UNIVERZITA PARDUBICE
ÚSTAV ELEKTROTECHNIKY A INFORMATIKY

APLIKACE PRO SPRÁVU INTERNETOVÝCH STRÁNEK A PROPOJENÍ FIREMNÍ DATABÁZE S INTERNETOVOU PREZENTACÍ

BAKALÁŘSKÁ PRÁCE

2007 **Lukáš Sirůček**

UNIVERZITA PARDUBICE
ÚSTAV ELEKTROTECHNIKY A INFORMATIKY

APLIKACE PRO SPRÁVU INTERNETOVÝCH STRÁNEK A PROPOJENÍ FIREMNÍ DATABÁZE S INTERNETOVOU PREZENTACÍ

BAKALÁŘSKÁ PRÁCE

AUTOR PRÁCE: Lukáš Sirůček
VEDOUCÍ PRÁCE: RNDr. Josef Rak

2007 **Lukáš Sirůček**

UNIVERSITY OF PARDUBICE
INSTITUTE OF ELECTRICAL ENGINEERING AND INFORMATICS

APPLICATION FOR WEBSITE ADMINISTRATION AND INTERCONNECTION BETWEEN COMPANY DATABASE AND WEBSITE

BACHELOR WORK

AUTHOR: Lukáš Sirůček
SUPERVISOR: RNDr. Josef Rak

2007 **Lukáš Sirůček**

Vysokoškolský ústav: Ústav elektrotechniky a informatiky
Katedra/Ústav: Ústav elektrotechniky a informatiky
Akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Pro: Lukáš Sirůček

Studijní program: Informační technologie
Studijní obor: Informační technologie

Název tématu: Aplikace pro správu internetových stránek a propojení firemní databáze s internetovou prezentací

Zásady pro zpracování:

- Vytvoření designu firemních stránek Nokia Servisu HK
- Rozparsování designu do validních www stránek při užití css, xhtml, php, mysql
- Naprogramování administrační aplikace (v Delphi), která bude pomocí php skriptů, ftp a konfiguračních souborů pracovat s obsahem webových stránek
- Vytvoření webového administračního rozhraní stránek
- Návod k používání aplikace a webového administračního rozhraní
- Možné způsoby propojení firemních dat (databáze oprav mobilních telefonů) s webovou prezentací
- Vytvoření propojení s databází – uživatel www po zadání IMEI telefonu a čísla opravy zjistí aktuální stav jeho zakázky

Seznam odborné literatury:

- Lacko, L.. Web a databáze. ISBN: 80-722-6555-5
- Riordan, R. M. *Vytváříme relační databázové aplikace.* ISBN: 80-722-6360-9
- Kadlec V. *Delphi - Hotová řešení*. ISBN: 80-251-0017-0
- Svoboda L. *1001 tipů a triků pro Delphi.* ISBN: 80-722-6488-5
- Teixeira S., Xavier Pacheco, X. *Mistrovství v Delphi 6*. ISBN: 80-7226-627-6

Rozsah: 30 stran

Vedoucí práce: RNDr. Josef Rak
Vedoucí katedry (ústavu): prof. Ing. Pavel Bezoušek, CSc.

Datum zadání práce: 31. 10. 2006
Termín odevzdání práce: 12. 5. 2007

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 14. 05. 2007

Lukáš Sirůček
(vlastnoruční podpis)

ABSTRAKT

Tato práce se zabývá problematikou softwarového řešení pro malé společnosti, které si přejí jednoduše a pohodlně měnit obsah své internetové prezentace pomocí aplikace pro prostředí operačního systému Windows. Vzorová aplikace, pro tuto bakalářskou práci, je naprogramována ve vývojovém prostředí Delphi 2005. V této práci se zabývám i možnostmi publikace některých firemních dat z interního softwarového řešení, vytvořeného na zakázku jinou společností. Tato práce se konkrétně dotýká autorizovaného Nokia servisu v Hradci Králové a její součástí jsou i internetové stránky s katalogem produktů a zobrazením stavu konkrétní opravy pro zákazníka.

Obsah

Seznam obrázků

Seznam zkratek a akronymů

ASP ..... Active Server Page
CGI ..... Common Gateway Interface
CLX ..... Component Library for Cross Platform
CSS ..... Cascading Style Sheets
DLL ..... Dynamic-Link Library
DSN ..... Database Source Name
FTP ..... File Transfer Protocol
HDD ..... Hard Disk
HTML ..... HyperText Markup Language
HTTP ..... Hypertext Transfer Protocol
IMEI ..... International Mobile Equipment Identity
Indy ..... Internet Direct
IP ..... Internet Protocol
ODBC ..... Open Database Connectivity
OS ..... Operační Systém
PDF ..... Portable Document Format
PHP ..... Hypertext Preprocessor
POP3 ..... Post Office Protocol
RAD ..... Rapid Application Development
RFC ..... Request for Comments
SFTP ..... Secure File Transfer Protocol
SMTP ..... Simple Mail Transfer Protocol
SQL ..... Structured Query Language
VCL ..... Visual Component Library
WWW ..... World Wide Web
XHTML ..... Extensible HyperText Markup Language
XML ..... Extensible Markup Language

1. Úvod

Dnes je Internet nedílnou součástí všedního života a samozřejmostí pro rozvoj každé firmy. Proto, aby byl Internet k užitku, musí nám poskytovat stále aktuální údaje. Každá webová stránka by měla mít snadný způsob aktualizace jejího obsahu. Existuje několik možností, jak toho dosáhnout. Pokud jsme zruční v psaní jazyku *html*, postačí nám například pouhý *FTP klient* a poznámkový blok, ve kterém si změníme vše, co potřebujeme a poté to nakopírujeme na server, aby si každý uživatel Internetu mohl zobrazit aktuální data. Další, a dnes nejvíce používanou, možností je aktualizace obsahu přes tzv. redakční systém, nebo jiný (jednodušší či specializovaný) aktualizační systém, dostupný přímo přes webové rozhraní. Toto řešení je velice elegantní a efektivní. Ale avšak pro náš předmět zkoumání, jsme si vybrali aktualizaci a správu internetový stránek přes offline aplikaci, dostupnou z uživatelského počítače.

Hlavním cílem této práce je vytvořit demonstrativní aplikaci, kterou bude možné měnit strukturu jednoho menu na www stránce, a k jednotlivým položkám menu přiřadit libovolný text, jenž se bude zobrazovat v internetové prezentaci. Aplikace bude pojata jako do budoucna rozšiřitelná o další funkce.

Dalším cílem je návrh vytvoření propojení konkrétních vnitřních firemních dat s internetovou prezentací pro autorizovaný Nokia servis v Hradci Králové.

2. Základní popis a možnosti řešení

2.1. Proč aplikaci místo webového rozhraní

Aplikace, běžící pod operačním systémem, nám dává více možností než webové rozhraní. Podívejme se například na jakýkoliv moderní textový nástroj, pro vytváření dokumentů. Co vše můžeme dělat s textem a obrázky. Pokud to vezmeme teoreticky, bylo by možné implementovat do aktualizační aplikace veškeré prvky textového nástroje a dosáhnout tím pohodlného a maximálně komfortního ovládání, na které je většina nás již zvyklá.

Ne každý si na první pohled oblíbí webové rozhraní nějakého redakčního systému. Webové rozhraní v některých směrech omezuje funkčnost a možnosti klienta. Metody známé z desktopových aplikací, jako je například vykreslování na obrazovku či obecné techniky jako *drag and drop*, nejsou standardními technologiemi prohlížečů podporovány. Uživatele to může odradit, protože není zvyklý na ovládání „aplikace“ přes internetový prohlížeč. Pro tu část veřejnosti, která pracuje s počítačem pouze jako s psacím strojem, je mnohem známější pracovní prostředí podobné oknům a aplikacím ve Windows, kde si ikonkou na ploše spustí program a následně v jeho okně začne se svou prací, kterou si před ukončením aplikace uloží.

