Jednosouborový databázový klient. Databáze typu klient-server.

Single file database client.
Client-server database

Eduard Zeman

Bakalářská práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Eduard ZEMAN**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Jednosouborový databázový klient. Databáze typu klient-server.**

Zásady pro vypracování:

1. **Důvody pro realizaci jednosouborových aplikací.**
2. **Výběr vývojového prostředí, databáze a jejich obecný popis.**
3. **Implementace databázového stroje MySql.**
4. **Návrh a jednoduchá programová realizace ukázkového klienta, bez lokálních konfiguračních direktiv.**
5. **Spojení klient-server realizované zásuvným modulem MyDac.**
6. **Ukázka konfigurace klienta.**
7. **Bezpečnost komunikace klient-server.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **SWAN, Tom. Mistrovství v delphi 4 : Kompletní průvodce pro tvorbu aplikací. Pavel Machek, David Hanousek, Luděk Hořčička. 1. vyd. Praha : Computer Press, 1999. 830 s. Programování. ISBN 80-7226-173-8.**
2. **KOFLER, Michael . Mistrovství v MySQL 5 : Kompletní průvodce webového vývojáře. 1. vyd. Brno : Computer press, 2007. 808 s. ISBN 978-80-251-1502-2.**
3. **GILMORE, W. Jason. VELKÁ KNIHA PHP a MYSQL 5 : kompendium znalostí pro začátečníky i profesionály. 2. přeprac. vyd. [s.l.] : Zoner Press, 2005. 864s. 323-331. ISBN 80-86815-53-6.**
4. **SVOBODA Luděk, et al. 1001 tipů a triků pro Delphi : 2. aktualizované vydání. Brno : Computer Press, 2003. 546 s. ISBN 80-7226-488-5.**

Vedoucí bakalářské práce: **RNDr. Ing. Miloš Krčmář**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ Fakulta aplikované informatiky

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

V současné době existuje celá řada technik, jak se připojovat na databáze informačních systémů. Obsahem této bakalářské práce je pojednání o jedné z možnosti napojení na databázi. Teoretizuje se zde problematika vybalancování aplikačního výkonu mezi klientskou částí a serverem. Uvádí výhody a nevýhody aplikací závislých na typu operačního systému. V praktické části je pak vytvořen klient, který je naprogramován jako jediný soubor. Nezasahuje do souborového systému, ze kterého je startován. Ukazuje elegantní možnost napojení na databázi a rovněž její využití pro uložení konfiguračních direktiv.

Klíčová slova: klient, databáze, připojení, klient-server

ABSTRACT

Nowadays there have been all kinds of ways how to connect with an information system database. Content of this thesis is an elaboration of one of the ways of connecting with a database. Questions about fiding balance of application achievments between clients and a server has been theoretically treated. Application advantages and disadvantages depended on a type of an operating system have been mentioned. In a practical part a client was created who is programmed as the only file. He does not intervene in a file system from whom he had been started. He illustrates an ellegant possibility of database connecting as well as its usage for saving of configuration directives.

Key words: client, database, connecting, client-server

Motto: „Maličkosti tvoří dokonalost, ale dokonalost není maličkostí.“

Chtěl bych poděkovat své rodině, za jejich trpělivost a podporu během mého studia.

Dále bych chtěl poděkovat panu RNDr. Ing. Miloši Krčmářovi, za jeho pomoc, cenné rady a připomínky při tvorbě této bakalářské práce.

Prohlašuji, že

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

Prohlašuji,

že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

V současnosti se neustále zvyšuje nárok na výkon a efektivitu jedince. Zasaženy jsou prakticky všechny oblasti života. Lidé se musejí přizpůsobovat stále novým technologiím, které ani zdaleka nejsou pro člověka uzpůsobené. Denně se potýkáme s množstvím informací, které musíme nějakým způsobem zpracovat. Tyto informace negeneruje pouze samo prostředí, ve kterém se pohybujeme a pracujeme, ale také nový fenomén posledních let, celosvětová síť Internet. V této síti je obrovské množství informací. Tyto informace jsou uloženy na serverech a poskytovány uživatelům. Uživatelé se pomocí sítě Internet a obslužných programů připojují na takovéto servery a získávají z nich potřebné informace. Internet je veřejná síť a existuje celá řada serverů, které nám umožní založit vlastní databázi. Tuto databázi pak poskytnout dalším uživatelům, kteří se na ni připojí. Pro připojení k databázím používáme programy, které umí tato data vhodně prezentovat. Prezentační programy se s databázemi spojují různými způsoby. Kooperace obslužného programu, databáze a jejich vzájemné spojení, určují úspěšnost celého projektu. Umožňují předkládat a sdílet data s obrovským množstvím potencionálních zákazníků. V následujícím textu bude popsána jedna z možností, jak vytvořit databázi, klienta a jak je vzájemně propojit.

Cílem této bakalářské práce je naprogramovat obslužný program databáze. Program musí splňovat následující podmínky:

- Jediný přímo spustitelný soubor bez nutnosti instalace.
- Napojení na vzdálenou databázi bez použití externích souborů.
- Uložení konfigurací do databáze.

I. TEORETICKÁ ČÁST

1 DŮVODY PRO REALIZACI JEDNOSOUBOROVÝCH APLIKACÍ

Výpočetní technika se rozvíjí rychlým tempem. Není to tak dávno, co měly domácí počítače kapacitu pevných disků v řádech jednotek MB a paměti **RAM** pouhých několik KB. Ohromný rozvoj v této oblasti způsobil, že drtivá většina softwarových aplikací nevyužívá plně možností dnešního hardware. Velikosti **HDD**, **RAM**, výkonů **CPU** a **GPU** příliš nemotivují programátory optimalizovat své aplikace. Výkon hardware tyto aplikace prostě podrží. Konkurenční tlak způsobuje, že aplikace nejsou od základu přepisovány, ale pouze opravovány a doplňovány. S masovým nástupem víceprocesorových systémů, na kterých stále běží sériové programy, se propast mezi hardwarem a softwarem ještě prohloubila. Na konci toho všeho stojí domácí počítač a domácí uživatel, který požaduje, aby počítač fungoval jednoduše a elegantně. Pokud možno bezpečně a bezchybně. Představa běžného uživatele je zcela jasná. Počítač má být jednoduchý jako mixér. Zapnu, kliknu a musí to fungovat. Pokud ne, tak je to špatný počítač. A mají jednoznačně pravdu. Na pevných discích současných domácích počítačů je veliké množství souborů. Většina z nich nebyla nikdy použita. S největší pravděpodobností ani nikdy nebude. Zaměřme se na osobní počítače a uživatele, kteří nemají o principech fungování hardware ani software nejmenší ponětí. Používají počítač k práci či zábavě. Takových uživatelů je drtivá většina. Většinou požadují, aby program jediným kliknutím spustili. Totéž požadují při odstranění programu. Nějaké instalační a odinstalační procedury je naprosto nezajímají. Na takovém počítači jsou desítky a stovky tisíc souborů, které nemají již žádné použití. Výkon a stabilita počítače klesá, fragmentace disku je značná. Nastane doba, kdy je nutný servisní zásah. Pokud je počítač připojen k Internetu a není účinně chráněn, je jeho softwarová životnost minimální. Dnes není problém počítač zabezpečit nejen proti virům a podobnému škodlivému software, ale i proti neznalosti samotného uživatele. Od toho jsou přístupová práva. Zisk však převažuje nad bezpečností. Stále se prodávají s novými počítači zkušební verze bezpečnostních systému. Jsou to programy, které po skončení expirační doby zastaví svoji činnost a oznámí uživateli, aby s tím cosi udělal. Oznámí to nezkušenému uživateli, který nemá nejmenší tušení, co se po něm chce. Zejména pokud aplikace komunikuje v cizím jazyce. Toto není výsměch nezkušeným uživatelům. V žádném případě ne! Kolik z nás po zvednutí kapoty svého auta dokáže provést víc operací, než doplnit kapalinu do ostřikovačů. Přiznejme si, kolik z nás dokáže vyměnit tak

banální věc jako je olej? A co teprve, když dostaneme vůz, který není náš. To se po otevření kapoty nestačíme divit. Pracovní plocha je najednou jiná. Najednou jsme nezkušení také my. Jenom v jiné oblasti. Výrobci automobilů to řeší naprosto elegantně. Nic víc, než doplnit ostřikovače a možná ještě dolít chladící kapalinu a dost! Odšroubujeme něco, co nemáme a je po záruce. Pokud by něco podobného bylo i v oblasti PC, tak by se zejména síti Internet značně ulevilo. Vraťme se ale k problému, který musí řešit běžný uživatel. Pro něho je například instalace hry téměř neproveditelná. Software (hra) se skládá z ohromného množství souborů, vyžaduje složité instalační procedury. PC se dostane do situace, kdy je na něm více než 500.000 souborů. A to hovoříme o běžném domácím počítači. Kolaps na sebe nenechá dlouho čekat. Zejména pokud uživatelé odstraňují robustní aplikace pouhým smazáním z disku. Pomineme-li tedy bezpečnost, zůstávají nám samotné aplikace a problémy s jejich instalací, spouštěním, konfigurací a následným odstraněním ze systému, pokud se nám přestanou líbit. Naštěstí již existují aplikace, které není nutné instalovat. Lze je spouštět přímo z **CD** a jiných read-only médií. Nebo aplikace stačí zkopírovat na disk a spustit. Pokud se nám přestanou líbit, tak je buď smažeme, nebo je necháme tam kde jsou. V prvním případě uvolníme místo na disku. V druhém ne. V obou případech ale nezpůsobíme systému žádný problém. Snad pouze to místo na disku nám může časem chybět a zpomalovat chod počítače, protože souborů bude na disku již mnoho a budou značně fragmentované. Skutečnost, že obsluha počítačů je nesmírně složitá a zcela se vymyká běžnému uživateli, je zřejmá. Toto si uvědomují už i výrobci OS a programů. Stále více se objevují aplikace, které migrují na server a uživatel si je doma spouští po Internetu. Například dokumenty se stěhují na servery, Google documents, Facebook atd. Budoucnost je jasná. Všechny programy i data na serverech a domácí počítače se omezí pouze na prezentaci těchto dat. Taková data jsou přístupná z kteréhokoli místa na světě a neomezují se pouze na domácí počítač. Rovněž sdílení těchto dat s ostatními členy rodiny i přáteli je příjemná záležitost. Pro běžného domácího uživatele je to vcelku ideální cesta. Nepoužívat složitý počítač, ale třeba televizor připojený na síť. Tam v podstatě není co pokazit. Snad pouze staré dobré kladivo při prohraném fotbalovém zápase by mohlo naši TV ohrozit. Nicméně manželky si svoje oblíbené telenovely jistě ubrání. Důvody pro používání aplikací, které nevyžadují žádné instalace, je zřejmá. Usnadnit jejich používání. V případě jednosouborových aplikací je situace ještě jednodušší. Takové aplikace není nutné ani přenášet na disk. Prostě se spustí. Ať jsou kdekoliv. Nic snadnějšího již realizovat nelze.

2 KLIENTI A SERVERY

Informace, kterých si ceníme, máme tendenci archivovat. Například rodinné fotografie. Uložíme je, či vytiskneme, aniž bychom měli snahu je nějak katalogizovat, či jinak detailně třídit. Není to potřeba. Použijeme k tomu vhodně pojmenovaný adresář či adresáře na záložním médiu a fotografie tam nakopírujeme. Nebo je vytiskneme a vložíme do pojmenovaného alba. Ke každé fotografii si uděláme poznámku a uložíme na bezpečné místo do šuplíku. Na druhé straně máme data, která také potřebujeme archivovat, ale navíc bychom v nich chtěli rychle vyhledávat konkrétní informace. Například mezi desetitisíci skladbami MP3 hledáme tu, která obsahuje v názvu skladby slovo „love“ a momentálně ji nemáme v počítači, ale na jednom z 250 CD. Docela těžká práce. Nyní nastává čas pro zavedení pojmu databáze. Pokračujeme stále s naším archivem **MP3** souborů. Jedna z možností, jak vytvořit databázi, je ruční zápis podrobných informací do sešitu. Seřazené podle interpreta apod. V tomto sešitě budeme mít i seznamy jednotlivých skladeb. Tedy jejich konkrétní názvy a co je na kterém CD. V podstatě nejjednodušší řešení, které je používáno již stovky let. Každý si jistě pamatuje na kartotéky u lékaře, v knihovně či videopůjčovně. To není nic jiného než databáze. Takovéto psané databáze nejsou vhodné pro vyhledávání informací, se kterými pro vyhledávání nebylo počítáno. Také nejsou vhodné pro jakékoli statistiky. Opět použijeme analogii s naší ručně psanou databází seznamu CD. Položme si otázku. Jak najít skladbu, která má v názvu písně slovo „love“. Naše ručně psaná databáze je seřazována abecedně pouze podle interpretů a názvů alb. Z tohoto problému vyplývá i nevýhoda ručně psaných databází. Tj. jejich ruční a velmi zdlouhavé prohledávání. Naše databáze CD je tragicky malinká ve srovnání s databází knihovny či nemocnice. Představíme-li si náročnost těchto vyhledávacích operací, je až s podivem, že existovaly podrobné statistiky porodnosti, výskytu chorob, škůdců, úrody, počasí atd. Mravenčí práce lidí, kteří toto dělali, je obdivuhodná. Neexistovala výpočetní technika a jinak to prostě nešlo. S nástupem výpočetní techniky došlo k odstranění těchto problémů s vyhledáváním a posléze i s prováděním analýz uložených dat. Začaly vznikat první digitální datové sklady. Říkejme jim databázové stroje či databáze. Jejich úkolem je udržovat data a poskytovat je těm subjektům, které o ně požádají. A to pokud možno vhodně seřazená či jinak vytříděná. Těmto subjektům říkejme klienti. Vezmeme náš ručně psaný sešit a opíšeme všechny informace do databáze. Jak se to dělá, není nyní podstatné. Jak už víme, tak databáze může s daty manipulovat. Dokáže řadit záznamy jednou podle

interpreta, jindy zase podle alba. Dokáže také prohledávat názvy skladeb. A to je to, co se nám hodí. Řekneme databázi, aby nám vypsala všechny skladby, které obsahují slovo „love“. Databáze zpracuje naši žádost a vypíše např. 26 písniček, které jsou uloženy na CD 25, 38, 156, 159. To nám ohromně usnadnilo práci. Z 250 CD nám stačí projít 4. Nyní už je zřejmé, co dělá databázový stroj. Klient zprostředkuje uživateli prostředí, ve kterém uživatel zadá svůj požadavek. Klient požadavek předá databázi a ta ho zpracuje. Výsledek pak databáze předá klientovi a klient zprostředkuje výsledky uživateli. Klient i databáze mohou být na jednom počítači nebo mohou být od sebe vzdáleni. Klient musí vždy navázat spojení s databází. Naopak databáze čeká (naslouchá), jestli po ní nějaký klient něco nepožaduje. Klient tedy s databází komunikuje. Musí být s databází propojen pomocí sítě. Databáze je navíc většinou umístěna na počítači, kterému říkáme server. Existuje několik technologií, jak spojit klienta s databázi. Asi nejznámější topologiií je **klient-server** a nejpopulárnější zase **peer-to-peer** Nás bude především zajímat architektura klient-server.

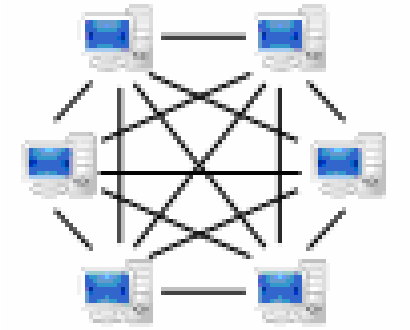
2.1 Síťové architektury.

V této kapitole se budeme zabývat dvěma způsoby propojení klientů s databázemi.

2.1.1 Architektura peer-to-peer

Tato architektura nezná pojem server (Obr.1). Všechny uzly takové sítě jsou si zcela rovny. Každý počítač v této síti může komunikovat se všemi ostatními. V podstatě je každý počítač klientem i serverem zároveň. Neexistuje zde specializovaný počítač, který by byl pouze serverem. Takové sítě mají jednu obrovskou výhodu. Čím více počítačů v síti je, tím stoupá její výkon. Vezměme např. jediný soubor o velikosti 1 GB. Tento soubor uložme na jeden počítač v P2P. Na začátku bude mnoho počítačů stahovat data z jediného počítače. Tento počítač má totiž omezenou kapacitu přenosového kanálu. Po chvíli už i na ostatních počítačích bude alespoň část tohoto souboru stažená. Další počítače už nemusí stahovat tento soubor pouze z prvotního počítače. Ale mohou stahovat ze všech ostatních. Mohou tedy používat více kanálů. Čím více počítačů, a alespoň fragmentů souborů je v P2P síti, tím je její výkon větší. Musí však existovat metoda, jak daný soubor správně rozporcovat tak, abychom mohli stahovat kousek z jednoho počítače a zároveň jiný kousek z jiného počítače. P2P síť naráží na problém s veřejnými a neveřejnými adresami cílových počítačů. Pokud mají cílové počítače neveřejné adresy, tak se navzájem nemohou běžným

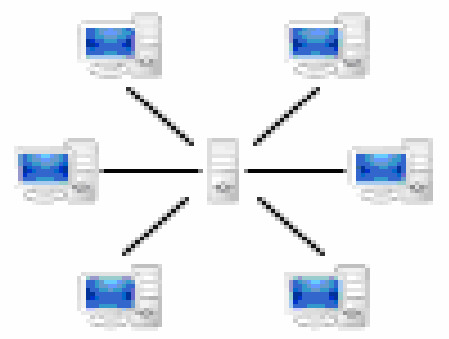
způsobem spojit. Proto se i v těchto sítích zavádí pojem server. Vložíme do takovéto sítě jeden počítač, který nám umožní navázat spojení i mezi neveřejnými adresami a tím seznámit navzájem klienty. Příkladem P2P sítě je např. GNUTELA.



