VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

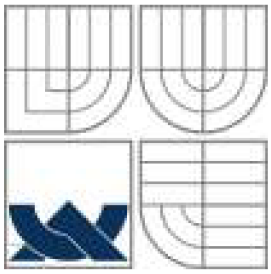
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM FINANČNÍ KONTROLY

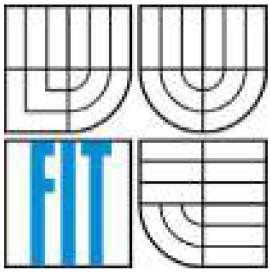
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Jiří Hanuš
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM FINANČNÍ KONTROLY

INFORMATION SYSTEM OF FINANCIAL CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Jiří Hanuš
AUTHOR

VEDOUCÍ PRÁCE — Ing. Lukáš Grullich
SUPERVISOR

BRNO 2007

Abstrakt

Práce se zabývá problematikou vývoje Informačního systému „na míru“ pro cílového zákazníka. Cílem softwaru je zefektivnit práci v oblasti finanční kontroly, evidence práce kontrolorů, správy rozpočtových položek a všech ostatních přidružených oblastí. Pro vývoj aplikace jsme zvolili kombinaci vývojového prostředí Delphi (pro uživatelské prostředí) a Oracle (pro databázi). Informační systém je třívrstvý a přístup k datům je umožněn přes tlustého klienta.

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 financial control, evidence of work, administration of budget item and all others associate areas. We choose for software development combination of development studio Delphi (for user interface) and Oracle (for database). Information system has three-level architecture and access to data is enabled by fat client.

Klíčová slova:

Informační systém, Finanční kontrola, Delphi, Oracle, třívrstvá architektura, tlustý klient

Keywords:

Information system, Financial control, Delphi, Oracle, three-level architecture, fat client.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáše Grulicha. Další informace mi poskytli spolupracovníci Ing. Michal Šos, Ing. Michal Hamsa a Ing. Ivan Tůma. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Poděkování

Rád bych poděkoval mému vedoucímu projektu, Ing. Lukáši Grulichovi, všem mým spolupracovníkům, jmenovitě Michalu Šosovi, Michalu Hamsovi a Ivanu Tůmovi, za jejich pomoc při vývoji aplikace a následně také zaměstnancům zvláštního oddělení pro finanční kontrolu za jejich informace ohledně funkčnosti programu.

Obsah

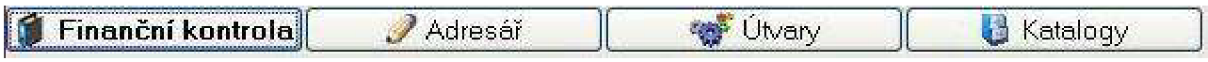
1. Úvod

Cílem této publikace je seznámit čtenáře s použitými postupy při vývoji Informačního systému finanční kontroly (software byl nedávno pojmenován Zofinka, dále se tedy budu držet tohoto označení). Zofinka vznikla na základě požadavku našeho zákazníka na zefektivnění jeho práce v oblasti finanční kontroly. S tím je spojena správa rozpočtových položek, správa šablon kontrolních listů, uchovávání záznamů o kontrolorech a evidence jejich práce na jednotlivých finančních kontrolách, adresář kontrolovaných firem a mnoho dalšího.
Vývoj softwaru začal v říjnu 2006 a vývoje se účastnili čtyři lidé. Prvotní analýzu vytvořil Ing. Ivan Tůma s ohledem na implementaci požadavků zákazníka do již existujícího informačního systému. Tento systém však nemohl poskytnout dostatečné možnosti pro úspěšné vyřešení zákazníkových požadavků a proto se rozhodlo o implementaci nového informačního systému. Pro samotnou implementaci systému jsme museli tuto analýzu značně pozměnit.
Další komunikaci se zákazníkem zajišťoval kolega Ing. Michal Šos, který byl také zodpovědný za vedení projektu po stránce harmonogramu a vývoje externích funkcí systému. Konečnou analýzu a návrh řešení jsem vedl za spolupráce Ing. Michala Hamsy, následná implementace byla rozdělena na část uživatelského prostředí, přenosu dat a aplikačního serveru na straně jedné a vývoje a správy databáze na straně druhé. Osobně jsem vytvářel první část.
Jak již z tohoto popisu vyplývá, systém byl navržen jako třívrstvý (klient – aplikační server – databázový server) s tím, že během vývoje přišel požadavek na offline verzi klienta. Proto také je implementován pouze tlustý klient pro práci v systému.
Samotná logická struktura byla během návrhu řešení rozdělena na čtyři jednotlivé moduly a to na modul Finanční kontroly, kde systém umožňuje funkcionalitu nad hlavními objekty typu Kontrolní list a Kontrolní spis, dále pak modul Adresář sloužící pro identifikaci všech kontrolovaných fyzických a právnických osob, modul Útvary pro správu vnitřní struktury organizace a jejich pracovníků a nakonec modul Katalogy pro uložení objektů, na které se ostatní moduly odvolávají (např. rozpočtové položky).

2. Upřesnění zadání

2.1. Moduly systému

Z analýzy zákazníkových požadavku vyplynulo, že systém musí být modulární. A to tak, že systém bude rozdělen do čtyř logických oblastí - modulů. Pro přehlednost budu popisovat moduly v pořadí, ve kterém na sebe navazují.
U každého modulu budu nejprve popisovat hlavní objekty, tzv. kořenové objekty, umístěné ve stromu objektů v aplikaci na nejvyšším místě. Na tyto kořenové objekty se mohou vázat další objekty modulu.



Obr. 1: Tlačítka pro přepínání modulů

Modul Katalogy

Tento modul se zabývá správou objektů, které se následně používají v dalších modulech. Mezi kořenové objekty patří **Rozpočtová položka.** Rozpočtová položka je obrazem reálné rozpočtové položky, tedy něčeho, za co byla proveden platba. Rozpočtová položka je popsána těmito atributy: Název, Kód, číselník Sektor (s hodnotami Ano/Ne), Popis. Kód je pro každou rozpočtovou položku vždy 15 číslic. V systému jsou rozpočtové položky tříděny hierarchicky podle pořadí: Kód podpory, Sektor, Rozpočtová položka.
Kód podpory je objekt spravovaný externě mimo tento modul, popíšu jej tedy podrobně později v kapitole 1.3.
Sektor slouží jako skupina rozpočtových položek a je také kořenovým objektem tohoto modulu. Má stejné atributy jako Rozpočtová položka, liší se pouze v hodnotě číselníku Sektor a v kódu, který musí být dlouhý osm číslic. Těchto osm číslic následně identifikuje všechny rozpočtové položky patřící do tohoto sektoru tak, že prvních osm číslic kódu rozpočtové položky je shodných s osmi číslicemi sektoru.

Druhým kořenovým objektem v modulu Katalogy je **Typový kontrolní list.** Tento slouží jako předloha pro kontrolní listy v modulu Finanční kontrola. Pro účely popisu modulu Katalogy popíšu pouze jeho atributy a jejich význam pak bude k nalezení v informacích o modulu Finanční kontrola. Typový kontrolní list má atributy Sektor a Název. Na tento kořenový objekt se odkazuje objekt Typový oddíl, který je popsán atributem Název.

Modul Útvary

Modul útvary je určen pro správu **Útvarů** společnosti (případně oddělení) a **Pracovníků** společnosti. Tyto dvě reálné předlohy jsou obrazem kořenových objektů v modulu Útvary. Každý z nich je popsán řadou informativních atributů, důležité je říci, že mezi nimi byla požadována vazba 0..1 a na každý kořenový objekt se odkazuje několik informativních číselníků. Ve své podstatě je tento modul nejjednodušší na implementaci.

Modul Adresář

V tomto modulu je cílem popsat a schraňovat informace o subjektech kontroly. Subjekt kontroly může být jak **Právnická osoba,** tak **Fyzická osoba**. Tyto dva objekty jsou v modulu Adresář použity jako kořenové. Oba tyto objekty jsou popsány velkým množstvím atributů, které na chod systému nemají význam a opět se čerpají hodnoty některých atributů z číselníků. Požadavkem také bylo, aby měla právnická osoba seznam kontaktů, se kterými se jednalo. Objekt Kontakt se tedy váže na právnickou osobu.

Modul Finanční kontrola

Zatímco ostatní moduly plní převážně informativní charakter, modul Finanční kontroly je středem aplikace, který všechny ostatní moduly spojuje. Kořenové adresáře jsou zde tyto dva: **Kontrolní spis** a **Kontrolní list.** Jsou hierarchicky tříděny tak, že jeden Kontrolní spis obsahuje Kontrolní listy. Kontrolní listy jsou pak obdobně řazeny odshora podle kontrolního období a typu kontrolního spisu (s hodnotami Hlavní, Vyžádaná, Třetí osoby). V podstatě jde pouze o filtrování podle atributů kontrolního spisu.

Kontrolní spis je kořenový objekt, popisující jeden případ finanční kontroly. Je popsán 32 atributy, které určují jak identifikační údaje kontroly, tak i její stav a následný vývoj (systém v daných fázích průběhu kontroly upozorňuje uživatele na termíny následných kroků). Každý kontrolní list má tři identifikační čísla: číslo spisu, číslo jednací a číslo pověření. Pro uživatelský komfort je také popsán atributem Věc, který kontrolní spis identifikuje v systému a není omezen pouze na čísla. Důležitým atributem je dále Subjekt, který je odkazem do modulu Adresář. Může odkazovat jak na fyzickou, tak na právnickou osobu. Pro ulehčení práce uživatelů je zde několik atributů automaticky počítáno z ostatních atributů kontrolního spisu (např. Kontrolní období nebo Skutečné zahájení).
Na Kontrolní spis se vážou následující ostatní objekty Finanční kontroly:

- **Kontrolor**
  Objekt Kontrolor je vytvořen zvolením požadovaného pracovníka modulu Útvary. Mezi atributy popisující kontrolora patří binární číselníky Vedoucí, Externista. Existuje omezení počtu vedoucích na jednoho. Dále pak jsou to informace o tom, kolik kontrolor vykonal na dané kontrole práce. Tyto údaje se počítají podle objektu Evidence práce, popsaného dále. Díky vazbě mezi uživatelem a pracovníkem se také touto cestou určují práva na jednotlivé finanční kontroly. Pokud není uživatel mezi kontrolory nebo nemá práva správce, má na danou finanční kontrolu práva pouze pro čtení (pokud tedy nejsou dále omezena).

- **Dokument**
  Dalším navázaným objektem je objekt Dokument. Ve své podstatě se jedná o metadata, popisující připojený soubor. Pro připojené soubory není žádné omezení na typ nebo velikost, systém je dokáže automaticky sám otevřít, pokud je pro danou příponu souboru přiřazen spouštěcí program. Soubory se dají nahrávat a zpětně stahovat do počítače. Systém sám dokumenty verzuje a je možné se tedy vrátit ke starší verzi souboru. Pro účel zefektivnění služeb byl také naimplementován systém šablon, který po vybrání požadované šablony dokumentu tuto šablonu naplní daty ze systému a uloží ji do informačního systému.