2.2. Nevýhody offline aplikace

Hlavní a zásadní nevýhodou aplikací je jejich přenositelnost. Pokud budeme webové stránky spravovat internetovou on-line aplikací, stačí nám pro aktualizaci (úpravu) stránek pouze internetový prohlížeč a jakýkoliv počítač připojený k internetu. Když však sáhneme po řešení offline aplikací, již stránky neaktualizujeme kdekoliv. Budeme si muset aplikaci vzít s sebou nebo si ji napřed stáhnout z internetu a poté ji nainstalovat nebo nastavit. Pokud je aplikace naprogramována pouze pro jedno prostředí operačního systému, na jiném operačním systému nám nebude fungovat vůbec, nebo jen omezeně.

Druhou velikou nevýhodou je práce s databází na serveru, kde jsou www stránky umístěny. V dnešní době nám téměř žádný poskytovatel webhostingu neumožní připojení na databázi z jiného počítače než z toho, na kterém jsou stránky přímo umístěny. Je to z bezpečnostního hlediska, kdy firewall na serveru zahazuje každý neoprávněný přístup do databáze, nejen pro zamezení přehlcení databázového serveru. V případě, že bychom s naší aplikací chtěli pracovat i s databází, museli bychom vlastnit tzv. *dedikovaný server* nebo případně *virtuální server*, což je řešení podstatně dražší než obyčejný webhosting. Proto se s aplikací omezíme jen na práci se soubory uloženými na serveru, kde jsou stránky umístěny, přes protokol *FTP*.

2.3. Možnosti řešení

2.3.1. Přímé propojení s databází

V ideálním případě by aplikace komunikovala přímo s databází na serveru. Po připojení, na vzdálenou databázi, by si prostřednictvím *SQL* dotazů zjistila potřebné nastavení, aktuální strukturu textů a webových stránek. Poté by vše zobrazila v uživatelsky přijatelném prostředí a reagovala by na podněty uživatele. Data by na server mohla odesílat buď přímo, po každé změně, nebo jen na požadavek uživatele, hromadně. Vzhledem k problematice připojení se na databázový server z jiného počítače, toto řešení ztrácí větší význam.

2.3.2. Nepřímé propojení s databází

Tato možnost je využívána například v prostředí vývojového nástroje *Macromedia Flash*[1], kde pro připojení do databáze potřebujeme „prostředníka“ (například *php* skript), který odešle SQL dotaz databázi, ta mu odpoví a pošle zpět výsledek. Prostředník výsledek zpracuje a zobrazí, dle požadavků aplikace, zformátovaný výstup, který si aplikace přečte. Musíme kontrolovat správný běh propojovacího skriptu a dodržovat formátování výsledků, získaných z databáze, aby nám jej mohla aktualizační aplikace správně číst a používat. Vždy musíme čekat na vykonání propojovacího skriptu, a až poté získaný výsledek načíst k dalšímu zpracování.

[1] dostupné z: http://www.adobe.com/products/flash/

Tomuto modelu se říká třívrstvý model. V té nejběžnější formě je webový prohlížeč (nebo aplikace) první vrstvou (prezentační), nástroje pro dynamické generování stránek (např. CGI, PHP, javové servlety nebo ASP) je vrstvou střední (logickou) a databáze je vrstvou třetí (datovou). Webový prohlížeč posílá požadavky střední vrstvě, která je obsluhuje prostřednictvím dotazů do databáze (resp. její aktualizací) a generováním uživatelského rozhraní. Ne vždy to může být jednoduché na naprogramování. Např. *Macromedia Flash* ve svém programovacím jazyku (*Action script*) již obsahuje metody, přímo související s tímto problémem, protože jako prvek *ActiveX* z bezpečnostních důvodů neumožňuje přistupovat k databázi přímo a ani měnit nebo zapisovat soubory umístěné na serveru.

2.3.3. Editace a vytváření souborů, FTP

Poslední možností je práce přímo se soubory uloženými na FTP. Aktualizační aplikace si po připojení na FTP server, kde jsou uloženy www stránky, stáhne všechny soubory s texty a nastaveními, které je možné editovat. Po dokončení editace, na lokálním disku, se všechny soubory uloží zpět na FTP a přepíšou se jimi soubory původní. Pro zvýšení bezpečnosti přenosu dat se dá použít protokol *SFTP* (Secure FTP).

Toto řešení lze kombinovat s lokální databází, konstruovanou pro aktualizační (publikační) aplikaci. Na lokální databázi můžeme mít např. celý internetový obchod nebo rozsáhlou internetovou prezentaci s fotogalerií, katalogem, apod. V aplikaci provedeme aktualizaci a editaci textů (případně fotografií) a poté vše vyexportujeme do html souborů a odešleme na FTP.

Jinou možností je editace textů a jiných prvků www stránek, které jsou pak na serveru dynamicky načítány do předem vytvořené šablony v html. Můžeme použít jakýkoliv skriptovací jazyk (např. php, asp, apod.), který si načte požadované soubory, které jsme po editaci nahrály na FTP, a zobrazí je formátované na webu. Tímto řešením se budeme dále zabývat podrobněji.

3. Návrh konkrétního modelového řešení

3.1. Použité technologie

Pro lokální aplikaci, uloženou na HDD počítače, ze kterého budeme chtít aktualizovat stránky, je použita technologie *VCL*, z vývojového prostředí *Delphi*. K této technologii se dále vrátíme podrobněji, a vysvětlíme ji v některé z následujících kapitol. Spustitelná aplikace je naprogramována a přeložena pro operační systém Microsoft Windows.

Serverová část, kde jsou uloženy stránky, umožňuje spouštění skriptů psaných v *php*. Veškeré dynamické části www stránek, jsou psány v tomto populárním jazyku. Statické prvky stránek jsou naprogramovány ve standardu *xhtml* a používají kaskádové styly (*css*), linkované z externího souboru (vysvětlím v kapitole **Vytvoření validní prezentace**, kapitola 7). Pro zpestření vzhledu jsou použity interaktivní prvky *Adobe Flash* (zanimování *headeru* stránek) a javascript (postupné zobrazování krátkých textových novinek).

Konfigurační soubory pro aplikaci, a datové struktury jsou uloženy (exportovány) do *xml*. **XML** je univerzálním řešením pro uložení strukturovaných dat, použitelným v mnoha programovacích jazycích, byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů.

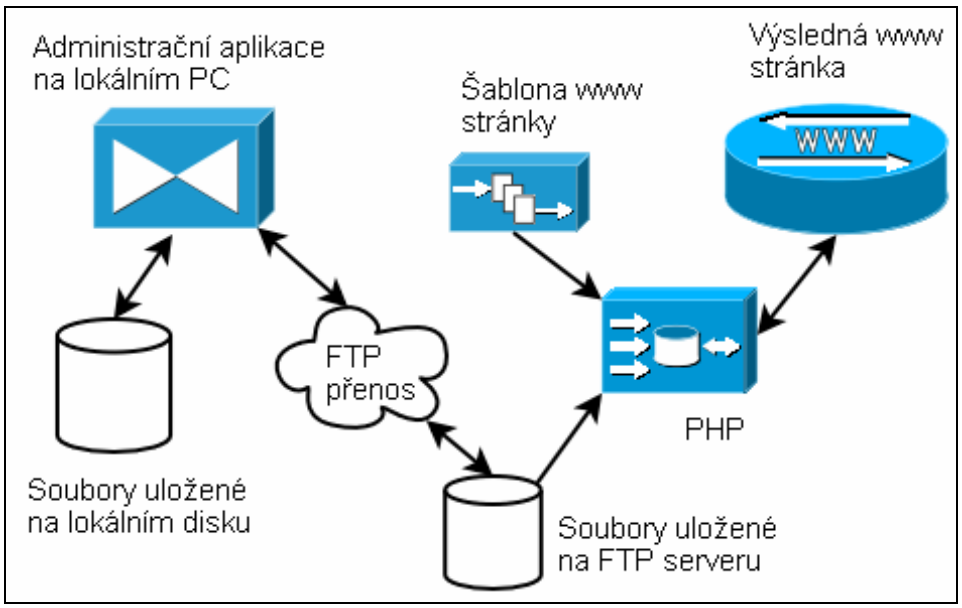
Pro komunikaci mezi aplikací a serverem je použito protokolu *FTP*, který poslouží ke stahování souborů ze serveru a k jejich ukládání zpět.