Obr.1. P2P - schéma sítě

2.1.2 Architektura klient-server

Klient-server je síťová architektura, která striktně odděluje klienta a server, kteří spolu komunikují přes počítačovou síť (Obr.2). Je třeba si uvědomit, že tato architektura popisuje vztah mezi dvěma počítačovými programy. První program, kterému říkáme klient, žádá o nějakou službu druhý počítačový program, kterému říkáme server. Typickým příkladem je webový prohlížeč. Např. Mozilla je klient, který žádá od webového serveru, aby mu poslala webovou stránku. Takový klient může požadovat data od jakéhokoliv webového serveru na světě. Model klient-server je nejpoužívanější architekturou vůbec. Jsou na ní založeny aplikace, jako např. internetové obchody, emaily, protokoly http, pop3 apod. Celá architektura funguje tak, že máme jeden server, kterého se na něco ptá mnoho klientů. Např. pokud uložíme jeden soubor o velikosti 1GB na jeden server architektury klient-server, tak ostatní klienti si tento soubor začnou stahovat pouze z tohoto serveru. Čím více klientů, tím větší vytížení přenosového kanálu a tím také klesá výkon na jednotlivých klientech. To je zásadní rozdíl oproti architektuře peer-to-peer. Zvýšit výkon můžeme např. tak, že budeme obsah serveru dělit na víc serverů, které se navenek budou chovat jako server jediný. Výhodou této architektury je právě jediný zdroj právě aktuálních dat. Klienti tedy mají k dispozici vždy aktualizovaná data v reálném čase.



Obr.2. Klient-server schéma sítě.

2.1.3 Aplikace klient-server

Pojem aplikace klient-server znamená, že aplikace obsahuje jak klienta, tak server. Nemusí se tedy nikde vzdáleně napojovat. Ideálním příkladem jsou programy pro domácí účetnictví. Nemusí se jednat o malé databáze, ale v podstatě data, která jsou v nich uložena, není třeba sdílet s jinými uživateli v reálném čase. Nás budou zajímat aplikace, kdy je klient a databáze zvlášť a je mezi nimi jistá spojovací vrstva. V podstatě máme dvě možnosti.

1. Klient a databáze jsou na stejném počítači.
2. Klient a databáze jsou na různých počítačích.

První případ, kdy je klient na stejném počítači, nás nebude zajímat. Situace je prakticky stejná, jako u zapouzdřené databáze, s tím rozdílem, že takovýto počítač je i serverem. Mohou se na něho připojovat vzdálení klienti. Typickým příkladem jsou samotné servery. Administrátor občas potřebuje pracovat s databází pomocí klienta přímo na serveru. Zajímavější je druhá varianta. Typicky internetový prohlížeč je klient a jakákoli webová stránka, na kterou se pomocí něho díváme, je poskytnuta webovým serverem. Tady nastává více variant kooperace klienta a serveru. Je třeba si uvědomit, že klient je samostatný program, který umí zobrazovat speciální data, která mu poskytne server. Avšak pouze speciální data, na jejichž interpretaci je naprogramován. Např. webový prohlížeč se na naši domácí databázi CD podívat nedokáže. Naopak náš specializovaný klient pro prohlížení CD si neporadí s webovými stránkami. Ještě než přejdeme k hlavní části této bakalářské práce, je třeba zdůraznit, že budeme považovat databázi za součást serveru. Náš klient přímo nepřistupuje na databázový stroj, ale na server. Na serveru běží jiný klient

(v podstatě se mu říká právě server), který nás s databází spojí. My ale budeme považovat server jako celek. Co se na něm detailně děje, nás nemusí zajímat.

2.2 Výkon klient-server

Server přijímá požadavky od klienta. Ty zpracuje a pošle zpět klientovi. Záleží, jak dalece server ve zpracování požadavků zajde. Jde o to, kolik práce udělá klient a kolik server. První možností je, že klient nebude dělat nic. Pouze zobrazí výsledky – data a grafiku aplikace. Veškerá práce padá na server. Říkáme, že skriptování provádí server. Typickým příkladem může být **PHP** [1]. V takovém případě server generuje naprosto vše. Formuláře, tlačítka a ty naplní příslušnými daty. Celý formulář poté zakóduje do **HTML** jazyka a pošle výsledek klientovi. Pokud by takový server obsluhoval mnoho klientů, mohl by se dostat do problémů. Abychom takto zatížený server odlehčili, přesuneme část práce na klienta. Tedy na klientský počítač. Proč by měl server počítat např. matematické výpočty, když to může udělat klient. Část aplikace přeneseme na klienta. Webový prohlížeč nedostane čisté HTML. HTML bude obsahovat i kusy programového kódu, které musí vykonat klient sám. Například pošlu na server dotaz. Kolik zákazníků si koupilo zelenou tužku a kolik jsme za to utržili peněz? Server pošle klientovi pouze surová data a klient si celkovou sumu spočítá sám (sečte prodané zelené tužky). Nyní již skriptování provádí klient. Můžeme zajít ještě dál. Můžeme si napsat vlastního klienta. Takový klient může požadovat po databázi pouze surová data a veškeré výpočty, třídění apod. si provede sám. Tímto způsobem velice odlehčíme serveru. Na druhé straně nám server bude posílat kompletní sady dat. Tedy i ty, které zrovna nepotřebujeme. Tím zase zatěžujeme přenosový kanál. Kompromisem je ponechat na serveru pouze základní rozhodování, o zbytek se potom postará náš klient. Klienta můžeme naprogramovat tak, že bude nebo nebude závislý na **OS**. Např. Java aplikace jsou multiplatformní. Jejich zdrojové kódy lze spustit s vhodným kompilátorem na libovolném OS. Na druhé straně jsou klienti přímo naprogramovaní jako spustitelné exe soubory(aplikace). Ti jsou pak závislí na OS nebo dokonce na verzi OS. Která z cest je lepší, je pouze věc názoru. Podívejme se na tuto problematiku pozorněji.

2.2.1 Specializovaný klient

Jedna z možností je použít nějaký programovací jazyk. Například Pascal, VisualBasic, C atd. Naprogramujeme aplikaci a zkompilujeme ji tak, aby šla spouštět například na operačním systému Windows 2000 a vyšším. Kompilaci lze provést tak, že do výsledného souboru exe přibalíme i všechny potřebné knihovny, sestavy reportu atd. To může mít za následek narůst výsledného souboru. V každém případě budeme mít k dispozici jediný exe soubor, který nebude nutné instalovat, ale pouze spustit. Klient sám se postará o vše další.

Na druhé straně naprogramujeme klienta v nějakém univerzálním programu. Například Java. Postup kompilace je nyní trošku jiný. Zdrojový kód je nejprve předkompilován na takzvaný byte-kod. Pro nás uživatele je již nečitelný, ale pro počítač je stále hardwarově nezávislý. Takový kód mohu předložit libovolnému operačnímu systému, který bude mít příslušný koncový Java kompilátor. Protože je program již předkompilovaný, je jeho konečná (nyní už hardwarově závislá) kompilace velice rychlá. Multiplatformní výhoda Javy je nesporná. Můžeme programovat na jakémkoliv OS a výsledek předkládat jakémukoliv jinému OS. Tato skutečnost ale není úplně pravdivá. Je třeba si uvědomit, že konečná kompilace je závislá na cílovém procesoru. A bez konečného kompilátoru nelze Java aplikace spustit.

2.2.2 Klient závislý na OS.

Klient naprogramovaný v Delphi (object pascal).

- **Nevýhoda:** -> Vývojové prostředí obvykle generuje (kompiluje) program do stejného operačního kódu v jakém samo pracuje. Takže pokud programujeme ve Windows XP, tak náš výsledný zdrojový kód bude zase pouze pro operační systémy z rodiny Windows. Tato nevýhoda padá, pokud je náš zdrojový kód psaný obecně. V takovém případě můžeme výsledný zdrojový kód nechat zkompilovat u někoho, kdo programuje aplikace v tomto vývojovém prostředí třeba pro UNIX.
- **Výhoda:** Výkon takto napsaného programu bude jednoznačně vyšší než u Javy. Nebude nutná ani žádná další kompilace. Klient je přímo spustitelná aplikace. Nepotřebuje ke své činnosti žádné další podpůrné soubory. Klient je jediný soubor, který není nutné instalovat. Jeho odstranění je realizováno pouhým smazáním.

Shrnutí:

Závislost na operačním systému:	ano
Počet souborů klienta:	1
Počet souborů potřebných pro spuštění klienta:	0
Potřeba instalace/odinstalace:	ne

2.2.3 Klient nezávislý na OS.

Klient naprogramovaný v Javě.

- **Výhoda**: Klient je multiplatformní. Lze spouštět zdrojové soubory na jakémkoliv operačním systému.
- **Nevýhoda**: Výkon aplikace bude nižší než u specializovaného klienta, protože Java musí provést konečnou kompilaci. Klient musí mít podpůrné soubory pro svou činnost. Minimálně konečný kompilátor závislý na OS. Tedy instalace jvm Java.

Shrnutí:

Závislost na operačním systému:	ano (konečný kompilátor)
Počet souborů klienta:	1
Počet souborů potřebných pro spuštěni klienta:	3300 (v instalaci obecné Javy)
Potřeba instalace/odinstalace:	ano (při nepřítomnosti Javy)

Opět se musíme dívat na problematiku z pohledu běžného uživatele. Pokud takový uživatel stáhne program napsaný třeba v c++ (přímo spustitelný), tak ho pouze uloží na disk. Zpravidla na plochu (autorova zkušenost se zákazníky) a klikne myší. Soubor se spustí a uživatel pracuje. Pokud se mu program přestane líbit, tak ho přesune do koše. Častěji však ponechá na ploše (opět autorova zkušenost se zákazníky) až do doby, kdy zavolá někoho, kdo mu plochu pročistí. Běžný uživatel rád přidává, ale má strach mazat. Jsou i tací, co mažou všechno, co jim přijde pod ruku.

Druhá situace. Běžný uživatel stáhne aplikaci napsanou třeba v Javě. Uživatel si stáhne takový soubor na plochu a klikne. Co se stane. Pokud není na cílovém počítači nainstalována Java, tak si ji program vyžádá. To většinou znamená konec pro běžného uživatele. Slušně vychovaný program si Javu doinstaluje sám. Zpravidla balíček JDK Java. Do počítače se nainstaluje zhruba 250MB ve 3000souborech. Toto proběhne pouze, pokud už Java není na cílovém PC přítomna. Nicméně to, že chci spustit 5MB program a potřebuji na to dalších 200MB ve 3000 souborech, je trochu moc. Je úplně jedno, že jsou v instalaci i helpy a obrázky. Běžného uživatele toto nezajímá. Neví o tom naprosto nic. Prostě se to děje. Běžný uživatel bude chtít časem aplikace taky smazat. Otázkou zůstává, co má smazat, kde a jak. Laik nemá žádnou šanci. Každý si tedy poradí, jak umí. Vede to pouze ke dvěma věcem. Servis a servis. V prvním případě pomoc s odstraněním programu. V druhém zprovoznění počítače.

3 VÝBĚR VÝVOJOVÉHO PROSTŘEDÍ

3.1 Databáze MySQL

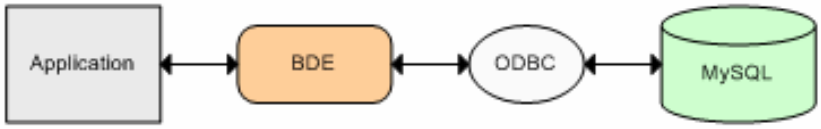
Jako databázi si zvolíme MySQL [7]. Tato databáze je pro nekomerční účely zdarma. MySQL je relační databázový server architektury klient-server. Je kompatibilní s jazykem **SQL**. Aktuálně ve verzi SQL2003. Volba padla právě na tuto databázi, protože je většinou obsažena i na veřejných hostingových serverech. Migrace databáze je pak poměrně snadná. Pro klienta to při přihlašování k databázi znamená přepsání jediného řádku. A to řádku s názvem serveru, kde je teď naše databáze umístěna. Můžeme mít i více databází, které budou replikované, a potom si klient může vybrat méně vytížený server.

3.2 Vývojový program klienta - Delphi 2007

Výběr programovacího jazyka záleží na nasazení naší aplikace. Na předpokladech využití, na destinaci používání, na výkonu sítě apod. Např. pokud budeme chtít používat naši firemní informační databázi kdekoliv na světě, tak těžko můžeme předpokládat, že budeme mít přístup k počítači, kdykoli to bude situace vyžadovat. Budeme nuceni vyrobit klienta tak, aby fungoval např. na mobilním telefonu. Na základě tohoto požadavku již provedeme potřebný výběr programovacího jazyka. My zvolíme programovací jazyk Delphi 2007 [5]. Jedná se v podstatě o objektově orientovaný programovací jazyk Pascal. Navíc v prostředí s použitím vizuálních komponent. I bez velkých znalostí programování v tomto jazyce je možné vytvářet pěkné a funkční aplikace.

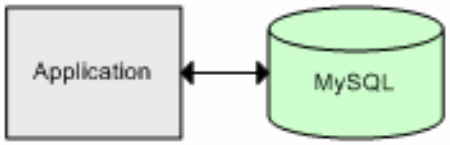
3.3 Spojení klient – server modulem MyDAC

Pro napojení klienta na databázi používáme zásuvný modul MyDAC [6]. MyDAC po integraci do Delphi umožňuje přímé napojení na databázový stroj bez použití překladače ADO/ODBC. Po kompilaci nevyžaduje žádné další externí soubory. Klient je přímo napojen na databázi, ať je kdekoli na světě. Není tedy nutné mít na straně klienta žádné další podpůrné aplikace jako v případě BDE (borland database engine) [4] (Obr.3).



Obr.3. Schéma spojení databáze s klientem technologií BDE.

MyDAC poskytuje dvě metody pro napojení na databázi. Přímé napojení (obr.4) a nepřímé napojení. (obr.5) V případě přímého napojení se MyDAC napojí na databázi přímo. Nepotřebuje k tomu databázového klienta v podobě knihovny, která je s MySQL serverem dodávána. V druhém nepřímém napojení použije dodaného klienta mysql.lib.



Obr.4. MyDAC - přímé napojení.

Obr.5. MyDAC - nepřímé napojení.

Naše aplikace bude používat přímé napojení. Na počítači, kde poběží klient, nepotřebujeme žádné další soubory. Takže klient bude absolutně přenositelný, nebo ho ani nemusíme kopírovat na disk. Kam přijdeme, tam zasunu do PC např. flash disk a přímo z něho klienta spustíme. Nicméně bude záležet na platformě. Klienty lze zkompilovat do nejpoužívanějších OS. Windows, MacOS a Linux.

3.4 Konfigurace klienta

Každý počítačový program má své specifické chování. Jsou programy, které po ukončení zapomenou vše, co jsme v nich prováděli. Takovéto programy nemají žádné vlastní konfigurace. Neuchovávají žádné údaje o své činnosti, ani o svém nastavení. Jiné programy si pamatují změny, které s nimi za běhu uživatel provede. Např. webový prohlížeč si pamatuje poslední navštívené stránky nebo třeba změnu svého vzhledu.

Programy si tyto informace ukládají obecně pro všechny uživatele, kteří s nimi pracují, nebo na základě autentizace pro každého uživatele samostatně. Při opětovném spuštění aplikace si tedy natáhne konfigurace patřící danému uživateli a nastaví prostředí tak, jak si uživatel určil. Možností, jak ukládat informace o těchto změnách, je celá řada. Například uložení do souboru či registru atd. U aplikací, které pracují s databázemi, se řešení přímo nabízí. Proč ukládat informace na lokální disk, když si je můžeme uložit do databáze na serveru, se kterou pracujeme. Metoda, kdy se konfigurace ukládají přímo na server, je velice výhodná. Aplikace klienta může být spuštěná z jakéhokoliv místa na světě a uživateli se po připojení na databázi nahrají jeho konfigurace. Toto v případě ukládání na disk nelze. Jiný počítač, jiný disk.

3.5 Bezpečnost komunikace klient server

Klient se připojí na databázi po síti. Na sítích obecně není bezpečno. Pokud nás případná ztráta dat příliš nemrzí, tak se o nějakou bezpečnost spojení klienta se serverem nemusíme příliš starat. Jsou však databáze, ve kterých jsou citlivé údaje. Např. bankovní účty apod. Takové aplikace vyžadují precizní a nezpochybnitelnou autentizaci uživatele, ale také přenosový kanál mezi klientem a serverem musí být zabezpečený proti odposlechu. Respektive odposlech je možný, ale získaná data musí být pro narušitele nečitelná. Přenosový kanál je nutné zašifrovat. K tomu slouží různé metody. Náš klient může používat technologii SSL (Secure Societ Layer). Je přímo součástí komponenty MyDAC i samotné databáze MySQL [6,7]. Firma Devart, která dodává MyDAC, nabízí také pokročilejší metodu zabezpečení kanálu. Komponentu rodiny Secure Bridge [7]. Avšak zabezpečení databáze a přenosového kanálu není předmětem této BC práce. Proto se jím nebudeme více zabývat.

II. PRAKTICKÁ ČÁST

4 NÁVRH A PROGRAMOVÁ REALIZACE KLIENTA

4.1 Implementace databázového stroje MySQL

Databázový stroj MySQL můžeme instalovat samostatně nebo spolu s webovým serverem. Pro naše účely je nejjednodušší použít balíček, který obsahuje jak databázi MySQL, tak webový server a rovněž webové rozhraní, které nám umožní vytvářet a spravovat databázi. Balíček se jmenuje XAMPP. Webový server není nezbytně nutné instalovat, ale právě z důvodů ovládání databáze MySQL webovým programem PhpMyAdmin je vhodné ho také použít. Už proto, že na veřejných serverech je PhpMyAdmin zpravidla používán. Po případné migraci na vzdálený server se nám jeho znalost bude hodit. XAMPP má přednastaveny všechny potřebné a důležité konfigurační atributy nejenom v MySQL, ale také na webovém serveru Apache. Můžeme tedy na svém lokálním počítači pracovat stejně, jako bychom byli připojeni na vzdálený server. Můžeme si dovolit manipulovat s nastavením jak databázového stroje, tak webového serveru. A tím provádět různé pokusy. Toto nám žádný provozovatel veřejného webu rozhodně nedovolí.

