak nebylo použito šifrování ukládané databáze. Tento krok by vedl spíše ke zbytečným komplikacím, nutnému zpomalení práce a jeho efekt by tak byl dosti rozporuplný.

Jiným způsobem zabezpečení aplikace je její digitální podpis, který ověřuje identitu autora a zaručuje, že aplikace nebyla změněna. Tohoto způsobu zabezpečení aplikace je vhodné využít při široké distribuci pomocí internetu, kdy může dojít ke zneužití.

Použití digitálního podpisu podporuje přímo vývojový nástroj Flash Builder, který při exportování výsledné verze podpis vyžaduje (Obrázek 28). V souvislosti s nasazením aplikace na jeden počítač, a to přímo programátorem aplikace, postrádá digitální podpis význam a pro vytvořenou aplikaci byl použit pouze digitální podpis, vytvořený autorem, který nelze ověřit.



Obrázek 28: Vytvoření certifikátu pro digitální podpis aplikace

Celkové zabezpečení aplikace je tedy závislé na zabezpečení celého počítače a odráží tak skutečnost, že informace v aplikaci používané nejsou citlivými, ale pouze osobními údaji.

10 NASAZENÍ APLIKACE

Uvedení aplikace do provozu proběhlo v sídle Oblastní charity Břeclav na počítači paní Hany Jagošové. Vytvořená aplikace má možnost být instalována buď ze souboru s příponou *.exe* nebo *.air*. První možnost je speciálně pro operační systémy firmy Microsoft, druhá pak poskytuje instalaci přímo v prostředí Adobe AIR a tudíž nezávislou na operačním systémem. Protože na cílovém počítači je operační systém Windows XP, byla použita instalace z prvního typu souboru.

Po nasazení běžel program v testovacím režimu, během něhož byl čas pro odstranění možných chyb, které nebyly nalezeny při sérii testů. Žádná taková chyba objevena nebyla a reakce na aplikaci byla pozitivní.

ZÁVĚR

Zadaná práce byla splněna vytvořením funkční aplikace, jejíž přínos je v lepší práci s informacemi, které Oblastní charita Břeclav používá pro svou činnost. Vytvořená aplikace je na poměry dnešních informačních systémů velmi odlišná, což je důsledek speciálních požadavků a také tím, že zaměstnání v tomto oboru je velmi specifické.

Zatímco jiné oblasti používající výpočetní techniku si mohou dovolit klást různé požadavky na své zaměstnance, oblast sociálních služeb je primárně zaměřena na pomoc lidem, a tak ji často vykonávají osoby s různými schopnostmi a dovednostmi. Schopnost pracovat s počítačem je ale na nižší úrovni, a proto vytvořený systém je na pohled co nejjednodušší. Tomu také odpovídá hlavní menu, které je uživateli stále zobrazeno a usnadňuje tak jeho pohyb v aplikaci. Ostatní části aplikace pak byly vytvářeny tak, aby si byly co nejvíce podobné stejné druhy vykonávané činnosti. Výsledkem této snahy je barevné odlišení prováděných operací a vždy stejná tlačítka pro správu dat.

Na dnešní poměry v oblasti informačních systémů také působí nezvykle využití lokálního úložiště dat. Protože každá oblastní charita má na starosti různé území, spolupracuje s různými lidmi a dělá různé akce, je potřeba společné databáze nulová. Taková databáze by pak stejně musela být členěna dle struktury celé organizace a ukládaná data by tvořila samostatné celky, které by spolu neměly žádnou souvislost.

Vytvořená aplikace splnila svůj účel a nahradila nesystémové nakládání s informacemi, které bylo vedeno formou papírových záznamů. Protože byly údaje vedeny pouze na papíře, není potřeba řešit zpětnou kompatibilitu aplikace. V tomto pohledu pak jako zpětná kompatibilita slouží funkce tisku záznamů.