3.2. Aktualizační model

Aplikace i php skripty užívají stejné adresářové a souborové struktury. Předem si určíme, které soubory co znamenají a co je v kterých adresářích uloženo. Toto bude nadále společným pro serverovou část (php skript) a lokální část (aplikace pro Windows).

Na požadavek uživatele (požadavkem může být i spuštění aplikace, nebo jen vybrání příslušné akce z menu programu), se aplikace připojí na předem definovaný FTP server, kde jsou uloženy stránky. Zjistí, zdali existují příslušné soubory a adresáře, a soubory si stáhne na lokální disk, do stejné adresářové struktury jako je na serveru, relativně k umístění programu. Po provedení úprav a změn v editovatelných souborech, popřípadě po vytvoření nových souborů s texty, dáme aplikaci pokyn k přenesení všech souborů zpět na server. Před přenesením souborů na server je potřeba smazat všechny soubory s texty, uložené na FTP serveru v aktualizačním adresáři, aby nám tam nezůstávaly tzv. plané soubory, které po editaci a uploadování souborů nových již nikam nepatří, jelikož jsme je při editaci např. záměrně smazali.

Php skript, vložený ve www prezentaci na serveru, při každém zobrazení stránek v prohlížeči, projde známé soubory v určených aktualizačních adresářích, přečte jejich obsah a vloží jej do xhtml šablony stránek. V prohlížeči se nám tím objeví aktuální obsah, který jsme editovali lokální offline aplikací.



Obr. 1: Aktualizační model

3.3. Vymezení společných pravidel

Pro relativně bezproblémovou spolupráci mezi programem a php skripty ve www prezentaci, si musíme vymezit několik základních pravidel:

1. **Co všechno můžeme měnit a editovat**
   - Pro náš modelový příklad budeme měnit pouze jedno doplňkové menu v prezentaci, a to kompletně s celou jeho strukturou
   - Ke každé položce, z doplňkového menu, můžeme přiřadit libovolný text, který se poté bude vkládat do šablony na internetové stránce
2. **V jakém formátu budeme ukládat data**
   - Pro uchování dat v *menu* použijeme formát *xml*, kde každá položka menu ponese svůj název, popis a relativní cestu k souboru s textem
   - Vytvořené texty budou uloženy jako běžný textový soubor a budou moci obsahovat i html tagy
   - Nastavení programu ukládáme opět do *xml* a obsahuje hlavně nastavení nutná pro připojení k FTP serveru, jako je jméno serveru, login, heslo a pracovní adresář
3. **Kde budou uloženy editovatelné a konfigurační soubory**
   - Pracovní adresář nazveme „**web**“ a bude obsahovat:
     - soubor „**nastaveni.xml**“ – základní nastavení programu
     - soubor „**menu.xml**“ –*menu* stránek
     - adresář „**text**“ – obsahuje vytvořené texty nalinkované k menu. Textové soubory jsou uloženy v následujícím tvaru: **jmenoPolozkyMenu_datumCasVytvoreni.txt**

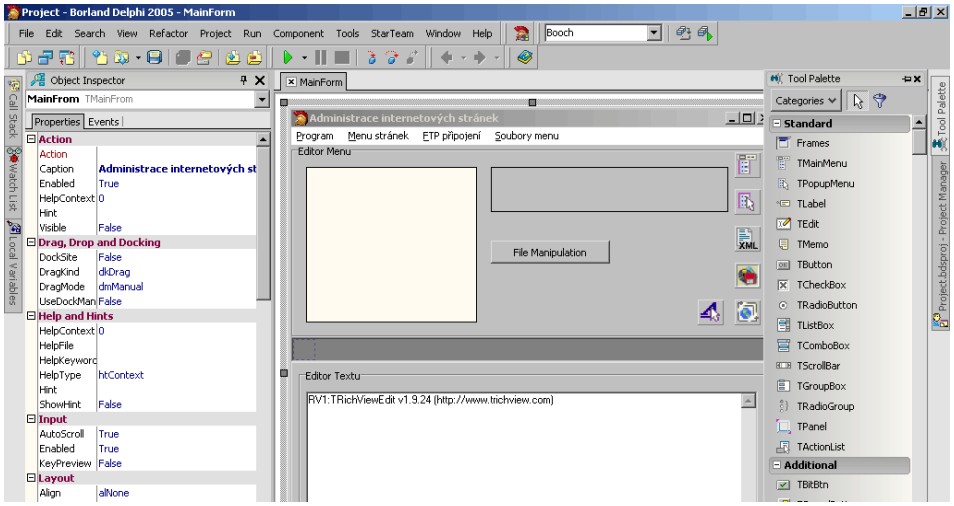
4. Vývojové prostředí aplikace

Předtím, než se dostaneme k hlavní části programování a konstruování programu, je třeba si říci něco o vývojovém prostředí *Borland Delphi* a programovacím jazyku *Object Pascal* (jazyk Delphi).

4.1. Borland Delphi

Borland Delphi[2] je vývojové prostředí, vytvořené společností Borland. V době psaní této práce byla k dispozici již desátá verze tohoto produktu, nazvaná Borland Delphi 2006. Tato verze podporuje programovací jazyk Object Pascal a C++ pro platformu Microsoft Windows, a navíc jazyk C# pro platformu Microsoft.NET[3]. Ale avšak ukázková aplikace, vytvořená jako součást této bakalářské práce, je naprogramována v prostředí Delphi 2005 (Borland® Developer Studio for Windows™ Version 9), s použitím *VCL* (*Visual Component Library*), pro platformu Win32.

Delphi je velice oblíbené pro vývoj desktopových a databázových aplikací. A to nejen pro jeho „jednoduchost“ a hbitost při designování aplikací, kdy máme k dispozici širokou škálu tzv. komponent, které umístíme pouhým přetažením do hlavního okna naší aplikace, ale také pro snadnost tvoření jednotlivých akcí, reagujících na vstup uživatele hotového programu.



Obr. 2: Náhled do vývojového prostředí Borland Delphi

[2] dostupné z: http://www.codegear.com/products/delphi/win32
[3] dostupné z: http://www.microsoft.com/cze/net/basics/

4.1.1. Historie

Delphi bylo první známé vývojové prostředí založené na tzv. *RAD* (*Rapid application development*) pro platformu Microsoft Windows. První verze Delphi byla vydána v roce 1995, a to pro 16bitová Windows 3.1. Delphi 2 přišly o rok později, ale to již pro 32bitová Windows 95. Tyto verze podporovaly pouze jazyk *Object Pascal*. Až o pár let později byla vydána varianta Borland C++ Builder, podporující jazyk *C++*. Hlavním vývojářem byl Anders Hejlsberg, který se podílel i na vývoji jazyku *Turbo Pascal*. Ten se v roce 1996 přesunul do Microsoftu, kde začal pracovat na jazyku *Visual J++* a později se stal šéfem vývoje *C#* a zasloužil se o vznik platformy *Microsoft.NET Framework*.

V roce 2001 se objevila mutace pro Linux, nazvaná *Kylix*. V době vydání byla první verze nestabilní a téměř nepoužitelná. *Kylix* měl smysl až od verze 3. S příchodem Kylixu přišla i knihovna komponent *CLX*, podporující jak platformu Windows, tak i Linux. Do Delphi byla tato podpora přidána ve verzi 6.