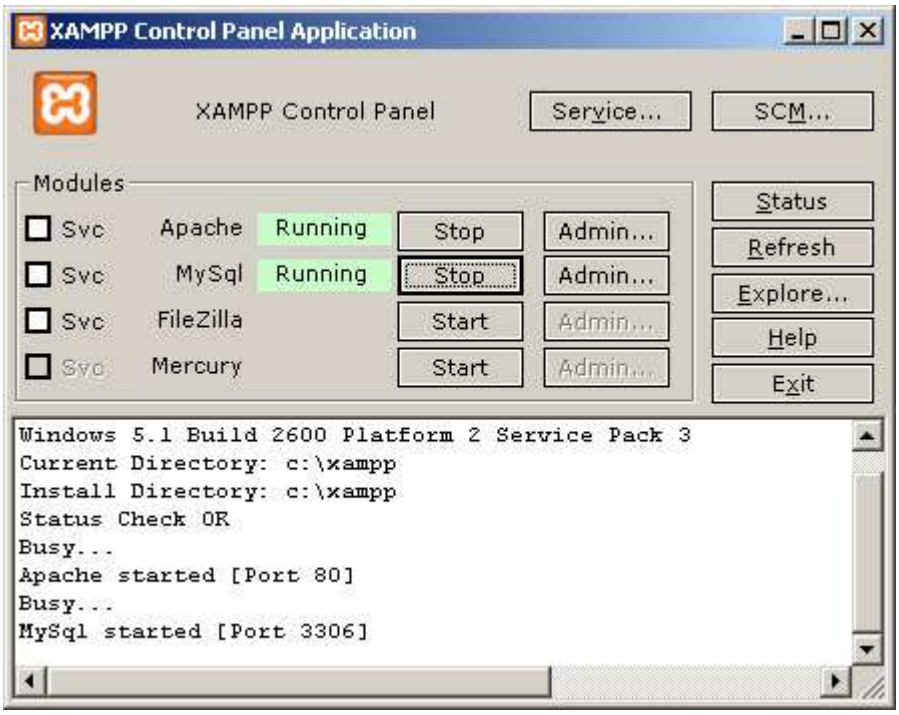
4.1.1 Instalace XAMPP

Balíček XAMPP je dostupný na webu. Stačí do vyhledávače zadat XAMPP. Po stažení balíčku (1.6.8) provedeme instalaci. Doporučuji ponechat všechna potřebná nastavení. Zejména cílový adresář instalace (destination folder) ponechat defaultní. Tedy c:\xampp. Je s tím v dalších krocích počítáno. V průběhu instalace dojde také na implementaci PHP interpreteru a konfiguraci serveru Apache (obr.6). Trvá to asi 2 minuty.

Obr.6. XAMPP - probíhající konfigurace serveru Apache.

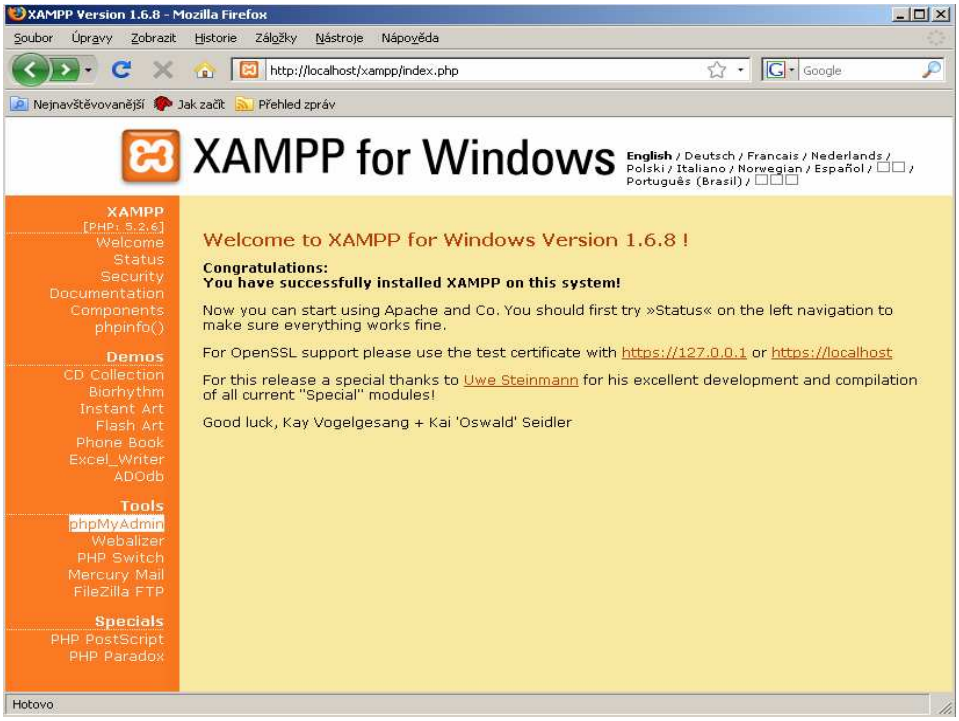
4.1.2 Vytvoření ukázkové databáze

Po instalaci se spustí konzola aplikace XAMPP (Obr.7). Zajímají nás pouze tlačítka start a stop u Apache a MySQL. Spouštíme tím a zastavujeme oba servery. Zaškrtávací políčko Svc nezaškrtávat. My budeme pracovat vždy ručně. Po ukončení práce vypneme oba servery, abychom uzavřeli potenciální přístup k našemu PC. Spustíme Apache a MySQL.



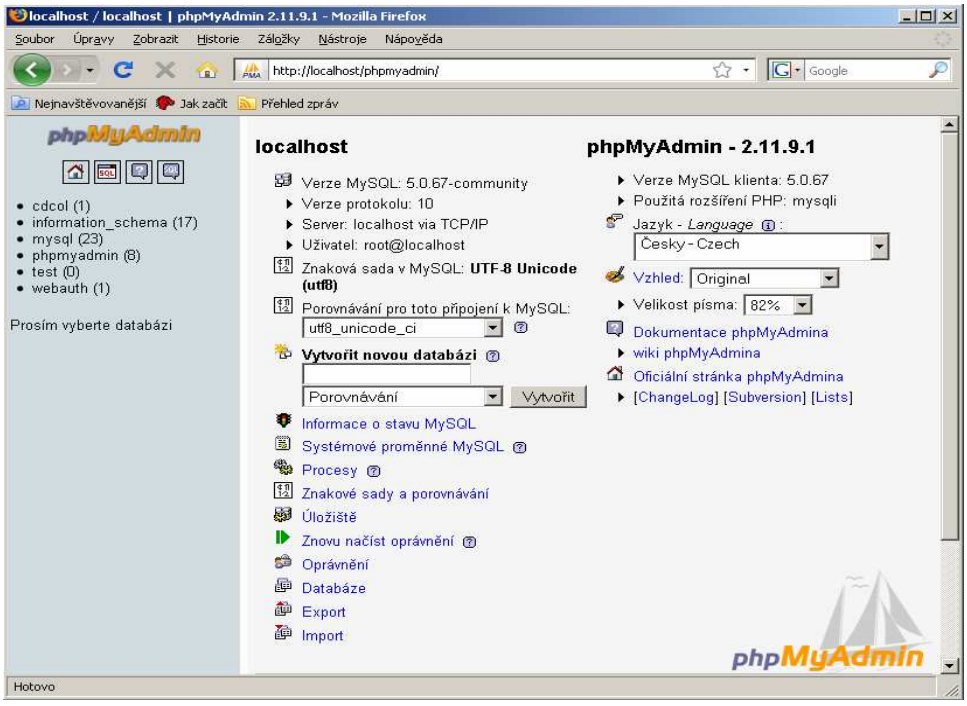
Obr.7. XAMPP - kontrolní a ovládací panel aplikace.

Nyní můžeme používat jak databázi MySQL, tak webový server Apache pro zobrazení HTML i PHP. Součástí instalace je i PHP interpreter, který nám umožní spolu s Apachem používat nástroj PhpMyAdmin, ve kterém provedeme vytvoření naší ukázkové databáze. Tuto aplikaci spustíme tedy v běžném internetovém prohlížeči. Z toho důvodu jsme si zprovoznili server Apache. Spustíme webový prohlížeč IE nebo Mozillu. Do řádku, kde obvykle píšeme adresu ve tvaru www.něco.cz, nyní napíšeme pouze localhost. XAMPP pro nás na této lokální adrese připravil uvítací stránku. Zdroje této stránky jsou v adresáři c:\xampp\htdocs. Stránka se startuje souborem Index.html, popřípadě Index.php. V prohlížeči máme nyní uvítání XAMPP a k dispozici několik jazykových mutací. Vybereme anglický jazyk. Získáme webové aplikační rozhraní XAMPP (Obr.8). Ne podobné tomu, co nám předkládají veřejné webové servery.



Obr.8. XAMPP - uvítací webová stránka aplikace.

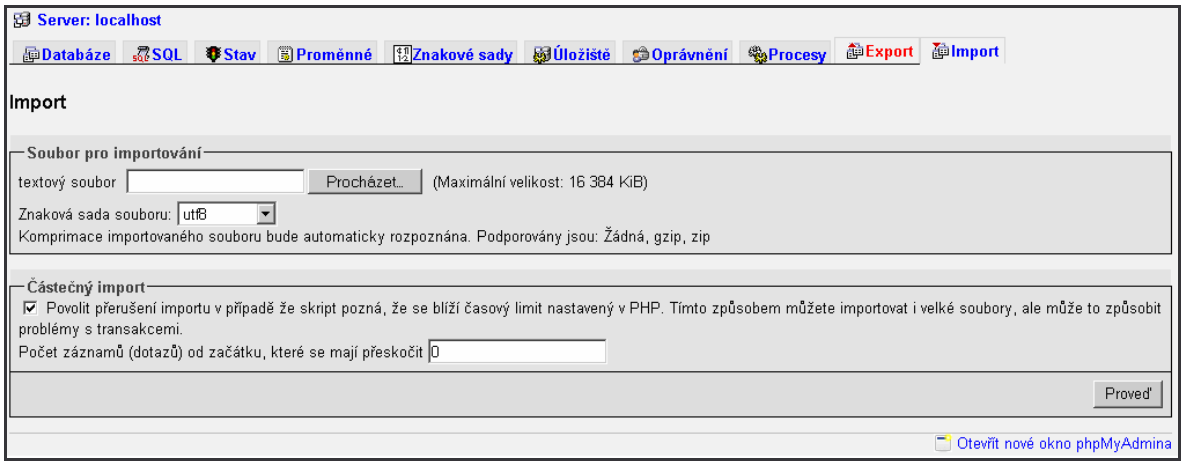
Na levé straně v sekci Tools, spustime webovou (PHP) aplikaci ***PhpMyAdmin*** (Obr.9).



Obr.9. Aplikace PhpMyAdmin

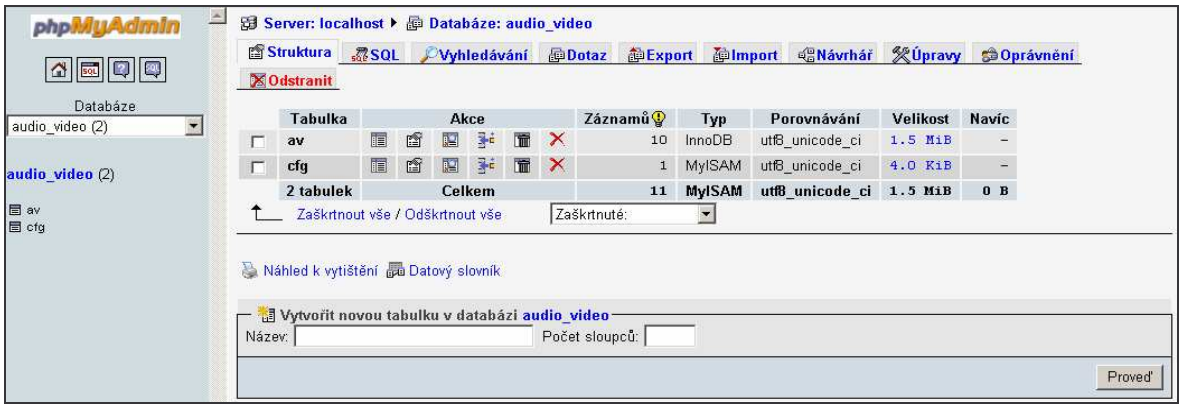
Toto je skutečně plnohodnotný klient, který pracuje s databází MySQL. Tedy v podstatě totéž, co deláme my. Jedná se o velice univerzálního klienta, který umí vše potřebné pro

administraci databáze. Naší snahou je naprogramovat klienta specialiovaného, který bude umět pouze prohlížet a editovat data jedné databáze, kterou si vytvoříme právě pomocí tohoto programu. Databázová sada pro naši ukázkovou aplikaci je již připravena na přiloženém CD v adresáři ***MYSQL***. Jedná se o soubor ***audio_video.sql***. Na obrázku 9, až úplně dole, použijeme odkaz import. Získáme stránku, pomocí které můžeme importovat databázový soubor (Obr.10).



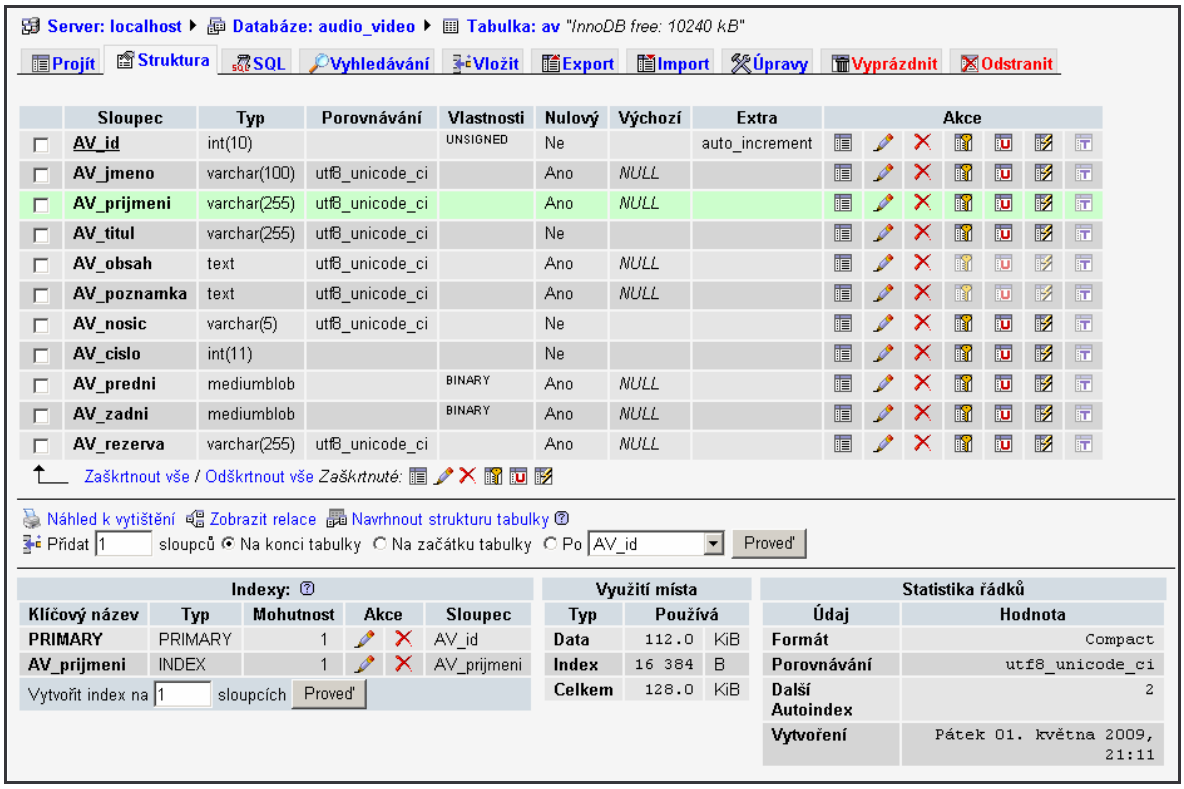
Obr.10. PhpMyAdmin - import dat.

Vložíme náš přichystaný soubor ***audio_video.sql*** a stiskneme tlačítko proveď. PhpMyAdmin provede importování databáze. Některé verze PhpMyAdminu generují při importu chybové hlášení o neexistenci databázových tabulek. Tím se nemusíme zabývat. Import projde a získáme opět úvodní stránku PhpMyAdmin, ve které bude v levém panelu o jednu databázi navíc. O naší databázi ***audio_video***. Klikneme na odkaz naší databáze a dostaneme okno podle obrázku (Obr.11).



Obr.11. PhpMyAdmin - seznam tabulek.

Získali jsme seznam tabulek, který naše databáze obsahuje. Vidíme, že naše databáze ***audio_video*** obsahuje dvě tabulky s názvem ***av*** a ***cfg***. Podívejme se nyní na strukturu tabulky ***av***. Pokud klikneme na odkaz vedle (symbol obálky) ***av***, získáme obsah tabulky. Nyní klikněme přímo na ***av*** a dostaneme struktury naší tabulky (Obr.12).



Obr.12. PhpMyAdmin - schéma tabulky „av“.

Tabulka ***av*** obsahuje celkem deset položek. Začínají názvem AV_id a končí AV_rezerva.

4.1.3 Struktura databáze audio-video

Nyní si pouze informativně řekneme o struktuře naší databáze [1,2]. Je třeba upozornit, že tato bakalářská práce nepojednává o databázích, ale především o napojování na databáze pomocí externích klientů. Naše ukázková databáze obsahuje následující sloupce.