Poslední částí aplikace je provádění záloh dat, která byla vytvořena opět s ohledem na co nejvyšší míru jednoduchosti. Zde je patrná nevýhoda lokálního běhu aplikace, protože nelze provádět zálohy například na jiný datový server. Navíc je nutné zadat uživatelem místo externího úložiště pro datovou kopii. Systém by sice mohl provádět automatické zálohy na disk počítače, ale význam zálohy umístěné na stejném disku jako jsou originální data, je rozporuplný.

ZÁVĚR V ANGLIČTINĚ

This work was fulfilled by creating the functional application whose benefit is to make the work with information used by Regional Charity Břeclav for its activities easier. This application is very different in comparison with the information systems used today which is caused not only by the special requirements but also the fact that the employment in this field is very specific.

While the other areas that use computer technology can make various demands on their employees, the social services are primarily based on helping people and so the persons with different abilities and skills do this work. The ability to work with a computer is at a lower level that's why the system is designed to look as simple as possible. This also corresponds to the main menu that is still displayed to the user and facilitates the orientation in the software application. The other parts of the application were then designed so that the same types of activities performed were as similar as possible. The result of this effort is the colour differentiation of operations and always the same keys for data management.

Compared with the current situation in the field of information systems, the use of local data storage also makes an unusual impression. As each regional charity is responsible for different areas, works with various people and does different activities, there is no need for a common database. Such a database would have to be structured according to the organization and the stored data would create separate units and these would then have no connection.

This application has fulfilled its purpose and replaced the unsystematic treatment of information which used to be conducted through paper records. As data were kept only on paper there is no need to deal with backward compatibility of the application. In this respect, the function of printing records serves as backward compatibility.

The last part of the application is the implementation of backup data which was again created with regard to the highest degree of simplicity. There is an obvious disadvantage of desktop application, because it is not possible to perform backups to another data server. Moreover, the user has to determine the external storage space for data copy. Although the system might make automatic backups to disk, the relevance of backups placed on the same disk as the original data is contradictory.

SEZNAM POUŽITÉ LITERATURY

1. Adobe Open Source. *Adobe*. [Online] © 2011. [Citace: 12. března 2011.] <http://opensource.adobe.com/>.
2. MXML. *Wikipedia*. [Online] 16. únor 2011. [Citace: 16. březen 2011.] <http://en.wikipedia.org/wiki/MXML>.
3. Adobe Flash Player. *Adobe*. [Online] © 2011. [Citace: 14. březen 2011.] <http://get.adobe.com/cz/flashplayer/>.
4. Flash Player penetration. *Adobe*. [Online] © 2011. [Citace: 12. březen 2011.] http://www.adobe.com/products/player_census/flashplayer/.
5. Adobe AIR. *Adobe*. [Online] © 2011. [Citace: 13. březen 2011.] <http://get.adobe.com/air/>.
6. Flash Builder 4 Standard. *Adobe*. [Online] © 2011. [Citace: 17. březen 2011.] <http://www.adobe.com/cz/products/creativesuite/flashbuilder/>.
7. Download a free trial of Flash Builder 4 Premium Edition. *Adobe*. [Online] © 2011. [Citace: 14. březen 2011.] https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash_builder.
8. Download Adobe Flex 4 SDK. *Adobe*. [Online] © 2011. [Citace: 14. březen 2011.] <http://www.adobe.com/cfusion/entitlement/index.cfm?e=flex4sdk>.
9. ActionScript® 3.0 Reference for the Adobe® Flash® Platform. *Adobe*. [Online] © 2011. [Citace: 11. 5 2011.] http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/.
10. ActionScript. *Wikipedia*. [Online] 9. prosinec 2010. [Citace: 16. březen 2011.] <http://cs.wikipedia.org/wiki/ActionScript>.
11. ELOoffice. *elo-digital*. [Online] © 2008. [Citace: 16. březen 2011.] <http://www.elo-digital.cz/cz/root/produkty/nase-systemy/elooffice/c64>.
12. Event and Festival Management and Database Software. *DataKal*. [Online] © 2011. [Citace: 16. březen 2011.] <http://www.kalendasoft.com/>.
13. Adobe Flex. *Adobe*. [Online] © 2011. [Citace: 16. únor 2011.] <http://www.adobe.com/cz/products/flex/>.