Od verze 8 je v Delphi podporováno *.NET* a tím i přichází konkurence pro Microsoft Visual Studio .NET. Delphi 8 podporovalo pouze tvorbu aplikací .NET. Podpora vývoje aplikací v klasickém Delphi Win32 je zahrnuta až v další verzi – Delphi 2005, jenž obsahuje v jednom vývojovém prostředí programovací jazyky Object Pascal (pro tvorbu aplikací na platformě Win32) i C# pro aplikace založené na .NET.

4.1.2. Hlavní výhody

Delphi je známo s těmito zásadními výhodami:

- Rapid application development (RAD)
- Silný objektově orientovaný programovací jazyk
- Veliká komunita vývojářů na internetu
- Možnost vytváření DLL knihoven
- Mnoho *VCL* komponet (často i se zdrojovými kódy)
- Propracovaná spolupráce s databází
- Rychlý optimalizující kompilátor
- Skvělá kompatibilita zdrojových kódů mezi verzemi Delphi

4.1.3. Nevýhody

- Limitovaná meziplatformní schopnost použití
- Přístup k systémovým funkcím vyžaduje načtení hlavičkových souborů, přeložených do Pascalu. To může způsobit zpomalení při překladu zdrojového kódu
- Méně vydaných knih, zabývajících se Delphi, oproti ostatním populárním jazykům jako je např. C++

4.1.4. Některé známé aplikace vytvořené v Delphi

Několik známých aplikací, které byly vyvinuty v Delphi:

- Databázové programy MySQL (Administrator, Query Browser)
- Skype pro internetovou komunikaci
- FL Studio – tvorba hudby
- Macromedia HomeSite – vytváření webových stránek
- Total Commander – souborový manažer

4.2. Rapid application development (RAD)

Rapid application development (RAD), neboli *rychlý vývoj aplikace*, je technologie, kterou začal popisovat James Martin v průběhu 80. let minulého století. Veškeré své poznatky shrnul a publikoval v knize, vydané v roce 1991. Je to metodologie popisující vytváření programů na základě konstrukčních prototypů, s výpomocí vývojového prostředí. Zjednodušeně to znamená, že spoustu programového kódu za nás napíše počítač sám, což se dá dobře využít při designu aplikací, kdy designér aplikace nemusí být programátor.

4.3. Visual Component Library (VCL)

VCL je technologie založená na vizuálně komponentovém frameworku pro vytváření aplikací na platformě Microsoft Windows, vyvinutá firmou Borland a používaná v jejich vývojových prostředích Broland Delphi a Borland C++ Builder. Tato knihovna je napsána v jazyce Object Pascal a teší se z popularity, získané díky použití na základě metodologie RAD.

VCL je objektově orientovaná knihovna zapsaná v Object-Pascalu, která poskytuje hierarchii tříd odpovídající prostředí MS-Windows. V rámci VCL jsou formou tříd popsány jednotlivé standardní prvky používané ve Windows (*Button*, *Menu*, *CheckBox*, *EditBox* atd.) společně s jejich vizuální reprezentací pomocí tzv. *komponent*.

VCL umožňuje podstatným způsobem zjednodušit vývoj aplikací pro Windows a dovoluje programátorovi provádět mnoho standardních operací automaticky a vyhnout se tak v programech dlouhým zápisům, vytvářet program částečně vizuální formou umístěním patřičné komponenty do nově vznikajícího programu.

VCL formuláře a komponenty používají objektovou hierarchii a dědění, kdy jsou všechny objekty přímo nebo nepřímo poděděny od „supertřídy" *TObject*. Toto řešní je nutné, neboť Delphi nepodporuje vícenásobné dědění, narozdíl od jazyku C++. Takto to bylo řešeno v jazyce *Smalltalk* – první opravdu objektově orientovaný programovací jazyk.

4.4. Object Pascal

Object Pascal je objektově orientovaný následník jazyku *Pascal*, známém jako základní programovací jazyk firmy Borland. Object Pascal byl firmou Borland později nazýván jako programovací jazyk Delphi, což vyjadřuje jeho hlavní požívání ve stejnojmenném vývojovém prostředí.

V roce 1985 byl vytvořen Niklausem Wirthem a Larry Teslerem pro společnost Apple Computer a rozšiřoval tehdejší jazyk Pascal o objektový přístup. Apple Computer jej přestala podporovat v roce 1994, když přešla z architektury procesorů Motorola na IBM PowerPC.

V prostředí Delphi se pro psaní objektů, v programovém kódu, používá klíčové slovo *class*, stejně jako např. v C++.

Příklad třídy napsané v Object Pascal (Delphi):

```
type TMojeTrida = class
        private
                atribut1: integer
        end;
```

5. Návrh a vytvoření aplikace

Administrační aplikace je navržena s ohledem na uživatelskou přívětivost a jednoduchou funkčnost. Dále byla snaha o co nejkratší a dobře členěný programový kód, jenž by přidal na jeho přehlednosti a čitelnosti. Jak jsem se již zmínil v kapitole **3**, aplikace bude pracovat se soubory a FTP přenosem, nikoliv s databází.

5.1. První vize o vylepšení aplikace – DLL

Vzhledem k tomu, že nejsnazší vytvoření aplikace v Delphi spočívá v tom, že si vytvoříme nový projekt, kde si umístíme hlavní okno programu (formulář VCL), na tento formuláři si přetáhneme jednotlivé komponenty z knihovny VCL, jim přiřadíme *events* (akce) a nakonec po úspěšné kompilaci programu nám vznikne jen jeden spustitelný soubor s koncovkou *exe*, jsem se pokoušel přijít na jiné řešení, které by na první pohled vypadalo více profesionálně.

Když si nainstalujete jakýkoliv program od známých vývojářských subjektů, většinou najdete ve složce, kde je program nainstalován, nejenom spustitelnou aplikaci, ale i další soubory, známé jako *DLL* (*Dynamic Linked Library*). DLL je samostatná aplikace. Z toho vyplívá základní rozdíl – *unita* je neoddělitelně spojena s programem, kdežto DLL jsou samostatné a mohou být užívány více aplikacemi. Chceme-li udělat novou verzi unity, musíme potom celý program znovu překompilovat, ale používáme-li DLL, stačí pouze jednu knihovnu nahradit novou verzí. Výhoda DLL knihoven je v tom, že můžeme načítat procedury a funkce do paměti v momentě kdy je potřebujeme a pak je opět z paměti vymažeme. Procedury jsou načítány z DLL hned po spuštění programu.

Nevýhodou DLL je, že mají pomalejší přístup ke svým procedurám. Oproti tomu je výhodou, že je jedno, v jakém programovacím jazyce jsou napsané, proto nám v Delphi budou fungovat i DLL napsané v C++ nebo ve Visual Basicu.

Moje aplikace nakonec obsahuje pouze jen dvě mnou naprogramované unity neobsahující příliš mnoho řádků programového kódu, tak jsem usoudil, že je zbytečné a zbytečně složité je přepisovat a kompilovat do DLL, jelikož by to na výsledné velikosti programu moc neubralo a použitelnost mých DLL knihoven by skončila u jen mé aplikace, přičemž jejich novější verze asi nepřipadají v úvahu – DLL knihovny by obsahovaly jen třídu, nesoucí nastavení programu.

5.2. *Vizuální koncepce*

5.2.1. Hlavní okno aplikace

Při spuštění aplikace se uživateli objeví okno dané velikosti, plně postačující pro ovládací prvky a pro práci s aplikací. Okno programu se spustí centrované vůči rozměrům pracovní plochy. Jeho velikost bude možné pouze libovolně zvětšovat, zmenšení okna je povoleno jen do minimální velikosti, abychom nezakryly ovládací a zobrazovací prvky formuláře (hlavního okna). Okno aplikace ponese název „Administrace internetových stránek“ a bude jej možné standardními tlačítky minimalizovat, maximalizovat a zavřít. Při přetažení okna k okraji obrazovky se aplikace automaticky „přichytne“ k okraji (vlastnost *ScreenSnap*). Barvy formuláře a jeho prvků jsou standardně převzaty z nastavení Windows.