- AV_id jednoznačná identifikace záznamu
- AV_jmeno jméno interpreta
- AV_prijmeni příjmení interpreta
- AV_titul název titulu
- AV_obsah seznam písniček nebo filmu atd.
- AV_poznamka poznámka k titulu
- AV_nosic na jakém mediu je titul uložen
- AV_cislo číslo nosiče
- AV_predni přední booklet (obrázek)
- AV_zadni zadní booklet (obrázek)
- AV_rezerva rezervní sloupeček

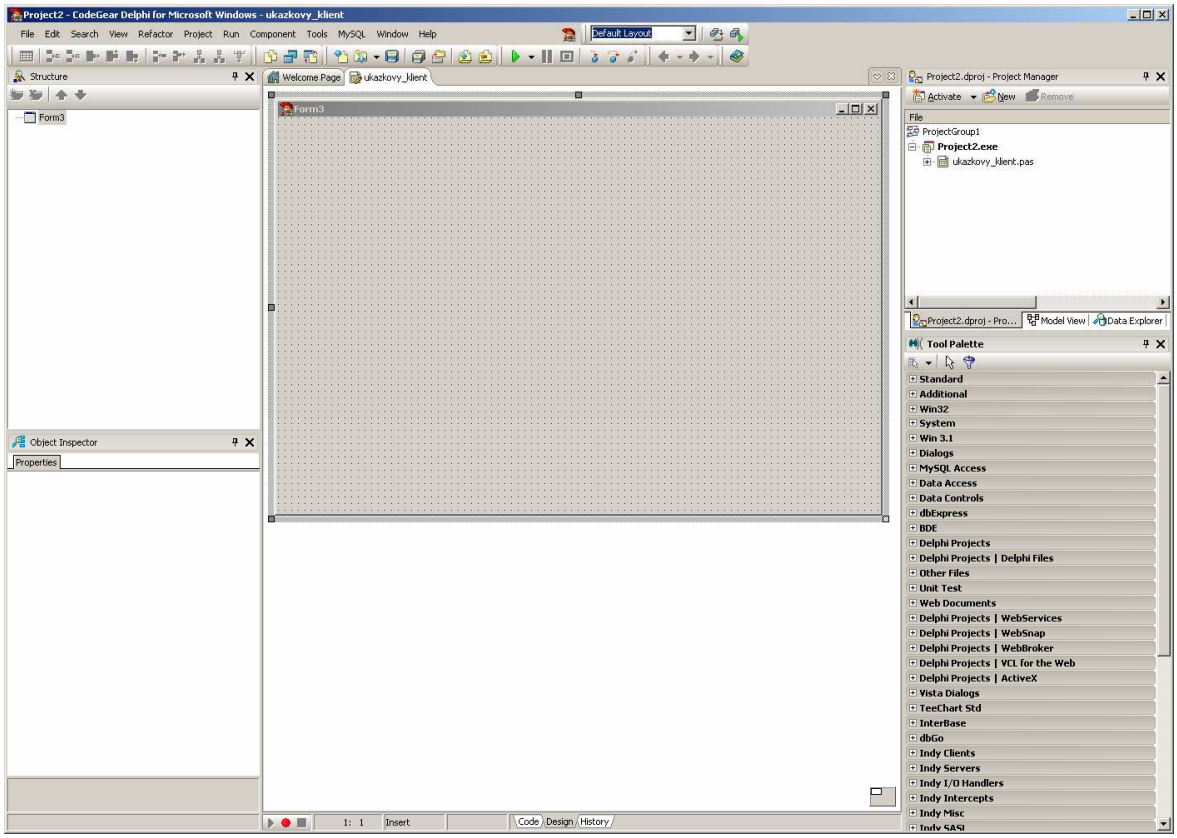
Každý sloupeček v tabulce musí mít svůj název a další doplňkové informace. Především datový typ. Tak například sloupeček AV_id je datový typ INT. Má vlastnost UNSIGNET(nezáporné číslo) a extra vlastnost AUTO_INCREMENT(automatické číslování). Další vlastností je PRIMÁRNÍ KLÍČ tohoto sloupce, který nám umožňuje spojovat více tabulek a jednoznačně identifikovat daný záznam tabulky. Více o strukturách tabulek v příslušné literatuře [1,2]. Vzhledem k maximálnímu zjednodušení naší ukázkové databáze používáme pouze jedinou tabulku. V praxi by návrh této databáze vypadal poněkud jinak. Minimálně bychom oddělili obrázky od ostatních dat a také bychom si vytvořili tabulku nosičů, abychom zabránili redundanci dat. Například proč ukládat do databáze stokrát typ nosiče SACD (Super-Audio-Compact-Disc), když můžeme do databáze vložit pouze číslo. Například 4. V tabulce nosiče budeme mít jednoznačně řečeno, že pod číslem 4 jsou SACD nosiče. Tím ušetříme 70 % kapacity u záznamu datového nosiče. Této technice se říká ***master-detail*** [1,2,4]. Nicméně pro jednoduchost a prezentační účely máme jedinou tabulku, která nám plně postačí pro naši ukázkovou

databázi ***audio_video***. Do naší databáze můžeme vkládat data i přímo pomocí PhpMyAdmina, ale to nyní dělat nebudeme. Pustíme se rovnou do programování designu našeho klienta.

4.2 Grafický návrh klienta v programu Delphi 2007

4.2.1 Delphi 2007 – CodeGear Rad studio

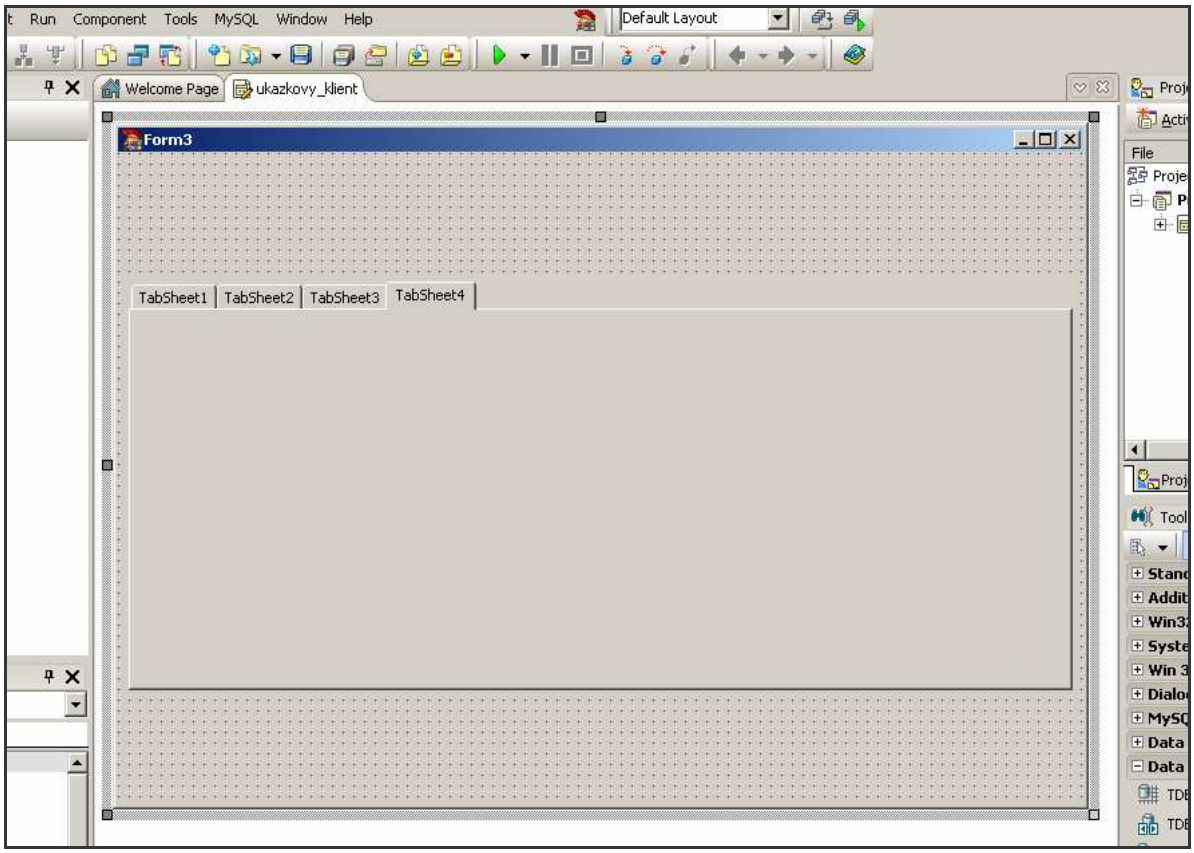
Spustíme Delphi 2007. Založíme nový projekt (File-New-VCL forms aplication Delphi for Win32). Na obrázku (Obr.13) je pracovní plocha vývojového prostředí Delphi. Šedá část uprostřed reprezentuje náš formulář, na který budeme postupně aplikovat jednotlivé komponenty potřebné pro realizaci našeho klienta. Komponenty najdeme na paletě komponent (tool palette), které se nachází v pravé dolní části. V levé části aplikace jsou potom seznamy komponent, které použijeme a také vlastnosti jednotlivých komponent, jako barvy, fonty, umístění atd.



Obr.13. Delphi - vývojové prostředí.

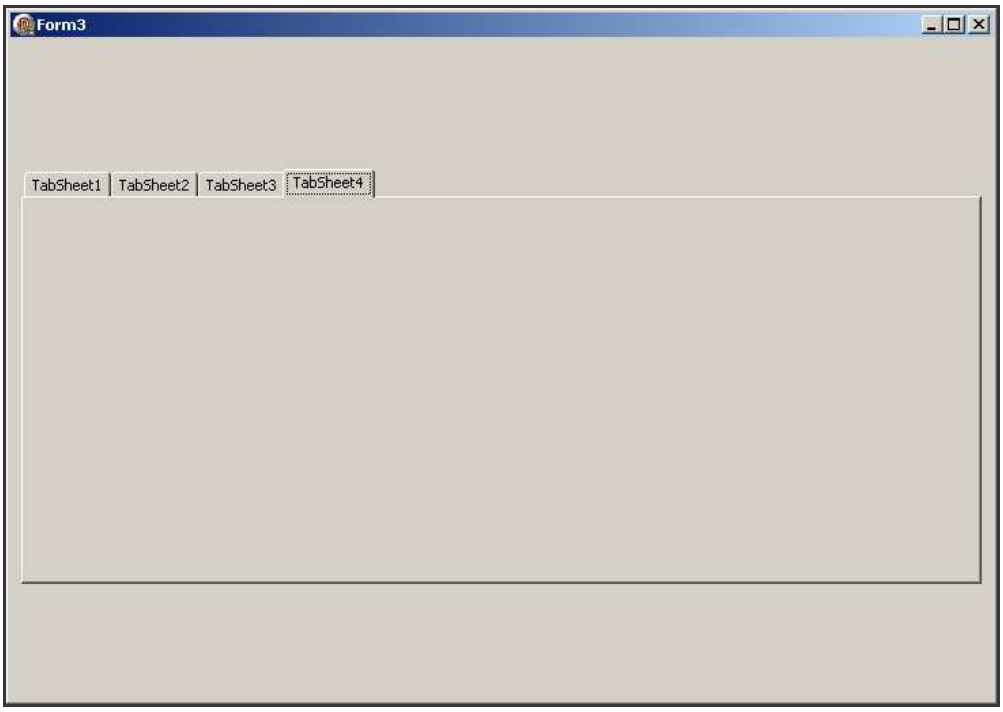
4.2.2 Vytvoření projektu a rozmístěni ovládacích prvků návrhu

Nyní provedeme velice jednoduchý návrh grafického rozhraní našeho klienta. Klient po kompilaci bude naprosto slepý. Nebude umět nic. Všechny ovládací prvky (komponenty), které použijeme, nebudou mít žádné vazby. Teprve s použitím další komponenty je oživíme. To ale až v další podkapitole. Grafický návrh rozdělíme do čtyř oblastí. Všechny oblasti ponecháme na jednom formuláři. Je to jednoduché a přehledné. Pro naše účely naprosto dostačující. Z palety ***Win32*** přetáhneme myší na náš formulář komponentu ***TpageControl*** a na ni pravým tlačítkem myši vytvoříme 4 záložky. Okno si podle potřeby roztáhneme a přizpůsobíme naším požadavkům. Výsledek bude vypadat jako (Obr.14).



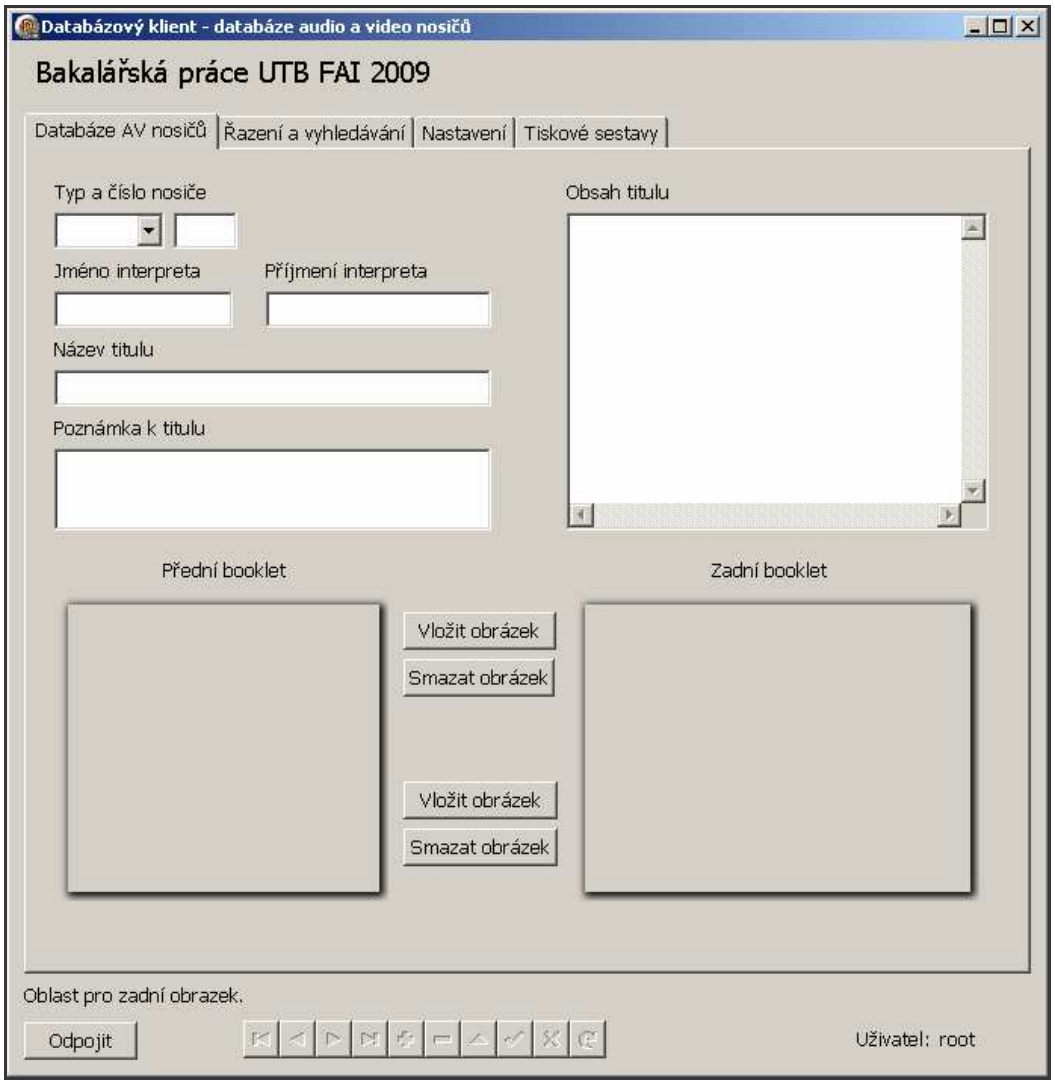
Obr.14. Delphi - Pracovní plocha budoucí aplikace.

Pokud nyní klikneme na zelenou šipku, kterou najdeme nad naším formulářem, dojde ke kompilaci našeho programu. Program vytvoří spustitelný exe soubor, který je následně spuštěn. Výsledek našeho dosavadního snažení vypadá následovně (Obr.15).



Obr.15. Delphi - Zkompilovaná aplikace.

Tímto způsobem budeme pokračovat vždy, když si budeme chtít prohlédnout výsledek naší dosavadní práce. Vygenerovaný formulář ukončíme křížkem vpravo nahoře. Nyní přejdeme k ukázce úpravy vlastností naší první komponenty. Na levé straně našeho vývojového prostředí je panel ***object inspektor,*** pomocí kterého můžeme s vybranou komponentou provádět všechny dostupné grafické i aplikační změny. Například TabSheet4 přejmenujeme na výstupní_sestavy tak, že v okně ***object inspektor*** na řádku ***Caption*** přepíšeme stávající název na nový název. Takto budeme manipulovat se všemi komponentami a jejich vlastnostmi. Nyní si tedy provedeme kompletní návrh vzhledu našeho formuláře. Barvy, velikost a rozmístění jednotlivých prvků je pouze na estetickém cítění každého jednotlivce. Nejprve si první záložku přejmenujeme na ***Databáze AV nosičů***. Potom na tuto záložku přeneseme postupně několik komponent a ty vhodně rozmístíme. Následně provedeme kompilaci a získáme např. následující vzhled (Obr.16).



Obr.16. Delphi - Výsledný grafický návrh aplikace.

