VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

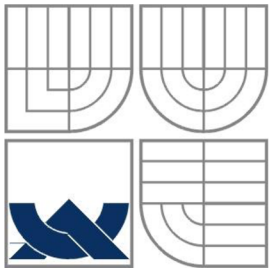
INFORMAČNÍ SYSTÉM PRO EVIDENCI PRACUJÍCÍCH CIZINCŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

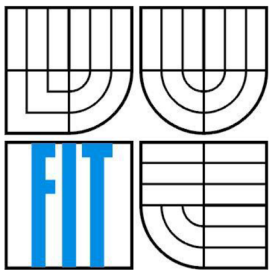
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ HANUŠ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INFORMAČNÍ SYSTÉM PRO EVIDENCI PRACUJÍCÍCH CIZINCŮ

INFORMATION SYSTEM FOR REGISTR OF WORKING FOREIGNERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ HANUŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. MAREK RYCHLÝ

BRNO 2009

Abstrakt

Práce se zabývá problematikou vývoje Informačního systému „na míru“ pro cílového zákazníka. Cílem softwaru je zefektivnit práci v oblasti evidence zahraničních pracujících cizinců na území ČR pro zaměstnance personální agentury a urychlit práci s dokumenty, jež se k této oblasti vztahují. Důraz je kladen na přehlednost a jednoduchost práce uživatele. Informační systém byl rozšířen o temporální data. Toto rozšíření umožňuje procházení historie stavů systému. Vývoj produktu byl veden pomocí metod projektového řízení.

Abstract

Work is considering problems of development Information system made-to-measure for target customer. Objective of this software is to fordize work in area of evidence working foreigners in the Czech republic for employees of the personal agency and speed up the work with the documents that are connected to this area. The main objective is the lucidity and simplicity of the software for users. Information systém was extended for temporal data. This extension allows to browse through historic states of system. Product development was led by methods of project managmet.

Klíčová slova

Informační systém, Delphi, uživatelské prostředí, temporální databáze, multimediální databáze, projektové řízení.

Keywords

Information system, Delphi, user interface, temporal database, multimedia database, project managment.

Citace

Jiří Hanuš, Informační systém pro evidenci pracujících cizinců, diplomová práce, Brno, FIT VUT v Brně, 2009

Informační systém pro evidenci pracujících cizinců

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Mgr. Marka Rychlého. Další informace mi poskytli spolupracovník Ing. Michal Šos, který se zčásti na projektu podílel v podobě komunikace se zákazníkem a také ředitel společnosti Ing. Miroslav Vacula, který mi byl oporou v podobě odborných rad.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Bc. Jiří Hanuš
26. 5. 2009

Poděkování

Rád bych poděkoval mému vedoucímu projektu, Mgr. Markovi Rychlému, mému spolupracovníkovi Michalu Šosovi, řediteli Ing. Miroslavu Vaculovi a následně také zaměstnancům personální agentury Staves za jejich informace ohledně funkčnosti programu.

Obsah

Předmluva

Tato práce vznikla s ohledem na reálné podmínky a postupy firmy, ve které jsem se pracoval po celou dobu mého studia. Jedná se o firmu, která se zabývá vývojem vlastního software, převážně pro správu firemních dokumentů, ale také menších operačních systémů a drobných programů ulehčujících kancelářskou práci našich zákazníkům. Projekt, kterému se budu věnovat, vznikl na zakázku dlouhodobého firemního klienta a jedná se o rozsahově střední informační systém řešící evidenci a tisk všech potřebných dokumentů pro pracující cizince, které náš klient zaměstnává.
Za dobu mé práce ve firmě je to třetí projekt, za který jsem kompletně odpovědný po stránce technické. To znamená provedení celého projektu, od počáteční komunikace se zákazníkem, přes plánování, realizaci, předání a následnou podporu. Většinu práce mám na starosti sám.
V této zprávě popíšu celý proces vývoje Informačního systému. V každé kapitole se zaměřím jak na obecné postupy související s danou fází, ale také na činnosti, které jsem musel vykonávat dál, aby projekt mohl úspěšně pokračovat.
Informační systém je navíc rozšířen o funkcionalitu procházení dřívějších stavů a údajů k danému datu s využitím temporální databáze. Tato funkčnost byla přidána na žádost vedoucího diplomové práce jako další rozšíření.

1 Úvod

Projekt, který tato práce popisuje, je vývoj informačního systému „na míru“ zákazníkovi. Vývoj na míru je v oblasti vývoje software jeden ze dvou majoritních přístupů k vývoji obecně. Vývoj na míru se zaměřuje na zmapování požadavků jednoho klienta (nebo skupiny klientů se stejnými požadavky) a vyvinutý software řeší potřeby právě tohoto jednoho zákazníka. Produkt, přestože obsahuje obecné postupy, je z velké míry upraven přesně na použití v jednom případě a to ho omezuje v nasazení u více zákazníků. Výhodou tohoto přístupu z hlediska kvality software je jednoznačně použitelnost (zvláště pak srozumitelnost a zvládnutelnost) software pro zákazníka, který si vývoj objednal.
Druhým přístupem je „krabicový“ vývoj software, kdy se software vytváří tak obecně, aby bylo možné jeho využití u více zákazníků. Takový produkt nemusí obsahovat všechnu funkčnost potřebnou zákazníkem a také obsahuje řadu funkcí, které zákazník nepotřebuje a které mohou mást jeho uživatele. Výhodou pro zákazníka je pak určitě cena produktu, neboť výrobce kalkuluje s větším množstvím prodejů a jednou kopií software nemusí zaplatit celý vývoj.
Z mých zkušeností, které jsem načerpal za dobu mé pracovní kariéry, bych řekl, že se často začíná s vývojem produktu na objednávku jednoho zákazníka (vývoj „na míru“). Pokud se hotový produkt následně zdá být vhodný pro širší využití, rozšíří se o další obecnou funkcionalitu a vytvoří se jeho „krabicová“ verze. Tento postup zaručuje zaplacení vývoje software a snižuje tak rizika spojená s přímým vývojem „krabicového“ verze. Z toho důvodu si myslím, že oba přístupy mají před sebou dlouhou budoucnost - ať už se jedná o vývoj software obecně, nebo o vývoj informačních systémů.
Diplomovou práci se snažím zpracovat přehledně tak, jak vývoj informačního systému na míru postupně přecházel jednotlivými plánovacími a realizačními fázemi. Realizace samotná je popisována v kapitolách Specifikace požadavků, Analýza, Návrh systému a Implementace. Tyto fáze implementace se několikrát opakovaly, neboť jsem zvolil vývojový iterační přístup. V této práci však záležitosti popisuji jako při vývoji metodou vodopád. Jsou pro to dva důvody. Celá zpráva je přehlednější a čitelnější. A požadavky klienta a následné změny v návrhu se během vývoje neměnily tak mnoho, jak jsem z předchozích zkušeností s projekty podobného rozsahu a zaměření znal. Na některých místech se však budu odkazovat na iterační metodu vývoje a bude to patřičně zdůrazněno.

1.1 Představení společnosti Výrobce

Vývoj informačního systému zajišťuje firma YDS u které jsem zaměstnancem na pozici projekt manažera se zaměřením na vývoj software. YDS je firmou zaměřující se především na komplexní správu firemních dokumentů - jak v dokumentové, tak v elektronické podobě. Hlavním firemním produktem je tak software digitální archiv Xenia, který zajišťuje popisování, ukládání a vyhledávání elektronických dokumentů s návazností na skenovací software ScanImage(tento software byl také dodáván jako rozšíření pro popisovaný informační systém).
Firma se však také zabývá vývojem software na míru. Mezi produkty, na kterých jsem se dříve podílel, patří software pro Firemní audit, IS Finanční kontrola (vyvíjeno pro Ministerstvo zemědělství, projekt jsem popisoval ve své Bakalářské práci) a software Autopůjčovna.
Společnost YDS má 30 zaměstnanců, centrálu umístěnou v Brně a pobočku v Praze.
Firmu YDS budu v práci dále nazývat jako Výrobce.

1.2 Představení Zadavatele

Firma Staves, pro kterou se systém vytváří, je brněnská personální agentura, zaměřující se na zaměstnávání cizinců (především z východu) na území České republiky. Pro firmu Staves budu dále v celé práci užívat označení Zadavatel. Firma zajišťuje kompletní služby pro tyto cizince. Připravuje pracovní povolení, víza, pojištění a registruje tyto osoby na úřadu práce. Po provedení vstupních operací jsou osloveni potencionální zaměstnavatelé, většinou průmyslové továrny nebo stavební firmy, s nabídkou těchto zaměstnanců. Firma následně zprostředkuje zaměstnání osob, resp. pronajme pracovní sílu těchto osob, neboť reálným zaměstnavatelem je stále Zadavatel. Firma také zajišťuje ubytování pracujících osob, pokud nemají možnost vlastního bydlení. Zjednodušeně řečeno, Zadavatel zajišťuje veškerou komunikaci s úřady a zaměstnavateli místo pracujících cizinců. Ty jsou pouze zaměstnanci Zadavatele.
Z výše uvedeného vyplývá, že firma eviduje velké množství informací a produkuje mnoho dokumentů pro komunikaci s úřady. Proto je jejím cílem tyto úkony co nejvíce zefektivnit, neboť se zvyšujícím se počtem pracujících osob je velmi časově náročné manuální zpracování vlastními zaměstnanci.

1.3 Zainteresované osoby

Ke zdárnému dokončení projektu je vždy zapotřebí kooperace většího množství osob. V tomto projektu byla situace poměrně jednoduchá, protože existovaly pouze dvě skupiny osob. Jednou skupinou osob byli zaměstnanci firmy Výrobce, druhou skupinu tvořili zaměstnanci firmy Zadavatele. V každém projektu je důležité pevně stanovit odpovědnosti jednotlivých osob. Určit vedoucí odpovědnou osobu, definovat kdo má jaká schvalovací práva, kdo může jaké dokumenty podepisovat a podobně. V tomto odstavci popíšu všechny zainteresované osoby a určím jejich odpovědnosti, které byly určeny na začátku vývoje projektu.
Ve firmě Výrobce platí pro každý projekt, který se týká vývoje nového software, že má přidělené vždy dvě odpovědné osoby. Jedna osoba zajišťuje technickou stránku projektu a reálně projekt řídí od chvíle podepsání smlouvy. Druhá osoba je odpovědná za obchodní stránku věci, od prvního kontaktu se zákazníkem přes předběžná obchodní jednání a podpis smlouvy, až po předání a ukončení projektu (zaplacení faktury). Tato osoba zodpovídá zákazníkovi za chod celého projektu. Oba tito odpovědní zaměstnanci jsou podřízeni řediteli firmy, který podepisuje ujednané smlouvy, ale jinak projekt nekontroluje. Pouze přebírá výsledky.
V tomto případě, jak jsem již uvedl výše, jsem byl technickým projektovým manažerem projektu. Po obchodní stránce se projektem zabýval výjimečně ředitel firmy Ing. Miroslav Vacula z důvodu osobních vztahů s firmou Zadavatele.
Jednali jsme především s ředitelem společnosti Zadavatele. Ten sice nebude se systémem pracovat, ale jako jediný má práva schvalovat verze, předávací testy a změnové protokoly. On také bude podepisovat smlouvu a projekt přebírat. Dále jsme jednali také s pracovníky společnosti a to s vedoucím evidence osob, pro kterého je systém převážně určen, a také s účetní, jež bude čerpat informace aktuálních zaměstnání cizinců. Tito se podílí na podrobné specifikaci zadání projektu a provádí přebírací testy.

1.4 Definice pojmů

Předtím, než přejdu k popisu systému, bych rád uvedl seznam pojmů používaných v projektu. Jedná se o seznam pojmů, který byl schválen Zadavatelem jako součást dokumentu Specifikace požadavků.

Výrobce – Firma YDS, která zajišťuje vývoj projektu pro Zadavatele.
Zadavatel – personální agentura Staves, která si objednala vývoj softwaru a která je přímým zaměstnavatelem osob.
Zaměstnavatel – rozumí se jím partneři Zadavatele. Od Zadavatele si tyto firmy pronajímají pracovní sílu v podobě osob cizinců.
Místo výkonu práce – adresa, na které se provádí práce osob. Nemusí se vždy jednat o adresu Zaměstnavatele.
Osoba – evidovaný cizinec, který je zaměstnancem Zadavatele a pracuje u Zaměstnavatele jako pracovní síla.
Úřad práce – sada dat, které jsou evidovány pro účely komunikace s úřadem práce.
Formulář – tisková sestava, která čerpá informace z evidovaných dat, případně vybízí před tiskem uživatele k vložení dalších dodatečných dat. Tyto informace jsou tisknuty do podoby papírového formuláře - předlohy.
Profese KZAM – úřední označení prováděné profese.
Kód KZAM – identifikační číslo prováděné profese.

2 Specifikace požadavků

V této kapitole se zaměřuji na popis očekávané funkcionality dodávaného produktu. Tato kapitola vychází z dokumentu Specifikace požadavků, který byl sepsán až po objednání produktu Zadavatelem, a na jehož základě se provádí návrh systému a implementace. Rozhodl jsem se zařadit tuto kapitolu jako druhou v pořadí, i přesto, že v reálném vývoji by nejprve předcházel hrubý návrh plánu projektu jako příprava pro cenovou kalkulaci a odhadu délky trvání projektu. Důvodem je lepší pochopení celé práce. Problematice plánování, startu projektu a jeho řízení se budu věnovat v kapitole 3.

2.1 Obecné požadavky

Program by měl sloužit pro správu informací o cizincích v ČR, jejich zaměstnání, o zaměstnavatelích a ubytování. Hlavní část programu bude tedy adresář osob, ze kterého bude možné přecházet do dalších oken a tisknout tiskové sestavy. Informační systém musí obsahovat řadu funkcí – například hlásit platnost důležitých dokladů cizinců, jako jsou víza, pasy a pojištění. Program musí zachovávat také fotografie dokladů a jednotlivé fotografie osob. Program umožňuje přidávání cizinců do družstva Zadavatele a kontroluje platnost členství. Pro potřeby uživatele bude muset být implementována řada náhledů a vyhledávání jednotlivých cizinců. Množství zpracovávaných dat je asi pro 1000 osob cizinců, program musí být přehledný i pro takovéto množství dat.
Program bude využíván také pro hromadný tisk dokumentů a formulářů, které se musejí vytvořit pro firmu Zadavatele na míru, dle přesných přepisů jejich formulářů. Zadavatel požaduje snadnou editaci formulářů po jejich naplnění metadaty, případně hromadný tisk pro více osob. Formuláře musí být zcela totožné s jejich papírovou předlohou.
Uživatelské rozhraní musí být velmi jednoduché a přehledné. Musí využívat intuitivní ovládání známé z MS Windows XP[1]. Program musí umožňovat simultánní náhled a editaci záznamů pro více uživatelů – až 10. Cílem je co nejvíce zefektivnit práci uživatelů ve firmě a odstranit zbytečné vyplňování formulářů.
Do programu bude také přidána funkcionalita procházení historických stavů systému. V historických stavech systému bude zakázána možnost přidávání a editace všech záznamů.

2.2 Požadavky na evidovaná data

Software eviduje převážně informace o osobách. Těchto informací je celá řada, a proto jsou rozděleny do logických bloků. Prvním blokem dat jsou obecné informace o osobě - jméno, příjmení, datum narození, rodné číslo (které je nepovinné z toho důvodu, že starší osoby cizinců často rodné číslo nemají přidělené) a trvalé bydliště v cizině.
Druhým blokem jsou rodinní příslušníci osoby. Mezi povinné informace patří příjmení otce a matky (převážně kvůli identifikaci cizince), nepovinné informace o dalších příslušnících rodiny jsou evidovány v tabulce rodina. V této tabulce je pro každou osobu uloženo jméno, příjmení, adresa, datum narození a vztah, který mají k evidované osobě.

[1] více na http://www.microsoft.com

Třetí blok dat jsou data vztažená k dokladům osoby - vízum, pas, pracovní povolení a zdravotní pojištění. Kromě čísel jednotlivých dokladů se zde také nachází temporální informace ohledně platností jednotlivých dokladů. V tomto bloku je také sada informací vztažených k úřadu práce.
Ve čtvrtém bloku dat jsou uchovávány informace o aktuálním bydlišti osoby na území ČR. Protože se Zadavatel zabývá také chodem vlastní ubytovny, jsou zde evidována data, zda je zde osoba ubytovaná. Pokud ano, jsou obsažena i data o ubytování.
V pátém bloku se eviduje vztah osoby k družstvu, informace o členství.
V šestém bloku dat se ukládají obrázky vztažené k osobě. Těmi jsou naskenovaná pasová fotografie, kopie občanského průkazu, vízum, pas, anketa v cizím jazyce pořízená při přijetí osoby Zadavatelem.
Evidují se také informace o zaměstnavatelích, které jsou však omezené pouze na identifikační atributy – název a IČ, protože jejich evidenci zajišťuje Zadavateli jiný software. Každý záznam o zaměstnavateli obsahuje seznam míst výkonu práce. To jsou adresy pracovišť, na kterých osoby pracují pro daného zaměstnavatele. Pro každého zaměstnavatele se eviduje seznam zaměstnání osob, které u něho pracovaly, s daty nástupu a odchodu.
Pro účely tisku formulářů se eviduje název, identifikační číslo, DIČ a adresa Zadavatele, stejně tak adresa a jméno jím provozované ubytovny.

2.3 Požadavky na uživatelské prostředí

Zadavatel požaduje jednoduché uživatelské rozhraní pro snadnou orientaci uživatelů. Hlavní řídící okno aplikace se otevírá při startu programu a svým během vymezuje jeho dobu spuštění. Jakmile je vypnuto toto okno, vypne se celá aplikace.
Zde je zobrazen seznam všech evidovaných osob, seřazený abecedně v tabulce. V této tabulce se zobrazují dále informace jednoznačně identifikující osobu – datum narození a stát. Následují údaje o platnosti důležitých dokumentů – platnost víza, platnost pracovního povolení z úřadu práce a platnost pojištění. Zadavatel dále požaduje dva sloupce, které definují zaměstnavatele a informaci od koho zaměstnanec pochází. Tabulka nabízí možnost abecedního třídění dle hodnot v jednotlivých sloupcích. V této tabulce jsou také jasně barevně označeni ti cizinci, jejichž platnost dokladů se blíží k datu vypršení. Tři měsíce před tímto datem se zviditelní hodnota ve sloupci platnosti daného dokladu. Poklepáním na vybraný záznam se otevře nové okno – detail osoby.
Okno pro detail osoby zobrazuje informace týkající se vybrané osoby. Všechny informace je možné editovat. Pro vybrání hodnot z číselníků jsou použity rolovací seznamy. V tomto okně se zobrazují náhledy připojených obrázků k osobě. Jsou to fotografie, kopie pasu, kopie pracovního povolení úřadu práce, naskenovaná ručně vyplněná vstupní anketa a kopie víza. Mezi posledními třemi se přechází výběrem zaškrtávacích políček. S každým obrázkem je možné provádět standardní operace načítání a mazání. Aplikace také spolupracuje s externím programem ScanImage, jež zajišťuje skenování obrázků, které jsou následně přidány do profilu osoby. Poslední uživatelskou potřebou pro práci s obrázky je procházení historie - pro každý obrázek se uchovává seznam všech dřívějších verzí tohoto obrázku s možností starší verze zobrazovat.
Na okno detail osoby jsou navázána okna pro práci s přidruženými rozšiřujícími objekty osoby. Těmi jsou Rodina – evidence rodinných příslušníků vybrané osoby, Úřad práce – podrobnější informace o pracovních záznamech cizince a Zaměstnavatel – partner Zadavatele, u kterého osoba pracuje, případně pracovala. Ve všech těchto oknech jsou pouze základní ovládací prvky pro přidávání, úpravu a mazání.
Z hlavního menu programu je možné zadat tisk dokumentů pro právě vybranou osobu, dokument se vybere volbou položky v menu Tisk. Položka zaměstnavatelé nabízí evidenci partnerů firmy, kteří zaměstnávají evidované osoby. Pro každého partnera je možné zadat několik míst výkonu práce.

Menu nastavení slouží pro definování obecných informací o firmě Zadavatele pro tisk do formulářů. Z hlavního menu je také možné editovat číselníky používané v okně detail osoby.
Okno pro práci a tisk osob přijatých do družstva se spustí z hlavního menu položkou P.m.p. „Staves“, družstvo. Toto okno eviduje pouze osoby přijaté do družstva a zobrazuje zde v tabulce informace týkající se přijetí do družstva. Z tohoto okna je možný tisk dokumentů související s komunikací osoby s družstvem - potvrzení členství, pro komunikaci firmy Zadavatele s úřadem práce – úřad práce, a seznam členů družstva. Je možné opět otevřít a upravovat kartu osoby.
Protože program pracuje až s tisíci evidovanými osobami, je zde implementováno snadné vyhledávání záznamů. Jedná se o jednoduché abecední třízení dle vybraného sloupce, případně kolonka vyhledávání, do které se po stisknutí klávesy znaku kdekoliv v hlavním okně tento znak zapíše a vybere se první osoba, jejíž začátek odpovídá prvnímu znaku. Klávesou *escape* se následně celý řetězec maže.
Pokročilejší vyhledávání v osobách se otevře v okně vyhledávání tlačítkem z hlavního okna. Pokročilé vyhledávání slouží uživatelům pro vytváření přehledných výběrů nad osobami omezenými dle Zadavatelem požadovaných filtrů. Tyto filtry jsou vždy jedním z atributů osoby a mají definované hodnoty, dle kterých si přeje zadavatel vyhledávat. Seznam filtrů je uveden v tabulce 2.1.

ID	Vyhledávací kritérium	Omezující kritéria	Výchozí vyhledané sloupce
1	Platnost víza, platnost úřadu práce	Měsíce	jméno, příjmení, narození, platnost víza, platnost ÚP, platnost pojištění
2	Platnost pasu	rok (2006 - 2019)	jméno, příjmení, narození, platnost pasu
3	Platnost pojištění	Měsíce	jméno, příjmení, narození, platnost VZP
4	Národnost	hodnota zadaná uživatelem	jméno, příjmení, narození, národnost, platnost víza, platnost ÚP
5	Ubytování	hodnota zadaná uživatelem	jméno, příjmení, narození, Ubytování
6	Místo výkonu práce	hodnota zadaná uživatelem	jméno, příjmení, narození, Místo výkonu práce
7	Od koho	rok (2006 - 2019)	jméno, příjmení, narození, od koho, přijetí do družstva

Tab. 2.1.: Tabulka filtrů podrobného vyhledávání

Z důvodu velkého množství dat, která se evidují pro jednu osobu, je nemožné přehledně zobrazit všechny sloupce. Vyhledávání je tedy omezeno některými výchozími atributy, které se vyhledají, a další si uživatel může vybrat v rozšiřujících podmínkách vyhledávání. Pro každý vyhledávací filtr je vždy definován různý seznam výchozích atributů, které se vzájemně liší. Uživatel má dále možnost třídit hodnoty abecedně podle kteréhokoliv zobrazeného atributu ve vyhledávací tabulce. Prvek, podle kterého se třídění provádí, je označen šipkou zobrazenou u názvu sloupce.