5.2.2. Menu aplikace

Hlavní menu (*main menu*) s položkami nutnými pro ovládání aplikace je klasicky umístěno v horní části okna, pod jeho titulkem. Bude obsahovat položky:

- Program (Ukončení programu, O programu)
- Menu stránek (Načíst již uložené menu ze souboru, Vytvořit nové menu, Uložit vytvořené nebo upravené menu do souboru)
- FTP připojení (Nastavení FTP připojení, Test funkčnosti připojení, Uložení nastavení do souboru, Načtení již uloženého nastavení ze souboru)
- Soubory menu (Jejich uložení na FTP a stažení z FTP serveru)

Všechny položky je možné ovládat pomocí akceleračních klávesových zkratek (Alt + písmeno) a jejich název bude zároveň zobrazen ve stavovém řádku aplikace.

5.2.3. Stavový řádek

Stavový řádek nám bude zobrazovat doplňkové informace k ovládacím prvkům programu. Bude fungovat jako malá nápověda, a v pozdějších verzích programu, bude zobrazovat i stav přenosu souborů na a z FTP serveru. Zobrazované rozšířené informace ovládacích prvků, jsou převzaty z vlastnosti *Hint* (neboli textového rozšířeného popisku, někdy zobrazovaného po najetí kurzoru myši, např. na tlačítko) u jednotlivých komponentů.

5.2.4. Pracovní plocha

Pracovní plochu aplikace rozdělíme, pro přehlednost a další podrobnější popis, do třech základních částí:

- Horní část pro práci s *menu internetových stránek* a pro přiřazování a otevírání souborů jednotlivých položek menu.
- Střední část bude, v této verzi programu, zobrazovat nápovědu a chybové hlášky při práci s programem. V budoucnu může být tento střední panel použit pro ovládací prvky na práci s textem, jelikož je umístěn nad textovým polem.
- Dolní část programu obsahuje textové pole, do kterého je načten text, patřící k položce menu, nebo v tomto poli vytváříme nový text, přiřaditelný k položce menu.

Jednotlivé části pracovní plochy jsou ve vývojovém prostředí tvořeny komponentou *Panel*, která nám umožňuje program libovolně členit do několika částí. Po spuštění programu je zobrazen pouze středový panel, zarovnaný v horní části plochy. Jakmile vytvoříme (nebo načteme) nové menu, nad středovým panelem se nám zobrazí panel pro práci s menu a středový panel se tím automaticky posune pod něj. Panel pro práci s menu a středový panel má danou nezměnitelnou výšku. Šířka všech panelů je automatická v velikosti okna aplikace. Dolní panel, pro psaní

textu, se zobrazí při otevření nebo vytvoření nového textu k položce menu. Jeho velikost se automaticky dorovnává do aktuální velikosti okna aplikace.

5.2.5. Horní panel pro práci s menu (Editor menu)

Při spuštění aplikace má tento panel nataven vlastnost *Visible* na hodnotu *false*, což znamená, že není zobrazen. Jakmile otevřeme soubor s již vytvořeným menu, nebo vytvoříme menu nové, vlastnost *Visible* se změní na *true* a panel se, i s ovládacími prvky, zobrazí. Středový panel se tím posune pod horní panel.

Horní panel obsahuje následující ovládací prvky:

- Komponentu TTreeView. V ní jsou zobrazeny jednotlivé položky a jejich podpoložky menu, ve stromové struktuře. Kliknutím pravého tlačítka, na tuto komponentu, se zobrazí kontextové (*popup* menu) s výběrem pro vytvoření nové položky (nebo podpoložky), případně s možnostmi pro označenou položku.
- Komponentu TListBox, které nám zobrazí informace o označené položce menu – pořadové číslo položky v TTreeView, název položky a název přiřazeného souboru.
- Tlačítko pro práci se souborem označené položky menu. Tomuto tlačítku se mění popisek a funkce, podle toho jaká práce se souborem by měla následovat: Otevření souboru, Vytvoření souboru, Uložení souboru.

Pozice a velikosti jednotlivých prvků na tomto horním panelu mají pevně dané hodnoty. Tzn., že při zvětšení okna aplikace, zůstávají tyto prvky stále na stejném místě, vzhledem k levému hornímu rohu aplikace. Ale horní panel se automaticky zvětšuje podle šířky okna aplikace.

5.2.6. Střední panel

Středový panel aplikace zobrazuje pouze tip pro práci s programem, popřípadě varovnou zprávu. V dalších verzích programu může být upraven, a posléze nést i některé ovládací prvky a tlačítka (například tlačítka pro ztučnění vyznačeného textu v editačním okně apod.).

Text uvnitř panelu je zarovnán na střed panelu. Pozice panelu se mění vzhledem k pozici horního panelu – takže pokud horní panel není zobrazen, středový panel je zobrazen hned pod hlavním menu programu.

5.2.7. Dolní panel (Editor textu)

Zobrazí se až po kliknutí na tlačítko pro otevření přiřazeného textu k položce menu, nebo po kliknutí na tlačítko pro vytvoření nového textu. Jeho velikost se automaticky dorovnává do zbylé velikosti okna (zbylé vůči hornímu a středovému panelu).

Tento panel nese doplňkovou komponentu *TRichViewEdit*, kterou popíši dále v této práci. Tato komponenta nám slouží k zadávání, editaci a exportu textu.

5.2.8. Ikona aplikace

Hlavní ikona aplikace byla vytvořena pomocí programu *Stardock.net IconDeveloper*[4] a programu *Adobe Photoshop*[5]. Druhým programem jsem vytvořil obrázek části okna aplikace a prolnul jej s ikonou globusu, vypreparovanou z jiného programu. V Adobe Photoshop jsem pracoval s obrázkem velikosti 96 x 96 pixelů a výsledný obrázek uložil do formátu PNG-24 s průhledností. Uložený obrázek stačilo importovat do programu IconDeveloper, kde se automaticky zmenšil do všech standardních velikostí používaných v ikonách, a následným uložením mi vznikla ikona použitelná do jakékoliv aplikace, a to v barevném 32bitovém formátu Windows XP (s alfa průhledností) i v 16bitovém formátu pro starší verzi Windows.

Obr. 3: Ikona aplikace

[4] dostupné z: http://www.stardock.com/products/icondeveloper/
[5] dostupné z: http://www.adobe.com/products/photoshop/family/

5.3. Důležité použité VCL komponenty

V této kapitole se pokusím popsat nejdůležitější VCL komponenty, použité v aplikaci. Co je to *VCL* (*Visual Component Library*) jsem již obecně naznačil v kapitole **4.3.** K tomu, aby aplikace měla nějaký vzhled a funkčnost, musíme používat komponenty z knihovny VCL. Ve vývojovém prostředí Borland Delphi je najdeme v panelu nástrojů *Tool Palette*.

Každá vizuální komponenta je ve VCL popsána jako třída (*class*) se svými:

- **vlastnostmi** (*property*): atributy tříd, které určují stav objektu (pozice, vzhled, chování atd.)
- **metodami**: procedury a funkce spřažené s objektem, které je možné volat za účelem provedení příslušné akce
- **událostmi**: speciální druh vlastnosti, která v průběhu programu reprezentuje výskyt určité (většinou uživatelské) akce.

Vlastnostmi přístupnými (v době návrhu aplikace) v nejběžnějších komponentách rozumíme např.: *Caption* (titulek komponenty), *Color* (barva povrchu nebo pozadí), *Enabled* (udává, zda je komponenta aktivní nebo nikoliv), *Hint* (řetězec, který bude použit pro plovoucí nápovědu), *Name* (jedinečné jméno komponenty), atd.

Metody dostupné ve většině komponent jsou např.: *Create* (vytvoří novou instanci), *Destroy* (zruší instanci, ale doporučuje se používat raději metodu *Free*), *Focused* (zjistí, zda ovládací prvek je zaměřen), *Invalidate* (dává požadavek na překreslení), *Update* (provede okamžité překreslení (pouze pokud byl předtím dán požadavek pomocí Invalidate), atd.