Použili jsme následující komponenty. Pro všechny popisky (Typ a číslo nosiče, Jméno a Příjmení interpreta, název titulu atd) Použijeme komponentu ***TLabel*** z palety ***Standard***. Komponenty s rozklikávací šipkou najdeme na paletě ***DataControls***. Komponenta se jmenuje ***TDBComboBox***. Na téže paletě najdeme komponenty ***TDBEdit*** a ***TDBMemo***. ***TDBMemo*** použijeme pro obsah titulu a poznámku. Další komponentou z palety ***Standard*** jsou tlačítka ***Tbutton***. Poslední komponentou z palety ***ImageEn*** je ***TImageEnDbVect,*** což jsou pochopitelně obrázky. Komponenty ***ImageEn*** si musíme nejprve doinstalovat do prostředí Delphi [7,8]. Jsou to komponenty pro práci s obrázky téměř všech formátů. Použití těchto komponent je výhodné, protože komponenta obsažena přímo v Delphi v paletě ***DataControls TDBImage*** je použita pouze pro formát BMP. Ostatní formáty se

musí načítat pomocí streamu a dalších programovacích technik. Což je zbytečné. Grafický návrh máme realizován a nyní přistupme k vlastnímu napojení na naši databázi. V databázi máme pro ukázku již několik záznamů. Ještě než přikročíme k oživení databáze, nezapomeneme přejmenovat naše 4 tlačítka a dodat popisky k těm dvěma obrázkům. Například „přední booklet“ atd.

4.3 Spojení klienta s databází modulem MyDAC

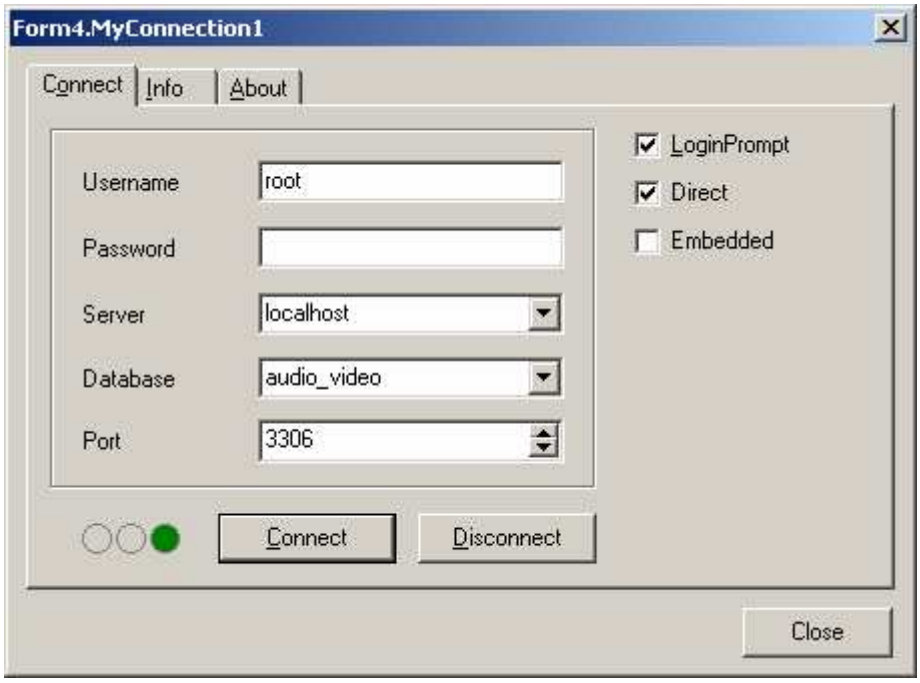
V této fázi vývoje klienta potřebujeme oživit jednotlivé komponenty. Tedy napojit komponenty na konkrétní databázový stroj, databázi, tabulku a na konkrétní sloupeček v tabulce. K tomuto účelu použijeme komponenty MyDAC [6]. Uvidíme, že napojení na databázi s použitím těchto komponent je velice snadné.

4.3.1 Instalace MyDAC

Komponenty MYDAC nainstalujeme běžným způsobem jako jakýkoliv jiný program do Windows. Instalátor se postará o integraci těchto komponent do prostředí Delphi. Komponenty se po restartu programu Delphi objeví v paletě komponent pod názvem ***MYSQLAccess***.

4.3.2 Propojení komponent MyDAC a DataAccess delphi

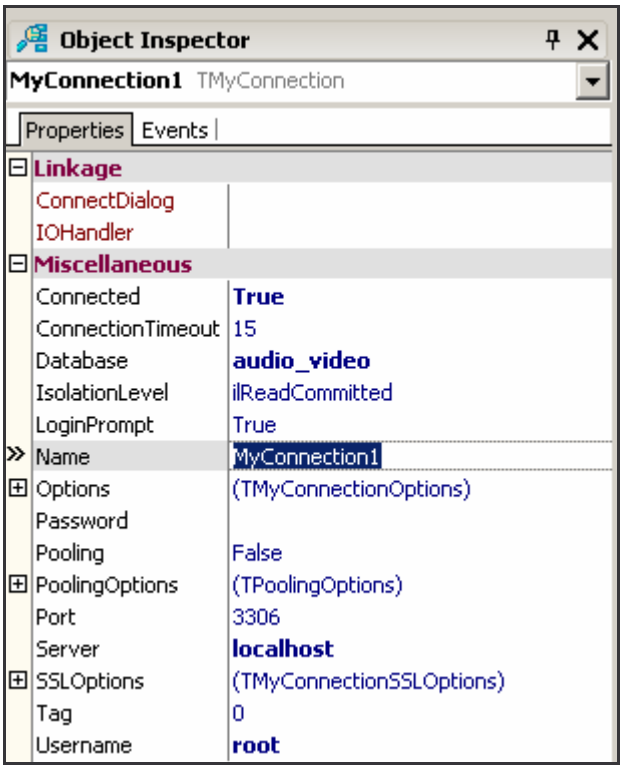
Rozklikneme si komponenty ***MYSQLAccess*** a přesuneme na náš pracovní formulář komponentu ***TMyConnection***. Je jedno, kde na formulář ji umístíme. Nyní máme několik způsobů, jak se napojit na naší databázi audio_video na serveru ***Localhost***. Jednak s použitím ***object inspektoru*** nebo 2x klikneme na komponentu ***TMyConnection***. Otevře se nám formulář, do kterého si můžeme přímo napsat konkrétní připojovací údaje. Samozřejmě to, co napíšeme do tohoto formuláře, se projeví i v ***objekt inspektoru***. Vyplněný formulář vidíme na (Obr.17).



Obr.17. MyDac - konfigurační formulář připojení.

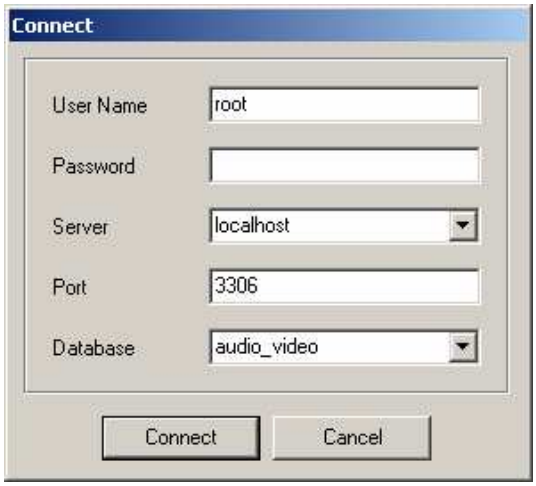
Databáze MySQL disponuje pokročilými technikami přístupových práv, ale to nás nyní nezajímá. Více se o přístupových právech můžeme dozvědět v literatuře [1,5]. My použijeme předdefinovaného prvotního uživatele. Mimochodem tzv. superuživatele. Navíc bez hesla. Bezpečnost takové databáze je naprosto nulová. V praxi nejenom, že speciálně pro ***root*** uživatelé použijeme silné heslo, ale pro běžný provoz databáze používáme jiného uživatele než ***root***. Např. ***pepa***. Uživateli ***pepa*** potom nadefinuje práva na databázi. Například může data modifikovat, přidávat nové záznamy, ale nesmí mazat záznamy apod. Nadefinujeme mu přístupová práva. V podstatě je to stejné jako přístupová práva do OS. Pokud totiž necháme uživatele ***root*** bez hesla, tak kdokoliv si s naší databází může dělat, co chce. Více informací o zabezpečení MySQL získáme v příslušné literatuře [1,5]. Naším úkolem je spojit naši databázi se serverem. Se serverem ***Localhost***. My máme k dispozici tento server, protože jsme si nainstalovali XAMPP, ve kterém je obsažena databáze MySQL a ta je nyní spuštěná. A ***Localhost*** není nic jiného, než náš počítač. ***Localhost*** má adresu 127.0.0.1. Je to adresa našeho počítače. Tedy pokud voláme server ***Lohalhost***, voláme sami sebe. A to je správné, protože databáze nyní leží na našem počítači. Název databáze je ***audio_video.*** Port je ***3306***, který defaultně využívá server MySQL. Tedy je to port, na kterém naslouchá speciální klient (démon mysqld) databáze MySQL. Naslouchá a čeká, jestli po něm někdo něco nebude chtít. A my chceme. Chceme se na něho napojit.

Chceme navázat spojení. Udělejme to. Stačí stisknout ***Connect*** a je hotovo. Nyní si všimněme ***object inspecotoru*** (Obr.18)



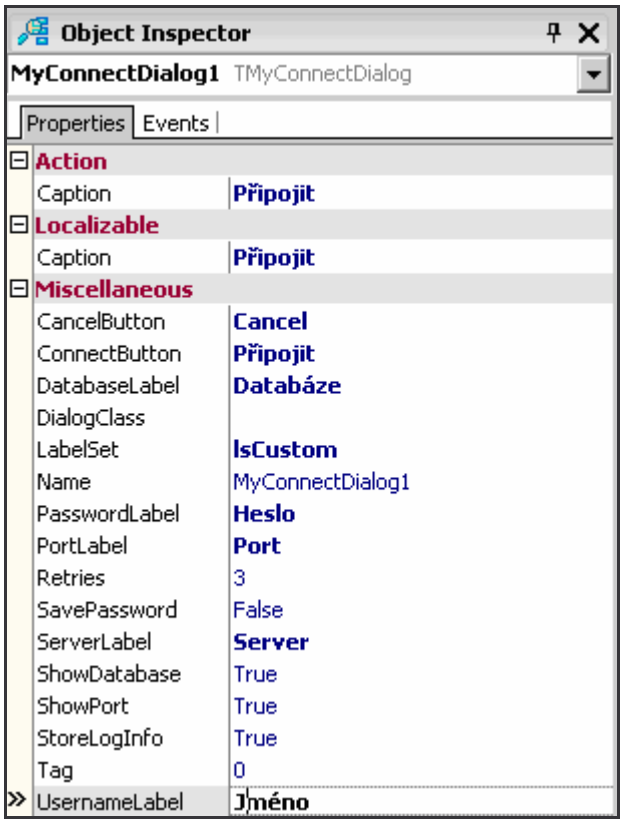
Obr.18. Delphi - Object Ispector.

Na řádku ***Connected*** je ***True***. Tedy jsme spojeni. Databáze ***audio_video*** jako ***Username root***. Poklepáním na ***Connected*** můžeme ***True*** změnit na ***False*** a tím se odpojit. Po opětovném ***True*** jsme vyzváni k zadání hesla pro nové spojení. Pokud nyní zkompilujeme program, tak náš klient použije přihlašovací insignie tak, jak jsme je zadali do ***object inspektoru***. Tedy natvrdo. To se nám ale moc nehodí. My se budeme chtít přihlašovat na jiný server pod jiným jménem a heslem atd. Potřebujeme tedy nějaký přihlašovací formulář. Ten si můžeme naprogramovat sami, nebo použijeme další komponentu z palety ***MySqlAccess***. A co jiného, než ***TmyConnectDialog***. Vložme tedy tuto komponentu na plochu a proveďme kompilaci. Před tím, než se spustí samotný klient, jsme vyzváni k zadání všech přihlašovacích údajů (Obr.19) A to je to, co potřebujeme.



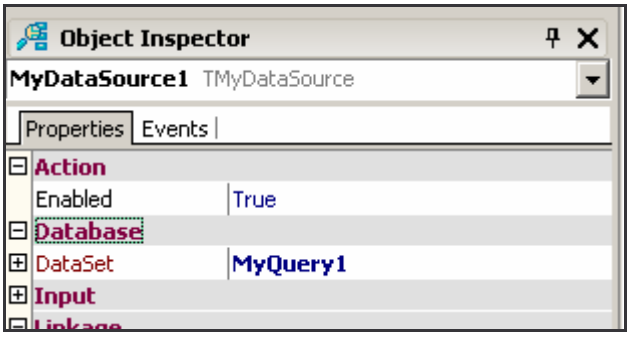
Obr.19. Aplikace - přihlašovací formulář.

Je možné říci klientovi, aby se vždy přihlašoval na konkrétní server s konkrétními přihlašovacími údaji. To lze velice jednoduše nastavit na ***object inspektoru*** u komponenty ***MyConnectDialog1***. My zatím přejmenujeme názvy na přihlašovacím formuláři do češtiny. Například takto. (Obr.20)



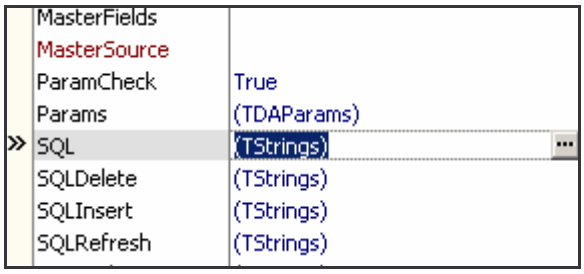
Obr.20. Delphi - nastavení přihlašování.

Provedeme kompilaci a vidíme, že je nám nabídnut přihlašovací formulář, ale bez našich úprav. Jak je to možné? Uvědomme si, že komponenta ***MyConnectDialog1*** je volně v prostoru. Zatím nikam neukazuje, na nic není napojena. K čemu je nám dobrá? My chceme, aby se komponenta ***MyConnection1*** připojila na naši databázi pomocí nějakého přihlašovacího formuláře. Musíme komponentě ***MyConnection1*** říci, pomocí jakého formuláře se tam má připojovat. Klikneme na komponentu ***MyConnection1*** a nahoře v ***object inspektoru*** u položky ***ConnectDialog*** vybereme ten dialog, který chceme použít. My máme pouze jeden reprezentovaný komponentou ***MyConnectDialog1***. Takže ji vybereme. Provedeme kompilaci a konečně dostaneme náš přihlašovací formulář, tentokrát již ve správném jazyce. Toto byla hezká ukázka použití objektového programování. V podstatě jsme řekli toto. Náš klient se připojuje na databázi přes připojovací komponentu, která nám k připojení nabídne náš přihlašovací formulář. Ten je k tomuto účelu přiložen k přihlašovací komponentě. Jsme tedy přihlášeni ke konkrétní databázi, ale nic moc se neděje. Žádná data nevidíme. Databázový server o nás ví. Splnil náš požadavek a vytvořil s naším klientem spojení. Nyní čeká, co s tím provedeme. Co vlastně po databázi chceme nejprve? Chceme naše komponenty, které jsme si rozmístili na náš formulář, spojit s konkrétními položkami naší tabulky ***av***. To nelze udělat přímo. Musíme použít další komponenty z palety ***MySqlAccess***. Konkrétně 2. ***TmyDataSource*** nám umožní připojit naše rozmístěné komponenty na konkrétní položku databáze. Tato komponenta to ale musí udělat přes další komponentu, ve které budou nejenom načtená data naší databáze, ale také informace o tabulce. Jako je počet a název položek atd. Tato komponenta se nazývá ***TmyQuery***. Obě komponenty vložíme na náš formulář. Klikneme na komponentu ***MyDataSource1*** a v ***object inspectoru*** rozklikneme položku databáze a tam nastavíme (vybereme) řádek ***dataset*** na ***MyQuery1*** (Obr.21)



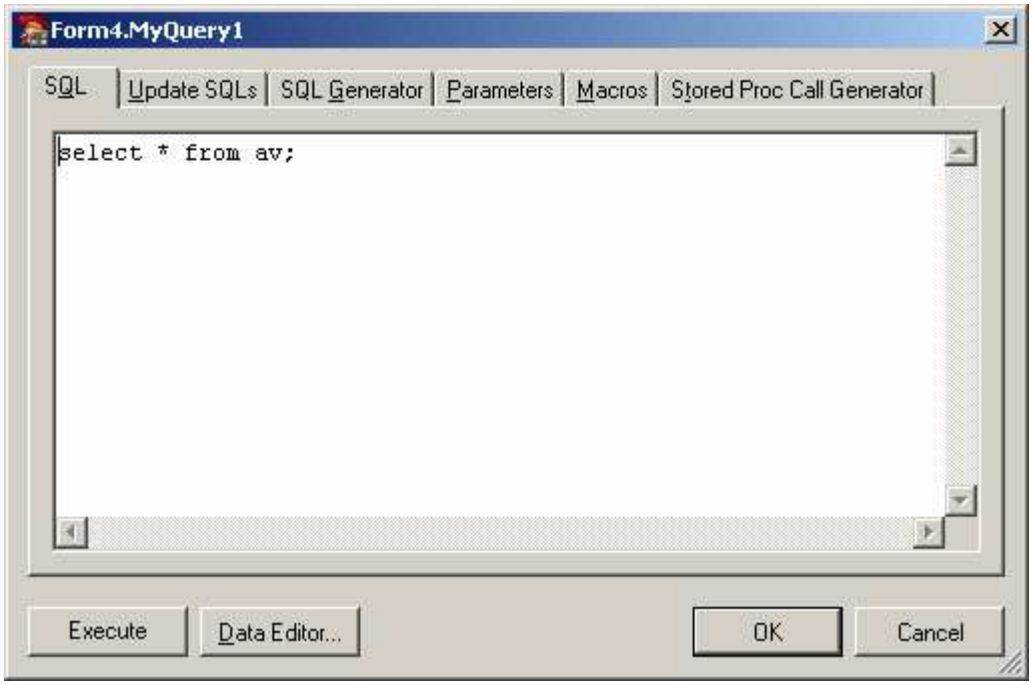
Obr.21. Delphi - napojování komponent.