2.4 Požadavky na formuláře

Prioritní funkčnost aplikace je tisk formulářů k evidovaným osobám a to jak jednotlivě, tak hromadně. Formuláře jsou vytvořeny jako tisková sestava a odpovídají originálním formulářům přijímaným například úřadem práce. Část formulářů se tiskne na čistý papír, část vyžaduje hlavičkový papír Zadavatele.
Data zapisovaná do formulářů se čerpají ze dvou zdrojů, prvním jsou evidovaná data programem, která se vztahují k právě vybrané osobě. Druhý zdroj dat je ruční zadání uživatelem v době tisku. Tento způsob je náročnější pro uživatele, ale nelze ho plně eliminovat. Uživatel musí vyplnit datum pořízení dokumentu, datum podání žádosti a podobně. Data se program snaží predikovat na základě určitých pravidel a nabízet je „před-vyplněné“ uživateli, ten je však musí vždy potvrdit.
Seznam formulářů je rozdělen na dvě části, první jsou formuláře tisknutelné pro všechny osoby, druhé jsou formuláře určené pro tisk osob v družstvu. Všechny tyto formuláře jsou připojené v příloze.

Obecné formuláře tisknutelné pro všechny osoby:

Žádost o výpis z trestního rejstříku
Formulář se tiskne vybraným osobám v hlavním okně. Všechny informace jsou z databáze osob a z nastavení informací o firmě Zadavatele. Jsou použity tyto atributy osoby: jméno, příjmení, rodné číslo, datum narození, místo narození, pohlaví, státní občanství, jméno a příjemní otce, jméno a příjmení matky, rodné příjmení matky, trvalé bydliště osoby, číslo pasu. Z nastavení firmy se čerpají data, ulice, město, PSČ.

Žádost cizince k povolení zaměstnání
Žádost cizince k povolení zaměstnání je dokument, který slouží jako podklad pro vystavení pracovního povolení pro cizince na území ČR. Tento dokument se předává k posouzení úřadu práce společně s vyjádřením zaměstnavatele nebo vyjádřením obchodní společnosti družstva. Formulář čerpá data z tabulky osoby, tisknutá data jsou: jméno, příjmení, místo narození, státní občanství, trvalé bydliště (ulice, město, oblast, PSČ, stát), číslo pasu, vydávající orgán. Další informace uživatel zadá před tiskem formuláře, datum zahájení a ukončení, místo výkonu práce, profese, profese KZAM – tyto hodnoty jsou nabídnuty uživateli v podobě výběrových seznamů. Do žádosti cizince se pak načtou informace z obecného nastavení tohoto formuláře, to jsou dosažené vzdělání, adresa pro doručování zásilek, adresa pobytu v ČR, zařazení dle OKEČ, vzdělání požadované pro výkon zaměstnání.

Vyjádření zaměstnavatele
Vyjádření zaměstnavatele je dokument vydávaný za účelem získání pracovního povolení od úřadu práce a podává se společně s žádostí cizince. Zaměstnavatelem je v tomto případě myšlen Zadavatel. V tomto formuláři se tisknou data z databáze z tabulek Osoba, Jméno, Příjmení, Místo narození, Státní občanství, Trvalé bydliště (ulice, město, oblast, PSČ, stát). Uživatel musí před tiskem zadat dále datum zahájení a ukončení, místo výkonu práce, profese, profese KZAM – tyto hodnoty jsou nabídnuty uživateli v podobě výběrových seznamů. Pro tento formulář je vytvořen seznam hodnot, jež se vyplňují pro každý formulář stejně. Jsou to název zaměstnavatele, IČ, sídlo, telefon, fax, odpovědný pracovník, uživatel, druh pracovního poměru. Tyto hodnoty uživatel nastavuje v nastavení aplikace a automaticky se používají pro každý dokument.

Vyjádření obchodní společnosti, družstva
Tento dokument je vydáván za účelem získání pracovního povolení od úřadu práce a podává se společně s žádostí cizince, pokud je osoba členem družstva Zadavatele. V tom případě tento dokument nahrazuje vyjádření zaměstnavatele. V tomto formuláři se tisknou data z databáze z tabulek Osoba, Jméno, Příjmení, Místo narození, Státní občanství, Trvalé bydliště (ulice, město, oblast, PSČ, stát). Uživatel musí před tiskem zadat dále datum zahájení a ukončení, místo výkonu práce, profese, profese KZAM – tyto hodnoty jsou nabídnuty uživateli v podobě výběrových seznamů. Pro tento formulář je vytvořen seznam hodnot, jež se vyplňují pro každý formulář stejně. Jsou to název zaměstnavatele, IČ, sídlo, telefon, fax, odpovědný pracovník, uživatel, druh pracovního poměru. Tyto hodnoty uživatel nastavuje v nastavení aplikace a automaticky se používají pro každý dokument.

Formuláře tisknuté pro členy družstva:

Potvrzení o členství, člen družstva
Formulář, který se tiskne při přijetí nového člena, nejčastěji evidované osoby, a zasílá se tomuto novému členu poštou. Všechny údaje se čerpají z tabulky Osoba a jsou to: jméno, příjmení, datum narození, číslo pasu, trvalé bydliště (ulice, město, oblast, PSČ, stát), datum přihlášky, datum přijetí.

Potvrzení o členství, úřad práce
Formulář, který se tiskne při přijetí nového člena, nejčastěji evidované osoby, a zasílá se jako vyjádření na úřad práce. Všechny údaje se čerpají z tabulky Osoba a jsou to: jméno, příjmení, datum narození, číslo pasu, trvalé bydliště (ulice, město, oblast, PSČ, stát), datum přijetí.

Seznam členů družstva
Formulář Seznam členů družstva se tiskne na vyžádání úřadu práce, jako prohlášení družstva a jeho aktuálních a platných členů. Aplikace tiskne všechny osoby, které jsou uloženy jako členové družstva a v seznamu se vyznačí vybraná osoba při tisku. Tisknou se informace o osobách, jméno, příjmení, datum narození, číslo pasu, trvalé bydliště (ulice, město, oblast, PSČ, stát), datum přijetí do družstva. Uživatel navíc musí vybrat datum, ke kterému chce seznam tisknout.

2.5 Rozšíření - Procházení historie

V rámci diplomového projektu jsme se rozhodli společně s vedoucím diplomové práce Mgr. Markem Rychlým zakomponovat do Informačního systému funkcionalitu umožňující procházení historie k vybranému dni (tato problematika nijak nesouvisí s archivem osob). Tímto rozšířením přibude potřeba pracovat v rámci projektu s temporálními daty.
Informační systém bude pracovat ve dvou módech:

Standardní mód - odpovídá požadavkům Zadavatele. Jsou zde povoleny všechny požadované funkce a uživatel není nijak omezován existencí rozšíření o procházení historie. Tento mód tak pracuje zcela autonomně.
Historický mód - při přepnutí informačního systému do tohoto módu se zobrazí okno pro výběr data, ve kterém je možné vybírat dny v minulosti. Po volbě dne dojde k obnovení informačního systému do stavu, ve kterém se v tomto dni nacházel. Mezi dny se jednoduše přechází šipkami umístěními pod výběrem dne. V tomto módu se zakáže přidávání, úprava a mazání všech údajů.

Z důvodu, že informační systém dodávaný Zadavateli tuto funkcionalitu obsahovat nebude, určil jsem si požadavek na snadné oddělení módu historie od informačního systému. Tato operace by neměla mít za následek přepisování vytvořeného kódu projektu.

3 Plán projektu

V následující kapitole se budu zabývat plánováním projektu. Plánování předpokládá, že jsou již známy požadavky klienta na konečný produkt. Plánování začalo po první „technické“ schůzce s klientem. Touto schůzkou se rozumí setkání za účelem projednání bližšího obsahu software, předtím než se určí cena systému a je podepsána smlouva o dodání díla. Toto „předplánování“ s obecnou znalostí hlavní funkcionality slouží pro určení předpokládané doby vývoje a z toho vyplývající ceny produktu.
Po podepsání smlouvy o dodávce následuje fáze projektu Specifikace požadavků, kdy se s klientem projde všechna požadovaná funkčnost do detailů. Po ukončení této fáze se provádí případná úprava plánu projektu a vytvoří se směrný plán, dle kterého se provádí zbylé fáze realizace.
Plánování samozřejmě dále paralelně postupuje s projektem v podobě kontrol plánu a v případě zdržení, změn požadavků a priorit klienta se plán musí přeplánovat a vhodně upravit.
Přestože se během čtyř plánovacích iterací funkčnost mnoho nelišila od požadované podoby, jsou v plánu uvedeny kontrolní schůzky se zákazníkem, během kterých proběhla pro danou fázi specifikace požadavků, a následovala analýza a návrh změn ve vyvíjeném systému.
Obecné informace použité v této kapitole jsem čerpal z literatury zabývající se projektovým řízením: [3] *Efektivní softwarové projekty* a [5] *Řízení projektů,* názvosloví vychází ze standardu [9] *Národní standard kompetencí projektového řízení.*

3.1 Start projektu

První předběžná specifikace zákazníkových požadavků proběhla za účasti všech odpovědných osob obou firem. Z první schůzky vzešel dokument předběžných zákazníkových požadavků, sepsaných přímo na místě během schůzky. Dokument byl nejprve zákazníkem potvrzen podpisem, následně z něj byla sestavena předběžná analýza požadavků klienta, která obsahuje případy užití a návaznosti obrazovek. Tento dokument byl podkladem pro vytvoření smlouvy mezi Výrobcem a Zadavatelem.
Projekt vývoje Informačního systému pro evidenci osob a tisk formulářů pro Zadavatele započal ve firmě Výrobce ke dni 16. 10. 2008. K tomuto datu objednal Zadavatel vývoj softwaru na zakázku. Vývoj projektu byl přidělen, s ohledem k jeho rozsahu, dvěma zaměstnancům Výrobce z technického oddělení s těmito odpovědnostmi: Jiří Hanuš – technický manažer projektu, komunikace se zákazníkem, návrh aplikace, vývoje aplikace, testování, předání. Michal Šos – zástupná komunikace se zákazníkem, vývoj propojení mezi existujícím programem ScanImage pro skenování a vyvíjenou aplikací.
První projektová schůzka následovala dne 23. 10. 2008. Jednalo se o schůzku, kde jsme detailně diskutovali nad požadovanou funkčností aplikace a vytvořili jsme tak základ pro oficiální dokument Specifikace požadavků, kterému se věnuji v kapitole 2. Po vytvoření specifikace požadavků tak bylo možné zahájit fáze Analýza a také podrobné plánování.

3.2 Dohoda o formě komunikace

Před samotným plánováním je povinností manažera projektu stanovit formu komunikace se zainteresovanými osobami. Důležité je určit, které osoby budou informovány, o jakých událostech a jak často. Vytvoří se dokument o domluvených komunikačních kanálech, kde je seznam zainteresovaných osob se všemi domluvenými kontakty.

Tento dokument jsme vytvořili během našeho prvního jednání dne 23.10.2008. Jak již bylo dříve uvedeno, ředitel firmy Zadavatele si vyhradil všechna práva pro potvrzování rozpracovaných fází a přejímaných dokumentů. V tomto dokumentu se však dále specifikovalo, kdo bude ze strany Zákazníka projekt testovat a kdo může mít upřesňující požadavky na jeho funkčnost. Ukázalo se, že bude třeba často komunikovat se zaměstnancem na pozici manažera evidence osob cizinců.
V dokumentu o dohodnuté komunikaci jsme také domluvili měsíční schůzky, kde bylo možné diskutovat o probíhajících pracích na projektu a řešit případné nejasnosti. Protože byl vývoj projektu plánován jako iterační, na těchto schůzkách se ukazovala funkční verze systému a zákazník tuto verzi přebíral potvrzením patřičných předávacích dokumentů.

3.3 Definování cílů projektu

V první fázi plánování projektu je dobré si uvědomit projektové cíle a přínosy. Cíle jsou primárně zamýšlené výstupy projektu. Přínosy jsou důsledky úspěšného dokončení a předání projektu zákazníkovi. V našem případě platí jako cíl projektu vytvořit Informační systém pro evidenci pracujících cizinců dle specifikace Zadavatele. Po dodání můžeme očekávat přínosy projevující se v podobě práce zaměstnanců Zadavatele - snadnější práce s údaji cizinců, méně času stráveném nad vyplňování dotazníků a třeba také zvýšení znalostí práce s PC.
Dále je také dobré si před zahájením dalšího plánování určit trojimperativ projektu a jeho určující osu pro Zadavatele. Trojimperativ se skládá ze tří os. 1. Co se dodává. 2. Do kdy se dodává. 3. Kolik to bude stát. Pro projekt platí tento trojimeprativ:

Co:	Informační systém pro evidenci pracujících cizinců
Kdy:	6.4.2009
Za kolik:	cena je v rámci desetitisíců

Pro úspěšný vývoj projektu je často dobré vědět, jaká osa je pro zákazníka nejdůležitější. Tuto informaci není jednoduché získat, protože zákazník samozřejmě chce mít všechno tak, jak bylo ujednáno (často i dříve, levněji, a s dvojnásobnou funkcionalitou), a nechce se vzdát ani jedné osy. Jde však o to zjistit, jestli je pro něj přednější mít systém, kde nebude např. implementována celá funkcionalita tisku formulářů, ale bude moci ke dni dodání již plnit databázi záznamy cizinců. Nebo zda raději v případě komplikací přeloží datum dodání na později a bude očekávat kompletně hotový software. Vždy je tedy určující jedna osa: kvalita, čas, nebo náklady.
Pro tento projekt jsme se zákazníkem určili během jednoho z prvních setkání prioritní osu čas. Bylo to z toho důvodu, že od data 13.4.2009 Zadavatel počítá s příchodem většího množství nových cizinců a nechce duplikovat práci v podobě ručního vypisování vstupních formulářů a následného přepisování dat do systému, jež plně tyto formuláře nahrazuje.

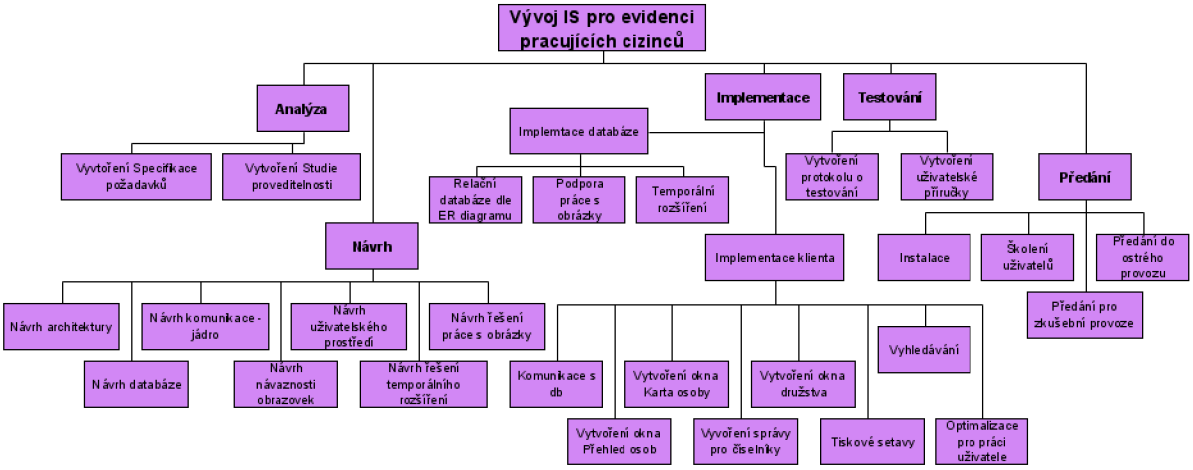
3.4 Určení činností

Samotné plánování projektu začíná při definici WBS (hierarchická struktura činností). Zde je uveden seznam činností, které při provedení vytvoří produkt.

WBS je vhodnou metodou pro rozdělení projektu do pracovních balíků, úkolů nebo činností. WBS snižuje pravděpodobnost, že vám něco vypadne. Jinak řečeno, účelem WBS je zajistit, aby všechny požadované projektové činnosti byly logicky identifikovány a propojeny.

[5] Řízení projektů

V rámci plánování se musí zohlednit všechny činnosti napříč všemi rolemi. Je zde nutné uvést práce analýzy, návrhu, jednání se zákazníkem, implementaci, provádění testů a podobně. Každou činnost je nutné evidovat. Pokud bychom se zaměřili pouze na plánování implementace, snadno bychom mohli vyvodit mylné závěry o době trvání projektu, to by mohlo mít neblahé důsledky v podobě zpoždění projektu i růstu nákladů. Obrázek (obr. 3.1.) ukazuje hierarchickou strukturu činností pro projekt do druhého stupně zanoření. Detailnější pohled je možné nalézt v příloze D.



Obr. 3.1.: Hierarchická struktura činností projektu

3.5 Určení návaznosti činností

Při určení jednotlivých činností nutných pro dokončení celého projektu je nutné určit jejich návaznosti. V této fázi plánování je důležité nezabývat se tím, kdo danou činnost vykonává, jak činnost trvá dlouho, případně jaké zdroje je třeba zajistit. Metoda se také nazývá vytváření síťového grafu. Tato metoda má celou řadu variant (činnosti umístěné v uzlech - AIN, události umístěné v uzlech - EIN, činnosti umístěné na hranách AOA atd.).

Síťový graf je jakékoli z několika grafických zobrazení, které navzájem spojují projektové činnosti (nebo úkoly) a události s cílem zobrazit jejich vzájemné závislosti. Každá činnost nebo událost má vzájemné vazby s předcházejícími, následujícími a souběžnými (paralelními) činnostmi nebo událostmi.

[5] Řízení projektů

Faktem je, že výsledkem této metody je jasná struktura postupu provádění celého projektu.
Po vytvoření samotného grafu přichází na řadu určování délky trvání jednotlivých činností. Je několik metod, jak určit správně délku trvání. Protože se ale jedná o projekt, kde mám již zkušenosti s vývojem a řízením několika podobných projektů, vystačím s metodou expertního odhadu (pohled vývojáře) a s použitím porovnání projektů (pohled projektového řízení).
Výsledkem je tedy síťový graf s určenými délkami činností. Po prozkoumání grafu je možné označit kritickou cestou (sled činností, kde součet délky jejich vykonávání se rovná minimální době provedení celého projektu) a k činnostem na kritické cestě v budoucím plánu přikládat speciální pozornost.
Z pohledu projektu je také dobré určit si v této fázi rozdělení vývoje do iterací. Iterace musí být již spustitelný program funkční tak, aby bylo možné jej předvést zákazníkovi. Výběrem vhodných

realizačních činností tak můžeme vytvořit iterační balíčky implementace a přidat mezi ně činnosti související s vývojovým iteračním postupem (činnosti: předání u zákazníka, analýza nových požadavků, návrh systému pro nové požadavky, schválení nového návrhu zákazníkem).
Síťový graf tohoto projektu je, z důvodu velkého rozsahu, možné nalézt na CD (viz. příloha E) společně se směrným plánem.

3.6 Vytvoření plánu projektu

Předešlé metody jsou prováděny jako fáze předcházející samotnému vytvoření plánu. Je to z toho důvodu, aby se různé pohledy na projekt prováděly logicky za sebou. Pokud bychom měli sestavit od začátku plán projektu s kompletním hierarchickým seznamem činností, jejich provázaností, délkou trvání, přidělením pracovníků a dalších zdrojů, rozdělení rozpočtu, a měli bychom do něj navíc zahrnout řadu omezení (například v podobě pevně daných termínů setkání se zákazníkem), bylo by to velmi složité a výsledek by s velkou pravděpodobností obsahoval mnoho chyb.
Díky výše uvedenému postupu máme v této fázi již uchopitelnou strukturu činností, společně s dobou jejich provádění. Nyní stačí pouze přidělit pracovníky, zavést činnosti do pracovního kalendáře a po drobných úpravách z hlediska požadovaných pevných termínů vytvořit platný směrný plán projektu.
Pro vytvoření plánu použiji kompletní projektový nástroj Microsoft Office Project 2007[1]. Výsledný plán pro tento projekt je přiložen na CD (viz. příloha E).

[1] více na http://www.microsoft.com/products/

4 Analýza

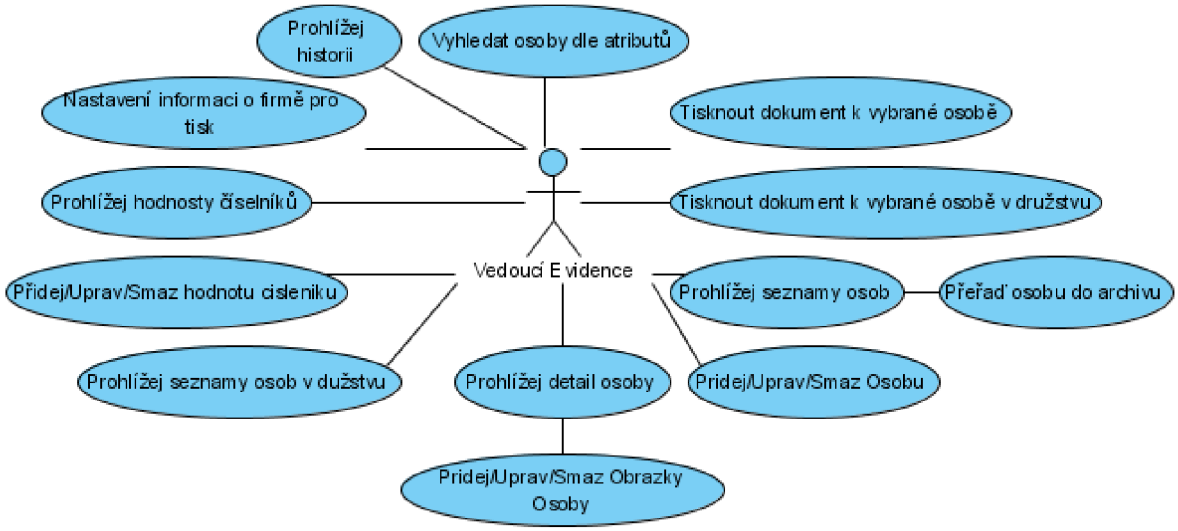
Z hlediska plánování jsou první dva kroky - Specifikace požadavků a Analýza - často zahrnuty do jedné fáze, protože popisují dvě navzájem úzce provázané práce. První je shromáždění informací o produktu od zákazníka, druhá je analýza toho, co zákazník požaduje. Aby bylo možné dále navázat na požadavky klienta v návrhu, je nutná jejich analýza. V této kapitole se tedy budu zabývat rozborem požadavků klienta po teoretické stránce. Zaměřím se na obecné principy a připravím několik možných postupů jak problematiku uvedenou v kapitole 2 řešit. Fáze analýzy probíhala ihned po skončení fáze Specifikace požadavků a zabývá se zpracováním dokumentů zapsaných při komunikaci s klientem.
Výsledkem analýzy byla studie proveditelnosti s doporučením technického oddělení, interní dokument předaný vedení firmy, který dle této zprávy schvaluje provádění dalších prací. Studie pro tento projekt byla schválena, protože jsme během analýzy nenalezli taková rizika, která by mohla reálně ohrozit úspěšné dokončení celého projektu.