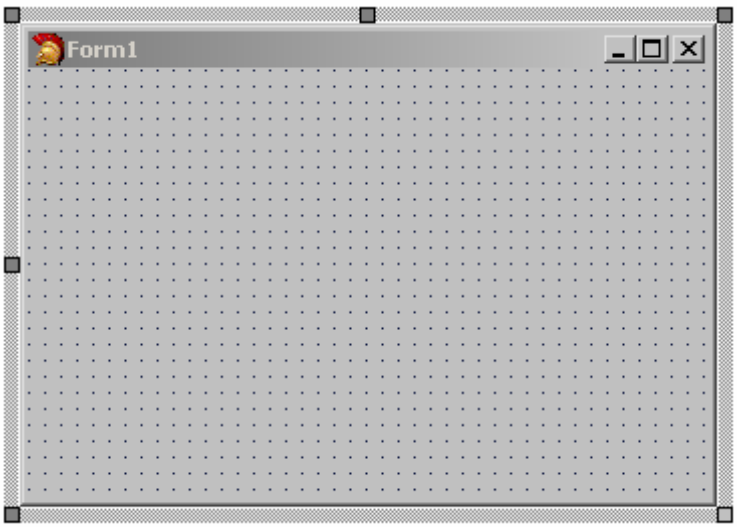
Události, které jsou nepoužívanější ve většině komponent: *OnChange* (dojde k ní při změně komponenty), *OnClick* (dojde k ní při klepnutím primárním (levým) tlačítkem na komponentu), *OnKeyPress* (dojde k ní při stisku klávesy, je poslána komponentě mající focus), atd.

5.3.1. Komponenta TForm

Mezi nejzákladnější VCL komponenty patří formulář (*TForm*), který nám slouží například jako hlavní okno aplikace. Tuto komponentu v panelu *Tool Palette* však nenajdeme, protože se vytváří automaticky s novým projektem typu *VCL Forms Application*. Pokud chceme do aplikace přidat další formulář (jako například dialogové okno s nastavením programu), musíme vybrat z menu položku *File* > *New* > *Form – Delphi for Win32*.

Formuláře mohou obsahovat další objekty (komponenty), jako je např.: *TButton*, *TMainMenu*, *TComboBox*, apod. Formuláře přebírají spoustu standardních vlastností, metod a událostí, jako jsou například některé popsané výše, na předchozí stránce.

Formuláře mají navíc například tyto vlastnosti: *ActiveControl* (specifikuje komponentu, která má mít focus jako první při zobrazení okna), *BorderIcons* (specifikuje, které ikony budou zobrazeny na titulním proužku okna – systémové menu, minimalizační a maximalizační tlačítko, tlačítko pro nápovědu), *Icon* (specifikuje ikonu pro minimalizované okno), *Menu* (specifikuje hlavní menu formuláře (okna), nastavuje se automaticky při použití komponenty *TMainMenu*), atd.

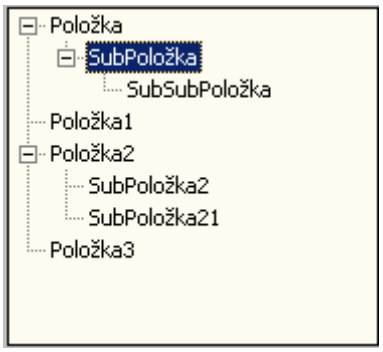


Obr. 4: TForm v návrhovém zobrazení v Delphi

Více o vlastnostech, metodách a událostech, jednotlivých komponent, se dočtete v helpu pro Delphi, kde jsou jednotlivé prvky podrobně popsány.

5.3.2. Komponenta TTreeView

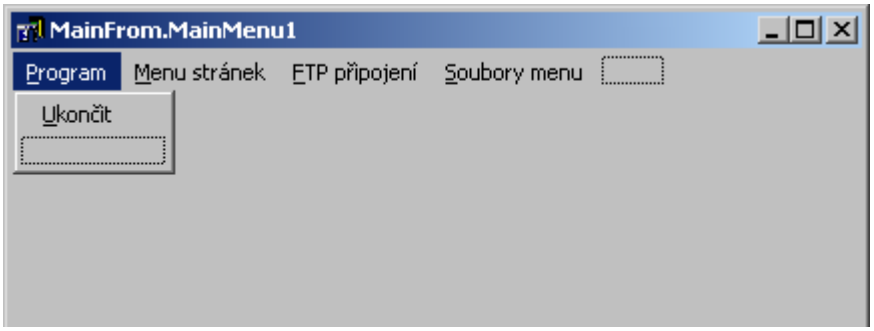
Tato komponenta hierarchicky zobrazuje položky, jako jsou například nadpisy stránky, obsah, nebo (nejznáměji) adresářová struktura na disku. Každá položka v *TreeView* může mít přiřazenou bitmapovou ikonu, nese s sebou svůj index v aktuální větvi stromu a může mít nastavenou vlastnost *Data*, což je pointer na libovolnou datovou strukturu. Jednotlivé větvě *TreeView* jsou položky typu *TTreeNode*.



Obr. 5: Ukázka použití komponenty TTreeView

5.3.3. Komponenta TMainMenu

TMainMenu je vlastně všem dobře známé hlavní menu aplikace, zobrazované pod titulkem (hlavičkou) aplikace. Obsahuje rozbalovací položky libovolné délky a každá položka může mít přiřazenu ikonu. Hlavní menu nelze umístit jinam než pod hlavičku aplikace. V návrhovém zobrazení formuláře v Delphi je tato komponenta identifikována svou ikonou. Po poklikání na tuto ikonu se nám zobrazí editace jednotlivých položek menu, jak je ukázáno na následujícím obrázku. Každá položka může mít definovaný tzv. *akcelerátor* (podtržené písmeno) pro použití klávesové zkratky. Akcelerátor se definuje znakem & (ampersant), před písmenem, které má být funkční a podtržené.



Obr. 6: Editace položek TMainMenu

5.3.4. Komponenta TIdFTP (pro FTP přenos)

TIdFTP je jednou z komponent projektu Indy.Sockets[6] (Internet Direct). Indy je Open Source soubor komponent, využívajících internetové protokoly a přenosy, jako jsou například protokoly SMTP, POP3, HTTP, a mnoho dalších. Jsou napsány v jazyce Delphi, ale jsou dostupné i pro jazyky C#, C++ a .NET. Tyto komponenty jsou standardně k dispozici v základní instalaci Delphi 2005 a obsahují jak klientskou část, tak i serverovou. Pro starší verze Delphi je možné tyto komponenty zdarma stáhnout z www stránek projektu Indy. Indy je dostupný i pro OS Linux.

Komponenta *TIdFPT* implementuje File Transfer Protocol (FTP), klientskou část, podle dokumentů internetového standardu RFC 959 apod. S FTP serverem komunikuje na bázi FTP příkazů a může jednoduchými metodami, po připojení k serveru, odesílat, přijímat, mazat, atd., soubory na FTP.

Pro stažení souborů z FTP serveru použijeme metodu *Get*, která je vlastně přetíženou procedurou pomocí níž FTP klient přijme soubor se jménem specifikovaným jeho názvem z FTP serveru, za použití FTP příkazu *RETR*.

Pro nahrání souborů na FTP slouží metoda *Put*, která je opět přetížená. Put nám odešle soubor, specifikovaný jeho názvem, z FTP klienta na FTP server za pomocí FTP příkazu *STOR*. Pokud je při volání metody Put uveden parametr *AApend*, bude odesílaný soubor připojen za již existující soubor FTP příkazem *APPE*.

Další použitou metodou je metoda *Delete*, která nám na FTP serveru smaže soubor se zadaným jménem, pomocí FTP příkazu *DELE*.

Nesmím vynechat ani metodu *List*, díky níž získáme z aktuálního adresáře na FTP (do adresáře se přepneme příkazem *ChangeDir*) výpis souborů a podadresářů na FTP. Ty pak s pomocí jednoduchého cyklu a property *DirectoryListing* můžeme vypsat nebo s nimi dále pracovat.