Klikneme na druhou komponentu ***MyQuery1*** a v ***object inspectoru*** klikneme na položku SQL. (Obr.22)



Obr.22. Delphi - vkládání SQL dotazu.

Získáme tím formulář, ve kterém můžeme činit dotazy na naší databázi a na konkrétní tabulku v ní. Po vyplnění dotazu (Obr.23) vlastně požádáme databázi, aby nám zaslala výsledek do naší komponenty ***MyQuery1***. K ***MyQuery1*** je připojena komponenta ***MyDataSource1.*** Teprve ke komponentě ***MyDataSource1*** připojíme naše komponenty na ploše našeho formuláře.

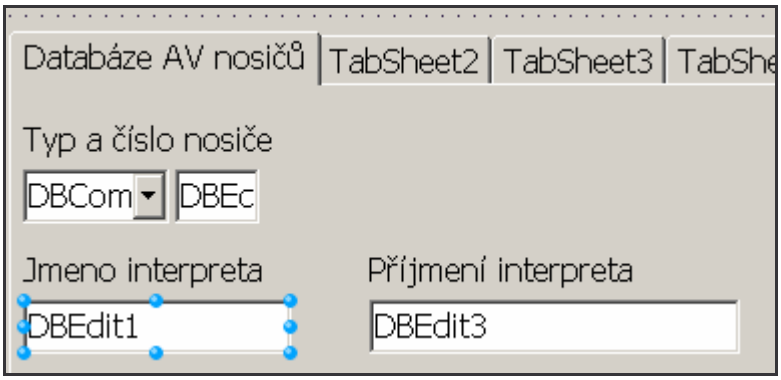


Obr.23. Delphi - formulář pro vkládání SQL dotazů.

Na obrázku v podstatě vidíme dotaz v jazyce SQL, který říká - pošli nám všechny záznamy z tabulky ***av***. Shrňme si dosavadní propojení komponent.

DATABAZE <---> MyConnection1 <>MyConnectDialog1<> MyQuery1<> MyDataSource1

A teprve nyní můžeme začít napojovat data v naší databázi na konkrétní buňky (komponenty), které jsme si přichystali na náš formulář. Abychom ihned po připojení viděli data, která máme v databázi uložena, můžeme říci komponentě ***MyQuery1,*** aby začala pracovat už teď v době návrhu. Klikneme na ni a v ***object inspektoru*** v řádku ***Active*** změníme ***False*** na ***True***. V této chvíli byl příkaz „***select * from av***;“ vykonán a můžeme s daty manipulovat, aniž by bylo potřeba aplikaci kompilovat. Přistupme nyní k propojení první komponenty. Vybereme si třeba komponentu ***DBEdit1*** (Jméno interpreta) (Obr.24).

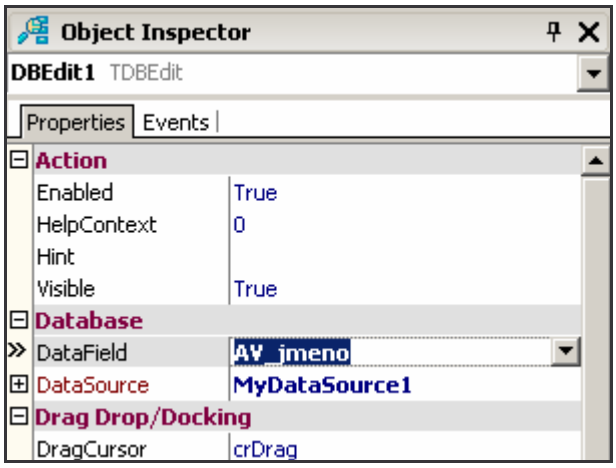


Obr.24. Delphi - výběr komponenty.

V ***object inspektroru*** máme rozklikávací položku **Databáse** (Obr.25). Tam máme řádek ***DataSource***. Vybereme zdroj dat. Samozřejmě tam najdeme ***MyDataSource1***.

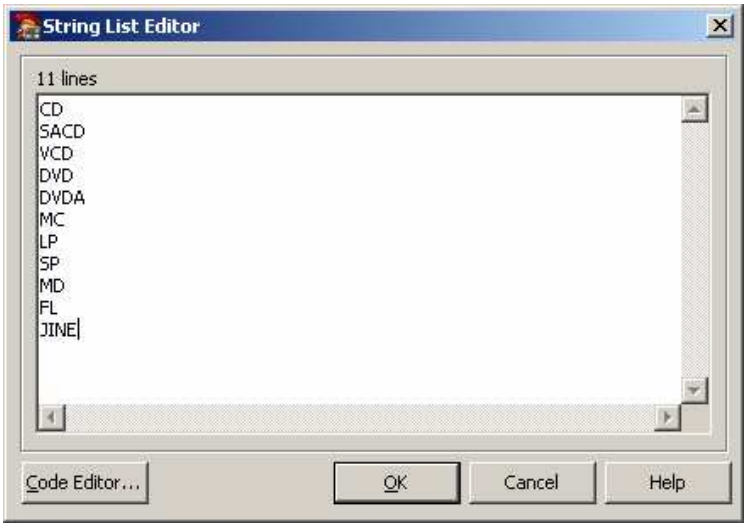
Obr.25. Delphi - vlastnost komponenty.

Tím si tato komponenta přečetla údaje o tabulce a nyní můžeme v řádku ***DataField*** vybrat konkrétní položku databáze. Vybereme jak jinak ***AV_jmeno*** (Obr.26).



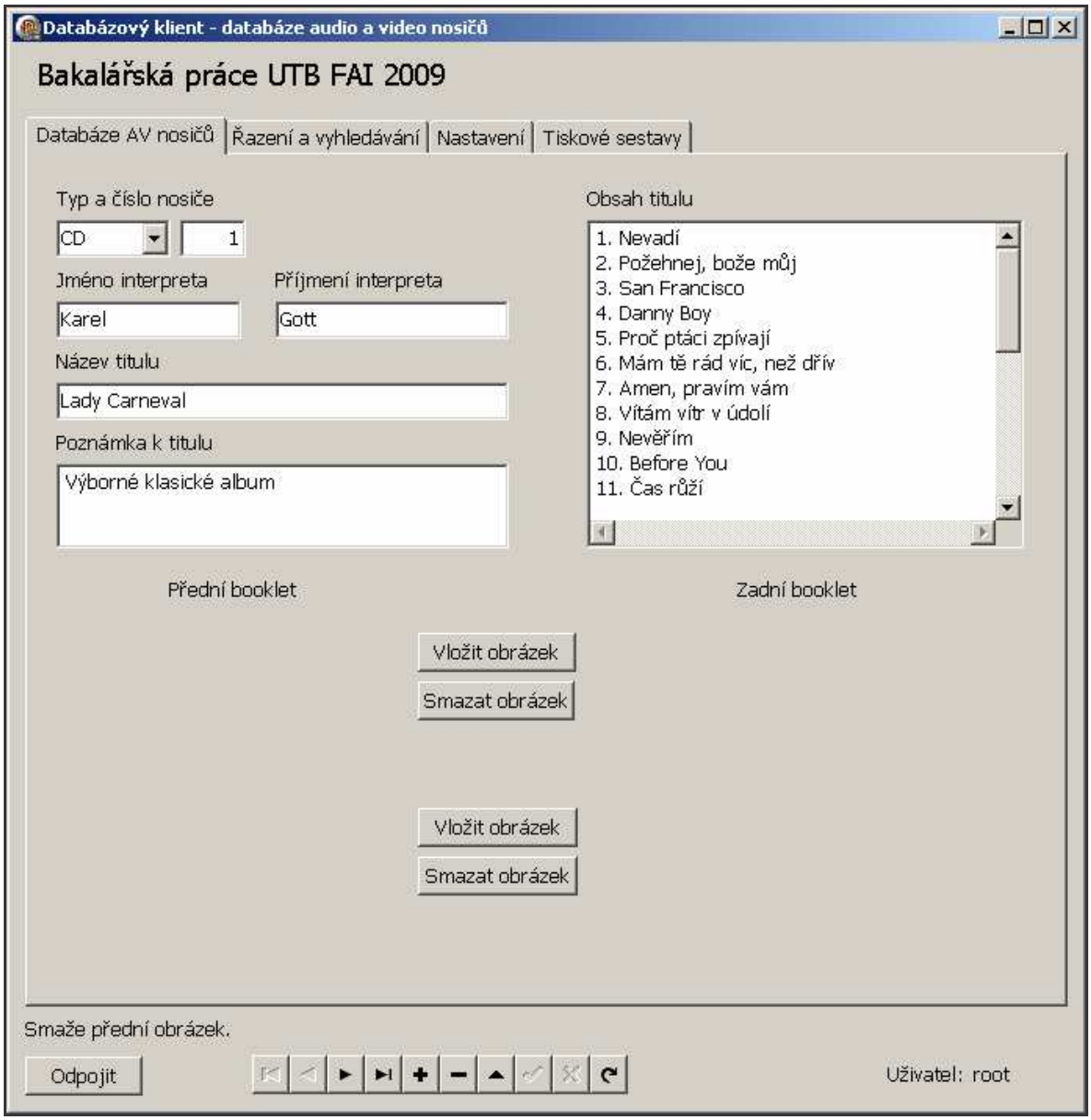
Obr.26. Delphi - oživení komponenty.

V okamžiku, kdy jsme toto udělali, se nám v naší komponentě ukáže konkrétní výsledek. První na seznamu databáze u položky ***AV_jmeno*** je Karel. Totéž provedeme analogicky s dalšími komponentami. Včetně obrázků. Za zmínku stojí ještě komponenta ***DBComboBox1***. Ta nám umožňuje vybírat položky, které chceme vkládat do databáze. Seznam položek předvyplníme v ***object inspektoru*** na řádku ***Items***. Vyjede nám formulář, do kterého zapíšeme to, co budeme chtít v seznamu (Obr. 27).



Obr.27. Delphi - Vyplňovací formulář pro seznamy.

Potvrdíme a zkompilujeme. Nyní již můžeme vybírat z nabídky nosičů. V ***object inspektoru*** si můžeme také zvolit, která položka se nám má objevit jako defaultní atd. Nemá smysl nyní cokoliv provádět. Mimochodem, copak se asi objeví v komponentě Příjmení interpreta? Pokud jsme postupovali správně, tak král českého popu Gott Karel. Včetně jeho alba Lady Carneval. Máme tedy napojené všechny komponenty. Provedeme kompilaci, připojíme se a vidíme výsledek (Obr.28).



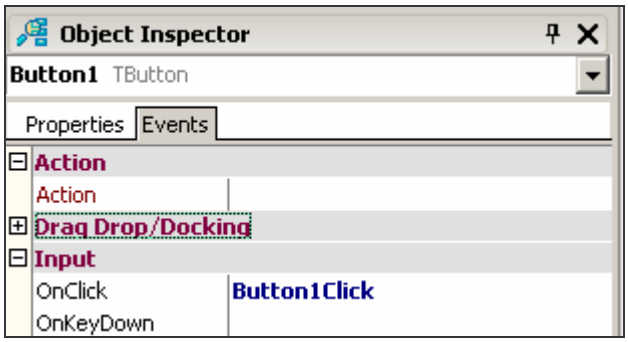
Obr.28. Delphi - zkompilovaná aplikace. Mezivýsledek konečného vzhledu.

Jak se ale dostaneme na další záznam? V databázi jich máme vyplněno několik. Můžeme to udělat zase s použitím komponenty, která umí manipulovat se záznamy v komponentě

MyQuery1. Tuto komponentu najdeme na paletě ***Data Controls*** a jmenuje se ***TDBNavigator***. Pomocí této komponenty se na další záznamy podíváme pomocí šipek. Vložme na formulář tuto komponentu a v ***object inspektoru*** ji napojme na ***MyDataSource1*** jako ostatní komponenty. Zkompilujme aplikaci a můžeme pomocí šipek procházet po záznamech databáze. A nejenom to. Nyní již máme plnou kontrolu nad záznamy databáze. Můžeme vkládat záznamy, upravovat je, mazat atd. To je naprostá fantazie. Naprogramovali jsme aplikaci klient-server. Slovo naprogramovali je přece jenom trochy silné slovo vzhledem k tomu, že jsme nenapsali jediný řádek kódu v Delphi. Přesto máme naprosto funkčního klienta naší databáze. To je mocný nástroj objektového programováni. V podstatě jsme pouze napsali jeden dotaz na databázový stroj. „***Select * from av***“. Nicméně tak úplně zadarmo to nebude. Například obrázky neumíme smazat ani je vkládat. Dále musíme ošetřit některé položky a vynutit si zápis. V databázi máme nastaveno, že je musíme povinně vyplňovat a aplikace by mohla za běhu zhavarovat, protože databáze pošle našemu klientovi zprávu o chybě. Náš klient si s ní nebude vědět rady a tím vyvstane programová výjimka, která může způsobit pád aplikace. Tomu musíme zabránit jednak tím, že dodržíme pravidla, která jsme si v tabulce databáze stanovili a také tím, že naší aplikaci naučíme výjimky řešit [4]. My se tady ošetřováním výjimek zabývat nebudeme, ale musíme se zabývat ošetřením našich komponent tak, aby uživatel zadával to, co měl a nedělal v databázi zbytečný nepořádek. Pokud si potenciálního uživatele nepohlídáme, tak to s naší databází skončí velice špatně. Jsme jenom lidé a ti mají sklony k lajdáctví. Příliš volnosti může být na škodu. Nejprve se pusťme do obrázku. To je asi nejzajímavější.

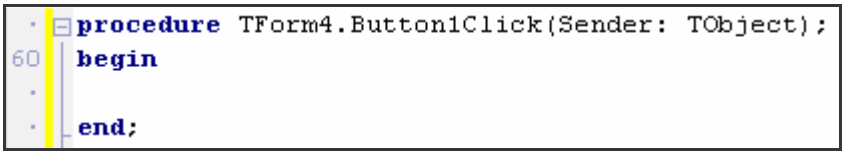
4.3.3 Ukázka programováni delphi 2007

Abychom mohli ovládat obrázkové komponenty, musíme se naučit používat události, na které tyto komponenty umí reagovat. Každá komponenta na něco může reagovat. Například na kliknutí myší nebo pouze na to, že myš přes komponenty přejede atd. Vyberme nyní například tlačítko, které jsme si přejmenovali na „Vložit obrázek“ V ***object inspectoru*** máme vždy dvě záložky (Obr. 29).



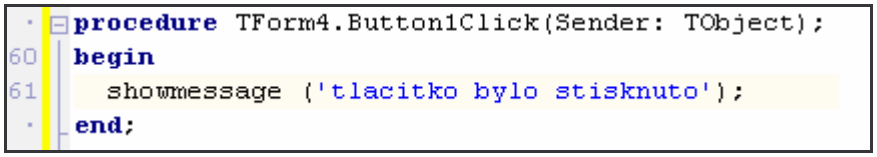
Obr.29. Delphi - Události.

Properties a ***events***. Právě v záložce ***events*** jsou události, na které naše tlačítko umí reagovat. Nás bude zajímat událost ***OnClick. OnClick*** znamená, že když na komponenty klikneme, tak se vykoná program, který je zapsán v této události. Nyní klikneme 2x na událost ***OnClick***. Dostaneme se do prostředí, kde můžeme ručně programovat. V našem případě nám Delphi předpřipravilo metodu ***Click*** (Obr. 30).



Obr.30. Delphi - předpřipravený kód události.

Mezi ***begin*** a ***end*** napíšeme programový kód dle obrázku (Obr. 31).



Obr.31. Delphi - hotový kód události.

Provedeme kompilaci a stiskneme tlačítko. Tlačítko zareaguje na událost ***OnClick*** (kliknuti) a vypíše nám hlášku „Tlačítko bylo stisknuto“. Takto tedy fungují události v Delphi. My budeme potřebovat komplikovanější kód, který nám dokáže vložit obrázek do příslušné komponenty a následně s ním dále manipulovat. Ještě než tak učiníme, je

třeba upozornit na to, že mezi prostředím pro programování a grafickým návrhem aplikace se přepínáme přes záložku úplně dole na stránce. Jsou tam 3 záložky (Obr.33).

Obr.33. Delphi - přepínací panel.

Nyní jsme v oblasti programování ***Code***. Do designu se přepneme přes záložku ***Design***. Přejdeme tedy k oživení komponenty - obrázek. Obě tlačítka naprogramujeme stejně, ale je třeba si dávat pozor, kam a na co se odkazuje programovací kód. Vždy musíme mít na paměti názvy komponent. Vložíme do formuláře další komponentu z palety ***ImageEn*** a to komponentu ***TopenImageEnDialog***. Tato komponenta se postará o výběr obrázku. Přepneme na záložku ***Code*** a vložíme do události ***OnClick*** následující kód. (Obr. 34)

```
procedure TForm4.Button1Click(Sender: TObject);
var
  fs: tFileStream;
  filesize_: int64;
begin
  with openimageendialog1 do
  begin
  openimageendialog1.Execute;
  if openimageendialog1.FileName <> '' then
    begin
    fs := TFileStream.Create(OpenImageEnDialog1.FileName, fmOpenRead + fmShareDenyNone);
    showmessage(fs.FileName + ' jmeno');
    try
    filesize_ := fs.Size;
    finally
    fs.free;
    end;
    if filesize_ < 100000 then    // maximalni velikost obrazku
      begin
        try
        Imageendbvect2.LoadFromFileAll(filename);
        Myquery1.Edit;
        TBlobField(MyQuery1.FieldByName('AV_predni')).LoadFromFile(filename); // ptame se na bunku AV_predni
        except
        showmessage ('Chybové hlášení 2 naší aplikace');
        end;
      end
    else
      showmessage ('Maximalní velikost souboru je 100KB, informaci o velikosti souboru je v náhledu.');
    end;
  end;
end;
```

Obr.34. Delphi - ukázkový kód obslužné události „Button1Click“.