Kontrolní list je také kořenovým objektem modulu finanční kontroly a je navázán na objekt Kontrolního spisu. Ve své podstatě se tedy nejedná o kořenový objekt doslova, ale uvádím ho pod tímto označením z důvodu popsání chování aplikace. Více se dozvíte v kapitole 1.2.
Kontrolní list symbolizuje jednu stránku kontrolního spisu, tedy jednu ucelenou část. Každý kontrolní list se odkazuje do modulu katalogů na objekt Sektoru. Rozpočtové položky v tomto Sektoru pak slouží jako podklady pro jednotlivé platby. Objekt kontrolní list je dále označen kódem, který se generuje z čísla kontrolního spisu a kódu kontrolovaného sektoru. Pro vytvoření kontrolního listu je uživateli dána možnost vytvoření kontrolního listu podle šablony, tzn. Typového kontrolního listu, který se jako kořenový objekt spravuje v modulu Katalogů. Kromě jména a vybraného sektoru se do nového kontrolního listu také vytvoří oddíly podle typových oddílů přidaných k typovému kontrolnímu listu.
Na Kontrolní list se následně vážou další objekty:

- **Oddíl**
  Objektem oddíl se značí oddíl otázek a odpovědí nad daným kontrolním listem. Oddíly pak mají samostatnou odpověď, kterou po výsledcích jednotlivých otázek určí manuálně uživatel. Oddíl má dále název, zkratku, identifikátor Veřejný a poznámku. Na oddíl se váže tedy objekt Otázka, který je definován atributy Název, Odpověď, identifikátorem Veřejný a pořadím v oddíle.

- **Platba**
  Z ohledu implementace pouze informativní objekt obsahující odkaz na danou Rozpočtovou položku v Sektoru Kontrolního listu a následně údaje o výši platby, o nedostatcích a nedoplatcích. Platba dále obsahuje odkazy na číselníky Kontrolovat a Kontrolováno, které ovlivňují vzhled dané platby v systému.

- **Evidence práce**
  Objekt Evidence práce slouží pro kontrolu práce kontrolorů. Po provedené práci se do systému zavede její popis a údaj v člověkohodinách pro časy přípravy, provedení a vyhodnocení. Z těchto informací se dále počítají přehledy o odvedené práci na daném kontrolním listu a přehledy vytíženosti jednotlivých kontrolorů kontrolního týmu kontrolního listu.

- **Dokument**
  Stejné jako Dokument u Kontrolního spisu

2.2. Uživatelské rozhraní

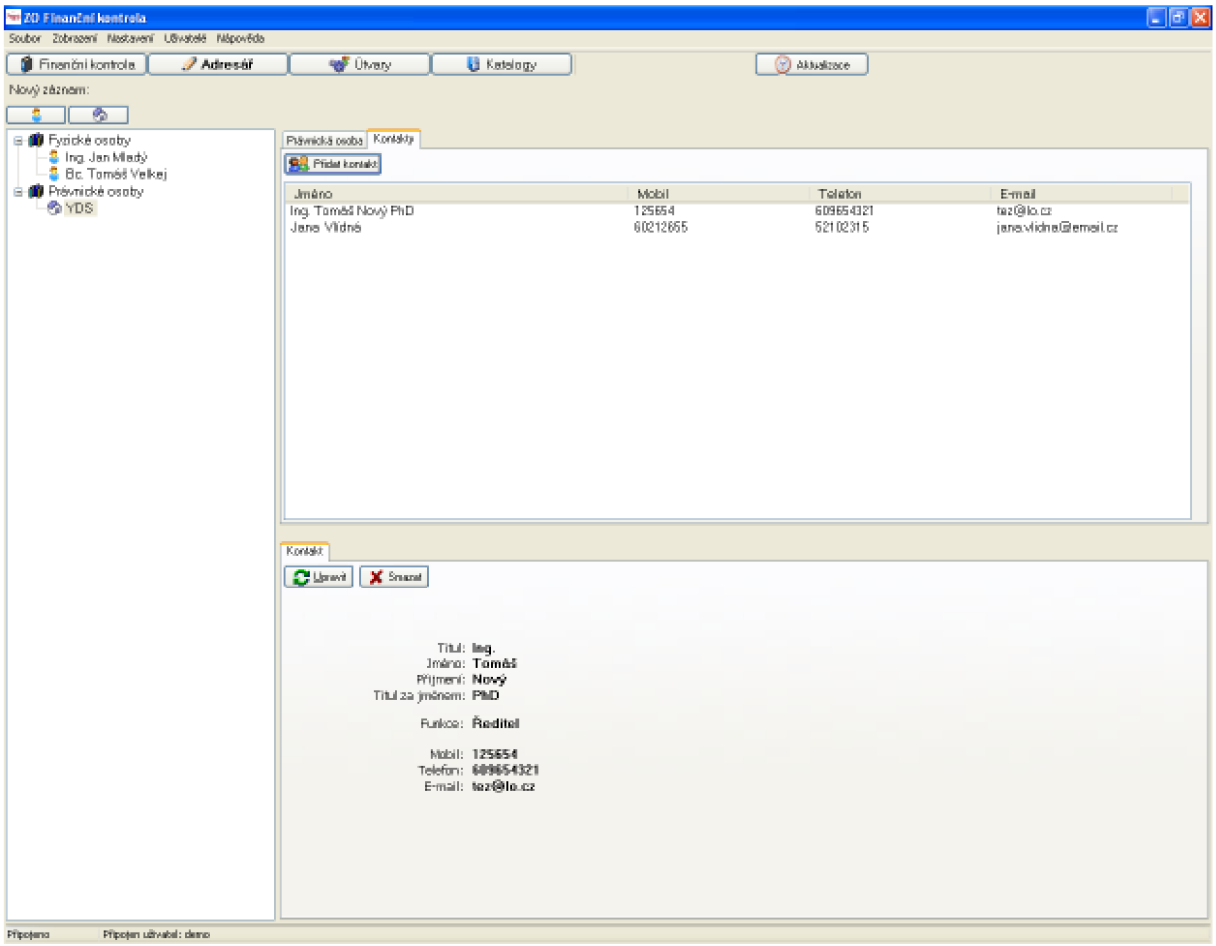
Obecné

Protože zákazník byl zvyklý používat již určité standardy, co se informačního systému týče, a stále ještě používá informační systém interního auditu, bylo nutné Zofinku přizpůsobit zvyklostem zákazníka v práci se systémem. To však bohužel nešlo udělat dokonale, protože používaný informační systém má pouze tenkého klienta. Výsledkem tedy je kompromis mezi zavedeným vzhledem a našimi potřebami pro přehledný chod systému.

Při pohledu na aplikaci je zřejmé rozložení ovládacích prvků. Tlačítka pro přepínání modulů jsou umístěny pod hlavní lištou menu. Hlavní navigační prvek, strom kořenových objektů, je zobrazen na levé straně okna a zabírá zhruba čtvrtinu celkové šířky okna. Nad ním jsou pak umístěny tlačítka pro vytvoření nových kořenových objektů. V horních pravých třech čtvrtinách okna je umístěn horní sešit

s informacemi o vybraném objektu a na něj navázaných dalších objektech. Informace o navázaných objektech jsou k dispozici v dolním sešitu.

Další funkcionalita je rozmístěna v horním ovládacím menu.



Obr. 2 Náhled na aplikaci – Modul Adresář

Strom kořenových objektů

Zde jsou ve stromové hierarchii zobrazeny kořenové objekty vybraného modulu. Strom často obsahuje atributové filtry, podle kterých se kořenové objekty řadí. Každý filtr i objekt je odlišen různou ikonou a ke změně ikony dochází při výjimkách daných objektů pro znázornění jiného stavu. Po vybrání kořenového objektu se zobrazí horní sešit, kde jsou na první straně umístěny informace o objektu a v dalších záložkách pak seznamy navázaných objektů.

Horní sešit

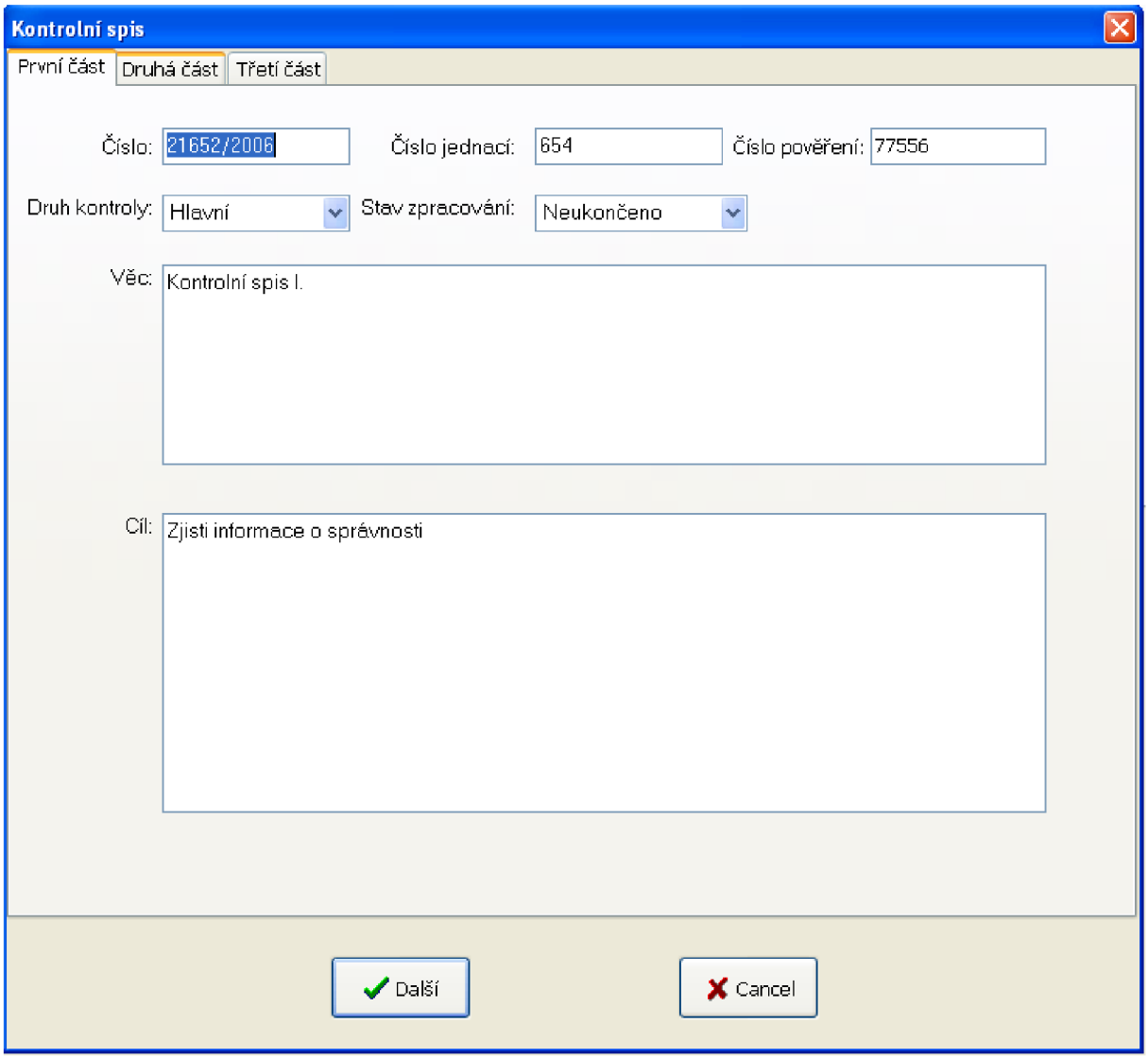
Horní sešit zobrazuje na svém prvním listě informace o vybraném objektu ve stromu, společně s ovládacími prvky pro úpravy a smazání. Na dalších listech jsou tabulky s navázanými objekty. Nad tabulkou jsou funkční tlačítka pro přidání nového navázaného objektu a případné speciální funkce.

Dolní sešit

Dolní sešit zobrazuje informace o vybraném navázaném objektu společně s tlačítky pro jejich editaci a smazání. Tento sešit je viditelný pouze tehdy, pokud je v horním sešitě vybrán navázaný objekt.

Zakládání a editace

Pro každý objekt je vytvořena dvojice oken pro zakládání a pro editaci. Okna obsahují kontrolu správnosti vkládaných dat. Při editaci platí, že se do polí pro vložení nahrají stávající hodnoty objektu.