4.1 Rozpracování potřeb zákazníka

Vytvořením specifikace požadavků tedy máme k dispozici jasný seznam toho, co informační systém musí obsahovat. Protože je specifikace sepsaná velmi podrobně, budu se zde zabývat pouze obecnými technikami analýzy a uvedu zde diagram případů užití, který ze specifikace požadavků vychází. Specifikace požadavků nám tak bude sloužit jako hlavní zdroj informací pro návrh systému.
Než pokročíme dále k diagramu případů užití, je důležité poznamenat, že Zadavatel nemá zájem o správu rolí a bezpečnost bude řešit interními prostředky, například omezením přístupu zaměstnancům k sytému.

4.1.1 Use-case diagram

Zadavatel požaduje řadu operací prováděné jednou rolí, která bude se systémem pracovat, tou je role Vedoucí Evidence. Tato role odpovídá každému zaměstnanci Zadavatele, který se systémem pracuje. Use-case diagram obsahující klíčové případy užití je uveden níže (obr. 4.1.), celý Use-case diagram je uveden jako příloha A.



Obr. 4.1.: Use-case klíčových vlastností Informačního systému

Operace, které může Vedoucí Evidence provádět, jsem roztřídil do příbuzných kategorií.

Prohlížení: **Prohlížení seznamu osob** - uživatel může prohlížet seznamy všech zadaných osob v systému. Jedná se o osoby aktivní i osoby v archivu.
Prohlížení detailů osob - pro každou osobu je možné otevřít detail osob a zde prohlížet její podrobné údaje.
Prohlížení seznamu osob v družstvu - osoby, které jsou přiřazené k družstvu, je možné prohlížet ve vlastním seznamu.
Prohlížení historie - uživatel může prohlížet historické stavy systému dle vybraných dní v minulosti. Systém pak zobrazuje data tak, jak vypadaly k vybranému dni.
Prohlížení hodnot číselníků - uživatel může zobrazovat hodnoty umístěné v číselnících.

Správa: **Přidávání/Úpravy/Mazání osob** - osobu v systému je možné plně spravovat.
Přeřazení osob do archivu - osobu v systému je možné přesunout do archivu a tím „zneaktivnit“ pro další výběry.
Přidávání/Mazání obrázků osob - pro každou osobu je možné spravovat vlastní obrázky. Obrázky se verzují, při vložení nového obrázku nebo při jeho mazání se jeho stará podoba uloží do historie.
Přidávání/Úpravy/Mazání hodnot číselníků - hodnoty v číselnících je možné spravovat.
Nastavení informací o firmě Zadavatele - je možné plně upravovat nastavení o firmě Zadavatele, které se následně používá při tisku formulářů.

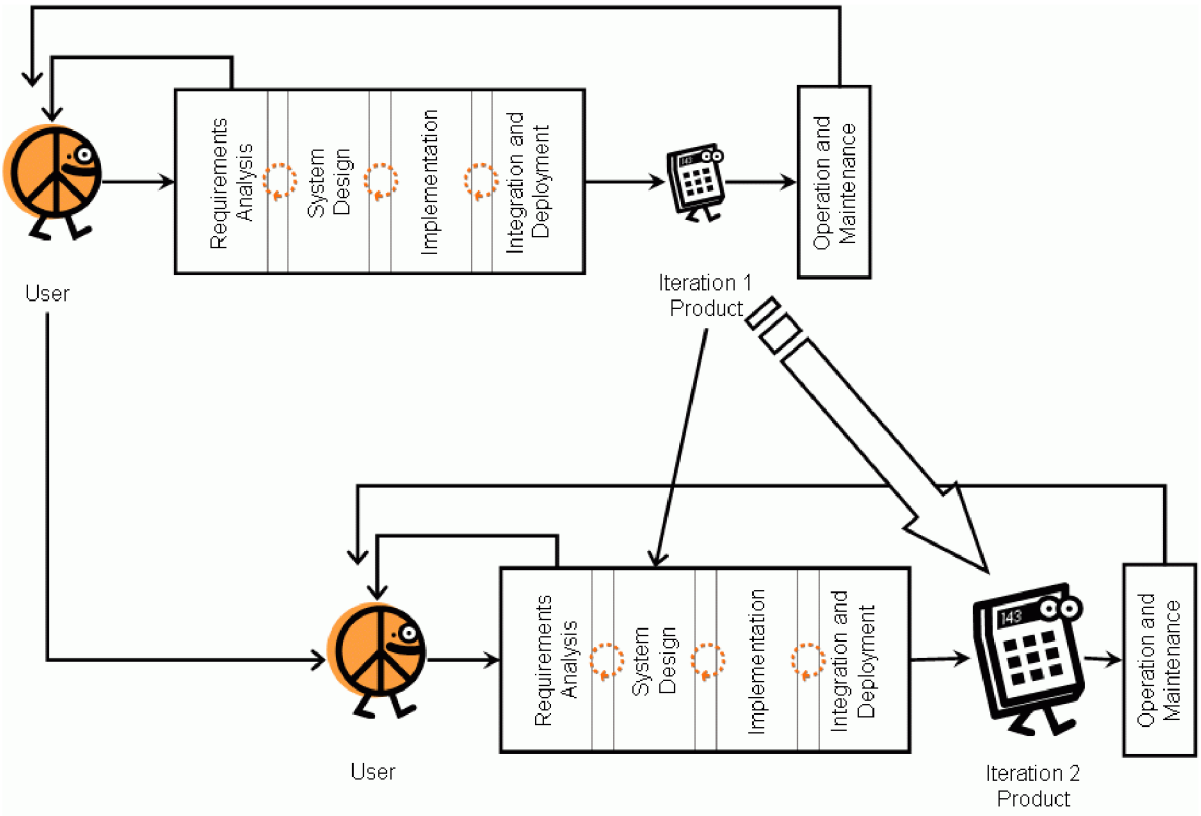
Tisk: **Tisknutí formuláře k vybrané osobě** - pro označené osoby je možné tisknout obecné formuláře. To jsou následující: Výpis z trestního rejstříku, Žádost o pracovní povolení, Vyjádření zaměstnavatele, Vyjádření obchodní společnosti/družstva. Pro tisk formulářů platí, že se do formuláře načtou vždy související údaje k vybrané osobě.
Tisknutí formuláře k vybrané osobě v družstvu - stejné jako obecný tisk formulářů. Pro osoby v družstvu je možné navíc tisknout formuláře: Seznam osob

v družstvu (s označením vybrané osoby), Potvrzení o členství v družstvu - osoba, Potvrzení o členství v družstvu - úřad práce.

Vyhledávání: **Vyhledávání osob** - systém umožní vyhledávání osob. Jednoduché vyhledávání se provádí nad seznamem zadaných osob a při psaní na klávesnici se mění označená osoba dle stisknutých kláves. Pokročilé vyhledávání je pak vyhledávání podle řady kritérií - atributů osob.

4.2 Iterační vývoj

Ze specifikace požadavků plyne, že řada funkcí systému je na sebe postupně vázána. Nemůžeme například tisknout formuláře osob předtím, než je možné osoby evidovat v systému. Z tohoto pohledu jsem postupoval při rozdělení projektu do jednotlivých iterací. Snažil jsem se dodržovat metodiku postupu iteračního vývoje dle iteračního vývoje s přírůstky (obr. 4.2.). Výhodou tohoto přístupu je to, že případné chyby jsou odstraněny rychle, ideálně v přírůstku, ve kterém byly způsobeny. Snadno se také odstraní nedorozumění způsobené špatným pochopením požadavků Zadavatele. Ten je vždy vyzýván, aby jednotlivé přírůstky prohlédl a schválil s případným seznamem úprav, které se musí promítnout do dalšího přírůstku.



Obr. 4.2.: Iterativní životní cyklus s přírůstky (převzato z [6])

*Pro **iterativní životní cykly s přírůstky** (označované někdy také jako evoluční nebo inkrementální) je charakteristické, že zahrnují iterace. Iterace znamená opakovaný průchod fázemi životního cyklu s cílem obohatit vytvářený systém v každé iteraci o nějaké rozšíření či vylepšení, které budeme nazývat přírůstkem. Možná vás napadá, že*

i v případě životního cyklu vodopád vytvářely zpětné vazby iterace. Svým způsobem ano. Ale zásadní rozdíl těchto iterací od iterací iterativních životních cyklů spočívá v tom, že jde pouze o korekci výsledku předchozí fáze. Iterativní životní cyklus vytváří v každé iteraci novou verzi vytvářeného systému. V případě vodopádu je výsledkem fází vývoje jediná verze, která je verzí finální.
Iterativní životní cyklus předpokládá, že výsledkem každé iterace je proveditelný kód, který se nazývá konstrukce (build). Nejde o podsystém vytvářeného systému, ale o celý systém, který ale nemá implementovanou veškerou požadovanou funkčnost, má implementovánu jenom část uživatelského rozhraní, neřeší otázku zabezpečení dat, víceuživatelského přístupu apod. Každá konstrukce přidává k předchozí něco navíc. Význam pojmů inkrement a konstrukce proto splývá.
Dalším důležitým rysem iterativních životních cyklů je, že iterace jsou krátké. Jejich délka se typicky počítá na dny, týdny, nejvýše měsíce. To umožňuje průběžné plánování a spolehlivé řízení projektu. V průběhu iterace lze měřit hodnoty důležité pro plánování (např. produktivitu vývojářů) a využít je při plánování iterace následující.

[7]Návrh Informačních systémů - AIS studijní podpora.

Pro tento projekt jsem se tedy rozhodl rozdělení do čtyř iteračních přírůstků. Přírůstky na sebe logicky navazují a každý přidává část požadované funkcionality Zadavatelem.

- **První iterace** spočívá ve vytvoření jádra aplikace (komunikace se serverem, čtení a zápis do databáze), návrh uživatelského rozhraní (hlavní okno, detail osoby) a vytvoření databáze včetně uložených procedur.
- **Druhá iterace** je rozšíření okna Detail osoby o multimediální data, propojení se skenovacím programem ScanImage, rozšíření Detailu osoby o číselníky a navázané objekty včetně jejich správy, dále pak připojení evidence členů družstva a správy údajů o Zadavateli. Druhá iterace obecně připravuje aplikaci na tisk formulářů.
- **Třetí iterace** se zabývá výhradně formuláři, jejich vytvoření, provázání s evidovanými osobami, plnění a formy tisku.
- **Čtvrtá iterace** je pak vytvoření uživatelsky zjednodušujících funkcí, jako je třídění, vyhledávání, vytvoření ovládání aplikace přes vstupy z klávesnice – rychlé klávesy pro nejdůležitější funkce programu, a další optimalizace pro uživatelskou práci i pro výkon aplikace.

Plán těchto iterací je uveden v kompletním plánu projektu v kapitole 3.

4.3 Temporální data

Většina případů využití databázových systému je v základu temporálního charakteru. To zahrnuje finanční systémy - nástroje pro managment portfolií, účetnictví a bankovnictví; evidenční systémy jako jsou evidence osob, lékařských záznamů a skladů; plánovací systémy pro rezervaci letenek, vlaků, hotelů a managment projektů; a další specifické aplikace jako je monitoring počasí. Takovéto

aplikace spoléhají na temporální databáze, obsahující data platná v závislosti času.
[8] Temporal Database Management.

Temporální databáze je druh databáze, kde jsou mimo statických dat uchovávány také temporální data, bez ztráty jejich temporálního charakteru. Temporální databáze (temporální databázový systém) je databáze zohledňující časové vlastnosti ukládaných dat. Časové vlastnosti obvykle zahrnují časové modely *valid time* (období, ve kterém byl fakt v modelovaném světě pravdivý) a *transaction time* (období, po které je fakt uložen v databázi). Tato dvě časová období by neměla být pro jeden fakt stejná. Příkladem je temporální databáze ukládající data o 18. století. *Valid time* těchto faktů je někde mezi roky 1700 a 1799, ale *transaction time* začíná okamžikem vložení faktů do databáze (například 21. ledna 1998) a nikdy nekončí (obvykle klíčové slovo *forever*).
Databáze se nazývá temporální, pokud uchovává alespoň jeden z těchto údajů. Pokud databáze obsahuje obě tyto dvě složky dohromady, tvoří *bitemporální* model dat.
Pro implementaci temporálních databází je možné využít rozšířené verze jazyka SQL (*Structured Query Language*) TSQL2, vycházejícího z původní verze SQL-92. Mezi hlavní přínosy tohoto rozšíření patří následující změny. Pro práci s temporálními daty využívá původních datových typů DATE, TIME, TIMESTAMP, INTERVAL ale přidává také nový datový typ PERIOD a SURROGATE. Model TSQL2 obsahuje nově celkem 3 základní časové osy. Jedná se o časové osy platnosti, transakce a uživatelsky definovaného času. Do TSQL2 je přidána kromě podpory *snímkové* tabulky (*snapshot*) také podpora *stavových* tabulek (*state*), tabulek *událostí* (*event*), a mnohé další. Obecně platí, že verze TSQL2 je vhodným nástrojem pro práci s temporálními daty, avšak základní funkcionalitu potřebnou pro práci v rozsahu tohoto projektu je možné provést i se základní verzí jazyka SQL (s rozšířením o databázové *triggery*). Vyhneme se tak problémům s komptabilitou systému používaného u zákazníka.

4.3.1 Temporální data v projektu

V projektu se počítá s ukládáním dvou typů temporálních dat. Prvním z nich jsou data vztažená k osobám - platnost dokladů, přijetí do družstva apod. Tyto údaje by se měly ukládat společně se záznamy o osobě, protože to odpovídá jejich logickému charakteru. Zákazník s těmito údaji bude moci provádět všechny základní operace přidávání, úprav, prohlížení a mazání. Systém nebude řešit integritu a logickou návaznost těchto dat. Tyto atributy budeme uvažovat pro další přístup jako pouze informační a nebudeme je zahrnovat do temporálního pohledu.
Druhým příkladem temporálních dat v projektu jsou data přidaná pro rozšíření funkcionality - procházení stavů informačního systému. Tato data budou odpovídat *transaction time* - uvedena v popisu temporální databáze. Jedná se o období, v jakém byla data uložena v databázi a budou obsažena ve všech záznamech ukládaných do databáze.

4.3.2 Pohled na práci s daty

Nejprve si musíme uvědomit, které atributy je nutné pro každý záznam uchovávat, abychom pokryli požadavky na správnou funkčnost. Každý záznam by měl obsahovat datum, kdy byl zaveden do databáze a datum, kdy byl z databáze smazán (rozumějme smazán z informačního systému). Tyto dva atributy nazývejme dále *platnost_od* a *platnost_do* a uvažujeme jejich datový typ TIMESTAMP (může být ale i jiného typu např. DATETIME).

V druhé řadě bychom měli rozmyslet, kdy dochází k změně těchto atributů a jaké operace databáze jsou tedy pro nás podstatné. Následuje tedy seznam instrukcí databáze, které budou ovlivňovat práci s temporálními daty v databázi a jakým způsobem s nimi musíme pracovat.

INSERT - pro INSERT platí, že vkládá nový záznam do databáze. To znamená, že vkládaný prvek dříve v databázi nebyl a jeho čas vytvoření odpovídá době volání této instrukce. Je nám tedy známa doba atributu *platnost_od* vytvářeného záznamu. Atribut *platnost_do* bude v tomto případě inicializován na *NULL* (odpovídá v našem kontextu chápání možnou hodnotu *forever*).

UPDATE - instrukce pro změnu atributů vybraného záznamu. Klíčová instrukce pro práci s temporálními daty, provádějící v podstatě dvě důležité operace. Zneplatnění údajů, které byly dříve platné, a vytvoření nových údajů platných od času zavolání instrukce. Dle tohoto postupu také musíme k instrukci přistupovat. První krok je tedy zavření (a tím zneplatnění) záznamu, nad kterým se instrukce provádí. To se provede změnou atributu *platnost_do* z hodnoty *NULL* na aktuální datum a čas. Druhý krok je následně vytvoření nového záznamu, kde postup je stejný jako u instrukce INSERT. Atribut *platnost_od* se bude rovnat aktuálnímu datu a času (a bude tak schůdný s atributem *platnost_do* právě zneplatněného záznamu), a atribut *platnost*_do bude mít hodnotu *NULL*. Instrukce UPDATE tak vede k velké redundanci záznamů v databázi, kterou budeme muset řešit patřičnými opatřeními.

DELETE - Instrukce mazající záznam z databáze. V našem projektu zcela nepřípustné chování, protože nás zajímají i neplatné údaje zpětně. Tuto instrukci tedy nemůžeme používat (resp. instrukce použita v projektu bude, vysvětlení dále) a nahradíme jí instrukcí UPDATE, která bude volána nad mazanou položkou a jejím atributem *platnost_do*, nastavující hodnotu této položky na aktuální datum a čas.

SELECT - instrukce pro vyhledávání záznamů. S ohledem na temporální data v databázi je však nutné specifikovat, k jakému datu nás záznamy zajímají. Obecně je myslím správné zvolit výchozí přístup vyhledávání aktuálně platných dat. To jsou data, která mají v atributu *platnost_do* hodnotu *NULL*. V případě, že chceme vybrat nějaké dřívější datum, je nutné toto datum specifikovat a vyhledávání provést omezením nad záznamy vyhovujícími výrazu:

platnost_od <= hledané datum <= platnost_do

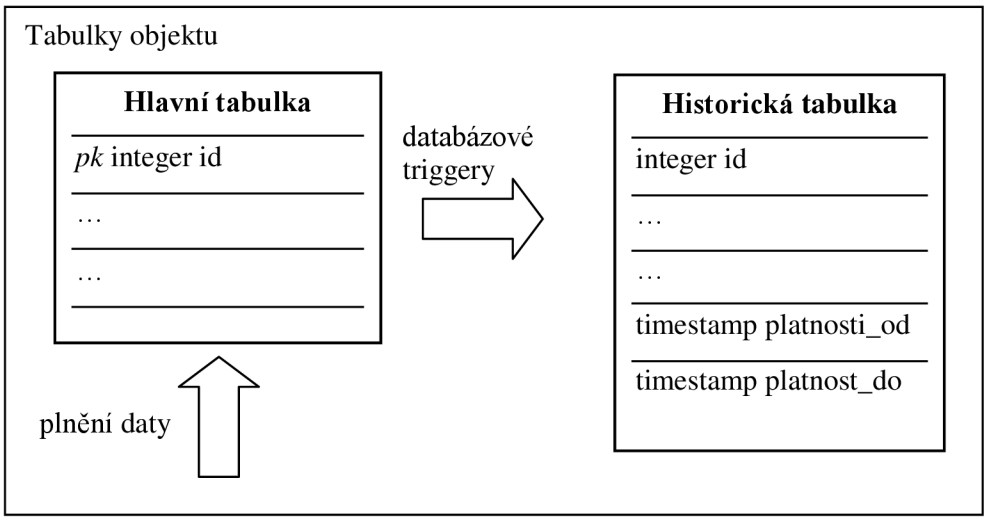
Tím se zajistí vyhledávání v požadovaném čase a také není problém určit požadovaný záznam, pokud používáme identifikaci záznamů identifikačním číslem.

Tímto jsme udělali výčet instrukcí, které pracují s daty v databázi, a jichž funkcionalitu budeme muset upravit směrem k příkazům volaným z aplikace ve výše zmíněném směru.

4.3.3 Uchovávání údajů v databázi

Předtím než začneme vyplňovat tabulky, musíme ještě vyřešit dva logické rozpory. Prvním z nich je fakt, že v případě použití primárních klíčů (nutné pro práci v relačních databázích) se v tabulkách budou duplikovat voláním instrukce záznamy se stejným primárním klíčem. Druhý fakt je ten, že musíme pracovat se smazanými údaji stále uloženými v databázi, které uživatel už nesmí v normálním provozu vidět. Oba problémy jsou si velmi blízké.

V první řadě bych rád zdůraznil použití primárních klíčů identifikačního čísla (atribut *id*) u každého ukládaného záznamu všech objektů bez výjimky. Hodnota atributu *id* objektu musí být unikátní v rámci tabulky a není tedy možné ukládat duplicitní kopie objektů v rámci historie, ani smazané údaje do stejné tabulky, jako aktuální data. Tento koncept velmi komplikuje použití temporálního náhledu na data, ale ukázal se z pohledu mého dřívějšího vývoje informačního systému jako správný a logický. V případě vyvíjeného projektu mluví pro toto řešení také fakt, že v dalším vývoji aplikace instalované u Zadavatele (kde nebude obsaženo rozšíření o temporální data) bude třeba jednotný přístup k záznamům databáze a to bez přepisování kódu dotazů k databázi v případě odstranění rozšíření o temporální data.
Pro řešení obou problémů jsem se inspiroval přístupem řady běžných profesionálních informačních systémů, kde je často žádoucí uchovávat data smazané uživatelem přes uživatelské rozhraní, v databázi pro budoucí případné kritické obnovy.
První z přístupů je v celku jednoduchý - ke každému záznamu přidat atribut pojmenovaný například status datového typu bit určující, jestli je záznam aktivní nebo smazaný. Protože v tomto případě opět dochází k problémům s unikátními klíči, je tento přístup nevhodný. Druhým řešením je vytvoření duplicitní tabulky, do které se smazaná data přesouvají, s příznakem kdy k mazání došlo. V těchto „odpadních“ tabulkách jsou smazány omezení na unikátnosti záznamů.
Pro řešení obou problémů spojených s temporálními daty jsem tedy využil přístup dvou tabulek. V první tabulce, budu ji nazývat dále jako Hlavní tabulka, se neřeší žádné problémy spojené s temporálními daty a záznamy se z databáze mažou přímo, jako by se jednalo o jednoduchý informační systém. Druhá tabulka, kterou dále budu označovat jako Historickou tabulku, obsahuje záznamy, u kterých jsou připojeny atributy *platnost_od* a *platnost_do*. V historické tabulce je také odstraněno omezení unikátních prvků.
Dostáváme tak dvě tabulky pro každý objekt, se kterým v projektu pracujeme. Tento přístup dobře odráží požadavky Zadavatele (resp. rozšíření požadované pro diplomový projekt), kdy jsou dva módy aplikace - v prvním módu se standardně pracuje s informačním systémem a s aktuálními daty (práce s Hlavními tabulkami), v druhém módu se prohlíží historie záznamu informačního systému (dotazy nad Historickými tabulkami). Tento přístup také jednoduše zajistí odstranění rozšíření - po odstranění modulu pro prohlížení historií, který se jako jediný odvolává na historické tabulky, je možné historické tabulky smazat.
Obě tabulky a jejich atributy jsou zobrazeny na obrázku (obr. 4.3.).