[6] dostupné z: http://www.indyproject.org/Sockets/index.EN.aspx

5.3.5. Komponenta TRichView (editace textu)

RichView je VLC komponenta pro Delphi, které umožňuje zobrazování, editaci a tisk hypertextových dokumentů, jejímž autorem je Sergey Tkachenko. Tato komponenta podporuje rozmanité textové možnosti (fonty, dolní/horní indexy, barvy textu a barvy pozadí, vkládání a zobrazování obrázků). Dokumenty napsané v *RichView* mohou obsahovat tabulky, obrázky, obrázky z komponenty *ImageList* a také některé Delphi komponenty. S *RichView* můžeme centrovat text, zarovnávat jej do bloku, vlevo, vpravo, můžeme nastavovat okraje, odrážky, číslované odrážky, ukládat text v Unicode, exportovat do RTF a exportovat do HTML.

RichView je kompletně napsán v jazyce Delphi, a nepoužívá externí DLL nebo ActiveX knihovny. Ani není založen na Microsoft RichEdit komponentě, která je standardní součástí prostředí Delphi. Bohužel je tato komponenta komerční, a její cena byla, v době psaní této práce, 189 Euro. Z internetových stránek[7], této komponenty, se dá stáhnout balíček (package) do Delphi, díky kterému můžeme tuto komponentu v Delphi používat. Ale po spuštění aplikace, která *RichView* obsahuje, se v levém horním rohu obrazovky zobrazuje červený rámeček s textem „TRichView unregistered“. Existuje i freeware verze, ale je to starší verze, která neumožňuje tolik možností a nemá oficiální podporu.

Na www stránkách je možné stáhnout doplňující komponenty pro *RichView* a to například pro import html dokumentů, pro konvertování obsahu *RichView* do PDF, pro zvýrazňování syntaxe, pro práci s XML soubory, apod.

Tuto komponentu najdeme například v programech Skype (program pro telefonování po internetu), The Bat (emailový klient).

Ve vzorové aplikaci, určené touto bakalářskou prací, bude tato komponenta pracovat ve svém nejjednodušším použití. Tzn., že nám bude sloužit k psaní prostého textu, který pak budeme exportovat do html a ukládat do formátu této komponentě vlastnímu – rvf (rich view file).

[7] dostupné z: http://www.trichview.com/

6. Práce s aplikací

6.1. Získání a instalace programu

Aplikaci je možné stáhnout z internetových stránek, a pak kamkoliv přenést na médiu nebo flash disku, popřípadě odeslat emailem. První verze aplikace se dodává jako zabalený archiv *ZIP*, který stačí extrahovat do Vámi zvolené složky a z ní poté spustit soubor *Web-amin.exe*. Při extrahování archivu nám vznikne i základní adresářová struktura programu. Při používání programu se nám, po uložení struktury menu, vytvoří soubor menu (*menu.xml*) a po uložení nastavení FTP připojení vytvoří soubor *nastaveni.xml*. Program si po spuštění nic neukládá do registrů ani do jiných než vlastních složek, tak pro odinstalování programu stačí jen smazat adresář, ve kterém je program umístěn. Pokud chceme urychlit jeho spuštění, doporučuji vynést si zástupce programu např. na pracovní plochu.

V dalších verzích aplikace se plánuje samoinstalační balíček, který nám s asistencí průvodce instalace (všem dobře známý *Setup Wizard*, používaný u instalací většiny aplikací) program nainstaluje do vybrané složky a vynese zástupce na plochu i do nabídky Start.

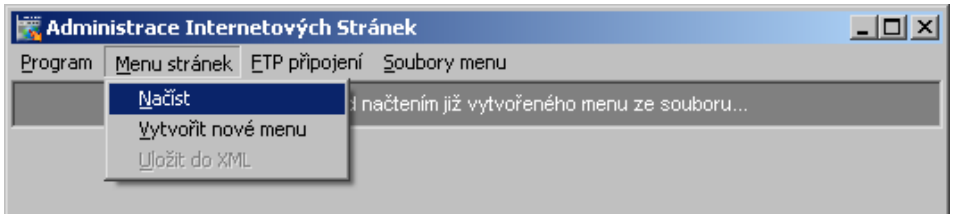
Po rozbalení (instalaci) programu jeho kořenový adresář vypadá následovně:

```
...<adresář s programem>\
     └ <web>\
     |    └ <rvf>\        - zdrojové soubory textu
     |    └ <text>\       - exportovaný text
     |   (└ menu.xml)[8]   - exporované menu
     |   (└ nastaveni.xml)[9] - nastavení programu
     └ Web-admin.exe      - spustitelná aplikace
```

[8] Exportované menu z programu. Tento XML soubor se vytvoří až po uložení menu.
[9] Exportované nastavení programu. Vytvoří se až po uložení nastavení z programu.

6.2. Vlastní menu www stránky

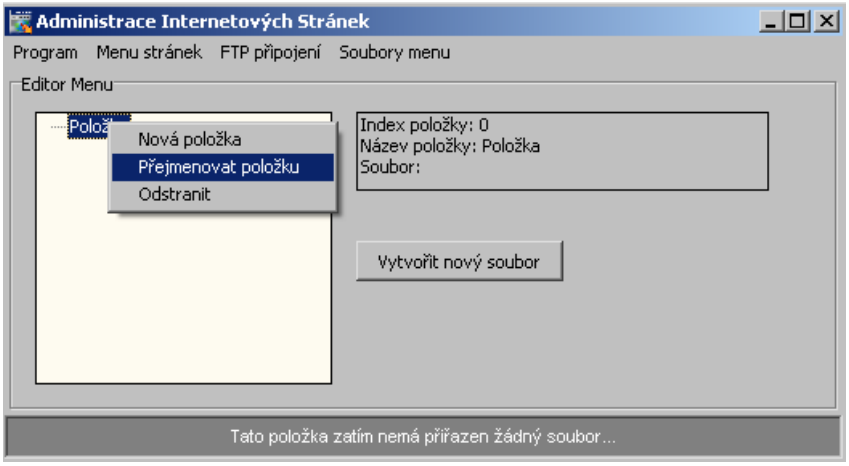
Pro práci s vlastním menu www stránek slouží položka hlavního menu: *Menu stránek* (Alt + M). Při prvním spuštění programu zatím nemáme žádné menu vytvořené, proto nemůžeme použít položku *Načíst* (Alt + N), ale musíme vytvořit menu nové – *Vytvořit nové menu* (Alt + V). Další možností je nastavit si FTP připojení, a menu a další soubory stáhnout z FTP serveru. Poté lze menu načíst ze souboru. K tomuto se vrátím v další kapitole.



Obr. 7: Položky hlavního menu pro práci s menu stránek

V případě, že použijeme položku *Načíst* a soubor *menu.xml* neexistuje, program nám nahlásí, že soubor s uloženým menu neexistuje.

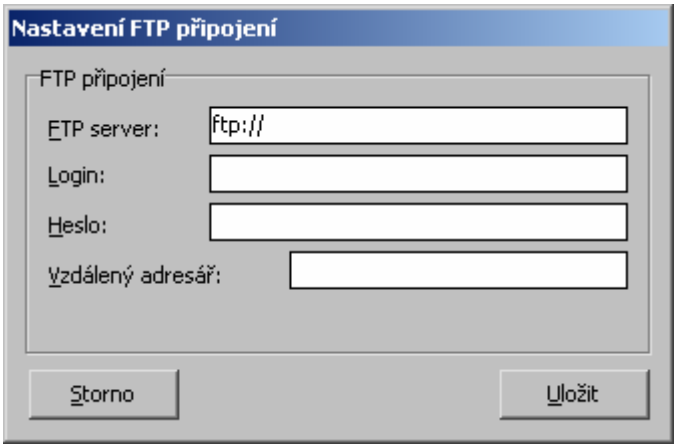
Po kliknutí na položku *Vytvořit nové menu* (nebo po načtení již existujícího menu) se v aplikaci zobrazí panel, pro práci s menu stránek. V levé části panelu je světlá plocha, kde jsou jednotlivé položky menu zobrazeny a my je můžeme libovolně přejmenovávat, mazat a přidávat. Pracujeme s nimi po kliknutí pravým tlačítkem myši, kdy se nám zobrazí popup menu s dalšími volbami.