Obr. 3: Okno pro editaci Kontrolního spisu

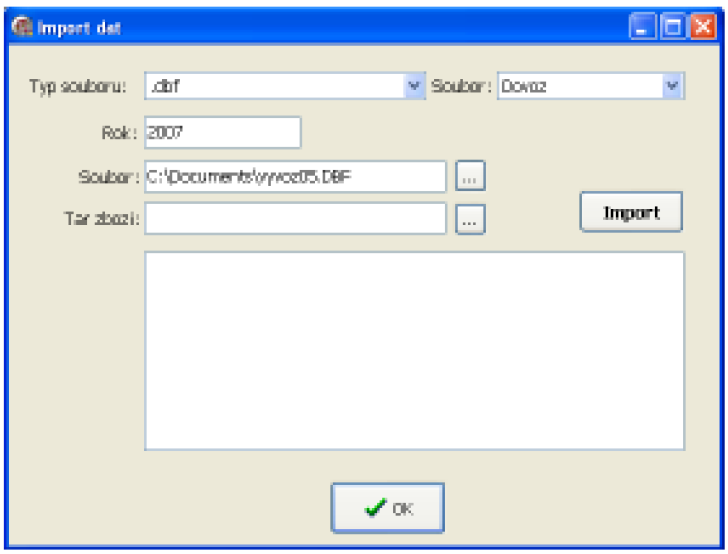
2.3. Přidružená funkcionalita

Importy a exporty

Požadavkem na systém bylo také velké množství importních a exportních formátů pro různé vstupní a výstupní informace.

Import:

- **Dbf**: Ve formátu dbf je možné importovat databázové soubory Dovoz a Vývoz. Tyto informace je možné exportovat do přehledu. Pro každý import je nutné vložit soubory Vývozu, resp. Dovozu a soubor Tar zboží.
- **Csv**: Ve formátu csv je možné importovat Xtabulku Seznam plateb. Z Xtabulky program čerpá informace o platbách a umožňuje je přiřazovat k vybraným kontrolním listům podle shodného záznamu IČO ve sloupci F200 v Xtabulce a záznamu ICO v subjektu dané kontroly.
- **Xls**: Ve formátu xls je možné importovat soubory tabulka 104 a propojené. Aby program pracoval správně, musí dojít k importu obou souborů pro daný rok. Z těchto souborů se čerpají rozpočtové položky.



Obr. 4: Okno pro Import

Export:

Exportovat je možné informace o všech objektech systému prostřednictvím okna pro vyhledávání (více v sekci Vyhledávání). Informace se ukládají do souboru csv.
Z importovaných databázových souborů je možné vytvořit uživatelem zvolené přehledy a uložit je opět do formátu dbf.
Poslední možný export je soubor ve formátu xls se jménem checklist. Jedná se o tabulku odpovědí na oddíly kontrolních listů.

Správa číselníků

Systém obsahuje nástroje pro editaci číselníků, použitých pro hodnoty vybraných atributů objektů, popřípadě použitých jako jednoduché objekty. Editace číselníků se provádí ve třech různých oknech.

- **Číselníky:** Uživatelsky editovatelné číselníky, použité jako hodnoty atributů objektů, je možné upravovat v jednoduchém okně se dvěma seznamy. První obsahuje jednotlivé číselníky a ve druhém jsou zobrazovány možné hodnoty vybraného číselníku.
- **Otázky**: Číselník otázek, který má navíc u každé otázky informaci, zda může být otázka použita obecně, nebo se jedná o otázku jedinečnou. Každá otázka zde vytvořená je automaticky schválena pro další použití v Kontrolních listech – sloupec Schváleno. Otázky kontrolorů, kteří je chtějí zařadit pro další použití, jsou označeny jako neschválené (Ne ve sloupci Schváleno) a dokud nejsou označené pomocí tlačítka Schválit jako schválené, nezobrazují se v nabídce otázek pro Kontrolní list.
- **Kód podpory**: Nejvyšší hierarchický objekt pro řazení rozpočtových položek je ve své podstatě pouze číselník se dvěma unikátními atributy, Názvem a Kódem.

Uživatelé a oprávnění

V systému je implementována správa uživatelů a jednoduchá správa práv. Do správy uživatelů patří vytváření nových uživatelských účtů, úpravy a generování nových hesel a mazání účtů. Každý uživatel musí mít nejprve svou identitu zavedenou v systému, tuto prvotní identifikaci provádí správce systému, když zavádí jednotlivé pracovníky. Vytvoření uživatele je pak přiřazení uživatelského účtu k pracovníkovi.
Práva a jejich spravování jsou zavedena co možná nejjednodušeji. Práva se určují pouze nad jednotlivými moduly a následně ještě nad funkcionalitou pro vyhledávání, importy a exporty, správou číselníků a správou uživatelů. Stupně oprávnění jsou následující: žádná, čtení, vytváření, editace, mazání, správce. Každé třída povoluje všechny možnosti práv nižších a přidává uživateli novou možnost pochopitelnou z názvu práva.

Šablony

Šablony slouží pro vytváření dokumentů kontrolního spisu, popřípadě kontrolního listu. Šablona je formulář, který se při připojení k danému objektu automaticky naplní jeho atributy. Šablony jsou rozděleny podle příslušnosti k danému typu kontrolního spisu. Podle toho se také omezuje výběr při vytváření šablony. K prohlížení šablon je vytvořeno vlastní okno.

Vyhledávání

Systém umožňuje vyhledávat jednotlivé objekty pomocí okna vyhledávání. Zde je k dispozici seznam vyhledatelných objektů a následně až tři možnosti pro omezení výběru pomocí hodnoty atributu. Atributy se automaticky načítají po vybrání objektu.
Nalezené objekty jsou zobrazeny v tabulce s jejich atributy. Tuto tabulku je možné exportovat do csv souboru či vytisknout. Pro každý objekt jsou připraveny tiskové sestavy.

2.4. Offline klient

V průběhu implementace systému přišel požadavek na vytvoření offline klienta, který by pracoval stejně jako tlustý klient aplikace připojené na server. Protože došlo k problému v komunikaci, implementovali jsme nesprávně automatickou synchronizaci dat při připojení z offline klienta na aplikační server aplikace. Toto naše řešení je plně funkční, avšak zákazník, který svůj požadavek na komunikaci mezi offline klientem a serverem specifikoval až později, tento systém využít nemůže, protože uživatelé s offline klientem se na aplikační server vůbec nedostanou.
Po vytvoření offline klienta tedy byl další požadavek na synchronizaci pomocí xml souborů.

2.5. Požadavky na chod systému

Požadavky na systém byly následující:

- Podpora Unix databázového serveru
- Tlustý klient
- Třívrstvá architektura
- Podpora operačního systému Windows na klientských stanicích
- Rozlišení 1280 x 1024 pixelů

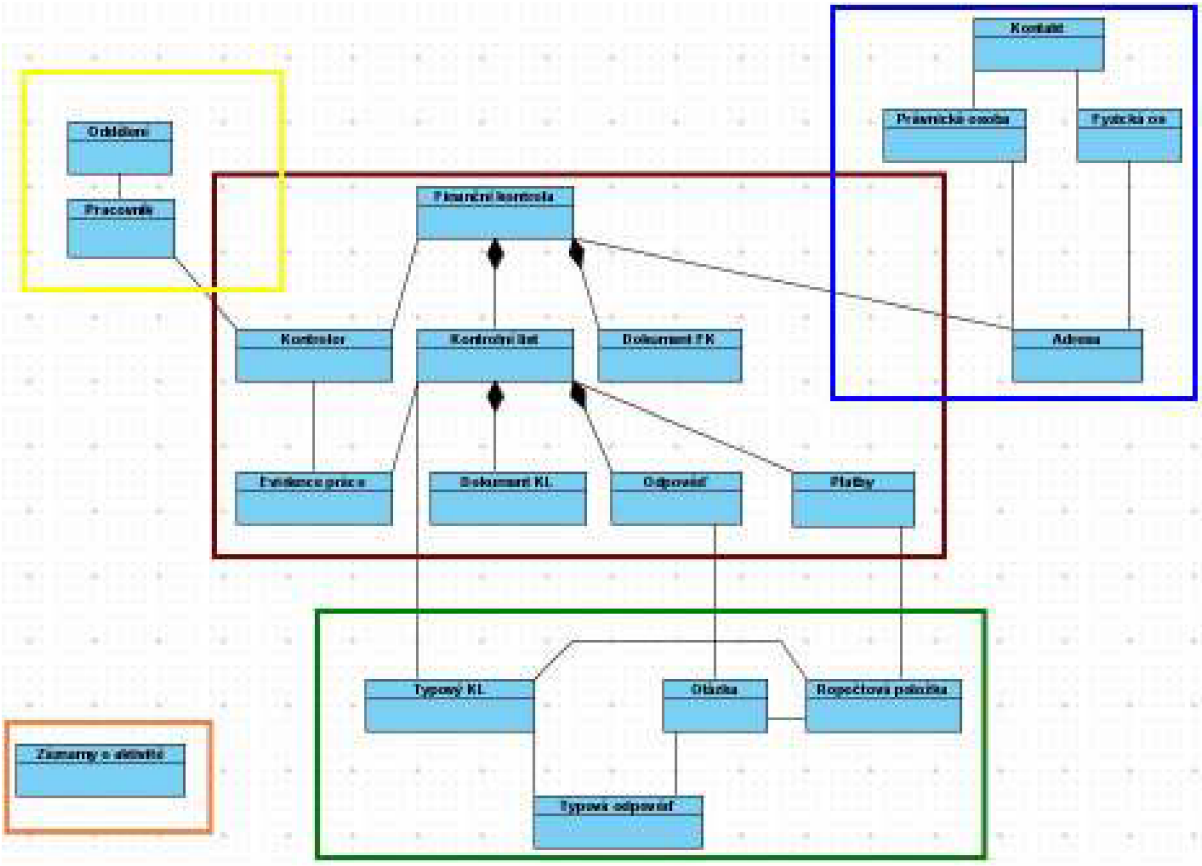
3. Teoretické řešení

3.1. Diagramy

Obecné

Pro rozsáhlost řešení zde uvádím pouze zjednodušené diagramy, ve kterých chybí jednotlivé atributy objektů. Také zde chybí odkazy na objekty číselníků a vedlejších objektů, které nepatří do hlavní kostry programu. Neúplnost je způsobena velkým množstvím objektů.