V tomto kódu si zjišťujeme velikost souboru tak, abychom do databáze nedávali zbytečně veliké soubory. Dále si všimněme klauzule ***Try,finally(except).*** Pokud píšeme cokoliv za ***try***, znamená to, že je to v takzvaném chráněném modu. V praxi to znamená, že při náhodné chybě aplikace nebo chybě programátora máme možnost ještě nějak reagovat a

zachránit aplikaci před pádem [4]. Tento kód není zcela dokonalý, ale pro demonstrační účely stačí. V budoucnu bychom například chtěli programovat vícejazyčné aplikace. My však máme hlášení naší aplikace přímo v kódu pomocí ***showmessage***. Lepší by bylo je uložit například do kontejneru ***memo*** a indexovat řádky. Potom bychom místo **showmessage** zavolali ***showmessage*** (memo1.indexX). Memo1 bychom uložili do databáze a volali na základě zvolené jazykové mutace.

Nyní oživíme i druhé tlačítko. Kód bude umístěn v události ***OnClick*** u druhého tlačítka a nesmíme zapomenout, že obrázek ukládáme do druhé komponenty pro obrázek. Takže změníme pouze řádek (Imageendbvect**2**.LoadFromFileAll(filename);) a řádek (TBlobField(MyQuery1.FieldByName('**AV_zadni**')).LoadFromFile(filename);) To je vše. Stejně tak naprogramujeme tlačítka pro smazání obrázku (Obr.35).

```
procedure TForm4.Button2Click(Sender: TObject);
begin
try
Myquery1.Edit;
TBlobField(MyQuery1.FieldByName('AV_predni')).Clear;
except
showmessage ('Chybové hlášení 1 naší aplikace.')
end;
end;
```

Obr.35. Delphi - kód pro vymazání obrázku.

Totéž pro druhý obrázek. Nezapomeňme přepsat ***AV_predni*** na ***AV_zadni***.

Provedeme kompilaci a uvidíme výsledek (Obr.36).

Obr.36. Delphi - výsledná aplikace.

Máme již plně funkční databázi audio video nosičů. Nyní ji můžeme již pouze vylepšovat. Provádět úpravy, které zvýší uživatelský komfort a zpříjemní práci s takovým klientem. V první řadě zbývá přinutit uživatele, aby zadával u nového záznamu minimálně název titulu, druh záznamového media a číslo záznamového media. Musíme tedy zařídit, aby se při jakémkoliv pokusu o uložení záznamu zkontrolovaly tyto tři editační položky. V případě, že jakákoliv z nich nebude vyplněna, tak bude uložení záznamu zamítnuto a uživatel bude upozorněn, že nesplnil nutnou podmínku. Vzhledem k tomu, že s databázovým strojem komunikujeme přes komponentu ***MyQuery1,*** tak zvolíme některou událost (**events**) právě u této komponenty. Hodit se nám bude událost ***BeforePost***. Tato

událost se nám vždy zavolá těsně před tím, než je záznam poslán na uložení. Při nedodržení našich podmínek tuto operaci zastavíme příkazem ***abort***. Na obrázku máme kompletní zdrojový kód této události (Obr. 37). Poklepeme tedy na událost ***BeforePost*** a vložíme níže uvedený kód. Použili jsme primitivní podmínky. Ptáme se, jestli je příslušná komponenta prázdná. Pokud ano, tak jí nastavíme fokus. Přeneseme editační kurzor na příslušnou komponentu. Nesmíme také zapomenout přerušit ukládání záznamů. Zavoláme metodu ***Abort***.

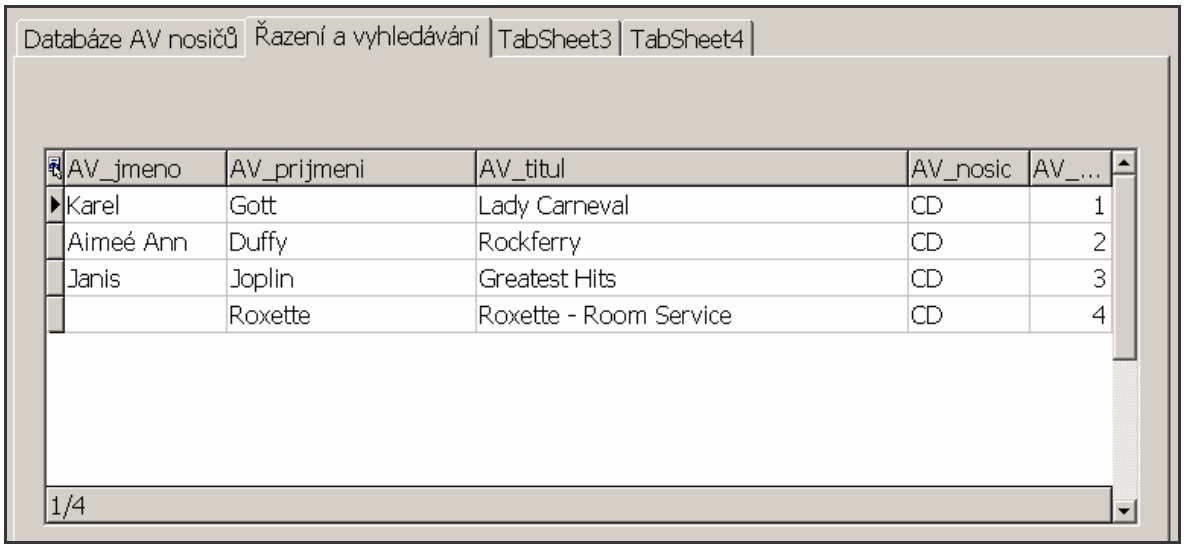
```
procedure TForm4.MyQuery1BeforePost(DataSet: TDataSet);
begin
if DBComboBox1.Text = '' then
  begin
    showmessage('Typ záznamového media musí být uveden');
    DBComboBox1.SetFocus;
    abort;
  end;
if DBEdit2.Text = '' then
  begin
    showmessage('Číslo záznamového media musí být uvedeno');
    DBEdit2.SetFocus;
    abort;
  end;
if DBEdit4.Text = '' then
  begin
    showmessage('Název titulu musí být uveden');
    DBEdit4.SetFocus;
    abort;
  end;

end;
```

Obr.37. Delphi - obsluha vstupních komponent pomocí událostí.

Toto byla poslední, téměř nezbytná, věc pro naši databázi. Nyní již plně funguje. Pro zvýšení komfortu obsluhy chceme ještě umět hledat a řadit záznamy. Řadit a vyhledávat data můžeme ručně tak, že budeme psát zdrojový kód ručně nebo použijeme komponentu, která už všechno vyřeší za nás. Pro ukázku použijeme obě metody. Při ručním postupu budeme modifikovat dotazy SQL. Při použití komponenty se na pozadí bude používat metoda ***locate*** komponenty ***MyQuery1*** na aktuální sadě dat ***recordset***.

4.3.4 Vyhledáváni a řazení záznamů

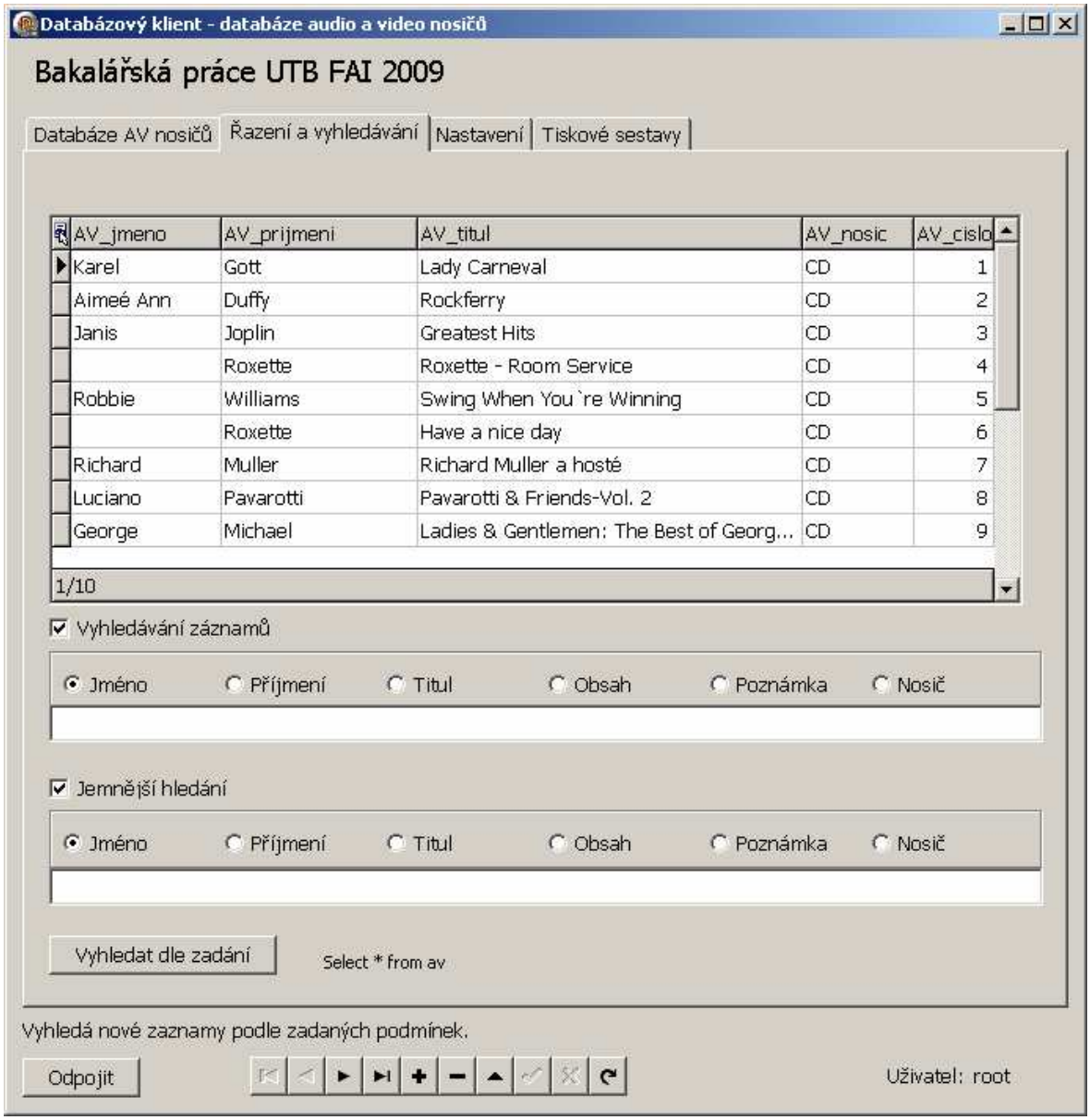
Na našem návrhu klienta se přepneme do druhé záložky a zároveň ji přejmenujeme na řazení a vyhledávání. Poté na ni umístíme komponentu ***TCRDBGrid*** z palety ***Data Controls***. Tato komponenta má nespornou výhodu. To, že jsme ji použili, nám umožňuje beze zbytku splnit požadavky na řazení a vyhledávaní dat. Na následujícím obrázku vidíme tuto komponentu již částečně upravenou (Obr. 38). V podstatě jsme si nechali zobrazovat pouze základní sloupce. Nemá cenu zobrazovat položky typu ***picture,*** protože komponenta je nedokáže zobrazit. Místo obrázku by vypsala pouze datový typ ***blob***. Co se tyká objektu ***memo,*** tak není v podstatě žádný problém se zobrazením, ale nám bude stačit pouze základní zobrazení.



Obr.38. Delphi - Vzhled komponenty DBGrid.

Vzhledem k našim současným znalostem Delphi jistě nebude problém upravit tuto komponentu. Nyní si přeložíme aplikace a vyzkoušíme její možnosti. V levém horním rohu najdeme rozklikávací položku, která umí filtrovat i hledat v záznamech. Pokud budeme chtít naše záznamy řadit například podle jména, tak klikneme na lištu ***AV_jmeno***. Řazení záznamů probíhá stejně jako to známe z operačního systému Windows. Druhá možnost, jak hledat a třídit záznamy, je upravovat dotazy SQL komponenty ***Myquery1***. Do této komponenty jsme napsali dotaz „***Select * from av***“. Vyzkoušejme si nyní přepsat tento dotaz na jiný. Například „***Select * from av where AV_jmeno = `Karel`;***“ Proveďme

kompilaci a podívejme se na výsledek. Naše databáze zobrazuje nyní pouze interpreta Karel Gott. Tedy za předpokladu, že v databázi máme pouze záznamy dle obrázku 38. Zvědavost nám velí, abychom nově naprogramovanou databázi ihned vyplnili dalšími daty. Můžeme také vyhledávat podle části slova. Vyzkoušejme tento příkaz. „***select * from av where AV_prijmeni LIKE '%tt%';***.“ Databáze nám pošle všechny záznamy, které mají ve sloupečku ***AV_titul*** kdekoliv výraz něco_tt_něco. Už tedy víme, jak můžeme získat hledané záznamy. Stačí nám tedy dynamicky přepisovat SQL dotaz v komponentě ***Myquery1*** v metodě SQL. Design stránky by mohl vypadat například takto (Obr.39).



Obr.39. Delphi - výsledný vzhled vyhledávací části.

Nastavit chování a rozmístění jednotlivých komponent je již pouze na uvážení programátora a není nutné zde již dále rozvádět. Snad pouze rutina, která je nutná pro změnu výsledné sady komponent ***Myquery1,*** by mohla dělat problémy. Jedna z možností je na obrázku (Obr. 40).

```
Myquery1.Close;
Myquery1.SQL.Clear;
Myquery1.sql.Add(retezec);
Myquery1.Execute;
```

Obr.40. Delphi - obsluha databázového dotazu.

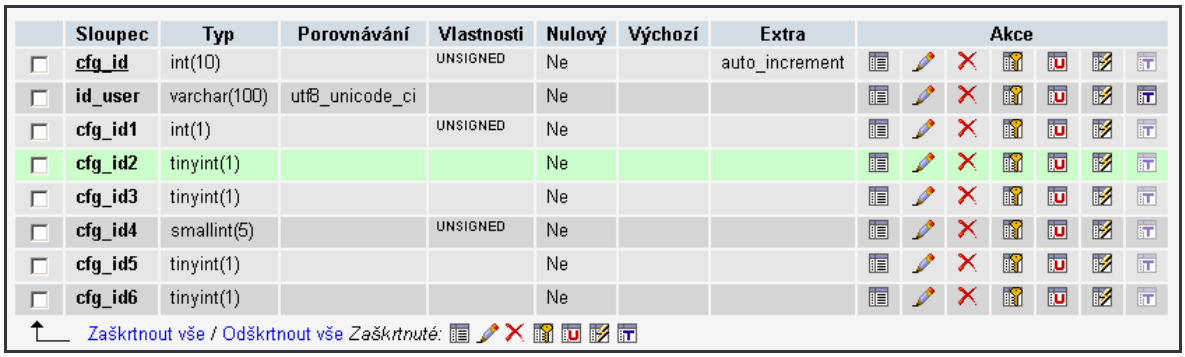
Komponentu nejdříve uzavřeme, potom vymažeme stávající SQL dotaz. Poté přidáme nový SQL dotaz (dle vyhledávacího formuláře) a nakonec necháme příkaz vykonat. Parametr ***retezec*** může být naplněn například takto:

retezec := 'Select * from av WHERE (' + bunka1 + ' LIKE ''%' + edit1.text + '%''); ';

Veškeré zdrojové kódy jsou na přiloženém CD. V podstatě si vytvoříme sadu tlačítek a editačních polí, kam uživatel zadá požadovaná data a aplikace nám z nich poskládá výsledný SQL dotaz. Také bychom mohli poskytnout uživateli možnost přímého zápisu dotazu SQL do nějakého editačního pole a posílat přímo do databáze. Kolik ale běžných uživatelů ví, co je to SQL, nebo databáze vůbec. Lepší bude provádět tyto operace na pozadí a uživatele tím netrápit. Pro ilustraci si na formulář přidáme komponentu ***TLabel*** a do ní budeme zobrazovat výsledný dotaz, který posíláme databázi. Výsledný dotaz jsme si zapsali do textové proměnné „***retezec***“. Stačí tedy, abychom komponentě ***Labelx*** nastavili metodu ***caption*** na tento řetězec. ***Labelx.caption := řetězec;***

4.3.5 Ukázka konfigurace klienta.

Až úplně nakonec našeho doposud úspěšného programování klienta musíme ještě splnit poslední úkol a tím je konfigurace našeho programu tak, aby si konfigurace ukládal do databáze. To je při našich současných znalostech již velice snadné. Budeme muset v databázi vytvořit další tabulku, do které budeme konfigurace ukládat. Tabulku nazveme ***cfg.*** Tabulka je již naimportovaná společně s tabulkou ***av***. Podívejme se v ***PhpMyAdminu*** na její strukturu (Obr. 41).