Obr. 4.3.: Tabulky objektů pro práci s temporálními daty

4.3.4 Práce s temporálními daty

Funkčnost celého navrženého modelu je závislý na převodu převádění instrukcí uvedených v podkapitole 4.3.2. Navíc je nutné vzít v úvahu řešení se dvěma tabulkami. To znamená, že příkaz se musí provést dle klasického významu na tabulku Hlavní, nad tabulkou Historickou se musí provést modifikovaná funkce instrukce popsaná výše. Pro metodu INSERT to například znamená vložení záznamu do Hlavní tabulky a vložení stejného záznamu do tabulky Historie společně se zadáním atributu *platnost_od* rovnající se času provedení instrukce.
Je možné využít několik přístupů k řešení této problematiky. Protože se jedná o aktivity zcela autonomní od práce uživatele (který samozřejmě příkazy vyvolává), je myslím vhodné celou logiku provádění přenést na stranu serveru. Zde máme k dispozici databázové *triggery*, které se mohou provést vždy při zavolání dané instrukce. Každá z instrukcí INSERT, UPDATE, DELETE nad hlavní tabulkou tak má vlastní *trigger*, který zaručí provedení požadované operace také nad tabulkou Historickou. Zachováme tak integritu údajů, protože všechna práce je automatizována a spojena do databázových transakcí.
Prohlížení dat v informačním systému může probíhat v rámci jednoho příkazu SELECT. V případě přepnutí do módu historie se do speciální pomocné tabulky HISTORY zapíše datum, ke kterému se má provádět procházení, a při přepnutí do standardního módu se pak toto datum nahradí hodnotou *NULL*. Pokud v době volání instrukce SELECT je hodnota data v tabulce HISTORY *NULL*, provádí se hledání v Hlavní tabulce. V opačném případě se vyhledává v Historické tabulce a hledané datum je v tomto případě datum uložené v tabulce HISTORY. Tímto se dosáhne programově elegantního řešení vyhledávání v obou tabulkách, bez přidávání velkého množství zdrojového kódu.

4.3.5 Čištění databáze

V databázi v historických tabulkách dochází k velké redundanci dat. Z toho důvodu je nutné provádět operaci čištění, která stará data vhodně promaže. Z hlediska užitečnosti se mi zdá nejvhodnější metoda mazání dat starých již několik měsíců a zanechání platného stavu pouze k jednomu dni daného měsíce. Např. data budou mazána po třech měsících stáří v systému a bude se zanechávat pouze stav platný k prvnímu dni měsíce.
Mazání pak probíhá zavoláním databázové *stored procedure* s atributem typu *datetime*. Z parametru se vybere hodnota měsíce a roku. Pro tento měsíc se smažou všechny záznamy neplatné k prvnímu dni tohoto měsíce. Smažou se tedy záznamy, pro něž platí následující podmínka:

platnost_od >= 2.vybrany_mesic.vybrany_rok
AND platnost_do < 1.vybrany_mesic+1.vybrany_rok

Při pravidelném volání této procedury s frekvencí jednoho měsíce a s parametrem data o tři měsíce starším, zůstanou v databázi pouze ty starší záznamy, které budou platné k prvnímu v měsíci. Volání této procedury se musí provádět automaticky, vždy k prvnímu dni v měsíci během nočních hodin.

4.3.6 Praktické řešení popsaného postupu

Pro lepší představu o použité metodice uvedu kód pro práci nad testovacím objektem *Osoba*. Mějme tedy jednoduchou tabulku *Osoba*. Vytvoření této tabulky je následující:

```
CREATE TABLE Osoba {
        id              INTEGER NOT NULL,
        jmeno           VARCHAR (30) NOT NULL,
        prijmeni        VARCHAR (30) NOT NULL,
        plat            INTEGER,
        PRIMARY KEY (id)
}
```

K této Hlavní tabulce v databázi musíme vytvořit také tabulku Historickou, do které se následně ukládají informace o temporálních datech. Přidáme proto dva nové atributy a odstraníme primární klíč. Kód pro vytvoření:

```
CREATE TABLE History_Osoba {
        id              INTEGER NOT NULL,
        jmeno           VARCHAR (30) NOT NULL,
        prijmeni        VARCHAR (30) NOT NULL,
        plat            INTEGER,
        platnost_od     DATETIME,
        platnos_do       DATETIME
}
```

Tyto dvě tabulky na sebe samozřejmě nijak nevidí - nejsou propojeny pomocí cizích klíčů. Obsah tabulek se kopíruje přímo. Data je možné spravovat (vytvářet, upravovat, mazat) pouze v standardním módu aplikace. Instrukce INSERT/UPDATE/DELETE se tedy volají pouze nad Hlavní tabulkou. Při tomto volání se však vždy spustí příslušný databázový *trigger* a provede upravenou funkci i nad tabulkou Historickou. Následuje kód pro vytvoření těchto tří *triggerů*:

```
CREATE TRIGGER H_OsobaInsert ON Osoba AFTER INSERT
AS
BEGIN
        DECLARE @id INTEGER
        DECLARE @jmeno VARCHAR(30)
        DECLARE @prijmeni VARCHAR(30)
        DECLARE @plat INTEGER

        SELECT @id = (SELECT id FROM Inserted)
        SELECT @jmeno = (SELECT jmeno FROM Inserted)
        SELECT @prijmeni = (SELECT prijmeni FROM Inserted)
        SELECT @plat = (SELECT plat FROM Inserted)

        INSERT INTO H_Osoba(id, jmeno, prijmeni, plat, platnos_od, platnost_do)
                VALUES (@id, @jmeno, @prijmeni, @plat, getdate(), NULL)
END
```

Trigger nad instrukcí INSERT. *Trigger* se volá po provedení INSERT nad tabulkou *Osoba*. Vytvoří proměnné, do kterých uloží vkládané hodnoty. Hodnotami těchto proměnných pak naplní nový záznam v tabulce H_*Osoba*. Hodnota atributu *platnost_od* je generován funkcí *getdate(), platnost_do* je nastaveno *NULL*.

```
CREATE TRIGGER H_OsobaUpdate ON Osoba AFTER UPDATE
AS
BEGIN
        DECLARE @id INTEGER
        DECLARE @jmeno VARCHAR(30)
        DECLARE @prijmeni VARCHAR(30)
        DECLARE @plat INTEGER

        SELECT @id = (SELECT id FROM Inserted)
        SELECT @jmeno = (SELECT jmeno FROM Inserted)
        SELECT @prijmeni = (SELECT prijmeni FROM Inserted)
        SELECT @plat = (SELECT plat FROM Inserted)

        UPDATE H_Osoba SET platnost_do = getdate()
                                WHERE (@id = id) AND (platnost_do IS NULL )

        INSERT INTO H_Osoba(id, jmeno, prijmeni, plat, platnos_od, platnost_do)
                VALUES (@id, @jmeno, @prijmeni, @plat, getdate(), NULL)
END
```

Trigger nad instrukcí UPDATE pracuje velmi podobně jako *trigger* první. Jediným rozdílem je to, že před vložením nového záznamu do tabulky *H_Osoba* (odpovídajícímu nově editované hodnotě v tabulce *Osoba*), je zneplatněn platný záznam předchozí. Ten je identifikován pomocí *platnost_do* (který musí být *NULL*), a shodným *id* s modifikovaným záznamem.

Trigger nad instrukcí DELETE:

```
CREATE TRIGGER H_OsobaDelete ON Osoba AFTER DELETE
AS
BEGIN
        DECLARE @id INTEGER
        SELECT @id = (SELECT id FROM Deleted)
        UPDATE H_Osoba SET platnost_do = getdate()
                                WHERE (@id = id) AND (platnost_do IS NULL )
END
```

Po zavolání instrukce DELETE nad tabulkou *Osoba* (kde opravdu dojde k vymazání záznamu), se provede jeho „zneaktivnění“ v tabulce *H_Osoba*. Zneplatněný záznam má *id* shodné s právě smazaným id a v *platnost_do* je hodnota *NULL*. Hodnotu atributu *platnost_do* tedy modifikujeme, aby v ní bylo uloženo aktuální datum a čas (opět funkce *getdate*()).

Uvažujme dál, že proběhlo plnění databáze, a byly nad ní provedeny určité modifikace. Sled těchto operací by mohl být následující:

Datum 1. 1. 2009

```
INSERT INTO Osoba(id, jmeno, prijmeni plat) VALUES (1, “Jan“, “Novák“, 10000);
INSERT INTO Osoba(id, jmeno, prijmeni plat) VALUES (2, “Petr“, “Procházka“, 25000);
```

Stav tabulek:

Tabulka:	**Osoba**	**Datum:**	**1.1.2009**
Id	*jmeno*	*prijmeni*	*plat*
1	Jan	Novák	10000
2	Petr	Procházka	25000

Tabulka:	**H_Osoba**	**Datum:**	**1.1.2009**		
id	*jmeno*	*prijmeni*	*plat*	*platnost_od*	*platnost_do*
1	Jan	Novák	10000	1.1.2009	NULL
2	Petr	Procházka	25000	1.1.2009	NULL

Datum 2. 1. 2009
INSERT INTO Osoba(id, jmeno, prijmeni plat) VALUES (3, "Tomáš", "Mladý", 50000);
UPDATE Osoba SET plat = 55000 WHERE id = 1;

Stav tabulek:

Tabulka:	**Osoba**	**Datum:**	**2.1.2009**
Id	*jmeno*	*prijmeni*	*plat*
1	Jan	Novák	55000
2	Petr	Procházka	25000
3	Tomáš	Mladý	50000

Tabulka:	**H_Osoba**	**Datum:**	**2.1.2009**		
id	*jmeno*	*prijmeni*	*plat*	*platnost_od*	*platnost_do*
1	Jan	Novák	10000	1.1.2009	2.1.2009
2	Petr	Procházka	25000	1.1.2009	NULL
1	Jan	Novák	55000	2.1.2009	NULL
3	Tomáš	Mladý	50000	2.1.2009	NULL

Datum 3. 1. 2009
UPDATE Osoba SET plat = 15000 WHERE id = 3;
DELETE FROM Osoba WHERE id = 2;

Stav tabulek:

Tabulka:	**Osoba**	**Datum:**	**3.1.2009**
Id	*jmeno*	*prijmeni*	*plat*
1	Jan	Novák	55000
3	Tomáš	Mladý	50000

Tabulka:	**H_Osoba**	**Datum:**	**3.1.2009**		
Id	*jmeno*	*prijmeni*	*plat*	*platnost_od*	*platnost_do*
1	Jan	Novák	10000	1.1.2009	2.1.2009
2	Petr	Procházka	25000	1.1.2009	3.1.2009
1	Jan	Novák	55000	2.1.2009	NULL
3	Tomáš	Mladý	50000	2.1.2009	3.1.2009
3	Tomáš	Mladý	15000	3.1.2009	NULL

Z tohoto příkladu plynou dvě výhody celého řešení. V tabulce *Osoba* jsou umístěna pouze platná data, se kterými je možné v Informačním systému volně pracovat bez nutnosti řešit temporální logiku těchto dat. Aktuální seznam všech záznamu dostaneme pomocí volání *SELECT * FROM Osoba*.
Naopak kompletní informace jsou uloženy v tabulce *H_Osoba*, kdy platí, že pokud se jednou zadá informace do Informačního systému, už nikdy se nemůže ztratit. Pokud bychom chtěli obnovit stav ke dni 2.1.2009, voláme instrukci SELECT takto:

SELECT * FROM H_Osoba WHERE platnost_od <= 2.1.2009 AND 2.1.2009 <= platnost_do

Získali bychom tyto údaje o osobách:

Jan	Novák	55000
Petr	Procházka	25000
Tomáš	Mladý	50000

Pokud bychom přidali do příkladu také navázanou tabulku *Auto*, ve které by byl povinný cizí klíč *id_osoba* směřující na *id* v tabulce *Osoba* (objekt *Auto* by byl závislý na objekt *Osoba*), nebylo by nutné tento přístup nijak modifikovat. Klíčové je v tomto případě mazání dat, které se provede vždy nejprve nad tabulkou Hlavní, a pokud je zde zajištěna integrita dat, projeví se změny v tabulkách Historických.

4.3.7 Zhodnocení metodiky

V mém řešení rozšíření informačního systému o temporální data používám stejný systém ukládání dat, který je použit v předcházející podkapitole. Metoda je bezpečná do té míry, do jaké jsou bezpečné příkazy prováděné nad hlavními tabulkami. To je otázka návrhu celého jádra informačního systému, tímto se budu zabývat v kapitole 5.

4.4 Multimediální data

Multimediální databáze je databázový systém, který dokáže spravovat multimediální data. Multimediální data jsou nestrukturovaná data s často velkým objemem. Příkladem jsou 2D obrázky, 3D modely, video, audio, dokumenty a jejich kombinace. Tato data jsou označena metadaty, jež specifikují, o jaká data se jedná, případně jaký je jejich logický název, k čemu se reálně váží apod.
Vyhledávání v multimediálních - obrázkových databázích je rozdílné vůči relačním databázím. Především jsou možné dvě metody vyhledávání.

První metodou je vyhledávání na základě textového popisu obrázku, podle klíčových slov. Vyhledávání obrázku se provádí pomocí vyhledávání v metadatech popisující obrázek a následným vybráním obrázku, jehož popis odpovídá vyhledávanému klíčovému slovu. Výhodou této metody je vyhledávání dle „sémantiky obrázku“ (dle toho co na obrázku je). Tato metoda také předpokládá platný obsah připojených metadat. Nevýhodou této metody pak je pracnost tvorby popisu a vliv subjektivního popisu tvůrce popisu obrázku. Nepředejde se situaci kdy i vhodně vytvořený popis obrázku zapříčiní nesprávné výsledky vyhledávání. Použití této metody jsou oborově specializované a ne příliš rozsáhlé databáze.

Druhou metodou vyhledávání je podobnostní vyhledávání obrázků. Dotaz má typicky podobu vzorového obrazu nebo náčrtku (obrazu dotazu) a z databáze se vybírá jeden nebo několik obrazů, která jsou mu nejvíce podobné (nevyhledává se rovnost). Výhodou této metody je fakt, že hodnoty vlastností, které se porovnávají, jsou odvozeny z obrazu dotazu i obrazů v databázi automaticky. Tuto metodu vyhledávání však limituje absence obecně použitelné metody, která by odvodila reprezentaci vyjadřující sémantiku obrazu. Pracuje se pouze s vizuálními vlastnostmi. Dva základní přístupy tohoto vyhledávání jsou metrický a transformační. Metrický - porovnání dvou obrazů se provádí vyhodnocením podobnostní funkce u reprezentace těchto obrazů (typicky tzv. vektory rysů (feature vector)). Výhodou je jednoduchost a výpočetní efektivnost – indexace. Tato metoda však vytváří stejnou metriku podobnosti pro všechny aplikace a uživatele – nerespektuje subjektivnost vnímání. Transformační - podobnost dvou obrazů je vyjádřena cenou transformace prvního obrazu na druhý nebo naopak. Tato metoda je pružnější a do určité míry lze promítnout subjektivnost vnímání podobnosti, má však větší výpočetní složitost a nižší efektivnost.

[4] Pokročilé databázové systémy - Multimediální databázové systémy

4.4.1 Řešení multimediálních dat v projektu

Vyvíjený informační systém se zabývá problematikou multimediálních - obrázkových dat a jejich databází. Pro každou evidovanou osobu uchovává fotografie, kopii víza, pasu, občanského průkazu a naskenovaný obraz dokumentu Anketa. Ten je vyplněn při vstupním pohovoru ručně cizincem často v cizím jazyce. V prvotních požadavcích na aplikaci byl předpoklad externího úložiště těchto obrázků (nejednalo by se tedy o multimediální databázi), protože uchovávat obrázky v databázi má za následek vysoké zvýšení jejího obsahu. Tento problém je o to závažnější z pohledu finanční nákladnosti, neboť při překročení velikosti databáze 4GB je nutné zakoupit placenou licenci MSSQL[1]. V průběhu vývoje jsme se však rozhodli přesto přejít na tuto variantu z důvodu ochrany dat před neautorizovaným přístupem uživatele a jeho možnou manipulaci s fotografiemi uloženými v externím archivu.

Ukládání obrázků se v prostředí MS SQL provádí pomocí datového typu BLOB (*binary large object*). Jedná se o datový typ blíže nespecifikovaných binárních dat. Jako BLOB se ukládají obecně také kromě obrázků zvukové záznamy, videa a jakákoliv jiná data, která jinak bývají uložena v samostatných souborech. BLOB je tedy obecný datový typ, který explicitně vyjadřuje, že databáze nemá informace o tom, jak jeho obsah interpretovat. V prostředí použité verze MS SQL 2008 je možné pro uchování obrázků použít datový typ IMAGE, který se od typu BLOB neliší.

V databázi bude tedy vhodné vytvořit oddělenou tabulku s vlastním primárním klíčem id a s atributem názvu obrázek datového typu IMAGE. Tyto záznamy pak propojíme přes tabulku verzování (kde bude uložen také typ obrázku a verze) k tabulce *Osoba*, a připojíme tak k obrázku vztažená *metadata*.

V případě vytvářeného informačního systému není nutné implementovat vyhledávání nad podobností obrázků (a ani případné rozšíření se Zadavateli nezdá pro jeho potřeby využitelné). Zůstává tedy vyhledávání v závislosti na *metadatech* - a to je samotné vyhledávání osob. Proto pro naše účely bude dostačující, pokud vyřešíme ukládání obrázků v databázi a jejich následné čtení.

[1] více na http://www.microsoft.com/cze/windowsserversystem/sql

5 Návrh systému

V této kapitole se budu zabývat podrobným návrhem informačního systému. Zaměřím se na samotnou architekturou aplikace, navržením vhodné databázové struktury, definováním patřičného uživatelského prostředí, návrhem formulářů a vložením temporálních a multimediálních dat do systému. Pro práci jsem využil vývojového prostředí pro UML programu Visual Pararadigm for UML[1]. Ukázky diagramů související s návrhem budu vkládat k patřičným částem textu, v příloze se pak nachází tyto diagramy kompletní pro celý informační systém.

5.1 Návrh architektury projektu

Pro informační systém je navržen jednoduchý model klient – server. Jedná se o síťovou architekturu, která odděluje klienta (aplikaci s grafickým uživatelským rozhraním) a server (který často obsahuje sdílené prostředky). Klient - server popisuje vztah mezi dvěma aplikacemi, v nichž klient žádá o služby server. Klient je tak aktivní (posílá žádosti a čeká na odpověď) a server pasivní (vyřizuje vzniklé požadavky klienta). Tento typ architektury je také označován jako *two-tier* (dvouřadová).
Klientem můžeme rozumět *tenkého klienta* - aplikaci umožňující práci pomocí grafického rozhraní, která se provádí na serveru, případně *tlustého klienta* - aplikace vykonává logiku systému a se serverem komunikuje za účelem získání sdílených prostředků. *Tenký klient* často nevyžaduje instalaci na klientských počítačích, většinou je využíván webový prohlížeč pro připojení k serveru. U tlustého klienta je instalace většinou nezbytná, zato však může nabídnout komfortnější uživatelské prostředí.
Budu tedy využívat architekturu klient - server. Zvolil jsem přístup pomocí *tlustého klienta* s databázovým serverem (model tenký klient - server je však využíván dle potřeb zadání a to pro temporální rozšíření).

5.2 Návrh databáze

Nejprve přejdu k návrhu databáze. Jedná se o vlastní základní logiku návrhu, která poskytne pro další práci jasný ucelený přehled o navrhovaném systému v podobě návaznosti použitých objektů. Vývoj databáze je součástí řešení první iterace projektu - jedná se tedy o jednu z prvních plánovaných implementačních úloh.
V první řadě si musíme uvědomit, které objekty budou v systému použity. Objektům následně přiřadíme potřebné atributy. Druhým krokem bude určení vztahů mezi objekty a jejich kardinalita.

5.2.1 Objekty databáze

V návrhu objektů jsem dodržoval pravidlo, že každý objekt má vlastní id. Toto id je interní a je otázkou bezpečnosti, aby toto číslo neznal uživatel. Pomocí *id* se na sebe odkazují různé objekty. *Id* je generováno databází pomocí *sekvence*.

[1] více na http://www.visual-paradigm.com/product/vpuml/

Hlavním objektem celého systému je objekt Osoba. Vyplývá to ze specifikace požadavků (kromě toho také ze jména systému). Objekt Osoba má velké množství atributů sloužící z velké části pouze jako zdroj informací uživateli. Seznam atributů je možné vidět na obrázku objektu (obr. 5.1.):

Osoba		
+id_osoba	integer(10)	Nullable = false
jmeno	varchar(100)	Nullable = false
prijmeni	varchar(100)	Nullable = false
rodneprijmeni	varchar(100)	Nullable = true
pohlavi	varchar(20)	Nullable = false
ulice	varchar(200)	Nullable = false
mesto	varchar(100)	Nullable = false
psc	varchar(7)	Nullable = true
statZkratka	varchar(50)	Nullable = false
smercislostatu	varchar(20)	Nullable = false
ucelpobytu	varchar(1000)	Nullable = true
cisloprofese	varchar(200)	Nullable = false
datumnarozeni	date	Nullable = false
rodnecislo	varchar(20)	Nullable = true
jmenootce	varchar(200)	Nullable = false
jmenomatky	varchar(200)	Nullable = false
rodnematky	varchar(200)	Nullable = false
rodinnystav	varchar(50)	Nullable = false
statniobcanstvi	integer(100)	Nullable = false
mistonarozeni	varchar(300)	Nullable = false
vydavajiciorgan	varchar(300)	Nullable = false
archiv	integer(2)	Nullable = false
telCR	varchar(50)	Nullable = true
telCizina	varchar(50)	Nullable = true
cislopasu	varchar(50)	Nullable = false
platnostpasu	date	Nullable = false
platnostpasuod	date	Nullable = false
platnostviza	date	Nullable = true
datumPrijeti	date	Nullable = true
datumStanov	date	Nullable = true
cislopojisteni	varchar(50)	Nullable = true
platnostpojisteni	date	Nullable = true
cisloup	varchar(100)	Nullable = true
platnostup	date	Nullable = true
platnostupod	date	Nullable = true
vydanehoup	date	Nullable = true
vydal	varchar(200)	Nullable = true
odkoho	varchar(255)	Nullable = true
clenemDruzstva	bit	Nullable = true
odchodzdruzstva	date	Nullable = true
#cis_Profeseid_cis_profese	*integer(10)*	*Nullable = false*
#cis_Zadavatelid_cis_zadavatel	*integer(10)*	*Nullable = false*
#cis_Ubytovanid_cis_ubytovani	*integer(10)*	*Nullable = false*

Obr. 5.1.: Výřez z ER diagram: Objekt Osoba

Seznam atributů s jejich popisem je uveden ve Specifikaci požadavků. Z obrázku je patrné, jaké datové typy byly atributům přiděleny a také jestli je atribut povinný. Atributy, které budou ovlivňovat chod systému:

jmeno + prijmeni - identifikace položky uživatelem systému
rodnecislo / datumNarozeni - protože ne všechny osoby budou mít rodné číslo, je povinné pouze datum narození. Společně s celým jménem by tyto údaje měly stačit pro jasnou identifikaci osoby (při shodě obou údajů je vždy možné ověřit jejich totožnost v detailu osoby).
archiv - určuje zdali je osoba aktivní.
datumPrijeti, odchodzDruzstva - pokud je vyplněno datum přijetí a není vyplněno datum odchodu (případně je datum odchodu starší než datum přijetí) je osoba uvedena jako člen družstva. V opačném případě v družstvu uvedena není.

platnostViza, platnostPojisteni, platnostup - platnosti dokladů, jež se kontrolují v seznamu osob. Pokud do data v tomto atributu zbývá méně než tři měsíce, je příslušný atribut označen.