Diagram hlavních objektů a jejich rozdělení do modulů

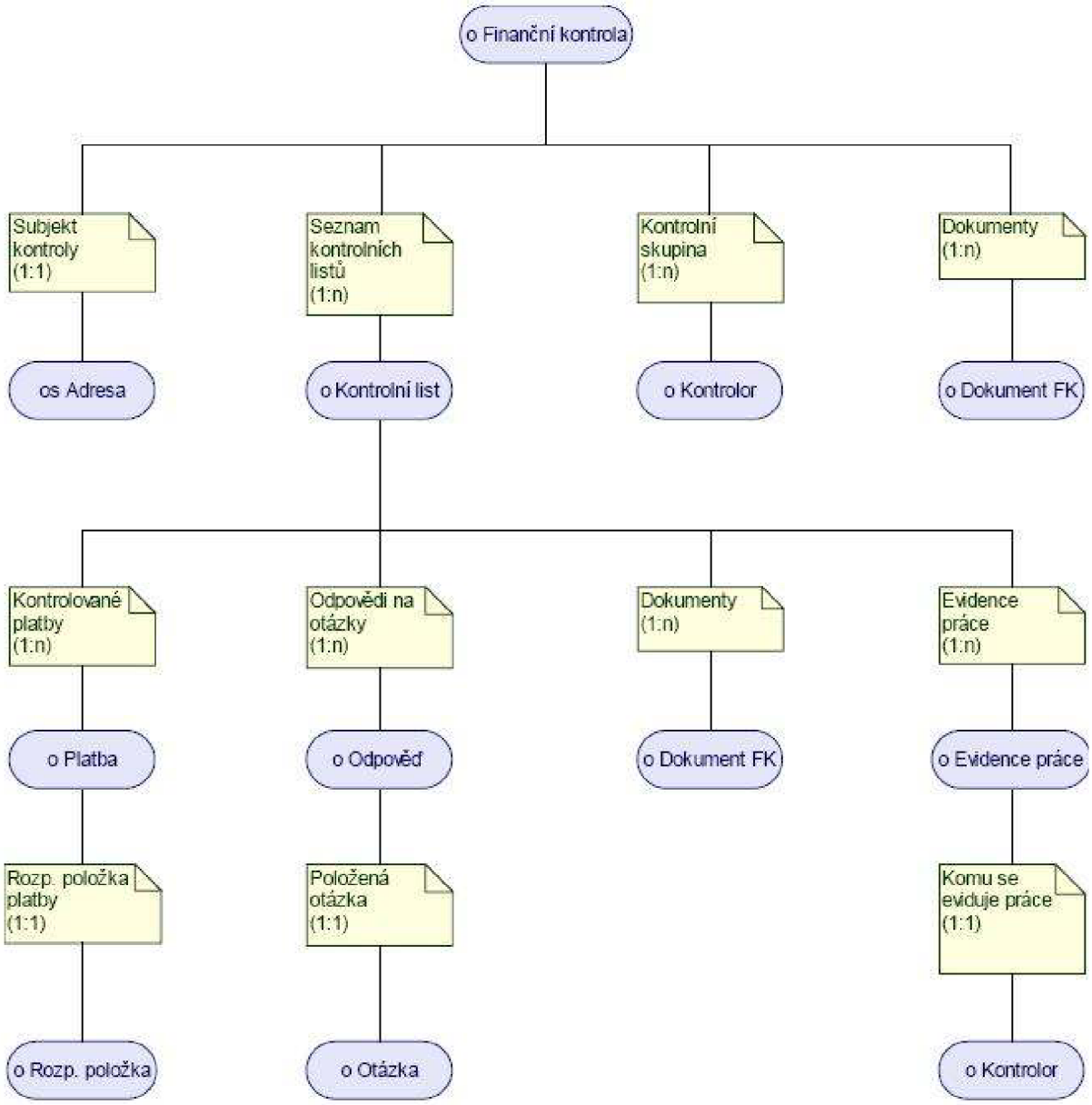


Obr. 5: Diagram kořenových objektů programu

Na tomto diagramu je možné vidět spojitost jednotlivých objektů. Příslušnost objektu do modulů je zde vyznačena pomocí barevných čtverců.

- Červený je pro Finanční kontrolu
- Zelený pro Katalogy
- Žlutý pro Útvary
- Modrý pro Adresář

Digram objektů ve finanční kontrole



Obr. 6: Hlavní objekty modulu finanční kontroly

Schéma znázorňuje všechny důležité objekty datového modelu, které jsou potřeba k úplné evidenci finančních kontrol.

Tabulka atributů finanční kotnroly

atribut	zdroj	typ	délka	poznámka
Číslo	výraz	text	0	číslo/rok
Věc	hodnota	text	500	Převzít název subjektu
Číslo jednací	hodnota	text	20	
Cíl	hodnota	text	1000	
Subjekt	odkaz	identifikátor	0	
Otisk subjektu	hodnota	text	500	
Druh kontroly	číselník	celé číslo	0	hlavní, křížová, vyžádaná
Kontrolované období	hodnota	text	50	
Oznámeno subjektu	hodnota	datum a čas	10	
Plánované zahájení	hodnota	datum a čas	0	
Skutečné zahájení	hodnota	datum a čas	0	
Protokol předán	hodnota	datum a čas	11	
Skutečné ukončení	hodnota	datum a čas	0	
Plánované ukončení	hodnota	datum a čas	0	
Náprava do	hodnota	datum a čas	0	
Návrh nápravy	hodnota	text	200	
Informace ministrovi	číselník	celé číslo	0	
Námitky	číselník	celé číslo	0	
Námitky podány	hodnota	datum a čas	12	
Rozh. o námitkách	hodnota	datum a čas	13	
Rozh. o námitkách	hodnota	text	100	
Uznané námitky	číselník	celé číslo	0	
Skartační znak	číselník	celé číslo	0	
Skartační lhůta	hodnota	celé číslo	0	
Cizí čj.	hodnota	text	20	
Stav zpracování	číselník	celé číslo	0	
Místo uložení	hodnota	text	50	
Vyřízeno dne	hodnota	datum a čas	0	
Výsledek	číselník	celé číslo	0	

Obr. 7: Tabulka atributů kontrolního spisu

Pro příklad uvádím tabulku atributů objektu kontrolní spis – dříve pojmenovaným finanční kontrola.

3.2. Rozložení prací

Obecné

Protože se na programu podílelo několik lidí, bylo cílem rozložit práci tak, abychom co nejlépe využili zbývající čas k tvorbě projektu.

Analýza potřeb zákazníka

Za tuto část byl na začátku projektu zodpovědný Ing. Ivan Tůma. Protože však byl na začátku zamýšlen jiný postup implementace, výsledný dokument o analýze nebyl pro naše účely zcela v pořádku. Hlavním problémem bylo, že se analýza odkazuje na již hotové části systému a dokonce již hotové moduly. Jejich funkčnost tak nebyla dobře popsána nebo chyběla úplně.
Druhým problémem bylo také to, že se objekty vázaly na již vytvořené hierarchie objektů a nebylo zřejmé, jaké objekty jsou pro vývoj aplikace třeba a jaké ne.
Protože od této analýzy a naší implementaci uběhlo asi půl roku, ukázalo se, že představy zákazníka se také velmi změnily. Na začátku tvorby tedy bylo nutné tuto analýzu značně upravit. Pan Ing. Ivan Tůma se pak do projektu už dále nezapojoval.

Konzultace se zákazníkem a příprava externí funkcionality

Tuto část měl na starosti kolega Ing. Michal Šos. Nejprve byla jeho úloha pouze v roli analytika, v průběhu vývoj byl pak pověřen také implementací externí funkcionality, jako je přenos souborů, exporty a importy do různých formátů a vytvoření správce pro aplikační server.
Pan Šos dále fungoval jako odborný konzultant pro problémy spojené s vývojem v Delphi, protože za sebou má již několik jiných projektů v tomto vývojovém prostředí vytvářených.
V neposlední řadě se také zapojil do vývoje při programování offline klienta a to převážně na poli uživatelského rozhraní.
Po každé konzultaci se zákazníkem jsem od něj dostávali zprávu s novými požadavky zákazníka společně se seznamem úprav a oprav.
Komunikace s panem Šosem probíhala především e-maily, případně také s využitím programů ICQ a Skype. I proto se občas vyskytly problémy v komunikaci, které vyústily ve zpětné zbytečné opravy kódu.

Databáze a návrh

V oblasti implementace samotné jsem pracovali převážně dva. Ing. Michal Hamsa byl zodpovědný za návrh a funkčnost databáze. Zpracoval a přepsal část původní analýzy, převážně pak upravil atributy jednotlivých objektů, navrhl jejich typy a velikosti.
Sám pak vedl vývoj databáze, kde pracoval v systému Oracle s využitím PL/SQL pro psaní uložených procedur. Protože pan Hamsa má široké znalosti v této problematice, bylo možné převést část logiky na databázový server, čímž jsme dosáhli menšího vytížení sítě.
Po vytvoření tabulek, jejichž definici jsme čerpali z analýzy, přišly na řadu procedury a pohledy, které se nejprve připravily podle daného standardu a následně upravovaly podle potřeb vývoje logiky programu. Věřím, že při dalším projektu by bylo možné standardizovat vetší část těchto potřeb a vytvářet tak procedury nezávisle od aktuálního vývoje programu.
Protože jsme s panem Hamsou byli ve stejné kanceláři, byla komunikace spolehlivá a neviděl jsem v ní žádný problém. Důležité rozhraní a další požadavky jsem si předávali písemně.

Vedoucí projektu

Z důvodů znalosti softwaru, který jsme měli nahradit, a také znalosti fungování finanční kontroly jsem byl pověřen vedením tohoto projektu. To spočívalo v sestavení týmu, předložení nové funkční analýzy zákazníkovi, vedení implementace a dohlížení nad provedenými pracemi. Problémy, které s tím byly spojeny, se zabývá celá tato práce.

4. Praktické řešení

4.1. Aplikační server

Administrace aplikačního serveru

Aplikační server je důležitým prvkem systému spojující klienty s databází. Server běží jako služba operačního systému a pro nastavení a správu je vytvořen obslužný program. V tomto programu se uživateli nabízí možnost nastavení přístupu k databázi, přístupového jména a hesla. Také se zde definuje port, na kterém aplikační server naslouchá klientům.
V programu pro správu se také nachází informace o stavu aktuálního připojení s databází a klienty. Dvě tlačítka, viditelně umístěná na hlavním okně, pak slouží k vypínání a zapínání služby aplikačního serveru.

Obr.8: Nástroj pro správu aplikačního serveru

4.2. Síťová komunikace

Obecné

Aplikace byla navržena s třívrstvou architekturou, takže se komunikace rozdělila na části klient-aplikační server a aplikační server-databáze. Přináší to výhodu v jednotném spojení s databází a lepší kontrole nad možnými kolizemi v zápisech a editacích nad databází.

Komunikace s aplikačním serverem

O komunikaci mezi klienty a aplikačním serverem se stará funkce DoCommand. Ta má pouze jediný parametr a to String. Do něj jsou pak ukládány jednotlivé proměnné pro aplikační server, oddělené svislou čarou. Tímto se zajistí neomezené množství poslaných parametrů.Dochází však k typové konverzi. Obvyklá funkce DoCommand vypadá takto:

```
DoCommand('SELECT|P_KONTROLNI_LIST.R_KONTROLNI_LIST_SEL|2|' + id_kl);
```

První parametr udává proceduru, kterou aplikační server zavolá, druhý pak název procedury a balíčku na databázovém serveru, předposlední číslo udává identifikační kód prováděného selectu a poslední je ID kontrolního listu.

Po zpracování vrací tato funkce do globální proměnné Data, typu Array of string, řádky záznamů výsledku. Jednotlivé hodnoty jsou oddělené opět svislou čarou, takže je nutné zavolat funkci pro parsování. Ta zapisuje do globální proměnné RSeznam, také typu Array of string, hodnoty nalezené na daném řádku. Tímto postupem se provádí většina selectů z databáze. Použita je standartní komponenta pro přenos po síti TCP/IP.

Komunikace s databází

Databáze se volá přes komponentu ADO StoredProc. Po příchodu zprávy v podobě řetězce od klienta zavolá server proceduru pro parsování. Ve chvíli, kdy aplikační server dostane požadavek na práci s databází, rozčlení podle první proměnné, zda se jedná o select, insert nebo update. Tyto funkce jsou jednotné pro všechny objekty, vyjma číselníků. Pro ty je použita obdobná trojice procedur, avšak bez zbytečného posílání kontrolních proměnných o úspěchu zápisu.
Po zjištění, kterou proceduru má aplikační server použít, se z druhé poslané proměnné určí, jaká uložená procedura se na databázi zavolá. Ta je určena balíčkem, ve kterém se nachází a názvem.
Pokud je volán select, rozhodli jsme se pro zavedení volby, určené první vstupní proměnnou selectu. Ta určuje, podle čeho a jaké sloupce tabulky či pohledu select vrátí. První tři jsme určili následovně:

- **Volba 1**: Pro všechny záznamy v tabulce
- **Volba 2**: Pro jeden záznam z tabulky, identifikovaný pomocí ID
- **Volba 3**: Pro záznamy se stejným číslem nadřazeného objektu

Procedury pro select mohou pracovat až s pěti vstupními parametry, většinou jsou však využity pouze dva. Výsledné záznamy se uloží do řetězců po řádkách s hodnotami oddělenými svislou čarou.
Při insertu a updatu je situace opačná, procedura zpracuje předané parametry a vrátí zpátky informaci o výsledku vložení. Informace z databáze je rozdělena do tří proměnných, které aplikační server vrací klientovi. První číslo udává, jestli bylo správně vloženo, druhé se při chybě naplní číslem chyby a při úspěchu ID vloženého prvku, poslední je pak řetězec, ve kterém je při chybě uchováno chybové hlášení.