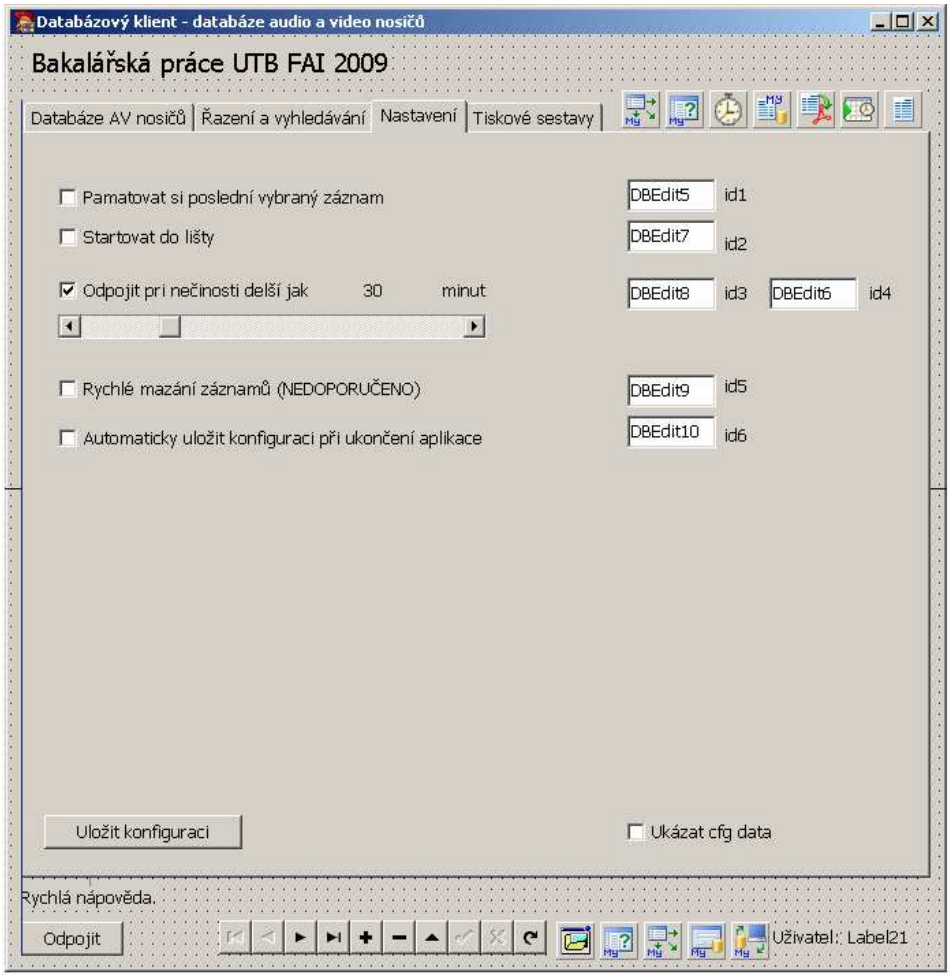


	Sloupec	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Extra	Akce
☐	cfg_id	int(10)		UNSIGNED	Ne		auto_increment	
☐	id_user	varchar(100)	utf8_unicode_ci		Ne			
☐	cfg_id1	int(1)		UNSIGNED	Ne			
☐	cfg_id2	tinyint(1)			Ne			
☐	cfg_id3	tinyint(1)			Ne			
☐	cfg_id4	smallint(5)		UNSIGNED	Ne			
☐	cfg_id5	tinyint(1)			Ne			
☐	cfg_id6	tinyint(1)			Ne			

Zaškrtnout vše / Odškrtnout vše *Zaškrtnuté:*

Obr.41. PhpMyAdmin - struktura tabulky cfg

Použili jsme celkem zbytečně sloupeček ***id_user***. Je to proto, že budeme do budoucna chtít naši databázi umístit na web a dovolit i ostatním lidem naše data sdílet. Každý uživatel si potom bude moci nakonfigurovat své vlastní specifické vlastnosti aplikace. Tím se zabývat nebudeme. Složitost celého návrhu databázového klienta by tím značně vzrostla. Museli bychom se postarat o víceuživatelský přístup ke stejným datům v reálném čase. Použili bychom techniky zamykání tabulek či řádků nebo transakce. Komponenty MyDAC podporují tyto operace. My se ovšem zaměříme pouze na uložení aktuálního záznamu, který si prohlížíme v okamžiku zavření aplikace, plus nějaké další údaje. (Obr. 42) Musíme si uvědomit, že celé prostředí našeho klienta je naprogramováno komponentovým způsobem. Tedy objektově. To znamená, že naše komponenty sdílí vlastnosti svých nadřazených komponent. Svých rodičů. Všechny komponenty mají své vlastnosti, na které můžeme přistupovat i za běhu programu. Je tedy snadné například změnit barvu písma, umístění prvků či jeho název. Stačí nám na to pouze zavolat příslušné vlastnosti těchto komponent. Vybrané vlastnosti uložit do databáze při zavírání aplikace (událost ***BeforeClose***) nebo vynuceně na uživatelův příkaz (událost ***OnClick***). Při otevření klienta si tento následně ověří uživatelé a podle toho načte příslušný obsah databáze. Použijeme událost ***FormActivate***, která se provede v okamžiku aktivace hlavního formuláře naší aplikace. Bude dobré použít další komponentu ***MyQuery2,*** protože se budeme napojovat na jinou tabulku. Toto spojení není nutné udržovat permanentně po celou dobu, kdy náš klient poběží, ale pouze v okamžiku načítání, změny a ukládání naší konfigurace. Nebudeme tedy zbytečně zatěžovat server dalším spojením. Kompletní programový kód je ve zdrojových souborech na přiloženém CD.

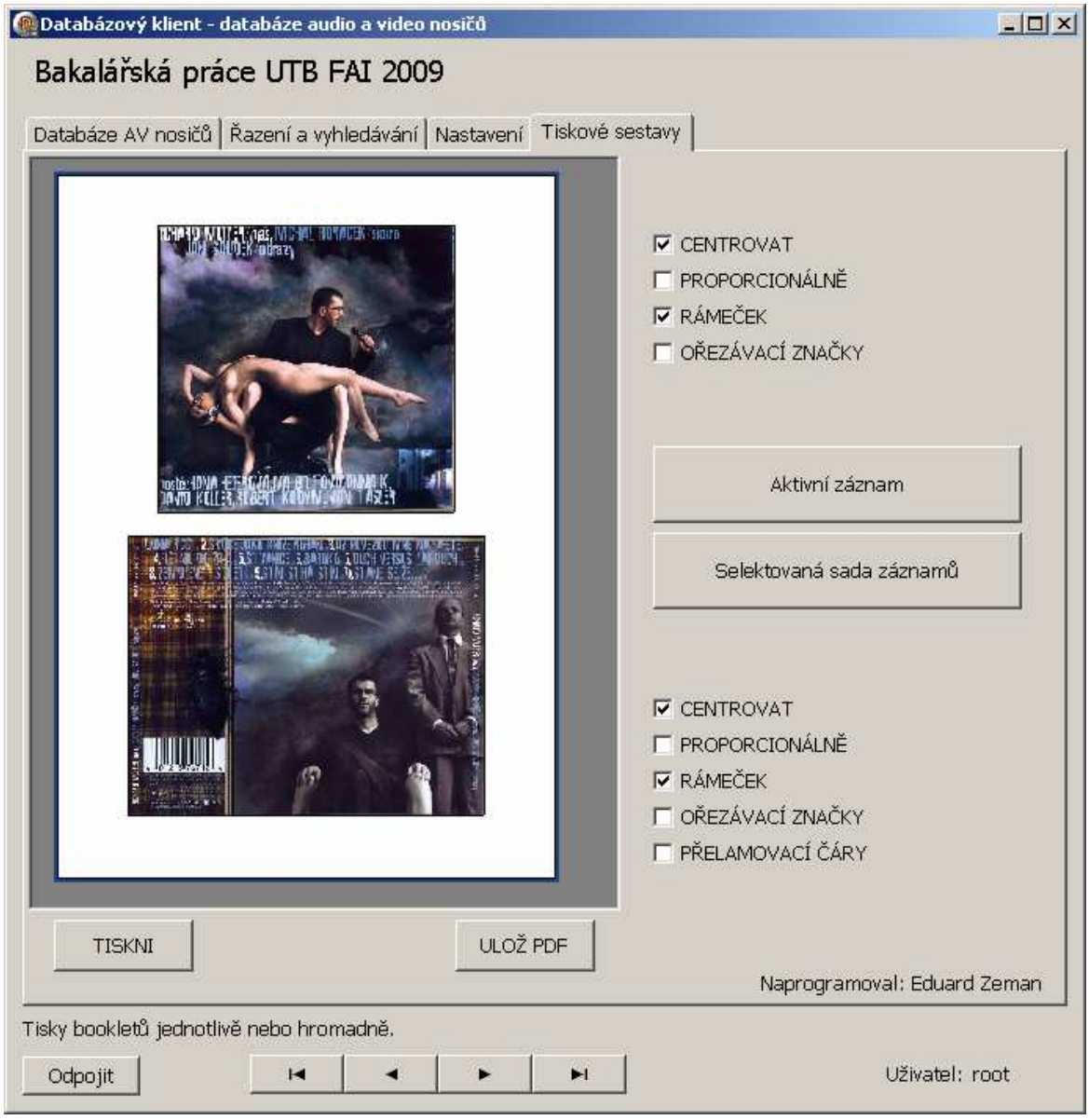


Obr.42. Delphi - výsledná aplikace nastavení konfigurací.

4.3.6 Výstupy z databáze.

Databázový klient by nebyl úplný pokud by neuměl vyhledaná data také vytisknout. Tento klient je naprogramován jako databáze audio a video nosičů a jejich bookletů. Proto ukládáme do databáze obrázky, které jsou vhodné svou kvalitou pro tisk. S tím souvisí i slabina naší databáze. Musíme načítat velké objemy dat. Minimálně v režimu tisku. Tady už narážíme na speciální techniky programování databází. Musíme řešit kompromis mezi kvalitou a rychlostí. Například tak, že obrázky budeme načítat až v okamžiku vlastního tisku. Na jedné straně tím získáme rychlost, na straně druhé snížíme uživatelský komfort. Také se nám může stát, že budeme načítat velké množství záznamů nevhodným výběrem a databáze překročí čas povolený k realizaci jednoho dotazu. Dále můžeme spotřebovat velké množství systémových prostředků. Museli bychom záznamy načítat po limitních hodnotách, které omezí počet načítaných záznamů atd. Můžeme také uživatele přibrzdit v jeho požadavcích tak, že na jeho dotaz nejprve aplikace zareaguje dotazem o počtu

záznamů, které by byly databázi vráceny uživateli a při překročení jejich počtu tento požadavek nepovolit. Možností je celá řada a svou specializací přesahuje rozsah této bakalářské práce. My se vraťme k našemu tisku. Vytisknout můžeme jeden konkrétní záznam, celou databázi nebo vybrané záznamy ze záložky řazení a vyhledávání. Například všechny obaly interpreta „Müller Richard“. Nalezen byl pouze jediný záznam (Obr. 43). Pro výstupní sestavy můžeme použít například komponentu ***QuickReport*** nebo ***FastReport***. ***QuickReport*** je přímo součásti Delphi, ale není implementován. Náš klient používá ***FastReport*** ve verzi 4.0 [9]. Veškerý kód je obsažen na přiloženém CD.



Obr.43. Delphi - výsledná aplikace tiskové sestavy.

5 PŘENOSITELNOST KLIENTA A DATABÁZE

Databázi máme naprogramovanou a teď co s její distribucí.

5.1 Migrace klienta

Použití klienta je snadné. Po spuštění stačí napsat přihlašovací jméno, heslo, server, na kterém je databáze a klienta spustit. Navíc můžeme ponechat na přihlašovacím formuláři pouze jméno a heslo. Ostatní přihlašovací údaje můžeme zapsat do databáze natvrdo a neobtěžovat tím uživatele. Pokud bychom změnili server, tak bychom uživatelům předložili klienta nového, již naprogramovaného, pro změněný server. Provedli bychom upgrade. Prakticky jenom prohodili exe soubor klienta. Přenositelnost klienta je také snadná. Nahrajeme si ho na flash disk a u kteréhokoliv kompatibilního počítače si sedneme, tam ho spustíme. Nemusíme ho nikam ukládat. Nebo ho spustíme přímo z webových stránek.

5.2 Migrace databáze

Migrace databáze je malinko složitější. My už ale víme, jak provést import a export databáze. Takže si databázi exportujeme a na jiném serveru zase neimportujeme. Popřípadě použijeme speciální programy pro archivaci a přenášení databáze.

5.3 Zálohování databáze

Totéž jako v předchozím případě. Zálohování je možné provést exportem databáze a jejím následným uložením na archivační médium. Tedy opět žádný problém. Navíc spousta serverů provádí zálohy automaticky jednou denně. S periodou točení 1 – 2 týdny.

ZÁVĚR

Naprogramovali jsme klienta, který umí obhospodařovat naši domácí sbírku hudební a filmové produkce. Ukázali jsme si jeden z elegantních způsobů, jak vytvořit klienta, který se nemusí instalovat a který se dokáže napojit na databázi odkudkoliv na světě, kde je přístup na síť Internet. Tento klient nepotřebuje žádné další podpůrné soubory pro svoji činnost a umí si i leccos zapamatovat ze svého nastavení. Je tak rychlý, jak rychlá je databáze a přenosový kanál. Je možné ho neustále doplňovat novými funkcemi a potencionálnímu uživateli ho předkládat ke stažení jako jakýkoliv jiný soubor v síti Internet. Tedy na jediný klik myší. Pokud se klient nebude uživateli líbit, tak ho jednoduše smaže. Přesune do koše. Tyto operace zvládne kterýkoliv uživatel, aniž by se musel bát, že svému počítači způsobí nějaké problémy. V teoretické části jsme si popsali prastaré ruční databáze. V souvislosti s databází CD nosičů jsme si položili otázku, jak najít všechny názvy písniček, ve kterých je obsaženo slůvko „love“. Použijme tedy vytvořeného klienta a položme mu tento dotaz. Uvidíme, jak dlouho bude trvat, než nám na dotaz odpoví. Databáze jsou kouzelná a velice užitečná věc. Doufám, že čtenářům této bakalářské práce přinesl alespoň střípek pohledu na dnešní možnosti, které mají samotné databáze a jejich klienti, kteří se na ně mohou připojovat různými způsoby.

Cílem této bakalářské práce bylo naprogramování obslužného programu pro databázi. Program měl splňovat následující podmínky:

- Jediný přímo spustitelný soubor bez nutnosti instalace. Splněno.
- Napojení na vzdálenou databázi bez použití externích souborů. Splněno.
- Uložení konfigurací do databáze. Splněno.

Cíle této bakalářské práce byly tedy **splněny**.

CONCLUSION

We programmed a client who is able to control our home collection of music and film production. We demonstrated one of ellegant ways how to create a client who does not need to be installed and who is able to connect with a database from any on-line place in a world. This client does not need any other support files for his work and is even able to remember his set up. He is as fast as the database and transmitting channel are. It is possible to keep supplying him with new functions and to render him to potential user for download as any other file on the Ineternet. That means with just one mouse click. If a user doesn´t like the client, he can erase him. Shift him into a trash bin. These operations are managable by any user without worries of making some problems in his computer. In a theoretical part we described old manual databases. In context of CD medium database we put question how to find all titles of the songs in which the word "love" is contained. Let us use the created client and lets put him this task. Lets see how long will it take to get his answer. Databases are magic and very useful thing. I hope the readers of this thesis got at least little sight on today´s possibilities for those who have those databases and their clients who can be connected with them by different ways. An achievment of this thesis was to programme a service programme for a database. This programme was supposed to honour these conditions:

- The only directly starting up file without necessity of instalation. Accomplished.
- Connecting with a remote database without usage of external files. Accomplished.
- Saving configuration to the database. Accomplished.

The achievments of the thesis were **all accomplished**.

SEZNAM POUŽITÉ LITERATURY

Monografie:

[1] GILMORE, W. Jason. *VELKÁ KNIHA PHP a MYSQL 5 :Kompendium znalostí pro začátečníky i profesionály.* 2. přeprac. vyd. [s.l.] : Zoner Press, 2005. 864s. ISBN 80-86815-53-6

[2] KOFLER, Michael . *Mistrovství v MySQL 5: Kompletní průvodce webového vývojáře.* 1. vyd. Brno : Computer press, 2007. 808 s. ISBN 978-80-251-1502-2

[3] SVOBODA, Luděk, et al. *1001 tipů a triků pro Delphi : 2. aktualizované vydání.* Brno : Computer Press, 2003. 546 s. ISBN 80-7226-488-5.

[4] SWAN, Tom. *Mistrovství v delphi 4: Kompletní průvodce pro tvorbu aplikací.* Pavel Machek, David Hanousek, Luděk Hořčička. 1. vyd. Praha : Computer Press, 1999. 830 s. Programování. ISBN 80-7226-173-8.

Internetové zdroje:

[5] *CodeGear Home Page* [online]. Embarcadero Technologies, Inc. c1994-2009 [cit. 2009-05-15]. Dostupný z WWW: <http://www.codegear.com/>.

[6] *Data Access Components for MySQL Overview* [online]. c1998-2009 [cit. 2009-04-12]. Dostupný z WWW: <http://devart.com/mydac/>.

[7] *MySQL* [online]. MySQL AB, Sun Microsystems, Inc., c1995-2009 [cit. 2009-04-12]. Dostupný z WWW: <http://www.mysql.com/>.

[8] *Quality Software Components* [online]. 2002-2009 [cit. 2009-04-16]. Dostupný z WWW: <http://www.hi-components.com/ndownloads_imageen.asp>.

[9] *Fast Report, Inc.* [online]. 1998-2009 [cit. 2009-04-20]. Dostupný z WWW: <http://fast-report.com/en/download/fast-report-4-download.html>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RAM	RANDOM ACCESS MEMORY - paměť s libovolným přístupem.
CPU	CENTRAL PROCESSING UNIT - centrální procesor počítače.
GPU	GRAPHICS PROCESSING UNIT - grafický procesor.
HDD	HARD DISC DRIVE - pevný disk.
CD	COMPACT DISC - audio nosič.
MP3	MPEG-1-LAYER 3 - formát ztrátové komprese zvukových souborů.
MPEG-1	MOTION PICTURE EXERTS GROUP - standardy používané na kódování informací audiovizuálního charakteru.
P2P	PEER To PEER - rovný s rovným. Síťová architektura.
Klient-server	Typ síťová architektury.
PHP	PERSONAL HOME PAGE - Hypertextový preprocesor. Jedná se o skriptovací jazyk na straně serveru.
HTML	HYPER TEXT MARKUP LANGUAGE - značkovací jazyk.
OS	OPERATING SYSTEM - operační systém.
SQL	STRUCTURED QUERY LANGUAGE - databázový jazyk.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA PI: CD disk obsahující:

- Bakalářskou práci
- Zdrojové kódy projektu
- Zkompilovanou aplikaci

PŘÍLOHA P I: NÁZEV PŘÍLOHY