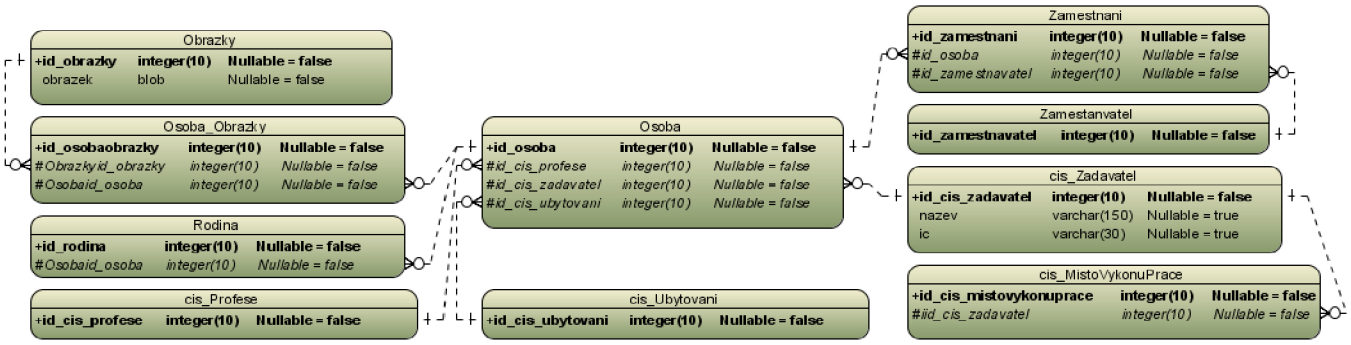
Objekt Osoba pracuje s řadou číselníků, pro každý číselník je třeba vytvořit vlastní objekt. Číselníkem zde označuji jednoduché objekty, které obsahují id a další hodnotu (hodnoty). Číselníky se často používají v kombinaci s ovládacím prvkem *dropdown* list, umožňující výběr právě jedné hodnoty. Smyslem číselníků je příprava často používaných hodnot „atributů" jiného objektu. Tyto hodnoty jsou uloženy v číselníku a uživatel je nemusí stále vyplňovat manuálně. Jedná se tak vlastně o „atributy" jiného objektu.

Ubytování - obsahuje adresy ubytování pro osoby.
Profese + KZAM - vykonávaná profese a její kód.
Zadavatel - název a IC firem Zadavatele. Každý záznam tohoto číselníku obsahuje ještě seznam adres míst výkonu práce:
Místo výkonu práce - adresy výkonu práce.

Dalším objektem použitým v systému je objekt **Rodina**. Objekt zastupuje jakéhokoliv rodinného příslušníka evidované Osoby. Objekt má atributy určující jméno, příjmení, datum narození, rodné číslo, národnost a profesi pro identifikaci daného rodinného příslušníka. Atribut *vztah* určuje v jakém je vztahu s evidovanou osobou (otec, matka, syn atd.).
Pro evidenci zaměstnání osob bude zapotřebí dvou objektů - **Zaměstnavatel** a **Zaměstnání**. Zaměstnavatel bude reprezentovat partnery Zadavatele a bude mít pouze dva atributy - *jméno* a *ic*. Zaměstnání pak bude objekt popisující všechna zaměstnání osob poskytovaná zavedenými Zaměstnavateli. Každé zaměstnání obsahuje atributy *odkdy* a *dokdy*, určující kdy byla Osoba zaměstnána.
V databázi budou také uloženy tabulky pro temporální data a tabulky pro obrázky. Každým z těchto se věnuji v patřičném odstavci níže.

5.2.2 ER - diagram

Následuje zjednodušený diagram ER (obr. 5.2.), který neobsahuje atributy jednotlivých objektů. Celý diagram je k dispozici v příloze B.



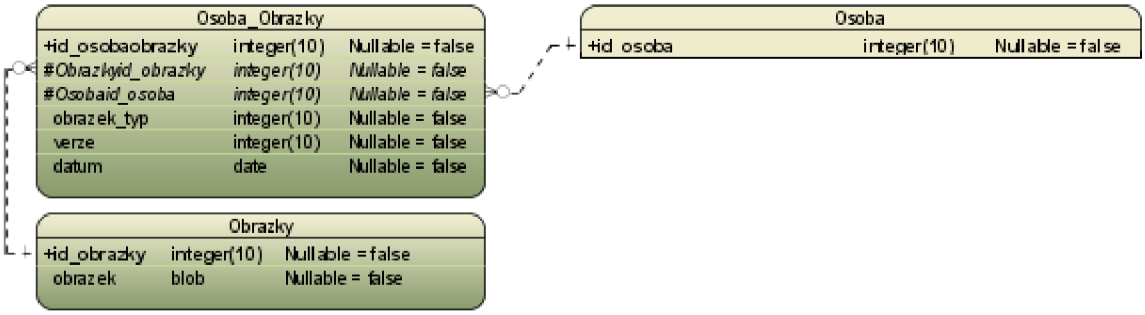
Obr. 5.2.: Zjednodušený ER diagram

Z obrázku (obr. 5. 2.) jsou patrné vztahy mezi entitami. Objekty číselníku jsou navázány na objekt Osoba vždy s kardinalitou *one to many* (vazby *cis_Ubytovani - Osoba, cis_Profese - Osoba*, *cis_Zadavatel - Osoba*). Odpovídá to ideologii číselníků - jedna hodnota je použita vícekrát. Tato vazba tedy přidává do objektu Osoba cizí klíče na tabulky číselníků.

Pro číselník *cis_Zadavel* platí také vazba *one to many* s objektem *cis_MistoVykonuPrace*.
V diagramu se počítá s využitím „propojovací“ tabulky, která řeší problematické spojení objektů *many to many*. Touto tabulkou je *Zamestnani*, které „propojuje“ objekty *Zamestnavatel* a *Osoba*. V tabulce *Zamestnani* tak musí být, kromě dalších dat, také cizí klíče na tabulku *Osoba* a *Zamestnavatel*. Vztah *Osoba - Zamestnani* a *Zamestnavatel - Zamestnani* je vždy *one to many*.
Podobným způsobem je také řešena potřeba verzování obrázků. Zde je „propojovací“ tabulkou Osoba_Obrazky sloužící pro propojení objektu *Osoba* a *Obrazky*.

5.2.3 Multimediální data

Obrázky se ukládají do tabulky *Obrazky*. V této tabulce jsou uloženy pouze datové typy image (resp. blob) označené *id*. Připojení obrázků k objektu Osoba řeší tabulka *Osoba_Obrazky* (obr. 5.3.).

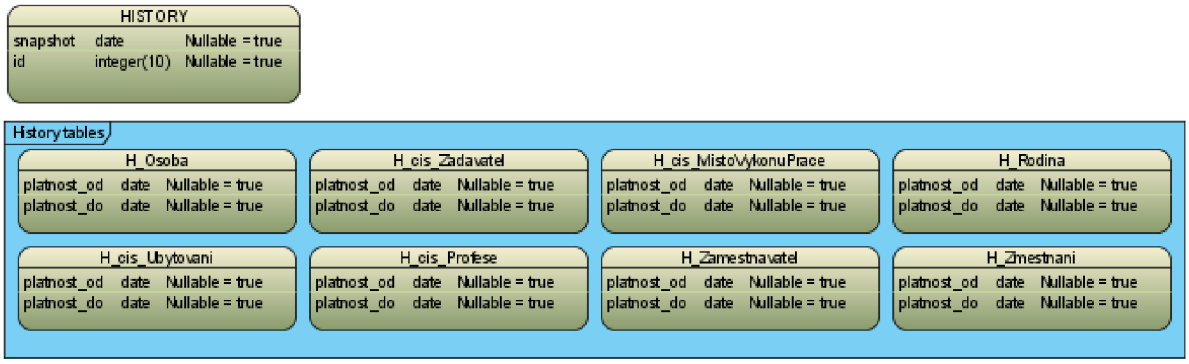


Obr. 5.3.: Výřez z ER - diagramu: Multimediální data

Propojení obrázku přes tabulku *Osoba_Obrazky* s Osobou umožňuje požadovanou funkcionalitu verzování obrázku. V tabulce *Osoba_Obrazky* jsou uloženy *metadata* k uloženému obrázku - to znamená o jaký obrázek se jedná (dle hodnoty *obrazek_typ* se určí, zdali jde o fotografii, vízum, anketu, pas, dokument pro ÚP), o jakou verzi (o kolikátý obrázek v pořadí se jedná, pro danou osobu) a datum, kdy k uložení obrázku došlo. *Osoba_Obrazky* a *Obrazky* jsou jediné dvě tabulky, pro které se nevedou historické záznamy. Vyplývá to z verzování, kdy se historie již uchovává.
Pro vložení nového obrázku je vytvořena *uložená procedura* v databázi provádějící automatické verzování - *Insert_Obrazek*. Ta má parametry: image *obrazek*, int *id_osoba* a int *obrazek_typ*. Vrací *id* přidělené novému obrázku. Procedura nejprve spočítá pro daný typ obrázku a danou osobu pořadí - verzi. Pak následuje vložení samotného obrázku do tabulky *Obrazky*, která přiřadí *id* nově vloženému záznamu obrázku. Tím jsou k dispozici všechny atributy pro vložení do tabulky *Osoba_Obrazky*, které se také provede.

5.2.4 Temporální data

Práce s temporálními daty vychází z rozboru provedeného v analýze v podkapitole 4.3. Každý objekt je popsán hlavní tabulkou, jak je uvedeno výše. Nad touto hlavní tabulkou jsou přidány 3 databázové *triggery*, které se spustí vždy po provedení instrukcí INSERT, UPDATE, DELETE. Tyto *triggery* provádějí mírně modifikované významy instrukcí nad tabulkami historickými. Jejich seznam je vidět na obrázku (obr. 5.4.).



Obr. 5.4.: Výřez z ER diagramu - Tabulky použité pro ukládání temporálních dat

Tabulky v poli *History tables* (na obrázku v modrém poli) jsou identické s jejich hlavními tabulkami (atributy zde tedy neuvádím) a jsou k nim navíc připojeny dva atributy platnost_od a platnost_do.
Pro účely jednotného přístupu do databáze z klientské aplikace je dále zavedena tabulka HISTORY. Ta obsahuje pouze jeden záznam s *id* = 1. V atributu *snapshot* je uložena hodnota *NULL* při standardním módu aplikace, a hodnota data ke kterému se provádí prohlížení historie při historickém módu aplikace. Změnu této hodnoty provádí klient, vždy při přepnutí do patřičného stavu (resp. vybrání patřičného data v historickém módu). Tento návrh počítá pouze s jednou klientskou stanicí (temporální rozšíření se nebude dodávat Zadavateli). Při práci více stanic by docházelo k přepínání módu mezi stanicemi. Tento problém by se odstranil více záznamy v tabulce HISTORY a přiřazením id záznamu každé stanici klienta.

5.3 Návrh komunikace

Důležitým prvkem v architektuře klient - server je forma komunikace mezi těmito aplikacemi.
V tomto odstavci bych rád navrhl řešení problematiky, jak celá komunikace bude implementována a s pomocí jakých komponent. Často se jedná o problematickou část vývoje informačního systému, kde může docházet ke zpožděním.
V Delphi existuje několik komponent umožňujících propojení s databází. Pro projekt jsem zvolil připojení pomocí ADO, vyvinuté společností Microsoft. Jedná se o sadu komponent pro práci s datovými zdroji.
Pro projekt se využije následujících komponent ADO:

ADOConnection - pro navázání spojení s databází
ADOQuery - umožňuje zasílání a provádění SQL dotazů nad připojenou databází
ADOStoredProc - volání uložených procedur na připojené databázi

Všechny tyto komponenty budou sdílené v hlavním formuláři aplikace (reprezentující okno Přehled osob). Při spuštění tohoto okna se provede nejprve pokus o navázání spojení s databází. Konfigurace připojení je uložena v *ini* souboru umístěném v adresáři programu, kde je možné ji editovat. S databází se komunikuje pomocí komponenty *ADOConnection*. První krokem je tedy nastavení této komponenty. Konfigurace nastavení je spouštěna z atributu *ConnectionString*, ten se skládá z hodnot atributů *Provider, Integrated Security, Persist Security Info, Initial Catalog, Data Source*.
Druhým krokem je zkouška připojení k databázi. To se provádí pomocí procedury *ADOConnection.Open*. Pokud jsme nastavili konfiguraci správně, měla by být následně komunikace

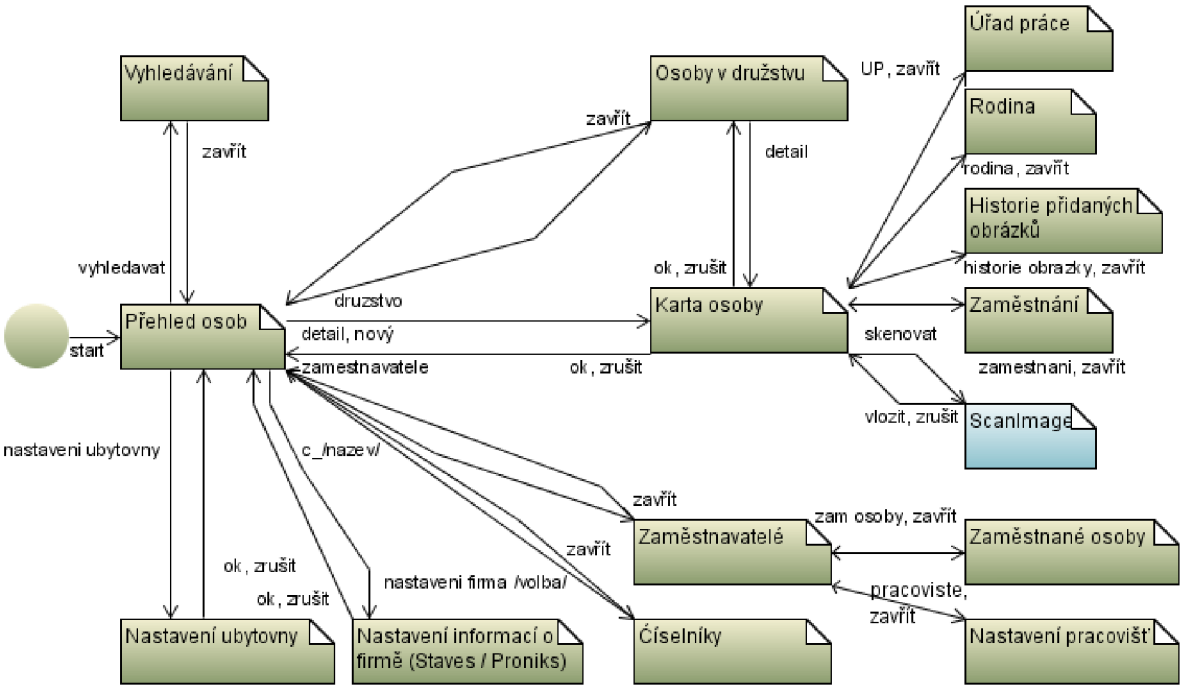
možná. Pokud došlo k chybě, musí dojít k re-konfiguraci nastavení připojení. Platnost připojení je možné kdykoliv ověřit pomocí atributu *ADOConnection.Connected*.
V informačním systému přistupujeme k databázi přímo pomocí dotazů SQL. Ty jsou zasílány databázi pomocí komponenty *ADOQuery*. *ADOQuery* obsahuje atribut *SQL.Text* - do kterého je možné nahrát řetězec požadovaného příkazu. Tento příkaz je zaslán databázi při volání procedury *ADOQuery.ExecSQL*. V případě provádění transakce nad daty v databázi je možné využít testování úspěšného projití transakce - to zajišťuje opět komponenta *ADOConnection*. Atribut InTransaction určuje, zda databáze aktuálně provádí transakci, která není ukončená. Pokud ano, můžeme zavolat proceduru *CommitTrans* pro její potvrzení. Pokud procedura vyvolá výjimku, je vhodné zavolat *RollbackTrans* pro vrácení změn (a následně vypsat chybové hlášení). Tento postup zajistí integritu dat i v případě, že dojde k chybě při vkládání dat do historických tabulek.
Volání instrukcí SELECT se provádí pomocí instrukce *ADOQuery.Open*, po vložení instrukce do *SQL.Text*. V *RecordCount* je uložen počet vrácených záznamů. K nim je možné přistupovat například pomocí *FieldByName(string name),* s parametrem jméno požadovaného sloupce.
Pro spouštění uložených procedur je určena komponenta *ADOStoredProc*. Atributy komponenty *ProcedureName, Parameters.ParamByName(name)*.Value, Parameters.ParamByName(name).Assign (pro obrázek) se nastaví jméno volané procedury a její parametry. Procedura se následně spustí procedurou *ExecProc*.
Tímto jsem vyčerpal metody připojení a komunikace s databází, které se v projektu budou nacházet. Z mého hlediska se jedná o jednoduchý a spolehlivý způsob práce s databází, který je pro požadavky projektu dostačující.

5.4 Ukládání nastavení

Data jsou uložená z velké části v databázi, některá data jsou však uložena v konfiguračních souborech, které se ukládají do složky s aplikací. Do konfiguračních souborů se ukládají údaje důležité pro konektivitu s databází – jméno serveru, databáze, a také data spojená s tiskem formuláře Žádost cizince o pracovní povolení, Vyjádření družstva. Je to z toho důvodu, že databáze neřeší tento typ dat a musela by se pro tato data zřizovat další tabulka o jednom řádku. Je zřejmé, že může docházet k různým verzím tohoto nastavení, vždy jedna verze pro jeden počítač Zadavatele. Ten však počítá s přístupem k serveru pouze z jednoho PC a není to pro něj problém. V případě, že bude třeba používat informační systém na více stanicích, bude muset dojít k distribuci souboru s nastavením.

5.5 Návrh návaznosti obrazovek

Studie proveditelnosti zohledňovala logiku návaznosti obrazovek. Zadavatel měl jasnou představu, jak by tato část aplikace měla vypadat, a protože jsem v jeho požadavcích neobjevil možné nekonzistence, držel jsem se přesně jeho požadavků. V rámci návrhu jsem zpracoval pomocný diagram návazností obrazovek, který specifikuje přechody mezi obrazovkami (obr. 5.5.).



Obr. 5.5.: Návaznost obrazovek s přechody

Diagram ukazuje všechny obrazovky a přechody mezi nimi. Popis vazby definuje, jaké volání spustí dané okno. Okna se mohou ukončovat dvěma možnostmi. První je jednoduché zrušení - okno se pouze zavře. Druhá možnost je jedna z variant *ok*, zrušit. Při variantě *ok* se nejprve uloží provedené změny, varianta zrušit zavírá okno bez uložení změn.
Hlavní obrazovka, která se spustí při startu informačního systému, bude **Přehled osob** a bude obsahovat klasické hlavní menu, seznam evidovaných osob a tlačítka pro další navigaci. Okno Přehled osob je hlavním oknem aplikace a běh informačního systému po jeho ukončení končí. Okna spustitelná z hlavního menu jsou:

Nastavení ubytovny - okno pro správu nastavení vztažených k ubytovně.
Nastavení informací o firmě - okno pro správu nastavení vztažených k firmě Zadavatele (uživatel má možnost zvolit možnost PRONIKS/STAVES - dvě přednastavené sady hodnot).
Zaměstnavatelé - okno se seznamem zaměstnavatelů, umožňující jejich kompletní správu. Z tohoto okna je možné spouštět navázaná okna:

Zaměstnané osoby - zobrazuje přehled osob, pracujících pro vybraného zaměstnavatele v okně Zaměstnavatelé.

Nastavení pracovišť - umožňuje správu pracovišť pro vybraného zaměstnavatele.

Číselníky - správa hodnot jednoho ze tří číselníků (dle vybraného).
Osoby v družstvu - zobrazuje seznam osob v družstvu a umožňuje tisk vybraných formulářů. Stejně jako z hlavního okna je možné poklepáním na osobu otevřít Kartu osoby (viz. dále).

Z hlavního okna je možné několika způsoby otevřít Kartu osoby. Tlačítkem Nová osoba se otevře nové prázdné okno **Karta osoby**. Při dvojkliku na osobu v seznamu osob, případně tlačítkem upravit osobu se otevře Karta osoby s vyplněným detailem označené osoby. Karta osoby také umožňuje otevření dalších oken:

Úřad práce - správa detailních informací o záznamech pro úřad práce vztažených k vybrané osobě.
Rodina - správa záznamů o rodině vztažených k vybrané osobě.
Zaměstnání - správa zaměstnání pro vybranou osobu. Umožňuje připojit zaměstnavatele k dané osobě.
Historie přidaných obrázků - u každé osoby je přidáno pět typů obrázků. Pro každý z těchto typů obrázků se provádí verzování. Starší verze je možné procházet kliknutím na tlačítko Historie umístěné pod příslušným obrázkem. Z tohoto okna, kde je seznam starších verzí, je možné obrázky otevírat.
ScanImage - otevře speciální software pro skenování. Po naskenování obrázku je možné tento obrázek přímo poslat do systému a vytvořit tak nový obrázek pro typ obrázku, od kterého se ScanImage spustil.

V hlavním okně aplikace se také zobrazuje archiv zneplatněných osob. Při přepnutí do archivu se znovu načte seznam osob, tentokrát osob umístěných v archivu.
Pro pokročilé vyhledávání se vybere tlačítko na hlavním okně - **Vyhledávání**. To otevře nové okno, ve kterém je naimplementována logika pokročilého vyhledávání.
Aplikace se ukončuje pomocí tlačítka Zavřít v hlavním okně, v menu stisknutím tlačítka Konec, případně zavřením okna Přehled osob.

5.6 Návrh uživatelského prostředí

Uživatelské prostředí vychází z požadované návaznosti jednotlivých obrazovek. Zaměřím se zde na dvě hlavní okna aplikace - Přehled osob a Karta osoby a zmíním také návrh jednoduchého okna pro správu číselníku. Zaměřím se také na temporální rozšíření - co bude nutné pro jeho běh na straně uživatelského prostředí.