Aktualizace

Aktualizace objektů v objektu se provádí automaticky každých deset minut. Je možné ji spustit také automaticky. Při přihlášení klienta na server se při úspěchu vrátí klientu identifikační číslo, kterým se při každé aktualizaci klient identifikuje. Server si uchovává informace o „živých“ klientech pomocí tabulky, která se každých deset minut přenastavuje. Každému klientu, identifikovanému zaslaným klíčem, se připíše při identifikaci jeho objektů ve sloupci Živý hodnota True. Při přenastavení serveru se všem klientům, majícím ve sloupci Živý True, přepíše True na False a všichni s False se smažou. Dalšímu klientovi, který se připojuje, se přidělí nejmenší volné číslo.
Při každém insertu, případně updatu, se všem ostatním klientům nastaví jejich příznak ve sloupci Aktualizovat na True. Při aktualizaci se tento příznak opět nastaví na False.
Tento systém kontroly zajistí dvě výhody. Pokud se v systému pouze informace prohlíží, nedochází k přenačítání objektů, které trvá několik vteřin, a také se toto přenačítání neprovádí u klienta, který změnu vyvolal – u něj se provede přenačtení ve chvíli potvrzení změny. Druhá výhoda je spíše obchodní a spočívá v omezení souběžných připojení.

Přenos a správa souborů

Po přidání metadat a připojení souboru je nutné tento soubor přenést do definovaného úložiště. To se definuje na aplikačním serveru a zde je zamýšleno, že úložiště pro dokumenty vznikne. Samozřejmě to

může být také jakýkoliv jiný síťový počítač. O problematiku přenosu souborů se stará komponenta pro správu toku dat mezi aplikačním serverem a klientem.
Ve chvíli, kdy aplikační server dostane požadavek pro vložení souboru, nejprve pošle databázi záznam o jeho vložení. Ten se zapíše do tabulky s metadaty. Obsahuje název souboru a generované číslo dokumentu - Verze. Verze i ID se vrací po úspěšném vložení těchto záznamů do tabulky a aplikační server je přidá do názvu souboru, který následně uloží do úložiště. Pokud jsou u záznamu o souboru již uloženy informace o názvu a o verzi, tyto staré informace se přesunují do tabulky Verze, kde je k nim přidružen cizí klíč ID tabulky soubor. Do tabulky Soubor se pak zapíší nové údaje.
Pro zpětné čtení a stahování souboru se tento soubor pošle pomocí stejné komponenty jako pro nahrání klientovi, který si následně opět zkrátí název o ID a verzi (tyto údaje jsou odděleny znakem #) a na konec se přidá náhodně generované číslo. To zabrání přepisování různých verzí na klientovi. Klient si soubory ukládá do adresáře temp.

4.3. Databáze

Obecné

Uživatelem zadané informace se uchovávají v databázi, kde jsou uloženy v celkovém počtu třiceti tabulek. Pro přístup k informacím se využívá nejen čtení z tabulek, ale také pohledů. Kromě několika málo případů, kdy je použito přímé čtení z databáze, se k informacím přistupuje pomocí uložených procedur, vytvořených v jazyce PL/SQL. Ty jsou podle tabulek roztříděny do balíčků.
V databázi se také mimo jiné nalézají již naplněné needitovatelné číselníky (např. Kraj).

Záznamy

Při analýze jsem zvolili několik pravidel pro přidávání a úpravy záznamů v tabulkách. Každý záznam bude mít automaticky přidělené ID, které se generuje pomocí sekvencí. Tímto ID je každý objekt identifikován a cizí klíče se odkazují právě na toto ID.
Každý záznam, vyjma číselníků, obsahuje také na posledních místech čtyři sloupce, do kterých se při každém zápisu nebo změně zapisuje ID uživatele, který vložení resp. změnu provedl, a také čas, kdy k tomu došlo. Tyto informace mohou složit k budoucímu monitoringu prací nebo k synchronizaci offline klientů se serverem.

Struktura

K informacím uložených v tabulkách se přistupuje pomocí procedur z balíčků. Balíčky jsou pojmenovány podle tabulky, nad kterou se provádí uložené procedury v nich sepsané. Pro přehlednost jsou jednotlivé procedury pojmenovány tedy takto:

P_KONTROLNI_LIST.R_KONTROLNI_LIST_INS

Kde P_KONTROLNI_LIST představuje název balíčku a R_KONTROLNI_LIST_INS název procedury. V balíčku jsou uloženy všechny procedury pro select a update, popřípadě speciální funkcionalitu (např. funkce pro zjištění, zda kontrolní list odkazuje na právnickou nebo fyzickou osobu v subjektu).
Funkcionalita, která se nedá přiřadit k žádné tabulce, je umístěna v modulu OTHERS.

Status

Každý záznam obsahuje také sloupec Status. Protože jsme se rozhodli, že uživatel může v krajním případě zajít tak daleko, že si sám smaže důležitá data, v systému se nikdy nic nemaže. Pro uživatele jsou viditelné všechny objekty, které mají status větší než 0.
V případě, že se z aplikace zavolá procedura DoCommand s prvním parametrem DELETE, na aplikačním serveru se tento požadavek přeloží jako jednoduchý update statusu daného objektu. Ten se nastaví na 0. Všechny selecty v aplikaci tedy počítají s kontrolou tohoto sloupce, stejně tak jako kontrola jedinečnosti Kdy, která vždy zohledňuje prvky se statusem větším než nula.

4.4. Nahrávání objektů

Obecné

Kdy a kolik načíst záznamů z databáze bylo rozhodnutí, které jsme museli učinit velice brzy a na záleželo na něm velká část funkcionality systému. Systém tedy funguje tak, že se při startu načtou všechny objekty do stromů. A po jejich zvolení a vybráním stránky s navázanými objekty se načtou navázané objekty do tabulky.
Pro každý objekt existuje v systému globální proměnná, která udržuje ID objektu právě vybraného.
V závislosti na tom musím říci, že tato myšlenka mohla být vedena ještě dále a v této proměnné mohly být uchovány všechny informace z databáze o aktuálním objektu. Bohužel, už jsem tento systém měli hluboko zapracovaný a nevyplatilo by se ho měnit.
Pro načtené objekty také existují pole uchovávající o nich základní informace – ID, název, případně další, které jsou zobrazeny ve stromu, resp. v tabulce. Pole je sepisováno tak, aby index každého prvku odpovídal jeho pořadí v dané komponentě.

Načítání do stromu

Načtení objektů do stromu se provádí při startu aplikace a následně také při každé aktualizaci (interval 10 minut). Strom se také načte při úpravě, smazání nebo vložení kořenového objektu.
Pro každý strom, jeden v každém modulu, platí, že má svou vlastní funkci, která zohledňuje načtení kořenových objektů podle dané hierarchie. Protože pro každý modul existují dva kořenové objekty, má každý z nich v databázi uloženou proceduru _TREE, která vrací pouze omezený počet atributů, nutných pro jednoznačnou identifikaci objektu, případně pro speciální kontrolu definovaných hodnot a následnou úpravu ikony v případě upozornění na nutný krok uživatele v dalším vývoji objektu. Příkladem může být uložená procedura R_KONTROLNI_SPIS_TREE, která vrací těchto osm atributů: ID, Typ kontroly, Kontrolní období, Číslo, Věc, Skutečné zahájení, Námitky podány a Výsledek.

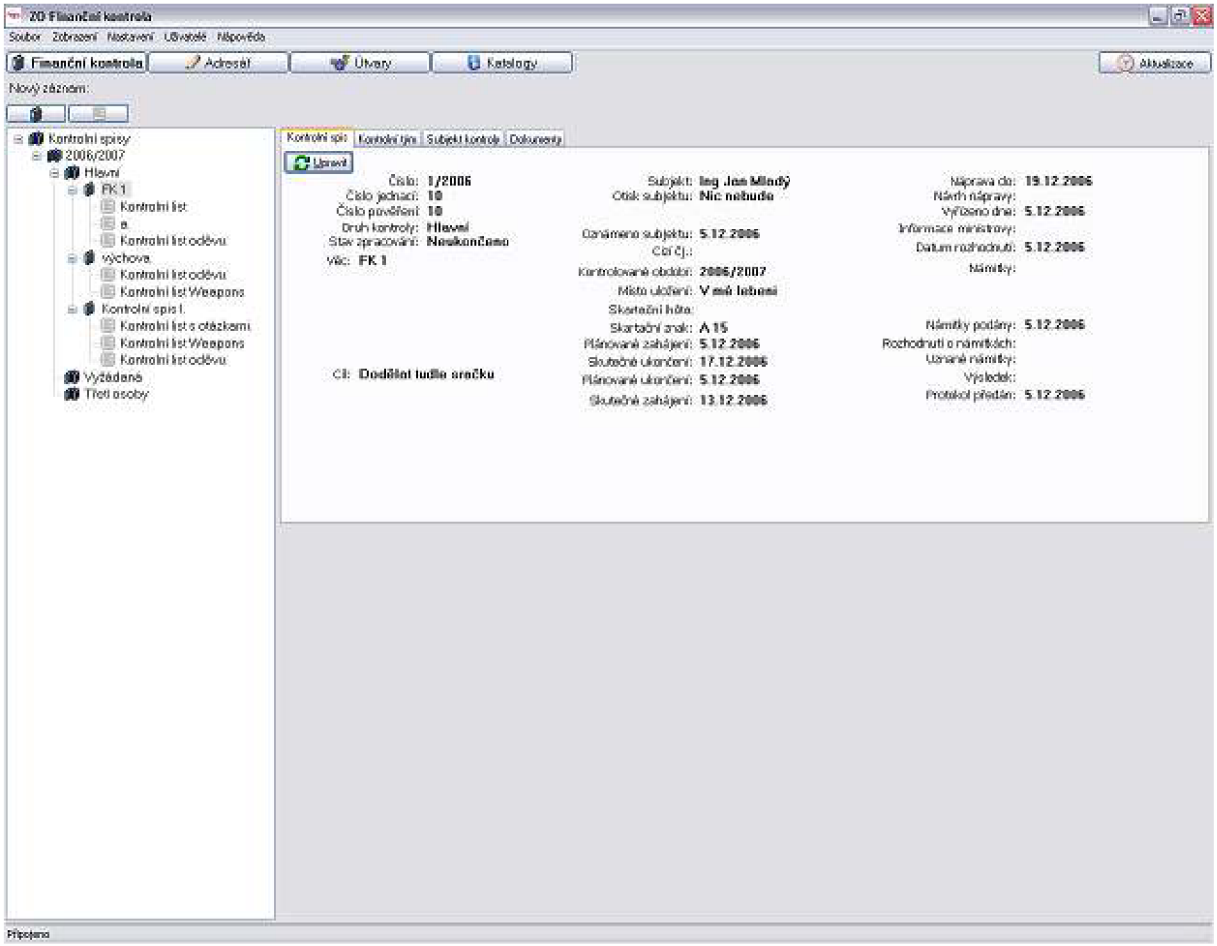
Načítání do tabulek

Načítání do tabulek je obsluhováno procedurou LoadTable s dalšími parametry. Prvním je řetězec, volající pomocí funkce DoCommand správnou proceduru na databázi. Druhým atributem je jméno ListView, do kterého se údaje zobrazují. Třetí atribut předává ID aktuálního objektu, na který se objekty v tabulce vážou, a posledním atributem je tabulka, do níž se informace budou ukládat. Funkce zavolá podle aktuálního nadřazeného objektu proceduru pomocí funkce DoCommand a výsledek uloží do předané tabulky a zobrazí v ListView. Na databázi jsou připravené funkce s koncovkou _TABLE pro načítání identifikujících položek, zobrazených v tabulce.
Procedura LoadTable se nahrává pokaždé, když dojde ke změně zobrazené stránky v horním sešitu, případně po vložení nebo upravení objektů v tabulce zobrazených.