14. SQLite Home Page. *SQLite*. [Online] © 2011. [Citace: 18. únor 2011.]

<http://www.sqlite.org/>.

15. Vývoj aplikací Adobe AIR 1.5 pomocí programu Adobe Flash CS4 Professional .

Adobe AIR 1.5. [Online] © 2008. [Citace: 5. května 2011.]

http://help.adobe.com/cs_CZ/AIR/1.5/devappsflash/.

16. Zákon č. 101/2000 Sb., o ochraně osobních údajů (účinné znění). *Úřad pro ochranu osobních údajů*. [Online] 15. ledna 20011. [Citace: 4. února 2011.]

<http://www.uoou.cz/uoou.aspx?menu=4&submenu=5&loc=20>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PR	Public Relations.
SQL	Structured Query Language.
ERP	Enterprise Resource Planning.
AIR	Adobe Integrated Runtime.
XML	Extensible Markup Language.
SDK	Software Development Kit.
MXML	Macromedia eXtensible Markup Language.
SWF	ShockWave Flash.
MySQL	My Structured Query Language.
E-R	Entity-Relationship.

SEZNAM OBRÁZKŮ

Obrázek 1: Členění Charity ČR	12
Obrázek 2: Informační systém ABRA používaný v organizaci ADRA	16
Obrázek 3: Informační systém Intuo používaný v organizaci ADRA	17
Obrázek 4: Rozšířenost softwarových nástrojů na počítačích připojených k internetu. (3)	21
Obrázek 5: Ikona programu Adobe Flash Player. (4)	20
Obrázek 6: Ikona programu Adobe AIR. (5)	21
Obrázek 7: Vývojové prostředí Flash Builder 4	22
Obrázek 8: Ikona programu Adobe Flash Builder. (7)	23
Obrázek 9: Ikona vývojového prostředí Flex SDK. (8)	24
Obrázek 10: Funkční požadavky na vedení databáze osob	28
Obrázek 11: Případy užití pro vedení databáze osob	29
Obrázek 12: Diagram tříd pro část aplikace týkající se správy osob	30
Obrázek 13: Sekvenční diagram přidání osoby	32
Obrázek 14: Diagram aktivit procesu přidání osoby	34
Obrázek 15: E-R diagram databáze	37
Obrázek 16: Úvodní obrazovka vytvořené aplikace	40
Obrázek 17: Tlačítka pro správu databáze osob	40
Obrázek 18: Formulář pro přidání nové osoby	42
Obrázek 19: Formulář pro vyhledání osoby při vymazání	43
Obrázek 20: Tabulka zobrazující vybrané osoby při vymazání	43
Obrázek 21: Přehled informací o vymazávané osobě	44
Obrázek 22: Formulář pro změnu údajů osoby	45
Obrázek 23: Formulář pro vyhledání osoby při zobrazení a tisku	46
Obrázek 24: Zobrazení informací při vyhledání osoby	47
Obrázek 25: Obrazovka pro vytvoření nového diplomu	48
Obrázek 26: Organizace zdrojových kódů	49
Obrázek 27: Chybové hlášení	60
Obrázek 28: Vytvoření certifikátu pro digitální podpis aplikace	62

SEZNAM TABULEK

Tabulka 1: Zadané vstupy při testování e-mailové adresy.....	59
--	----

SEZNAM KÓDŮ

Kód 1: Maximalizace obrazovky aplikace.....	50
Kód 2: Funkce pro zobrazení požadovaného formuláře.....	51
Kód 3: Vytvoření adresářové struktury pro uložení databáze.....	52
Kód 4: Funkce pro ukládání dat do databáze.....	54
Kód 5: Zápis objektu do souboru.....	55

SEZNAM PŘÍLOH

Příloha 1: CD obsahující zdrojové kódy a instalační verzi aplikace.

Příloha 2: Hodnocení aplikace pracovníci Oblastní charity Břeclav.