5.6.1 Okno Přehled osob

Hlavní okno informačního systému obsahuje seznam osob, hlavní menu a sadu ovládacích tlačítek. Pro seznam osob bude vhodné vybrat komponentu, jež zajistí snadnou práci s vyhledanými záznamy a bude v ní možné třídit záznamy dle různých sloupců a barevně vyznačovat záznamy, jak se požaduje ve specifikaci. Pro tyto účely se využívá komponenta *LMDListBox*. Pro tuto komponentu musí být v okně tolik místa, aby se do ní vešly všechny potřebné zobrazované informace. Zobrazované atributy budou Příjmení a Jméno (řetězec vzniklý spojením atributů příjmení a jméno objektu Osoba), Datum narození, Stát, Číslo pasu a Od Koho. Tyto slouží pro rychlou identifikaci osoby bez nutnosti otevírat její detail. V seznamu se pro každý záznam dále zobrazují tři sloupce platností (platnost víza, platnost pojištění a platnost ÚP), tyto je nutné pro každý záznam označit v případě, že k danému dni zbývá méně než tři měsíce.
V okně Přehled osoby bude hlavní menu aplikace. Menu bude rozděleno dle zvyklostí operačního systému Windows XP na tyto *submenu* - Soubor, Zobrazit, Nastavení a Nápověda. Obsah těchto menu bude tento:

Soubor: Nový - otevře okno Karta Osoby
Tisk - pro vybranou osobu v seznamu osob vytiskne vybraný formulář
Konec - ukončí aplikaci

Zobrazit: Prohlížení historie - přepínání mezi módy informačního systému
Družstvo - otevře okno se seznamem členů družstva Osoby v družstvu
Zaměstnavatelé - otevře okno Zaměstnavatelé

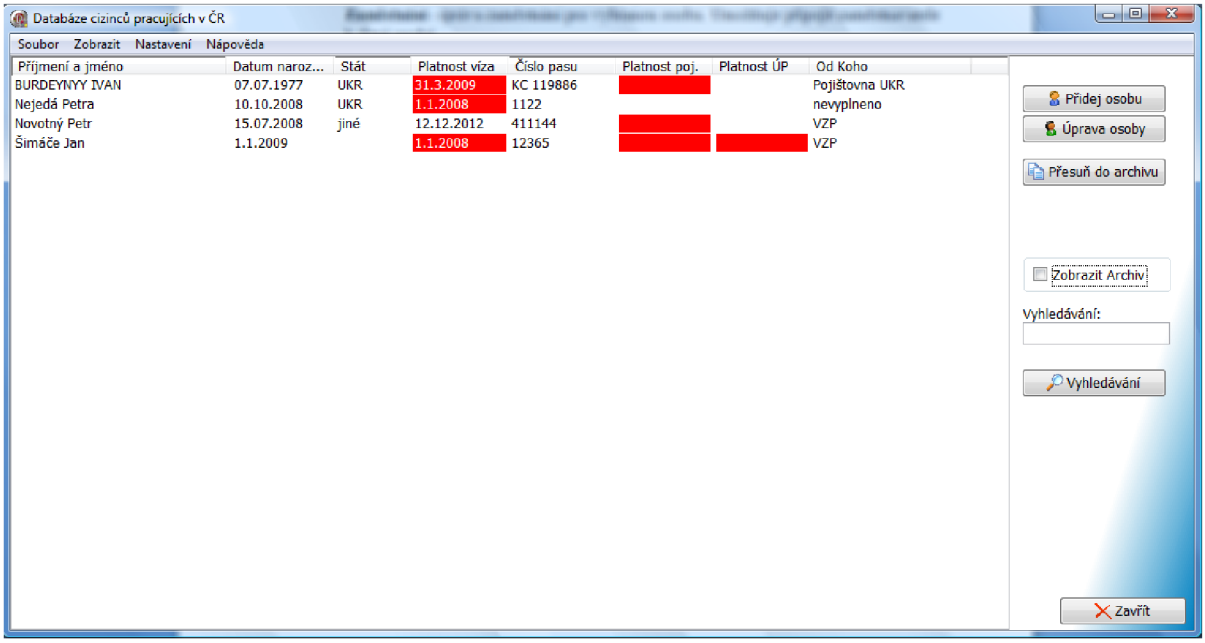
Nastavení: Číselníky - výběrem v tomto menu se zvolí číselník pro správu. Číselníky pro výběr: Ubytování, Profese a Zadavatelé.
Nastavení ÚP - otevírá okna pro nastavení tisku dokumentů dodávaných na úřad práce s údaji vztažených k firmě. Možnosti - Staves, Proniks (pozn. obě firmy Zadavatele), adresa ubytovny.

Nápověda: O programu - informační okno o programu - název, verze, výrobce.

Položky menu jsou reprezentovány komponentou *TMainMenuItem*, celé menu je komponenta *TMainMenu*.
Ovládací tlačítka bude vhodné umístit do jednoho panelu, který bude možné graficky upravovat. Vybral jsem komponentu *TLMDPanelFill*. V této komponentě tedy umístíme tlačítka (komponenta *TLMDButton*) pro správu:

Přidej osobu: Otevře nové okno Detail osoby.
Úprava osoby: Otevře okno Detail osoby pro vybranou osobu v seznamu osob.
Přesuň do archivu: Nastaví atribut Archiv u vybrané osoby na 1 a načte znovu seznam osob.
Obnov z archivu: Nastaví atribut Archiv u vybrané osoby na 0 a načte znovu seznam osob.
Odstraň osobu: Smaže vybranou osobu a načte znovu seznam osob.
Vyhledávání: Otevře okno pro vyhledávání.
Zavřít: Ukončí aplikaci.

V ovládacím panelu bude ještě přepínání mezi aktivními osobami a archivem (komponenta *TLMDCheckBox*) a aktuální vyhledávaný řetězec (komponenta *TEditBox*). Přepínač archivu bude ovlivňovat vlastnost *visible* tlačítek Přesuň do archivu, Obnov z archivu a Odstraň osobu.
Konečná podoba okna je vidět na obrázku níže (obr. 5.6.).



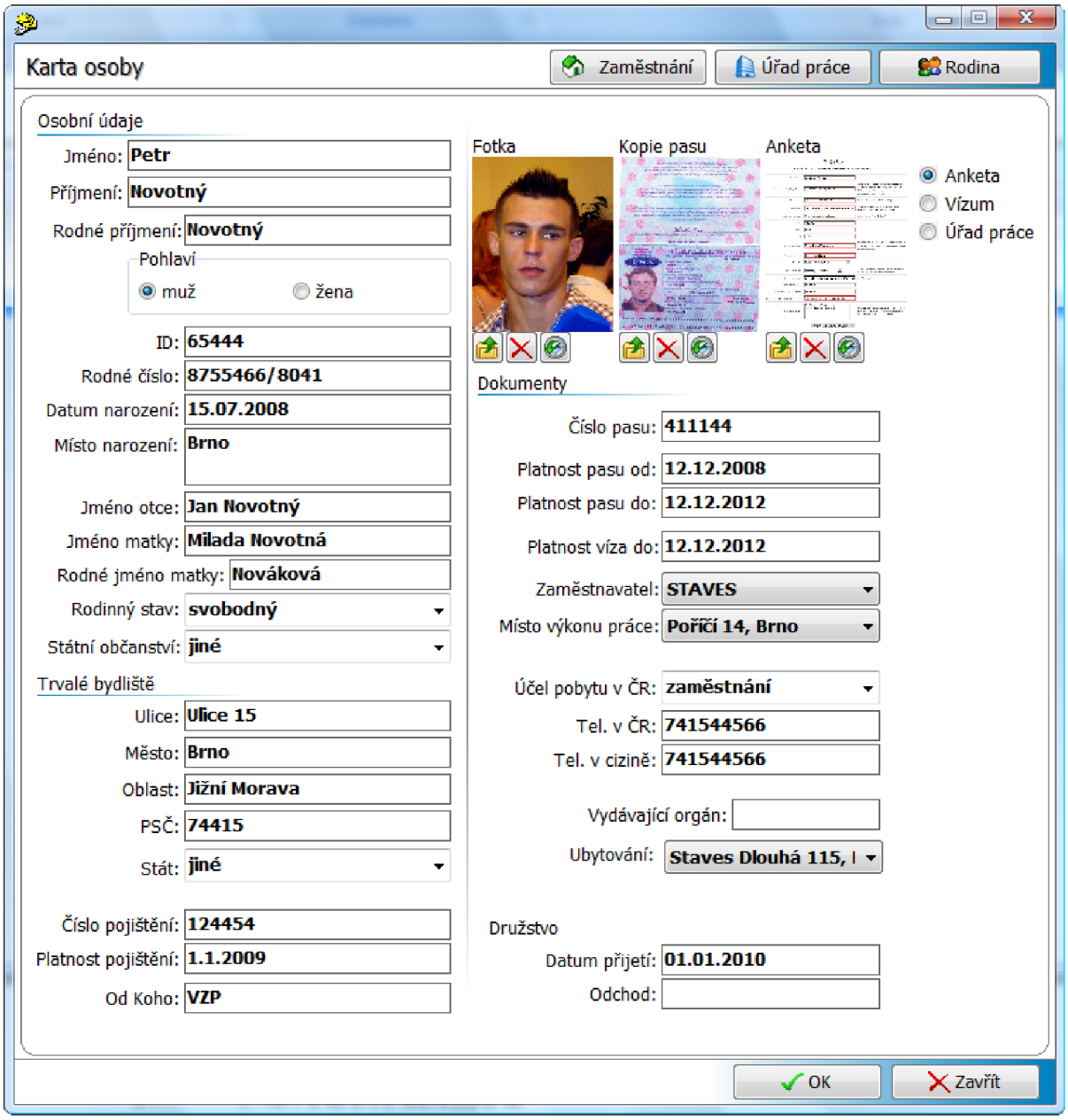
Obr. 5.6.: Náhled okna Přehled osob

5.6.2 Okno Karta osoby

V tomto okně se zobrazuje detail osoby. Protože objekt Osob obsahuje více jak třicet atributů, které je nutné v tomto okně spravovat, bude vhodné toto množství rozdělit do podskupin. Podskupiny pro atributy jsou Osobní údaje, Trvalé bydliště, Pojištění (přidané později), Dokumenty a Družstvo. Ovládací prvky v každé této podskupině budou umístěny v komponentě *TLMDSimplePanel*, která zajistí vizuální oddělení.
Pro vstup (resp. zobrazení) atributů v okně využijeme následující komponenty rozdělené podle typu daného atributu:

Řetězec, číslo: *TLabeledEdit*
Volba z omezeného množství možností: *TRadioBox*
Volba z číselníku: *TComboBox*

Na Kartě osoby bude nutné zobrazení tlačítek (komponenta *TLMDButton*), odkazujících na navázané objekty Osoby. Tato tlačítka budou otevírat okno pro správu daného objektu a po skončení vrátí aktivitu oknu Karta osoby.
Poslední podskupinou atributů osoby budou navázané obrázky. Obrázků je pro osobu pět typů, a protože každý obrázek bude obsahovat náhled (komponenta *TImage*), je nutné jejich počet na kartě mírně „omezit“. To provedeme zavedením přepínače (komponenta *TRadioBox*) mezi méně důležitými obrázky. Na kartě bude vždy při otevření vidět obrázek typu fotografie, číslo pasu a anketa, při potřebě zobrazovat obrázky úřad práce a vízum se po přepnutí náhled vymění s náhledem obrázku anketa. Přepínač tak bude ovlivňovat vlastnost *visible* komponent posledních tří obrázků. Tyto komponenty budou kromě náhledu obsahovat také tlačítka pro načítání obrázku (komponenta *TOpenPictureDialog*), mazání a zobrazení verzí (otevření okna - seznam verzí). Pro otevření plné velikosti obrázku se klikne na náhled obrázku a tím se spustí okno pro prohlížení obrázku. Náhled již funkčního okna Karta Osoby je na obrázku níže (obr. 5.7.).



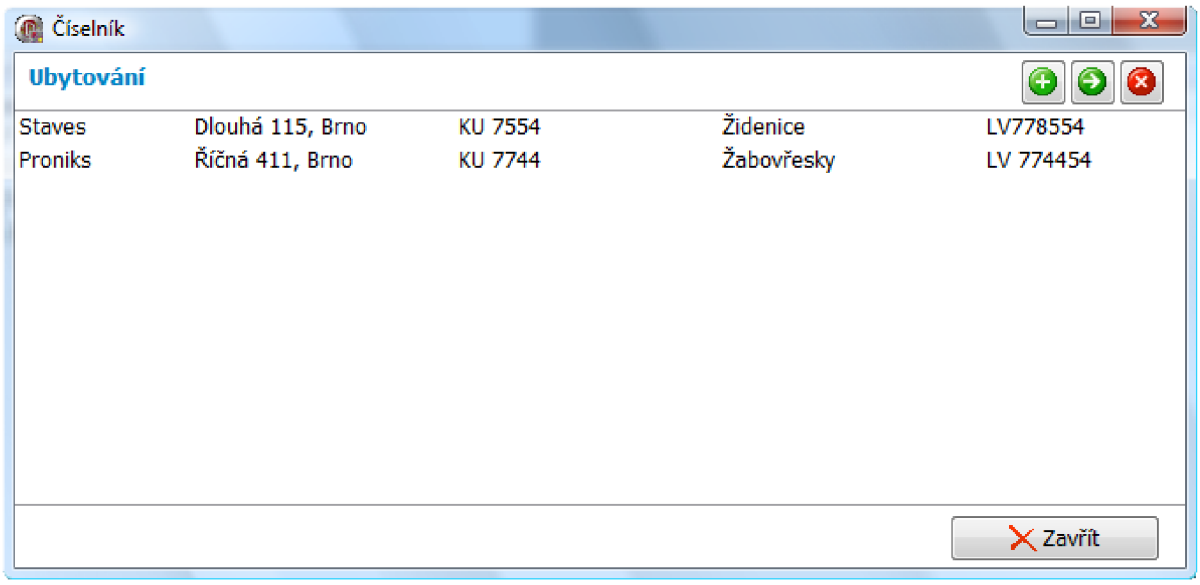
Obr. 5.7.: Náhled okna Karta osoby

5.6.3 Okno pro správu jednoduchého objektu

V systému bude potřeba několik oken pro správu jednoduchých objektů. Těchto oken bude v aplikaci několik - objekty číselníků a objekt Rodina. Okno musí obstarávat všechny povolené operace nad těmito objekty (vytváření, úprava, mazání, prohlížení). Pro každou operaci tak bude zapotřebí tlačítka (komponenta *TLMDButton*), ta by měla být umístěna na jednotném panelu (komponenta *TLMDPanelFill*). Pro výpis objektů z databáze je vhodná komponenta *TLMDListBox*, ve které se popíší do názvu sloupců názvy atributů objektu.

S tímto oknem musí být svázané ještě jedno pomocné okno. Toto okno se bude spouštět při stisku tlačítka pro vytváření a pro editaci a bude obsahovat takové komponenty, do kterých bude možné zapsat hodnoty atributů objektu (nejčastěji komponenty *TLabeledEdit* - text a *TComboBox* - výběr).

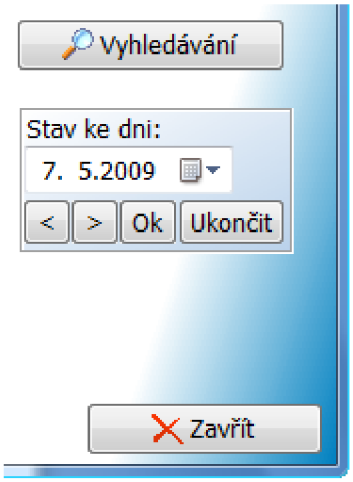
Toto okno musí obsahovat tlačítka ok a cancel (komponenta *TLMDButton*) pro potvrzení nebo zrušení provedených změn.
Okno pro správu obsahuje ještě tlačítko zavřít (komponenta *TLMDButton*) pro ukončení práce.
Náhled jednoho z použití okna pro správu objektů je vidět níže (obr. 5.8.).



Obr. 5.8.: Náhled okna Číselník Ubytování

5.6.4 Rozšíření pro prohlížení historie

Při rozšiřování systému o funkcionalitu prohlížení historie bude třeba vhodně upravit uživatelské prostředí. V informačním systému budou dva módy - standardní a historický. Pro přepínání mezi módy bude pro výběr využito přepínače v hlavním menu (komponenta *TMainMenuItem* s vlastností *RadioItem* nastaveno na hodnotu *true*).
V módu historie se zobrazí v dolním pravém rohu okna Přehled osob panel (komponenta *TLMDPanelFill*) pro prohlížení historie (obr. 5.9.).



Obr. 5.9: Výřez ovládacího panelu okna Přehled osob v módu historie

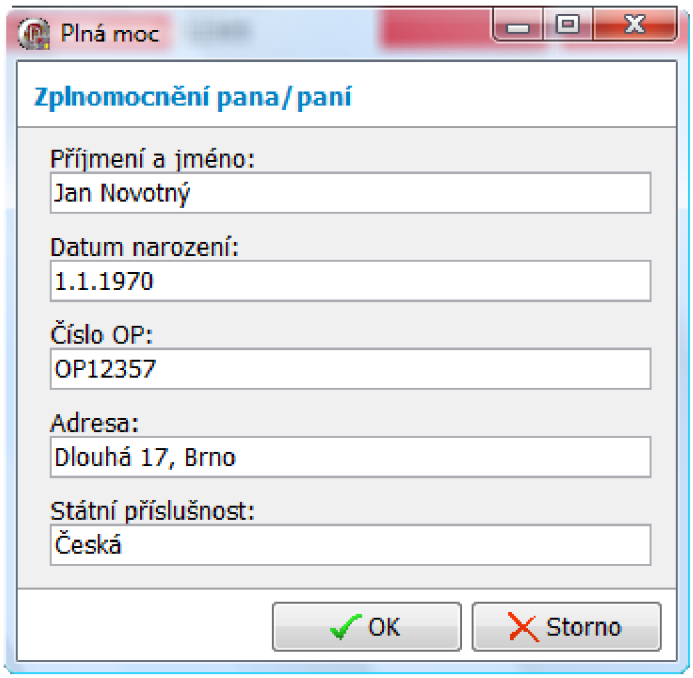
V ovládacím panelu bude editovací položka pro výběr z kalendáře (komponenta *TDataTimePicker*) zobrazovat k jakému dni se stav informačního systému vztahuje.
V ovládacím panelu budou dále tato tlačítka (komponenta *TLMDButton*):

„šipka vlevo“: posune den o jeden do minulosti od vybraného dne a aktualizuje stav
„šipka vpravo“: posune den o jeden do budoucnosti od vybraného dne a aktualizuje stav
Ok: aktualizuje stav systému k danému dni
Ukončit: přepne do standardního módu

V módu historie není možné vkládat a editovat údaje. Všechna okna v systému mají při módu historie znemožněno vkládání a potvrzování údajů (vlastnost *disabled* tlačítek pro vkládání /Nový …/ a potvrzování změn /Ok/ je nastavena na *true*).

5.7 Tisk formulářů

Tiskem formulářů popisuji celou posloupnost operací. Ta spočívá v označení osoby (resp. osob) pro kterou se tisk provede, označení příslušného formuláře a pokud je to nutné vyplnění dodatečného dialogového okna pro získání dalších údajů. Pak zbývá vložení všech dat do vytvořeného formuláře a tisk.
Označování osob se bude provádět dle příslušného indexu v seznamu osob v okně Přehled osob a v okně Družstvo.
Seznam formulářů pro tisk bude uveden pro Přehled osob v hlavním menu a pro formuláře v okně Družstva pod seznamem osob.
Pro každý formulář, kde je nutné vypisovat dodatečné informace, je nutné vytvořit speciální okno s příslušnými komponentami. Do těchto komponent se budou zapisovat další data pro tisk. Náhled okna otevíraného před tiskem je níže (obr. 5.10.):



Obr. 5.10.: Náhled dialogu před tiskem, pro vložení dodatečných dat - formulář Plná moc

Formuláře se budou vytvářet pomocí vývojového nástroje Rave Reports[1]. Zde se vytvoří jedna sada tiskových sestav dle tisknutých výstupů (označováno jako *Reports*). Propojení mezi vytvořenými tiskovými sestavami a projektem zaručují komponenty *TRvSystem* a *TRvProject*. *TRvSystem* je použit pro vytvoření spojení (tzv. engine) a *TRvProject* je vytvořen pro každý jeden tisknutý formulář. Pomocí instance komponenty *TRvProject* se předávají hodnoty parametrů pro tisk (procedura *SetParam*). Pole pro převzetí dat jsou v tiskové sestavě označeny jako *DataField* a nesou označení, pomocí kterého se do nich nahrávají data. Zahájení tisku se provádí pomocí procedury *Execute*. Požadovaná tiskárna pro výstup se vybere v komponentě *TPrinterSetupDialog*.

[1] více na http://www.nevrona.com/

6 Implementace

Fáze implementace je stěžejní fází projektové fáze realizace. V této fázi dochází k samotné výrobě produktu. Fáze implementace nemá žádné předepsané vytvářené projektové dokumenty, ty se vytváří ve fázi testování (např. protokoly o testování, uživatelská příručka). Výstupem fáze implementace je samotný kód produktu (může být doplněn o technickou zprávu).
V této kapitole se budu zabývat vývojovými nástroji, které jsem pro projekt použil a implementací klienta se zaměřením na často používané komponenty. Během implementace jsem čerpal z literatury *[1] Deplhi: Hotová řešení* a *[2] Komponenty v Delphi*. Tyto publikace se zaměřují na správné psaní kódu v použitém vývojovém prostředí a byly mi během implementace užitečným zdrojem informací a vhodných postupů.
Implementace probíhala ve čtyřech inkrementacích. Popíšu zde, jaké prvky se implementovaly pro jakou iteraci vývoje, a jak se případně v dalších iteracích rozšiřovaly.

6.1 Vývojové nástroje

Pro vývoj projektu jsem zvolil vývojové prostředí Borland Delphi 2005[1] jazyka Delphi. Tento jsem vybral především z důvodů standartu vývoje aplikací ve firmě.

Delphi je integrované grafické vývojové prostředí firmy Borland určené pro tvorbu aplikací na platformě MS Windows v jazyce Object Pascal (objektové nástavbě Pascal). Obsahuje systém RAD (Rapid Application Development), který umožňuje vizuální návrh grafického uživatelského rozhraní, na jehož základě je automaticky vytvářena kostra zdrojového kódu, což výrazně urychluje vývojový cyklus.
Programování v Delphi je z velké části založeno na použití komponent. Komponenta je malý program (balíček funkcí), který vykonává určitou činnost (například zobrazuje text nebo obrázky, přehrává multimédia, komunikuje s databází, zprostředkovává FTP přenos, atd…).
Velkou předností Delphi proti některým konkurenčním produktům jsou knihovny komponent, které jsou jejich součástí (např. VCL, CLX, Indy). Dodávané komponenty významně usnadňují tvorbu aplikací. Další komponenty lze stáhnout z internetu. V Delphi lze vytvářet vlastní komponenty.
[10] www.borland.cz

V Delphi je použita knihovna rozšiřujících komponent LMD-Tools Special Edition 7.0[2]. Knihovna LMD Tools obsahuje více jak 200 VCL komponent, které je možné zdarma použít. Pokrývají všechny oblasti - tlačítka, texty, dialogy, multimédia, kontejnery, rozšíření stávajících komponent (včetně odstranění chyb VCL). Tuto sadu komponent využívám především kvůli komponentě *TLMDListBox*, jež umožňuje snadné řazení zobrazených prvků v seznamu.
Pro vytváření tiskových sestav formulářů je použitý software Rave Reports[3]. Ten je nástupcem produktu *ReportPrinter*. Rave je sada komponent a nástrojů pro vytváření tiskových sestav a jejich správu a distribuci.
Vývojové prostředí databáze je SQL Server Managment Studio[1] pro MS SQL 2008[2]. Databáze byla vyvíjena pomocí integrovaného grafického prostředí a v databázovém nástroji *Query* použitím standardních příkazů SQL.