4.5. Tlustý klient

Moduly

Moduly jsou vytvářeny pomocí komponent frames. V hlavní liště programu jsou pak ovládací prvky pro jejich přepínání.



Obr. 9: Modul finanční kontrola

Finanční kontrola

Modul finanční kontroly umožňuje práci se samotnými finančními kontrolami a jejich kontrolními listy. Tyto dva prvky jsou umístěny ve stromě a jsou řazeny podle skutečného zahájení finanční kontroly a jejího typu. Kontrolní listy se pak vážou vždy na jednu finanční kontrolu.
Finanční kontrola zpracovává řadu informací o samotné finanční kontrole. Jméno, Cíl, identifikační čísla, důležitá data. Jedná se o hlavní prvek celého systému.
K finanční kontrole je nutné určit kontrolní tým s jedním vedoucím. Uživatelé uvedení v tomto seznamu mají právo do celé finanční kontroly nahlížet.
Finanční kontrola také obsahuje informace o kontrolovaném subjektu s možností ho editovat.
K finanční kontrole je možné také přidávat neomezené množství dokumentů. A ty následně přehrávat.
Kontrolní list znázorňuje jednotlivé práce na dané finanční kontrole. Kontrolní listy jsou vždy nad jedním sektorem rozpočtových položek. Při jejich vytváření je nutné mít ve stromě označenou finanční

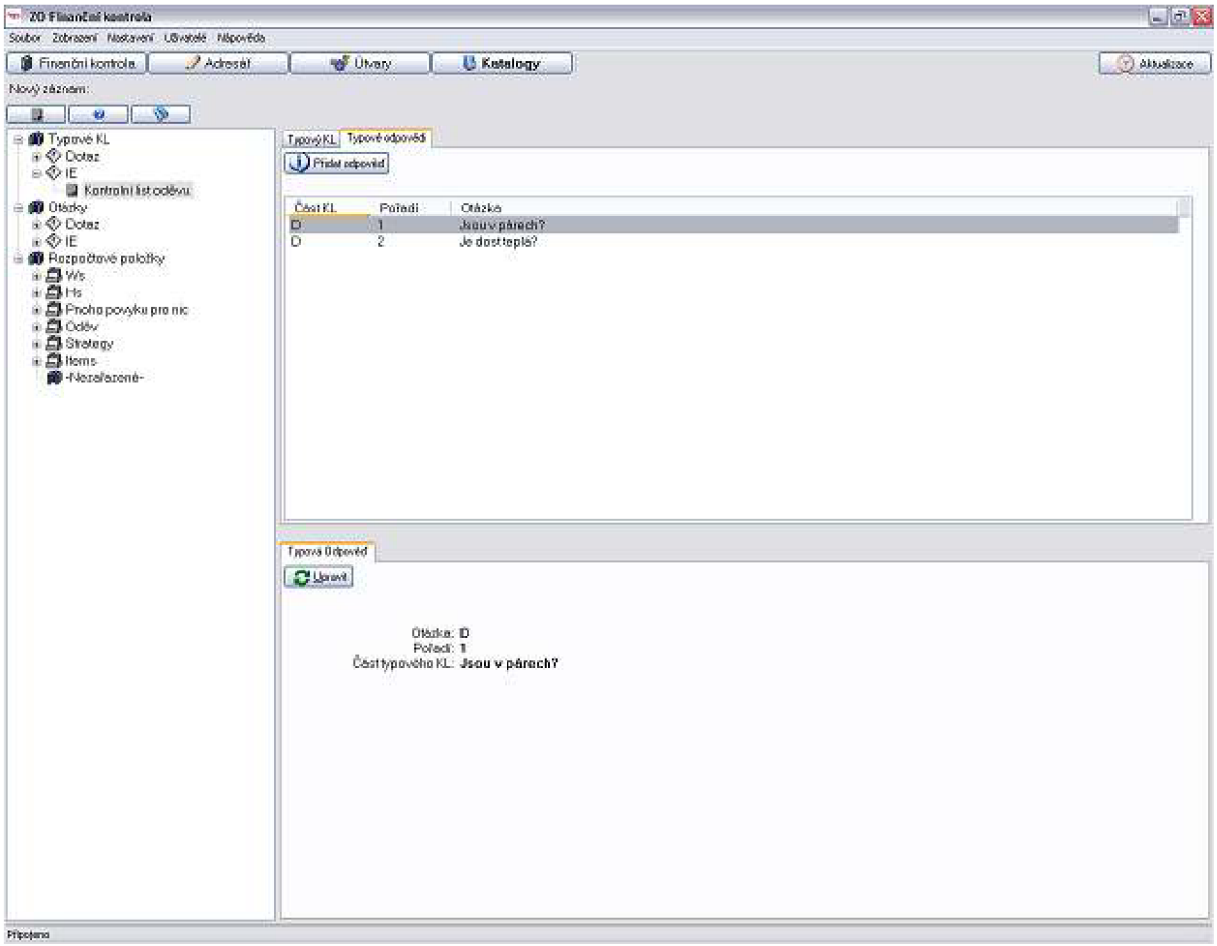
kontrolu se kterou se budou vázat. Kontrolní list je možné nahrát z katalogů a usnadnit si tak práci s výběrem otázek které se na kontrolní list přiřazují hned po jeho vytvoření.
Na kontrolní list se vážou skupiny otázek nad rozpočtovými položkami a odpovědí, kde má uživatel možnost přiřazování odpovědí k daným otázkám a jejich editaci.
Kontrolní list obsahuje informace o evidenci práce kontrolorů kteří na daném kontrolním listě pracují.
Na tomto místě je také seznam plateb které se ke kontrolovanému sektoru rozpočtových otázek vážou.
Stejně jako v u Finančních kontrol je zde možnost přiřazování dokumentů.

Adresář

Modul adresář slouží pro správu subjektů finančních kontrol a jejich správu. Subjekty jsou rozděleny podle hlediska **Fyzických a Právnických osob**. U každé takovéto položky jsou uchovávány informace o jméně, adrese apod. Právnická osoba má ještě k dispozici seznam kontaktů se kterými uživatel jedná a na které se bude odvolávat. Kontakt obsahuje informace o jméně a kontaktu(email, telefon, mobil).

Útvary

Modul útvary zpracovává informace o vnitřních záležitostech kontrolního oddělení. Je zde seznam **pracovníků** a informace o **útvarech** oddělení. K daným útvarům se přiřazují pracovníci a jim se přiřazuje funkce v útvaru. Ze seznamu pracovníků se dále vybírají uživatelé kteří mají přístup k systému.



Obr. 10.: Modul katalogy

Katalogy

Modul katalogy byl vytvořen pro práci s informacemi na které se IS odvolává především v modulu Finanční kontroly. V ovládacím okně spravuje: **typové kontrolní listy, otázky a rozpočtové položky.**
Typové kontrolní listy jsou kontrolní listy obsahující pouze informace důležité pro jejich identifikaci. Je to sektor rozpočtových položek a název. Dále je na typový kontrolní list navázaná skupina otázek s informací o jejich pořadí. Tyto otázky se vybírají podle kódu podpory typového kontrolního listu a řadí se podle části typového kontrolního listu do které patří.
Otázky je seznam otázek vázané na rozpočtové položky, na jejichž zodpovězení stačí ano/ne. Jsou třízené podle kódu podpory podle které jsou následně vybírány do kontrolních listů.
Rozpočtové položky je seznam všech rozpočtových položek se kterými se ve finančních kontrolách pracuje. Tyto mají daný patnácti místný kód a jsou řazeny hierarchicky podle kódu podpory a sektoru rozpočtových položek.

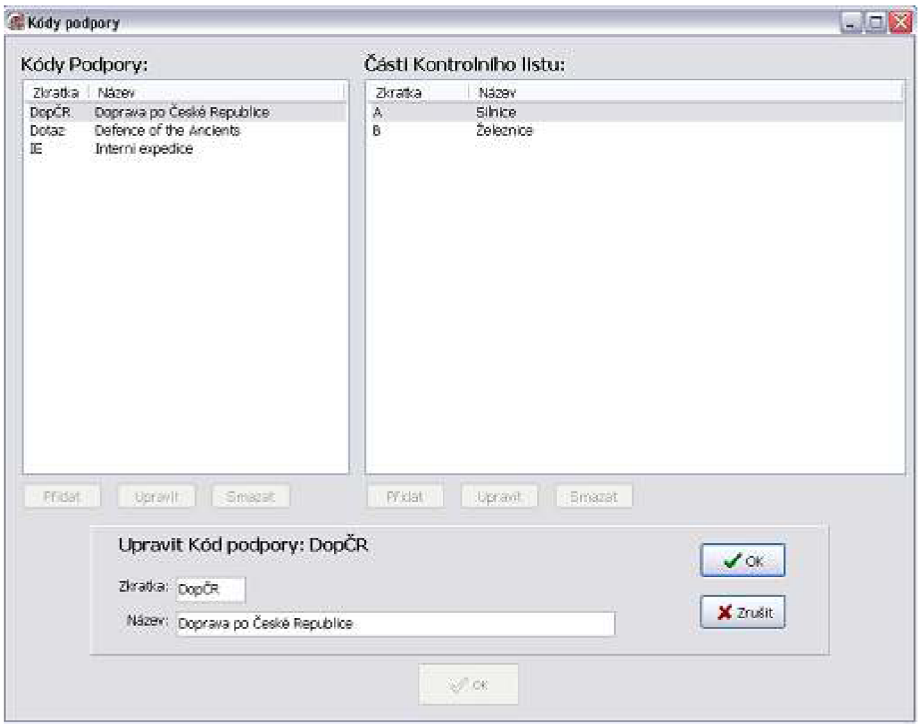
Práce se systémem

Přihlašování

Uživatel, jenž má právo administrátora přidělí pracovníkům v systému jejich přihlašovací údaje. Nikdo jiný nemá do systému možnost přístupu.

Kód podpory

Kód podpory stojí v hierarchii správy rozpočtových položek nejvýše. Určuje širokou skupinu položek se kterými je možné dále pracovat. Pro práci se systémem je nutné nejdříve zadat kódy podpory v okně pro jejich správu.



Obr. 11.: Okno pro správu kódu podpory

Na kód podpory se váže seznám částí kontrolního listu. Jejichž položky určují kolik a jaké části bude mít kontrolní list pod tímto kódem podpory.