[1] více na http://www.codegear.com/
[2] více na http://www.lmdinnovative.com/
[3] více na http://www.nevrona.com/

6.2 Implementace databáze

Databáze byla vyvíjena dle detailního rozboru uvedeného v kapitole 5. Tabulky databáze odpovídají objektům uvedeným v ER-diagramu (viz. příloha B). Zde byl až na změnu názvosloví u tabulky Zadavatel proveden přepis 1:1.
Pro každou tabulku byly implementovány *triggery*. Podrobný popis jsem uvedl v analýze v podkapitole 4.3..
Pro podporu multimediálních dat jsem vytvořil popsanou uloženou proceduru uvedenou v návrhu odstavec 5.2.3. Její podoba v SQL vypadá takto:

```
CREATE PROCEDURE [dbo].[Insert_Obrazek]
        @Obrazek image, @Obrazek_Typ int, @ID_Osoba int,
        @ID_Obrazek int output
AS
BEGIN
        DECLARE @Verze int;
        SET @Verze=(Select ISNULL(MAX(Verze)+1,0) From Osoba_Obrazky WHERE
        ID_Osoba=@ID_Osoba AND Obrazek_Typ=@Obrazek_Typ);
        INSERT INTO Obrazky (Obrazek) VALUES (@Obrazek);
        SELECT @ID_Obrazek = @@IDENTITY;
        INSERT INTO Osoba_Obrazky (ID_Obrazek, ID_Osoba, Obrazek_Typ, Verze) VALUES
        (@ID_Obrazek, @ID_Osoba, @Obrazek_Typ, @Verze);
END
```

SQL skript pro vytvoření databáze a vytvořenou databázi naplněnou testovacími daty je možné nalézt na přiloženém CD (viz. příloha E).

6.3 Implementace klienta

Implementace uživatelského prostředí probíhá s použitím vhodných komponent definovaných v DELPHI. Uvedu zde popis nejpoužívanějších komponent, jejich atributy a využívané události, které jsou nad komponentami používané v systému.

6.3.1 Rozdělení kódu

Při vývoji byl kód rozdělen do souborů dle okna aplikace, ke kterému se váží. Tyto soubory se nazývají *Forms* a po vytvoření obsahují platnou komponentu *TForm* popisující dané okno. Nad touto třídou je možné zpracovávat řadu událostí - *OnCreate* (při vytvoření), *OnDestroy* (při zániku), *OnActivate* (při získání „fokusu“ na okno) atd. Použití těchto událostí popíši u formulářů, jež je využívají. Projekt se tak skládá z množiny formulářů, kde je definováno, který formulář se spustí při startu programu.
Pro potřeby projektu bylo vytvořeno celkem 29 formulářů.
Formuláře použité v aplikaci se dělí do dvou skupin. První skupina významnějších oken (Přehled osob, Karta osoby, přehledy objektů, dialogová okna před tisky, apod.) se vytváří při startu, při

[1] více na http://www.microsoft.com/products/
[2] více na http://www.microsoft.com/products/

požadavku na jejich otevření se pouze otevírají a případně plní daty. Druhá skupina jsou méně významné formuláře (správa číselníků, okna nastavení, apod.), ty se vytváří vždy až po jejich zavolání a po jejich ukončení uživatelem se jejich instance uvolní.

6.3.2 Komunikace s databází

První implementační práce na straně klienta byla implementace jádra - neboli komunikace se serverem. Metodika a použité komponenty jsou popsány v návrhu (viz. odstavec 5.3.). Kód komunikace s databází je implementován ve formuláři *FCizinci*. Při aktivaci tohoto okna (událost *OnCreate*) se testuje připojení k databázi. Pokud připojení neexistuje, pokusí se klient připojit. Nastavení připojení se načítají ze souboru *ini*. Pokud se připojení nezdaří, aplikace se ukončí.

6.3.3 Přehled osob

Je hlavním oknem aplikace (formulář *FCizinci*) a je zde implementována logika připojení k databázi, čerpání dat z databáze pro tisky formulářů, přepínání mezi módy aplikace a kromě toho samozřejmě kód pro komponenty umístěné na hlavním okně. V první implementaci se vytvořil základ tohoto okna a obsahoval zobrazování seznamu cizinců a propojení s oknem Karta osoby. Hlavní okno se měnilo s každou iterací a do formuláře přibýval další kód.

seznam osob - seznam osob je tabulka s nejdůležitějšími informacemi o osobách. Při aktivaci okna Přehled osob (událost *OnActivation* formuláře) se provede načtení dat z databáze. Pro potřeby seznamu osob je vytvořena třída *TCizinecMini*. Tato třída obsahuje pouze data pro zobrazení v seznamu osob - jméno, příjmení, stát, datum narození, platnost víza, pojištění, úřadu práce a číslo pasu. Navíc je ještě přidáno id osoby pro její identifikaci systémem.
Načtená data z databáze jsou nejprve uložena do pole instancí třídy *TCizinecMini* jménem *PoleCizincu* a odtud se kopírují do seznamu osob. Při potřebě označení dané osoby se dle indexu v seznamu osob (komponenta *TLMDListobox* vlastnost *ItemIndex*) najde příslušná osoba v poli *PoleCizincu*.

archiv - přepínání mezi aktivními a neplatnými osobami se provádí přepínačem archiv (komponenta *TLMDCheckBox*). Při jeho označení (událost *OnClick*) se kontroluje, zda je zakřížkován (vlastnost *Checked*). V závislosti na výsledku se vyprázdní seznam osob a znovu se naplní osobami, které jsou nebo nejsou v archivu (atribut objektu osoba *Archiv*).

hlavní menu - obsahuje seznam oken a funkcí, které je možné v aplikaci spouštět a provádět. Dle typu okna se provedou patřičné kroky pro jeho vytvoření (resp. otevření). Okna jsou otevírána pomocí funkce *ShowModal*, která při otevření neumožňuje návrat k oknu, které tuto funkci volalo, až do ukončení otevřeného okna.

6.3.4 Karta osoby

Implementace okna pro Kartu osoby (formulář *FClovek*) se prováděla od první iterace (práce s atributy Osoby). V druhé iteraci přibyla funkcionalita otevírání obrázků, provázání s číselníky a správa navázaných objektů. V této iteraci se také přidaly do objektu Osoba další atributy vztažené k pojištění - o ně tedy bylo nutné prvky v okně rozšířit. Ve čtvrté iteraci došlo ke změnám některých

prvků (např. byly přidány ikony), upravilo se uspořádání komponent v okně a upravilo se pořadí *TabOrder* pro skoky označení mezi komponentami.

načítání detailu osob - při aktivaci okna Karta osoby (událost *OnActivation* formuláře) se provede načtení z databáze detailů osoby. ID osoby, pro kterou se okno otevírá, se předává jako argument funkce *Otevri*. Všechny načtené údaje se nejprve ukládají do lokálních proměnných, odpovídajících uloženým atributům objektu Osoba, a až následně se zobrazují v uživatelském rozhraní.

ukládání osoby - po otevření okna Karta osoby se kontroluje, zda došlo ke změně některého z atributů (proměnná *zmeneno* typu *boolean*). Před uzavřením okna (případně před potvrzením změn) je uživatel vyzván, zda chce změny opravu uložit. Uložení probíhá zavoláním příslušné instrukce INSERT případně UPDATE.

správa obrázků - s obrázky se pracuje pomocí komponenty *TImage*. Jejich načtení je možné provést voláním procedury atributu této třídy *Picture* - *LoadFromFile* a *Assign*. *LoadFromFile* slouží pro načtení z externího souboru na disku a *Assign* převádí uložený BLOB z databáze na datový typ *TImage*.Picture.
Obrázky je možné pomocí atributu *Scratch* komponenty *TImage* upravit do podoby náhledu (zmenšit) na Kartě osoby.
Po dvojkliku (událost *OnDblClick* komponenty *TImage*) na náhled obrázku se otevře nové okno (např. formulář *FotkaForm*), které slouží pouze pro zobrazení obrázku v plné velikosti.
Při smazání obrázku je obrázek nahrazen prázdným obrázkem *blank.jpg* a původní obrázek je možný prohlížet pouze v přehledu verzí (formulář *TImageHistory*). V tomto okně se načítají všechny obrázky z databáze pro danou osobu. Jsou seřazeny dle hodnoty atributu verze.

6.3.5 Správa jednoduchých objektů

Jedná se o okno, které zajišťuje správu jednoduchých objektů. Těchto oken bylo vytvořeno několik, každé pro určitý objekt (např. formuláře *FCISProfese*, *FCISUbytovani*, *FCISZamestnavatel*, *FRodina*). Tato okna jsou volána procedurou *Otevri* z jiného formuláře. V této proceduře se nastaví počáteční stav okna a provede se načtení dat z databáze (v případě, že je objekt navázaný na jiný, je proceduře předáno *id* „volajícího“ objektu - to se použije při dotazu na databázi). Data jsou načtena do seznamu objektů (komponenta *TListBox*), kde jsou jejich atributy popsány v názvech sloupců. Při načítání se ukládá id záznamů do lokálního pole. Pro získání id vybraného záznamu pak stačí načíst příslušný prvek v poli. Jako index slouží vlastnost komponenty *TListBox* - *ItemIndex*.
Formuláře obsahují tlačítka pro vytváření, úpravu a mazání (komponenta *TLMDButton*). Tlačítka pro vytvoření a úpravy otevírají další formulář - jednoduché okno se seznamem atributů daného objektu (např. formulář *FCISProfeseItem*, *FRodinaItem* apod.). Po vyplnění (resp. změně) těchto atributů a potvrzením tlačítka OK se vrací „fokus“ zpět na původní okno, ve kterém proběhnou akce vložení (resp. update) nového prvku a znovunačtení seznamu objektů.

6.3.6 Tiskové sestavy

Implementace tiskových sestav se prováděla během třetí iterace. Po vytvořeních dodatečných oken (např. formulář *FDoplneniTisku*) a předávání potřebných parametrů z databáze jsem se zabýval nejvíce správným vzhledem formuláře. Formuláře musely být vytvořeny přesně dle fyzických vzorů a

jejich převod do virtuální podoby byl otázkou důkladných měření a posuvů jednotlivých prvků v tiskových sestavách.

6.3.7 Vyhledávání

Vyhledávání se vytvářelo ve čtvrté iteraci. V informačním systému je možné využívat dvě podoby vyhledávání. Jednoduchá metoda vyhledávání pracuje se seznamem osob v okně Přehled osob. Do ovládacího panelu na levé straně byla přidána komponenta *TLMDEdit* se jménem *EVyhledavani*. Kdykoliv je „fokus“ nad oknem Přehled osob, je možné psát na klávesnici znaky, ty se zobrazují v *EVyhledavani*. Se stiskem každé klávesy (událost *OnKeyUp*) se provede procedura *Hledej*, která označí první osobu v seznamu (nastaví vlastnost *Selected* na hodnotu *true* pro odpovídající *ItemIndex* komponenty *LMDListBox* sloužící jako seznam osob), jejíž příjmení začíná psaným řetězcem. Při stisku *Esc* se vyhledávaný řetězec smaže. Při smazání řetězce a při neúspěšném vyhledávání se označení osob ztratí (vlastnost *Selected* nastaví *false*).
Druhou formou je pokročilé vyhledávání. To je implementováno v okně Vyhledávání (formulář FSearch). Okno Vyhledávání se spustí tlačítkem z levého ovládacího panelu v okně Přehled osob.
Okno pro pokročilé vyhledávání nabízí vyhledávání dle různých atributů osob. Pro výběr daného atributu je v okně komponenta *TComboBox*, ve které je umístěn seznam těchto atributů. Vyhledávání je možné omezit nad určitou hodnotou ve vybraném atributu (např. rok), požadovaná hodnota se vybere v komponentě *TEdit* jménem *EText*, případně v komponentě *TComboBox* jménem *ERok* a *EMesic*. U těchto komponent se dynamicky mění hodnota atributu *visible*, dle vybraného atributu hledání tak, aby byla viditelná v jeden okamžik pouze jedna.
Samotné vyhledávání se pak provádí po stisku tlačítka *Hledej* instrukcí SELECT v databázi. Tato instrukce patřičně zohledňuje vybraný atribut a hodnotu, dle které se hledá (odpovídá části příkazu SELECT: … WHERE atribut = hodnota). Výsledek vyhledávání je převeden do seznamu - komponenta *TStringGrid*.
Z okna vyhledávání je možné otevřít okno pro specifikaci zobrazených atributů výsledné kolekce (formulář *FSeznamSloupcu*). Ten obsahuje komponentu *TCheckListBox* se seznamem názvů atributů objektu Osoba. Zaškrtnutí daného atributu přidá sloupec do komponenty *TStringGrid* v okně vyhledávání.

6.3.8 Optimalizace pro uživatelskou práci

Čtvrtá iterace se týkala také uživatelské optimalizace programu. Zadavatel měl mnoho požadavků na drobné úpravy (změna názvů atributů, změna zobrazení, jiné zobrazované sloupce v záznamech, apod.). Také se implementovala podpora pro snazší práci s aplikací pomocí klávesnice. To obnášelo přidávat reakce na události *OnKeyPress*, *OnKeyUp* pro komponenty reprezentující seznamy osob, tlačítka a samotné formuláře.

6.3.9 Rozšíření o temporální databázi

Implementace rozšíření probíhala separátně během čtvrté iterace. Do této iterace by byla i logicky začleněna. Implementace spočívala v přidání tabulek a *triggerů* do databáze. Na straně klienta byly začleněny uživatelské ovládací prvky do uživatelského rozhraní a byl vytvořen mód historie (při přepnutí do módu historie je v uživatelském rozhraní zakázána možnost přidávat a upravovat jakýkoliv objekt; v informačním systému je možné jen prohlížet a nastavovat datum, ke kterému se stav načítá).

Při vybrání daného data se toto datum zapíše do databáze do tabulky HISTORY. Přestože by bylo možné testovat vybraný mód pro instrukce SELECT na existenci tohoto data v tabulce HISTORY, kvůli potřebě jednoduchého odstranění rozšíření jsem zvolil cestu testování přes globální proměnnou v klientovi. Tato proměnná se jmenuje *Historie* a je typu *boolean*. Následně před každou operací SELECT testuje stav této proměnné. Pokud je nastavena na hodnotu *true*, zavolá se modifikovaný SELECT nad příslušnou tabulkou s prefixem H_ patřičně omezený hodnotou data v tabulce HISTORY (*platnost*_od < *datum AND platnost_do* > *datum*).

7 Testování

Testování je řádná fáze vývoje jakéhokoliv software. V rámci vývojového plánu následuje po fázi implementace, ale v praxi se testování provádí zároveň s implementací. Hotové části kódu se testují a v případě nálezu chyby se musí kód vrátit zpět pro doladění. Testování tedy probíhá jak po úsecích kódu, tak v podobě testů nad celou aplikací při vyvinutí funkční verze. Při iteračním vývoji je to vždy při dokončení daného přírůstku. Po úspěšném otestování software interně v rámci firmy výrobce je produkt poslán k zákazníkovi pro zahájení zkušebního běhu systému.
Osoba, jež provádí testování, má většinou za úkol vytvoření uživatelské příručky. Tester má v nahlížení na produkt blíže k perspektivě uživatele, než vývojář. Tester je také často vhodná osoba na provádění uživatelské podpory, zejména při menším počtu zaměstnanců firmy výrobce.
Dokument uživatelská příručka je k dispozici na CD (příloha E). Obsahuje návod jak pracovat s aplikací, popis klávesových zkratek, popis ovládacích prvků v jednotlivých oknech a je doplněna názornými „*screenshoty*“ pro dobré pochopení aplikace.

7.1 Zkušební běh systému

Zkušebním během systému se rozumí běh systému u zákazníka po předání produktu. V případě iteračního vývoje to může být vždy po předání hotové iterace. Zákazník má čas na plné otestování systému, kdy kontroluje, jestli obdržel to, co požadoval. V této fázi často komunikuje s výrobcem a žádá ho o změny v systému. Zkušební doba je ukončena schůzkou s výrobcem, kde se diskutuje nad seznamem požadovaných změn v systému. Tyto změny mohou být požadavky na opravu (pokud se našla chyba logiky systému), úpravu (změna implementované funkcionality), případně na přidání nových funkcí. Po dohodě se vytvoří nová, změnová specifikace požadavků a může také dojít k úpravě ceny softwaru a lhůty dodání.
Zkušební doba na akceptaci iterací byla se Zadavatelem domluvena na 5 dní pro nehotové verze a 10 dní po předání hotového díla. Existovala nepsaná dohoda, že pokud se jednalo o malé změny, které Zadavatel požadoval, bylo mu vyhověno dříve než s dodávkou nové iterace. Zadavatel systém využíval od druhé iterace - plnil informační systém osobami, které už měl evidované.

7.2 Změny v systému

Během zkušebního vývoje zákazník sepisoval změny, které by rád zahrnul do další iterace. K častým změnám však nedocházelo a změny, které Zadavatel požadoval, jsou zahrnuty v návrhu systému. Mezi časté změny patřilo:

- přidávání nových atributů objektu Osoba
- jiné rozmístění ovládacích prvků v okně Karta Osoby
- změna vzhledu formuláře
- přidání prvků pro jednodušší ovládání (např. reakce na stisk enter)

Protože Zadavatel po druhé iteraci aktivně pracoval s programem a plnil databázi platnými daty, museli jsme některé požadované změny provést úpravami nad hotovým kódem. Tyto úpravy tak plně nekorelují s návrhem. První tato úprava byla ukládání zaměstnání podle roku. Zadavatel požadoval zrušení ukládání dle rozsahu a chtěl, aby systém ukládal pouze rok. Tento rok se v databázi tedy

ukládá jako rozmezí mezi aktuálním dnem (pokud se nejedná o probíhající rok je hodnota roku patřičně snížena) a dnem za jeden rok od prvního data. Došlo také ke změně specifikace požadavků, kdy zamýšlený číselník místo výkonu práce byl navázán na objekt Zaměstnavatel. Tato vazba byla zrušena a nahrazena novou vazbou na číselník Zadavatel. Také slovo Zadavatel bylo v rámci zkušebního běhu u zákazníka odstraněno ze systému. Požadavek zněl nahradit ho slovem Zaměstnavatel a partnery Zadavatele pak označit slovem Zaměstnavatelé (v návrhu však používám původní názvosloví, bylo ustanoveno na prvním společném jednání, a zdá se mi jednoznačnější).
Obecně však platilo, že změny byly převážně drobné a neovlivňovaly obecné navržení systému.

8 Předání a další vývoj

Fáze předání následuje vždy po ukončení fáze implementace a to v každé iteraci vývoje. Tato kapitola se však váže k předání celého projektu. To se uskutečnilo nakonec o dva dny později vůči plánu, převážně kvůli tomu, že se obě strany závazně domlouvaly na termínu až v průběhu samotného běhu projektu. Zdržení tedy nebylo způsobeno nedodržením termínu. Předání v sobě obsahuje řadu kroků popsaných níže a je zakončeno podepsáním předávacího protokolu (je nutný podpis Zadavatele). Předávání se provádí ve dvou fázích, první je nasazení produktu do zkušebního provozu a druhá fáze je předání do ostrého provozu. Po předání do ostrého provozu projekt končí.

8.1 Předání a ukončení

Předání proběhlo ke dni 8. 4. 2009. Předání v sobě zahrnovalo:

- nastavení počítače, určeného Zadavatelem, na kterém bude informační systém používán. To zahrnuje kontrolu a případné úpravy nastavení uživatelského prostředí, instalaci a konfiguraci databázového serveru (Zadavatel nemá vyhrazený server - bude využívat pracovní stanici).
- instalaci informačního systému a konfiguraci pro připojení k databázovému serveru.
- proškolení uživatelů, kteří budou vystupovat v roli Vedoucí Evidence.
- předání uživatelské příručky.
- potvrzení předávacích listů pro předání produktu a zaškolení obsluhy

Zadavatel byl s produktem spokojen a během předání neměl k produktu žádné výhrady. Na základě toho byl zahájen zkušební provoz systému u Zadavatele, který trval dalších 10 pracovních dní. Během zkušebního období měl Zadavatel několik žádostí o změny (převážně uživatelského rozhraní), které jsme provedli. Ke dni 22. 4. 2009, kdy jsme úspěšně předali opravnou verzi systému, byl projekt Vývoj informačního systému pro evidenci pracujících cizinců ukončen. Další podpora, údržba a rozšíření se budou dále řešit obecným přístupem v podobě *maintainance* a v případě rozšíření funkcionality dalšími projekty.

8.2 Zákaznická podpora

Projekt byl tedy ukončen k datu konce zkušebního období informačnímu systému u Zadavatele. Práce na systému tím ale zcela nekončí. Vývojové oddělení má za úkol archivovat předanou verzi systému. Pokud se u zákazníka objeví během provozu další chyby, musí se provést jejich oprava. V tom případě se vytváří nová verze předané verze systému. U výrobce tak vždy jsou k dispozici všechny předané verze, poslední verze navíc běží pro potřeby telefonických poradenství na vlastním serveru.
Pokud má Zadavatel zájem o další rozšíření funkcionality, přistupuje se k tomuto požadavku jako k novému projektu, a prochází se všemi fázemi vývoje software - analýza, návrh, implementace, předání.

8.3 Další rozšíření

Systém se nyní nachází ve stavu, kdy vyhovuje potřebám Zadavatele. V rámci dalších plánovaných rozšíření bude nutné implementovat další formuláře. Pro přidávání těchto formulářů bude třeba definovat opět jejich podobu, určit nové hodnoty záznamů, přidat další informace, které systém neobsahuje a také vytvořit doplňující dialogová okna spouštějící se před samotným tiskem. Toto rozšíření, vcelku nezajímavé pro vývojáře, je pro Zadavatele stěžejní, protože ulehčuje práci jeho zaměstnancům.

Nejdůležitější navrhované rozšíření, které se mi zdá velmi potřebné, je autentizace uživatelů. Zadavatel, i přes upozornění, o ně neprojevil zájem. Jednalo by se o jednoduché logování do systému, kde by měl každý uživatel své heslo. V systému by bylo možné lepší monitorování provozu. Pro každého uživatele by se monitorovalo poslední přihlášení a v souvislosti s historickými stavy systému by se pro každou změnu dal ukládat uživatel, který tuto změnu provedl. Při případných nejasnostech ve změnách by bylo možné určit případného „viníka“. Druhou výhodou by bylo definování rolí pro každého uživatele, s případnými omezeními funkcionality. Absence požadavku na tuto funkcionalitu také zapříčinila výběr jednoduché architektury klient - server. V případě potřeby autentizace bych asi volil pokročilejší tří-vrstvou architekturu, kde by aplikační server řešil právě toto účtování.

Z hlediska delšího časového období je aplikace zaváděna jako menší informační systém, který se může postupně rozšiřovat o další zcela odlišnou funkčnost, než je pouze evidence osob. Prvním náznakem je například evidence zaměstnavatelů, jež se může rozšířit o kompletní podrobný nástroj pro jejich správu. Další možnosti vývoje nabízí správa vlastních zaměstnanců firmy, evidence jejich práce a s tím spojené mzdy.

9 Závěr

Během komunikace se zákazníkem jsem se dozvěděl mnoho informací ohledně problematiky zaměstnávání cizinců v České republice, především pak cizinců pocházejících z rozvojových zemí Evropy jako je Ukrajina, Bělorusko, Rumunsko apod. Překvapující je pro mne zjištění, kolik informací je o těchto osobách schraňováno pouze pro účely úřadu práce, ale také kolik informací uchovává personální agentura. Seznámil jsem se blíže také s fungováním firmy a s jejími požadavky. Překvapil mne fakt, že firma Zadavatele je reálným zaměstnancem osob a jejich pracovní sílu pronajímá obchodním partnerům.

Nejužitečnějším přínosem projektu hodnotím získání dalších zkušeností s vývoje softwaru na míru zákazníkovi. Tímto je celý proces komunikace se zákazníkem, proces validace jednotlivých kroků zákazníkem a také předávání a testování produktu na pracovišti zákazníka. V projektu jsem si vyzkoušel implementaci softwaru, který má dvě základní funkcionality: 1. evidence osob, kterých jsou řádově stovky, 2. Tisk dokumentů a vytváření přesných obrazů formulářů k jejich reálnému použití na úřadech. První problematika mě naučila jednoduše a efektivně osoby vyhledávat a spravovat, používat různé pokročilé komponenty, které zákazníkovi opravdu ulehčí práci a ušetří čas. Druhý bod mě přiměl zamyslet se nad tím, kolik času se může ušetřit při takovémto plnění formulářů, kdy mnoho firem a úřadů stále vypisuje množství údajů manuálně, i přesto, že data jsou již vložena do databází jejich systémů.

Jako velmi zajímavé hodnotím také rozšíření o temporální logiku záznamů v databázi. Jedná se o mocný nástroj, který při správné implementaci může snadno fungovat i ve větších informačních systémech. Pokud případný zákazník má dostatečné kapacity pro velkou redundanci dat v databázi, tak mu tento systém zajistí plný přehled o tom, jak se s informačním systémem v jeho firmě pracuje.

V neposlední řadě jsem také rád, že celý projekt vývoje informačního systému pro evidenci pracujících cizinců dopadl úspěšně, a že je nyní plně využíván ve firmě, pro kterou byl vytvořen.

Literatura

[1] Kadlec, V. *Deplhi: Hotová řešení*. CP Books, a.s.. Brno. 2005. ISBN 80-251-0017-0

[2] Pirkl, J. *Komponenty v Delphi*. CP Books, a.s.. Praha. 2002. ISBN 80-7226-746-9

[3] Guckenheimer, S. *Efektivní softwarové projekty.* Zoner Press, Brno 2007. ISBN 978-80-86-815-62-6

[4] Zendulka, J. *Pokročilé databázové systémy - Multimediální databázové systémy*. FIT VUT. Brno. 2006

[5] Milton D. Rosenau Jr. *Řízení projektů*. Computer Press, 2003. ISBN 80-7226-218-1.

[6] Maciaszek, L. A., Liong, B. L. *Practical Software Engineering: A Case Study Approach*. Harlow, Addison-Wesley, 864 p., 2005, ISBN 0-321-20465-4.

[7] Zendulka, J.. *Analýza a návrh informačních systémů - Studijní opora AIS.* FIT VUT Brno. 2006

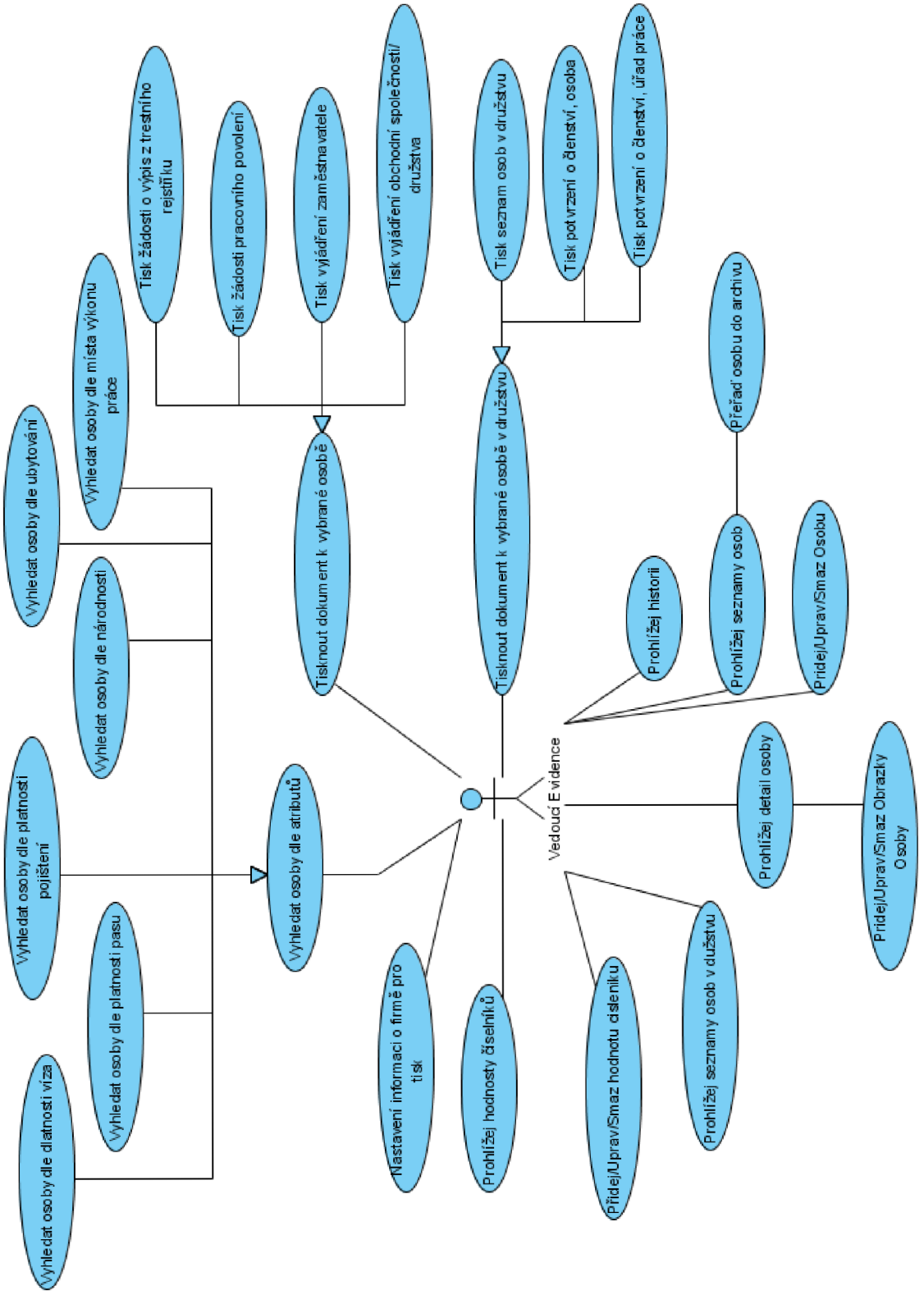
[8] Christian S. Jensen. *Temporal Database Management*. 2000. [http://www.cs.aau.dk/~csj/Thesis/]

[9] Pitaš, J. a kolektiv. *Národní standard kompetencí projektového řízení.* Společnost pro projektové řízení, o.s.. Brno. 2008. 978-80-214-3665-7

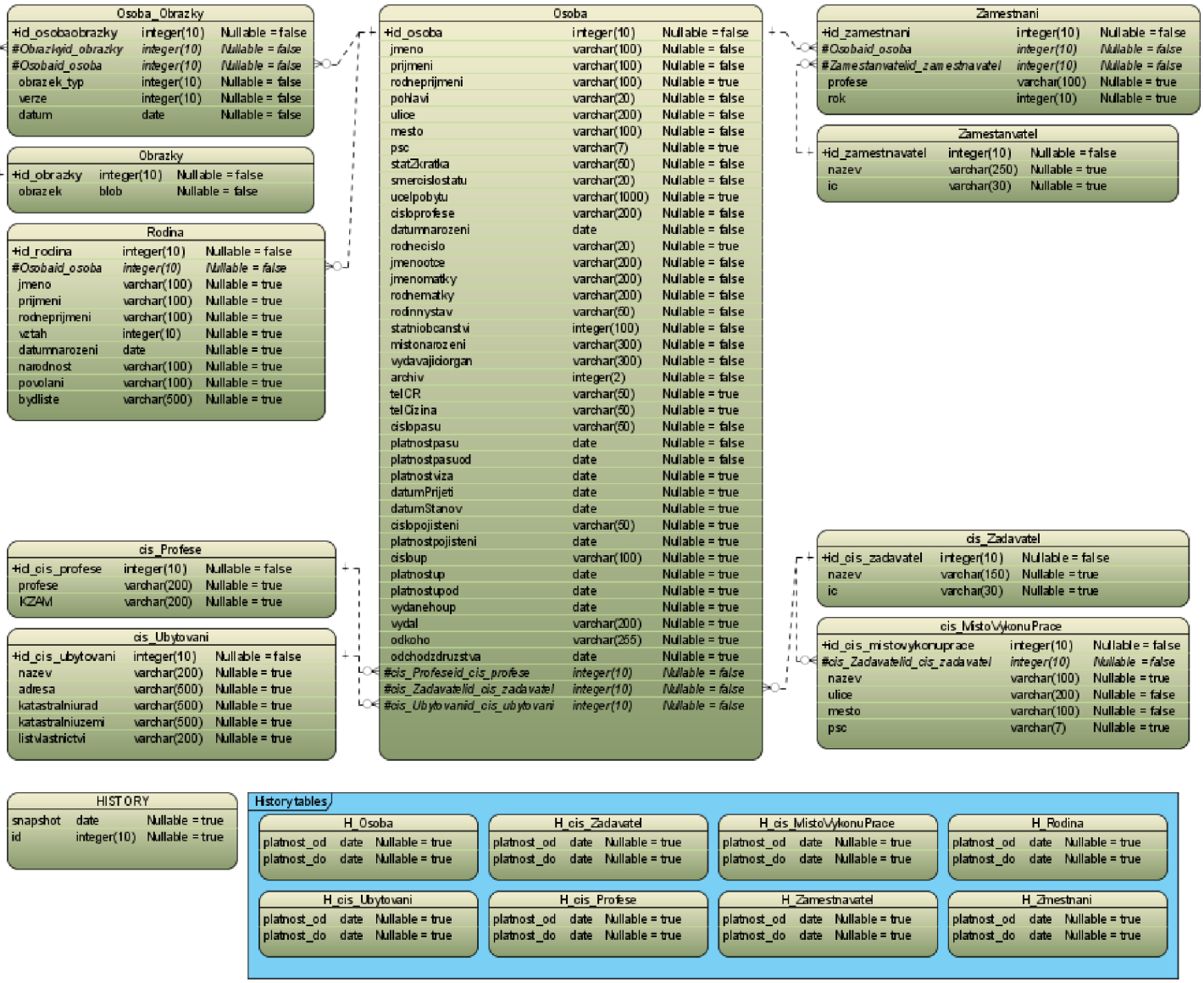
[10] www.borland.cz [online]. 2000 [cit. 2009-01-05].

Seznam příloh

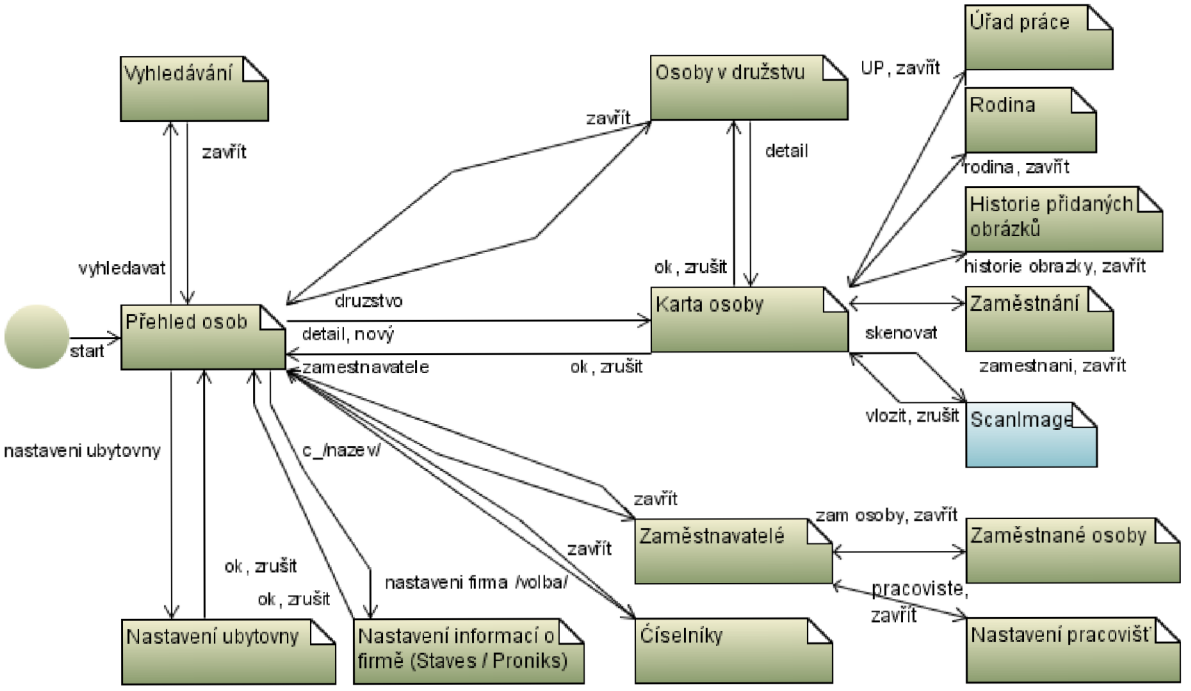
A. Use-case diagram



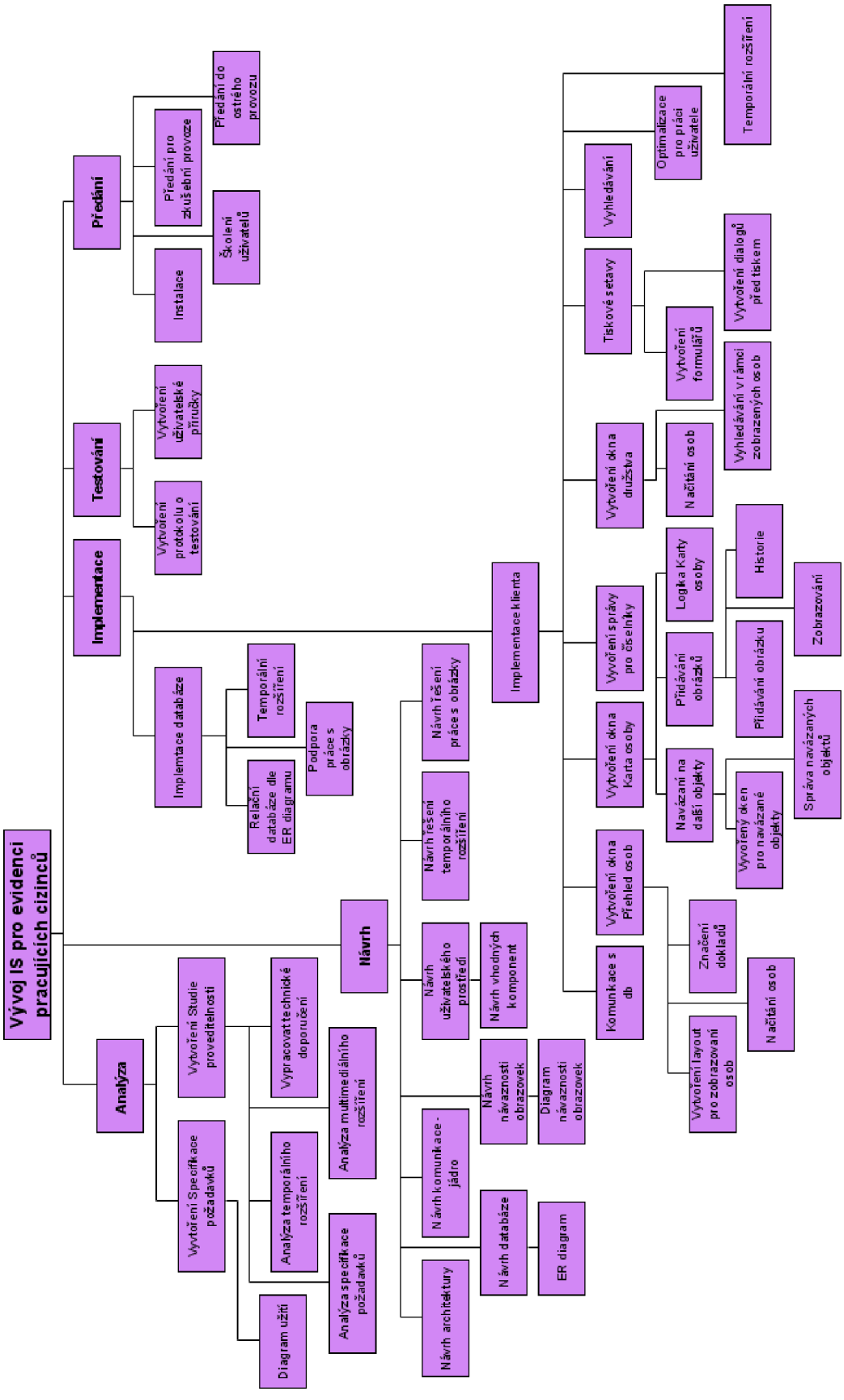
B. ER diagram


Osoba_Obrazky
+id_osobaobrazky integer(10) Nullable = false
#Obrazkyid_obrazky integer(10) Nullable = false
#Osobaid_osoba integer(10) Nullable = false
obrazek_typ integer(10) Nullable = false
verze integer(10) Nullable = false
datum date Nullable = false
Obrazky
+id_obrazky integer(10) Nullable = false
obrazek blob Nullable = false
Rodina
+id_rodina integer(10) Nullable = false
#Osobaid_osoba integer(10) Nullable = false
jmeno varchar(100) Nullable = true
prijmeni varchar(100) Nullable = true
rodneprijmeni varchar(100) Nullable = true
vztah integer(10) Nullable = true
datumnarozeni date Nullable = true
narodnost varchar(100) Nullable = true
povolani varchar(100) Nullable = true
bydliste varchar(500) Nullable = true
Osoba
+id_osoba integer(10) Nullable = false
jmeno varchar(100) Nullable = false
prijmeni varchar(100) Nullable = false
rodneprijmeni varchar(100) Nullable = true
pohlavi varchar(20) Nullable = false
ulice varchar(200) Nullable = false
mesto varchar(100) Nullable = false
psc varchar(7) Nullable = true
statZkratka varchar(50) Nullable = false
smercislostatu varchar(20) Nullable = false
ucelpobytu varchar(1000) Nullable = true
cisloprofese varchar(200) Nullable = false
datumnarozeni date Nullable = false
rodnecislo varchar(20) Nullable = true
jmenootce varchar(200) Nullable = false
jmenomatky varchar(200) Nullable = false
rodnematky varchar(200) Nullable = false
rodinnystav varchar(50) Nullable = false
statniobcanstvi integer(100) Nullable = false
mistonarozeni varchar(300) Nullable = false
vydavajiciorgan varchar(300) Nullable = false
archiv integer(2) Nullable = false
telCR varchar(50) Nullable = true
telCizina varchar(50) Nullable = true
cislopasu varchar(50) Nullable = false
platnostpasu date Nullable = false
platnostpasuod date Nullable = false
platnostviza date Nullable = true
datumPrijeti date Nullable = true
datumStanov date Nullable = true
cislopojisteni varchar(50) Nullable = true
platnostpojisteni date Nullable = true
cisloup varchar(100) Nullable = true
platnostup date Nullable = true
platnostupod date Nullable = true
vydanehoup date Nullable = true
vydal varchar(200) Nullable = true
odkoho varchar(255) Nullable = true
odchodzdruzstva date Nullable = true
#cis_Profeseid_cis_profese integer(10) Nullable = false
#cis_Zadavatelid_cis_zadavatel integer(10) Nullable = false
#cis_Ubytovanid_cis_ubytovani integer(10) Nullable = false
Zamestnani
+id_zamestnani integer(10) Nullable = false
#Osobaid_osoba integer(10) Nullable = false
#Zamestanvatelid_zamestnavatel integer(10) Nullable = false
profese varchar(100) Nullable = true
rok integer(10) Nullable = true
Zamestanvatel
+id_zamestnavatel integer(10) Nullable = false
nazev varchar(250) Nullable = true
ic varchar(30) Nullable = true
cis_Profese
+id_cis_profese integer(10) Nullable = false
profese varchar(200) Nullable = true
KZAM varchar(200) Nullable = true
cis_Ubytovani
+id_cis_ubytovani integer(10) Nullable = false
nazev varchar(200) Nullable = true
adresa varchar(500) Nullable = true
katastralniurad varchar(500) Nullable = true
katastralniuzemi varchar(500) Nullable = true
listvlastnictvi varchar(200) Nullable = true
cis_Zadavatel
+id_cis_zadavatel integer(10) Nullable = false
nazev varchar(150) Nullable = true
ic varchar(30) Nullable = true
cis_MistoVykonuPrace
+id_cis_mistovykonuprace integer(10) Nullable = false
#cis_Zadavatelid_cis_zadavatel integer(10) Nullable = false
nazev varchar(100) Nullable = true
ulice varchar(200) Nullable = false
mesto varchar(100) Nullable = false
psc varchar(7) Nullable = true
HISTORY
snapshot date Nullable = true
id integer(10) Nullable = true
Historytables
H_Osoba
platnost_od date Nullable = true
platnost_do date Nullable = true
H_cis_Zadavatel
platnost_od date Nullable = true
platnost_do date Nullable = true
H_cis_MistoVykonuPrace
platnost_od date Nullable = true
platnost_do date Nullable = true
H_Rodina
platnost_od date Nullable = true
platnost_do date Nullable = true
H_cis_Ubytovani
platnost_od date Nullable = true
platnost_do date Nullable = true
H_cis_Profese
platnost_od date Nullable = true
platnost_do date Nullable = true
H_Zamestnavatel
platnost_od date Nullable = true
platnost_do date Nullable = true
H_Zmestnani
platnost_od date Nullable = true
platnost_do date Nullable = true

C. Diagram návaznosti obrazovek



D. Hierarchická struktura činností (WBS)



E. CD příloha

Přiložené CD obsahuje:

- Zkompilovaný program
- Databáze s testovacími daty
- Skript pro vytvoření nové databáze
- Zdrojové kódy projektu
- Uživatelský manuál
- Ukázky formulářů
- Plán projektu
- Diagramy