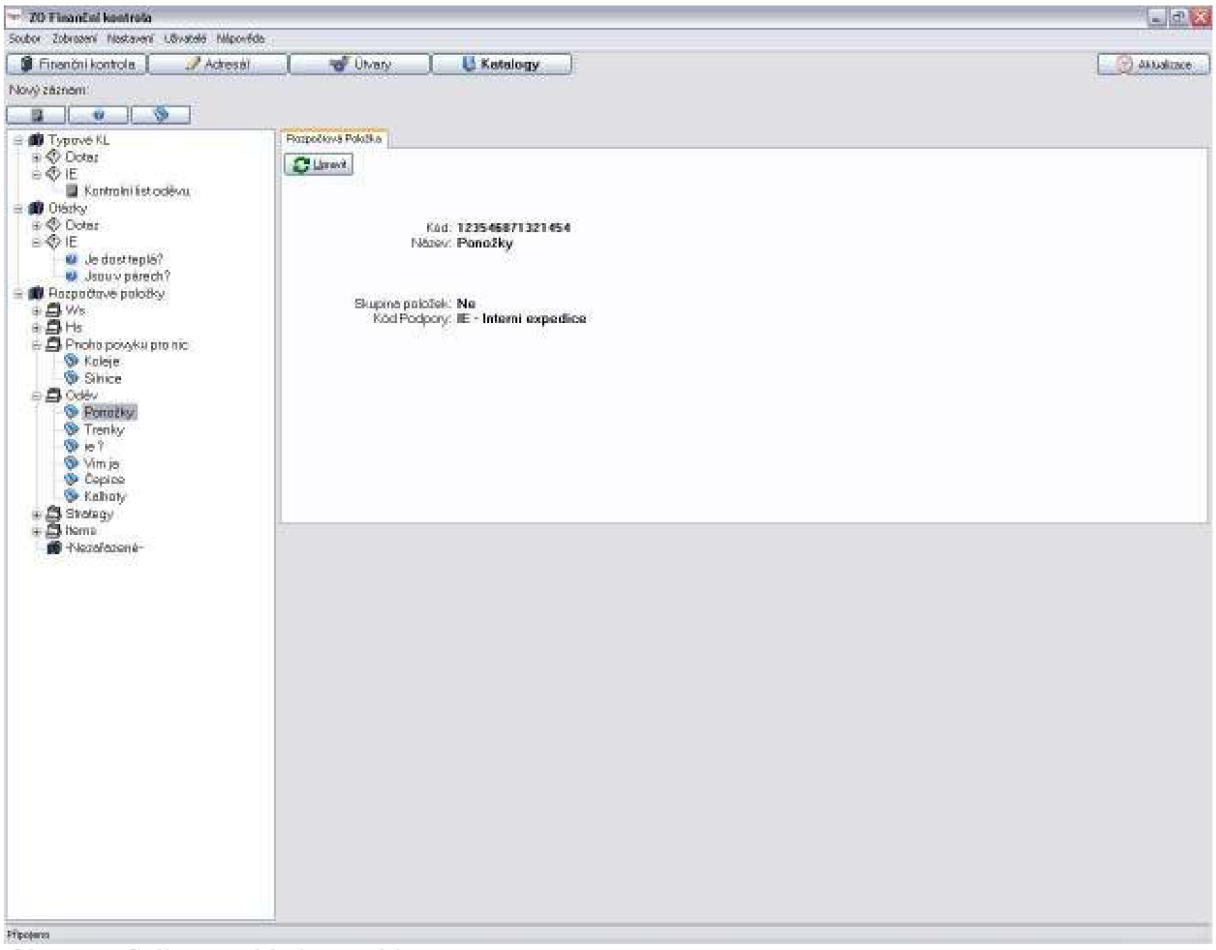
Číselníky

Okno pro správu číselníku zobrazuje informace pro dané rozšiřitelné číselníky a nabízí prostředky pro jejich správu. Je vhodné je pro další práci v systému naplnit.

Menu

Slouží pro přepínání mezi moduly. Je umístěné na komponentě toolbar. Aktuální modul je vyznačen tučným nápisem na tlačítku které modul reprezentuje.
Menu dále obsahuje tlačítko pro aktualizaci. Aktualizace se provádí automaticky každých 5 minut, pro okamžitou aktualizaci je možné použít toto tlačítko. Systém aktualizuje pouze v případě že některý z uživatelů provedl insert, update, nebo delete do databáze.

Uživatelské prvky na hlavním okně aplikace



Obr. 12.: Celkový pohled na aplikaci

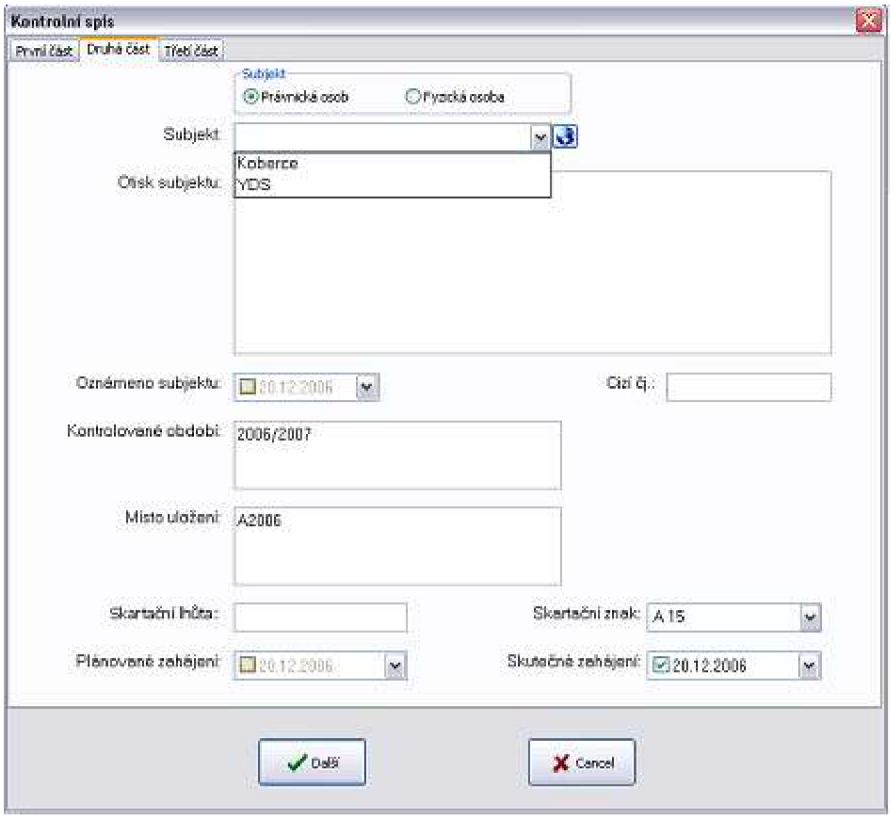
Strom

Ve stromě(TreeList) je uložen seznam hlavních prvků daného modulu. Do stromu je možné přidávat prvky pomocí ikonek umístěných přímo nad ním a označeným nový záznam.

Informační bloky

Pro daný prvek označený ve stromě je zobrazena komponenta PageControl s informacemi rozdělenými do TabSheetů podle druhu záznamu který je označen. Na prvním sheetu jsou vždy informace o daném záznamu a tlačítko pro úpravu vybraného záznamu na další jsou informace o prvcích které se na vybraný záznam vážou. Tyto jsou zobrazeny v komponentě ListView a na dané sheetu je tlačítko pro jejich přidávání. Informace k dané položce jenž není ve stromě je zobrazena na druhém PageControl umístěné v dolní části aplikace.

Nové položky a jejich editace



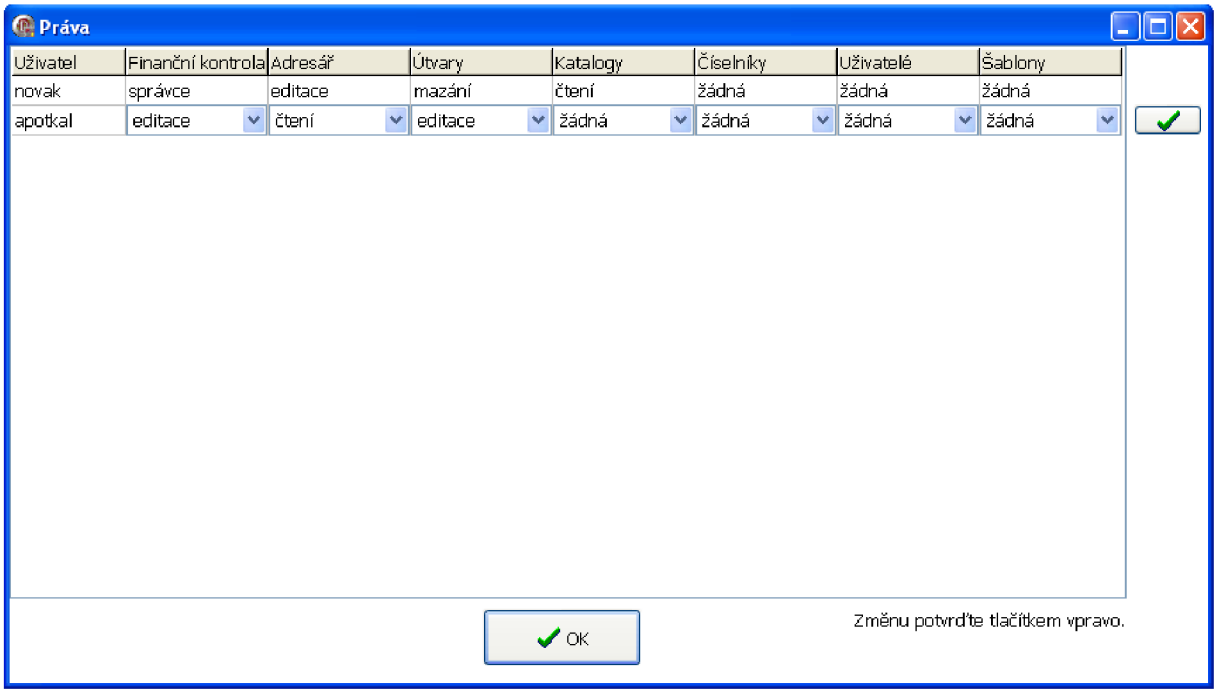
Obr. 13: Okno pro vytvoření nové Finanční kontroly

Nové položky se vytvářejí pomocí kliknutí na dané tlačítka a vyplnění požadovaných informací. Pro výběr z několika položek(číselníky, záznamy z katalogů) jsou použity needitovatelné comboboxy. Pro přímé zadávání záznamů z katalogů slouží speedbutton umístěný nalevo od komponenty pro výběr. Okna pro editaci a pro tvorbu jsou si velmi podobná, rozdíl spočívá v načtení informací do editovatelných položek z databáze v případě editu záznamu.

4.6. Práva

Jak již bylo řečeno v úvodu, požadavky na práva uživatelů nebyly tak rozsáhlé, aby se pro každý ovládací prvek definovala práva zvlášť, a tak jsme implementovali pro každý okruh funkcionality programu jeden číselník práv. Práva jsou tedy zapsána v jedné tabulce, kde jsou pro cizí klíč do tabulky Uživatel uvedena tomuto uživateli přidělená práva, rozdělená do sloupců podle kategorií. Program při úspěšném přihlášení tedy spustí funkci Nastav práva, v níž se nejdříve získají hodnoty práv jednotlivých kategorií a následně se postupně nastaví viditelnost jednotlivých ovládacích prvků.

Pro hodnotu 0 (žádné) platí zakrytí tlačítka pro celý modul, případně pro ovládací prvky v hlavním menu.
Pro hodnotu 1 (čtení) platí zakrytí všech tlačítek v celém modulu.
Pro hodnotu 2 (vytváření) platí povolení tlačítek pro vytváření objektů, ostatní tlačítka jsou zakryta.
Pro hodnotu 3 (úpravy) platí povolení tlačítek pro vytváření a úpravy objektů, ostatní tlačítka jsou zakryta.
Pro hodnotu 4 (mazání) platí zakrytí tlačítek pro speciální práce s objekty.
Pro hodnotu 5 (správce) platí plné využití systému.



Obr. 14: Okno pro editaci práv

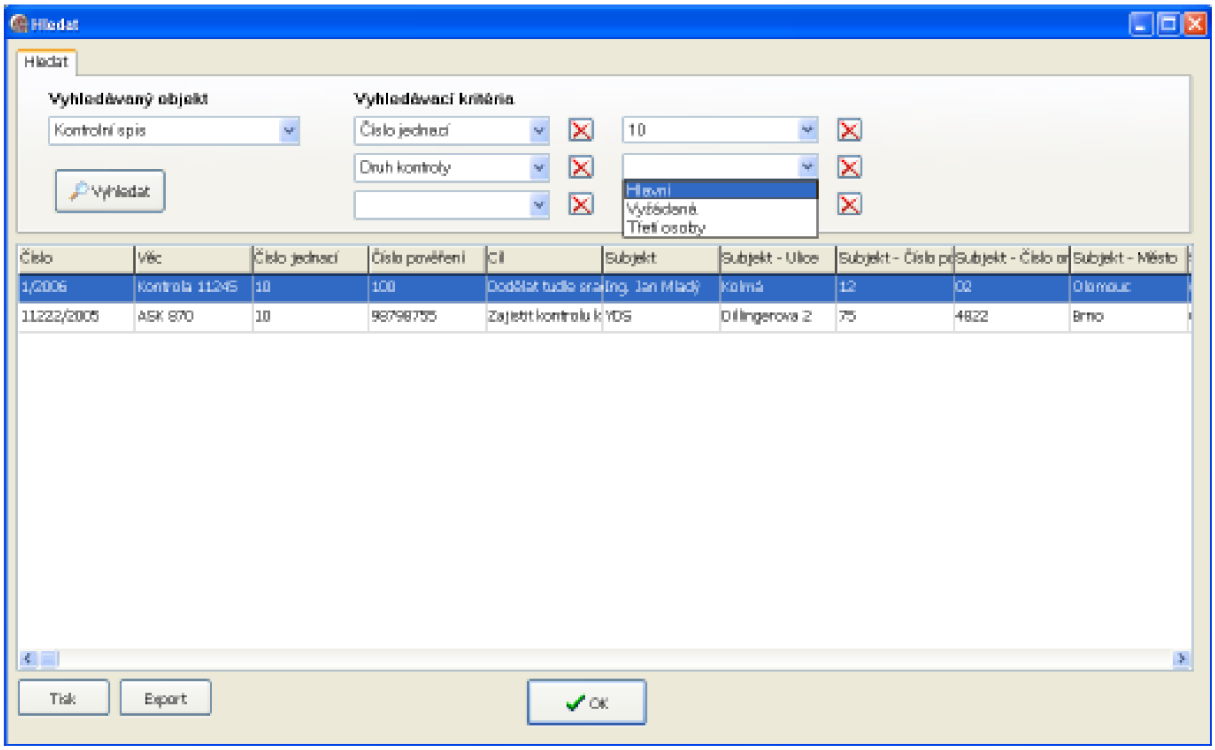
Práva jsou rozdělena do kategorií po modulech a dále pro správu číselníků, uživatelů a práv a importu a exportu. Pro kategorie správy je vždy jen 0 pro zakázáno a 1 pro povoleno.

Práva je také nutné určovat při načítání kontrolních spisů. Jde o jiné omezení, ale výsledek je v podstatě stejný – zakrytí nebo povolení jednotlivých ovládacích prvků. V tomto případě ještě s omezením k načteným právům.
Pro každý kontrolní spis platí, že pokud není uživatel uveden jako Kontrolor pro danou kontrolu (uživatel má cizí klíč pracovníka a stejně tak Kontrolor má odkaz na pracovníka), může ji nezávisle na

svých právech pouze číst. Pokud je v kontrole uveden jako Kontrolor, může s kontrolou pracovat podle svých oprávnění. Toto omezení neplatí pro správce modulu Finanční kontrola.
Programově je toto vyřešeno podmínkou: má uživatel správce modulu nebo je Kontrolorem v dané kontrole? Pokud ne, načítají se práva pro čtení, které přepíší práva aktuální. Pokud platí ano, nahrají se pro tento modul práva znovu.

4.7. Vyhledávání

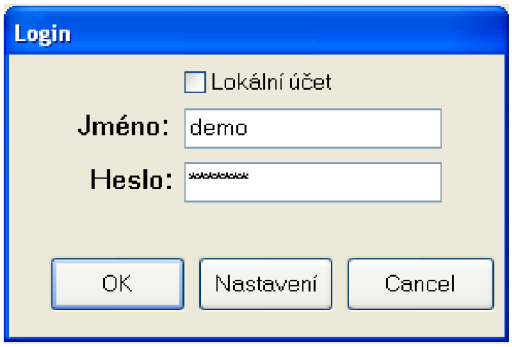
Abychom poskytli zákazníkovi možnost hledat jakýkoliv objekt nezávisle na jeho zařazení v hierarchii, vznikl nástroj pro vyhledávání. Okno pracuje na vzájemném propojení řady comboboxů. Prvním určujícím pravidlem je zde vyhledávaný objekt, ve své podstatě pouze seznam objektů, jež je možné vyhledávat. Po zvolení jednoho se načtou názvy atributů, podle kterých je možné tento objekt vyhledávat, do třech comboboxů v prvním sloupci, nazvaném Vyhledávací kritéria. Prvním z nich se také otevře pro uživatelův výběr.
Po výběru jednoho kritéria se automaticky otevře combobox druhý ve stejném sloupci pro zadání dalšího kritéria, a combobox v pravém sloupci. V této chvíli se do tohoto ovládacího prvku také načte celý seznam všech hodnot obsažený v databázi. Pokud uživatel píše svou vyhledávanou hodnotu, tento seznam se mu sám omezuje.
Vyhledávací kritéria jsou k dispozici maximálně 3, což stačí pro dostatečnou volnost uživatele.
Při samotném vyhledávání jsme použili v celé aplikaci jediný přímý select z databáze, prvním parametrem specifikovaný ve funkci DoCommand jako SELRAW. Po něm následuje příkaz SQL.
Vrácené výsledky se zobrazují v tabulce v dolní části okna. Tento seznam je možné jednoduše exportovat do .csv anebo tisknout připravenou tiskovou sestavou.



Obr. 15: Okno pro vyhledávání

4.8. Offline klient

Práce na offline klientovi začaly prakticky ihned po dokončení hlavní části programu. Bylo nutné tedy nalézt funkční model, při jehož implementaci by se nesahalo do již hotové části kódu a uživatelé by nemuseli pro práci s programem instalovat nic víc než samotnou aplikaci. Tímto řešením se nakonec stala komunikace mezi Klientem a databází Firebird. Po nalezení komponent pro komunikaci s tímto nástrojem jsme začali databázi z Oraclu převádět do Firebirdu, resp. do Delphi. Klient totiž musí být schopný sám vytvořit při prvním startu celou strukturu tabulek a pohledů. Z uložených procedur Oraclu rozložených do balíčků se tyto procedury staly funkcemi Firebirdu.
Rozlišení zda se klient bude připojovat k databázi, nebo zda se bude spouštět offline, je ponechané na uživateli aplikace. Tento výběr se provádí v přihlašovacím okně.



Obr. 16: Přihlášení do systému

Pro sdílení informací byl vytvořen systém synchronizace, který využíval zaznamenávání informací o čase změny záznamu v databázi. Následně se tato informace při připojení k aplikačnímu serveru srovnávala se změnou daného ID na hlavní databázi. Pokud byl čas novější, informace se aktualizovaly.
Tento systém ale nakonec zákazníkovi nevyhovoval z důvodu rozložení práce kontrolorů, kteří se k serveru ani připojovat nemohli. Musel se tedy vymyslet jiný systém předávání informací.
Posílání dat bylo tedy vyřešeno pomocí XML souborů. Zákazník chtěl pro svou práci posílat pouze objekty Oddíl, Kontrolní list a Kontrolní spis. Na tyto objekty je navázaná však řada dalších objektů a tak jsou XML soubory poměrně rozsáhlé.
K těmto souborům je však také třeba přidat dokumenty vázané na Kontrolní listy a Kontrolní spisy a také celý tento objem dat zašifrovat. Celou tuto problematiku řeší funkce, která z adresáře \temp\Export, kam se všechny dokumenty a XML soubor uloží, všechny tyto soubory uloží do archivu, jenž následně zašifruje 16 bitovou šifrou.
Po vytvoření tohoto modelu se však začaly objevovat problémy s ním spjaté. Každá úprava databáze musí být přepsána do offline klienta a ne všechny stavební prvky se takto dají přepsat. Pro další vývoj tedy bude nutné vytvářet záznamy pro každou změnu hlavní databáze a následně rozhodovat při úpravách offline klientů, zda jsou tyto změny proveditelné a také potřebné.

5. Závěr

Na prací, která trvala šest měsíců a v níž byli zapojeni čtyři lidé, jsem osobně získal velké množství zkušeností ve vývoji rozsáhlého informačního systém. Je jasné, že úspěšný vývoj závisí na spolehlivé komunikaci týmu, na plnění zvolených vývojových částí v určeném termínu a na přehledné dokumentaci a komentářích ke všem důležitým procesům. Problémy, do kterých jsme se při vývoji dostali, vyplývali často z těchto tří příčin. Dalším určujícím prvkem ve vývoji softwaru je čas strávený nad analýzou a návrhem řešení, této části jsme věnovali týden a to ještě s ohledem na to, že část analýzy už byla zpracovaná. Pokud bychom tuto část zanedbali, přidělali bychom si mnohem více práce v konečné implementaci. Následky tohoto nám přesto byly známé, protože jsme v analýze nebyli schopni postihnout další, zatím neznámé požadavky zákazníka. Při tvorbě offline klienta jsme tak například museli zasahovat do již hotových klíčových částí kódu a následně opravovat všechny návazné funkce a též funkce, které vykazovaly chyby.
Zanedbanou částí vývoje je ovšem testování. Ideální stav by byl tehdy, kdybychom měli za celý vývoj alespoň jednoho pracovníka, který by se zabýval testováním hotových částí aplikace a nezasahoval do kódu. Toho jsme však neměli a tak jsem testování prováděl ke konci vývoje osobně já a měsíc testování pravděpodobně neodhalil úplně všechny chyby systému.
I z důvodu krátkého testování je zřejmé, že nás čeká celá řada oprav a úprav systému, aby konečně padl zákazníkovy „na míru“, jak bylo zamýšleno. Zkušební doba softwaru byla u zákazníka prodloužena na dva měsíce, čímž získáme dostatečný prostor k doladění systému.
Úpravy systému bych však rád vedl ještě dál a využil navržené modularity k vytvoření modulů, nezabývajících se pouze finanční kontrolou. Prvním krokem tedy bude vytvořené funkční kostry bez modulů, které ve vývoji ještě vykazovaly jisté chyby. Následné psaní modulů bych pak rád unifikoval, aby bylo dosaženo flexibility v poskytovaných řešeních.
Systém má zabudovanou řadu možností, které zatím nejsou využity. Jednou z nich je možnost vytvořit monitorovací aplikaci, sledující činnosti všech uživatelů nad danými objekty. Uchovávány jsou informace o vytvoření a změně dokumentu, identifikující jak čas, tak i uživatele. Tyto informace by se následně mohly použít k vytváření automatických přehledů o vytíženosti jednotlivých kontrolorů a případně i o plnění jednotlivých termínů v průběhu jednotlivých finančních kontrol.
Problematika softwaru na míru je a bude určitě delší dobu aktuální z důvodu nemožnosti vytvoření standardu. Zákazník se totiž nechce učit věcem, které mu nabídnete. Často má vlastní představu o systému, popřípadě vaše řešení nepokrývá ani nepatrnou část jeho potřeb. Není tedy jiné zbytí, než vyjít zákazníkovi maximálně vstříc. Dalším faktorem, bránícím standardizaci, jsou také odlišné požadavky napříč státy, vycházející z různých zákonů. I když tyto problémy nenastanou, je zde také třeba jazykové konverze, která se nemusí vyplatit distributorovi systému či zákazníkovi, který by platil navýšenou cenu. Myslím si však, že tato situace může vývoji softwaru jedině prospět, protože s každou novou firmou je tu další škála potřeb nutných k zamyšlení.

6. Použitá 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