Obr. 8: Práce s menu www stránky

6.3. Nastavení a FTP přenos

Abychom moli přenést vytvořené menu se soubory na FTP server a naopak tyto soubory z FTP přijmout na lokální disk, je třeba si nastavit parametry pro FTP přenos. Najdeme je pod menu *FTP připojení* (Alt + F). Po kliknutí na položku *Nastavení FTP...* se otevře *modální* formulář pro zadání parametrů, který reaguje na stisk klávesy *ESC* (zavření formuláře s ponecháním původního nastavení) a klávesy *Enter* (zavření formuláře s přijmutím nového nastavení).



Obr. 9: Formulář pro nastavení FTP

Položka *FTP server* obsahuje vzdálený FTP server nebo jeho IP adresu, pro připojení. *Login* je přihlašovací jméno na FTP. *Heslo* je heslo nutné pro přihlášení na FTP server. *Vzdálený adresář* je root www stránek na serveru, od kterého se odvíjí stejná adresářová struktura jako na lokálním disku (*web/*, *web/rvf/*, *web/text/*). POZOR! Tato adresářová struktura MUSÍ na FTP serveru již být vytvořena a musí mít nastavené atributy pro povolený zápis do adresářů. Zvolené nastavení můžeme uložit do souboru, kliknutím na položku menu *FTP připojení* > *Uložit nastavení*.

Před odesláním nebo přijmutím souborů z FTP, musíme otestovat funkčnost připojení: *FTP připojení* > *Test připojení*. Teprve po úspěšném testu, můžeme soubory odeslat na FTP (*Soubory menu* > *Nahrát na FTP*) a stáhnout z FTP (*Soubory menu* > *Stáhnout z FTP*). Po dokončení těchto operací program nahlásí, že FTP přenos byl úspěšně dokončen.

7. Vytvoření validní prezentace

V této kapitole se pokusím stručně přiblížit nejdůležitější body a části, při vytváření www prezentace Autorizovaného Nokia servisu Hradec Králové[10]. Při tvorbě stránek bylo přihlíženo na dvě zásadní rozdělení aspektů: *uživatelské* a *technologické*.

7.1. Uživatelské aspekty

Do tohoto rozdělení patří především interakce stránek s uživatelem. Uživatelovo vizuální vnímání, jeho čas strávený hledáním a prohlížením dat na stránkách, komfort při prohlížení stránek apod. Jsou to vlastně věci, které bere uživatel, prohlížející stránky, jako samozřejmost a nemusí nad nimi přemýšlet. To co se pod těmito uživatelskými aspekty skrývá, bývají již *technologické aspekty*. Některé z nich spolu přímo souvisejí. Například uživatelův čas strávený prohlížením stránek může být závislý na použitých grafických objektech vložených do designu stránek a na dalších datově náročných technologiích (např. Flash). Ale ne všechny *uživatelské aspekty* jsou přímo podloženy těmi technologickými. Mezi ty nepodložené patří například rozmístění ovládacích a zobrazovacích částí stránky, členění textů, hierarchie hlavního menu, použité barvy…

7.1.1. Volba vzhledu

Celkový vzhled stránek Nokia servisu byl zvolen spíše formálně, s ohledem na přehlednost. Po konzultaci se zadavatelem jsem vycházel z jedné z klasických koncepcí: horní část stránek tvoří tzv. *header* (dominantní plocha stránek v horní části, nejčastěji obsahuje grafiku), pod headerem se nachází (v levé části) hlavní menu, a v pravé části zobrazovací plocha. Barvy použité v designu stránek měli korespondovat s barvami loga firmy Nokia (modrá). Byl vytvořen pouze jeden grafický návrh, který byl zadavatelem rovnou přijat. Jeho první verzi je vidět na následujícím obrázku. Pro potřeby stránek a potřeby zadavatele, byly nakonec položky doplňujícího menu (pod hlavním menu) přesunuty rovnou do menu hlavního, jako rozbalovací podpoložky. Rovněž byly upraveny drobné grafické prvky v headeru a logo. Konečnou verzi stránek je

[10] dostupné z: http://www.nokiahk.cz

možné prohlédnout na adrese www.nokiahk.cz (duben 2007) nebo na přiloženém CD u této bakalářské práce.

Obr. 10: První verze grafického návrhu

7.2. *Technologické aspekty*

Jsou tvořeny převážně použitými technologiemi při výrobě stránek a věcmi, který běžný uživatel nevidí nebo vidí, ale netuší, proč se tomu tak děje. Patří sem např. použití kaskádových stylů *CSS*, díky nimž je vzhled stránek a umístění jejich prvků takové jaké je. Ale uživatel neví, jak toho bylo dosaženo a ani to převážně nezkoumá.

7.2.1. Celkový grafický design

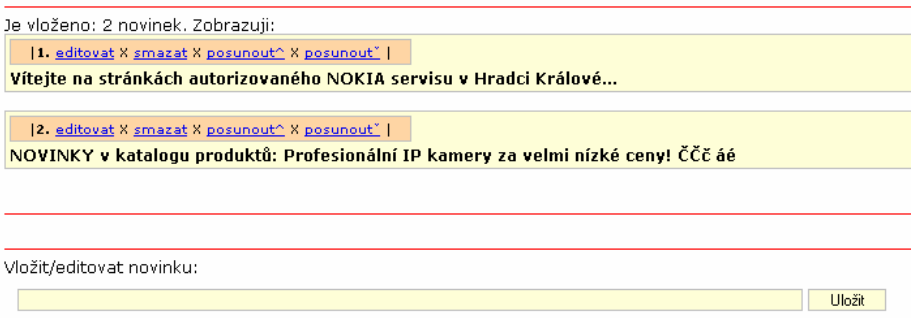
Design stránek byl zpracován programem Adobe Photoshop a následně, po schválení zadavatelem, *rozparsován* (postup, kdy z jednoho obrázku, který představuje celkové zobrazení stránek, dostaneme jednotlivé obrázky, co možná nejmenší velikosti, které budou použity při vytváření stránek jako objekty a části celkového designu). Veškeré obrázky byly, dle množství obsahujících barev, buď komprimovány do formátu *GIF* nebo *JPEG*, s ohledem na jejich vizuální kvalitu a datovou velikost.

Celkový design stránek je tvořen převážně pomocí html tagů *div*, kterým jsou přiřazeny vlastnosti *css* z externího *css* souboru. Doporučuje se nepoužívat inline css styly (css vlastnosti napsané přímo v *tagu*), a pokud již tak činíme, musíme v hlavičce *html* definovat *meta tag*: `<meta http-equiv="Content-Style-Type" content="text/css"/>`. Ve www prezentaci Nokia servisu nejsou použity žádné inline styly. Stránka je v prohlížeči horizontálně centrována tagem *div* s vlastností `align="center"` a na výšku se roztahuje podle délky svého obsahu.

7.2.2. Header

Header stránky obsahuje dominantní grafiku pro celou prezentaci a je ve všech částech prezentace stejný. Jeho součástí je i malá animace vytvořená technologií Macromedia Flash (dnes Adobe Flash). Pokud nemáme v prohlížeči povolen *ActiveX*, tak se místo animace zobrazí jen statický obrázek. Je toho docíleno tím, že flash animace je v *divu* s absolutní pozicí, nad obrázky headeru.

Součástí headeru je i postupné zobrazování textů (jako novinky), kterého je docíleno *javascriptem*. Texty novinek se vkládají skrz jednoduché webové administrační rozhraní a do stránek jsou načteny javasriptem generovaným pomocí *php*.



Obr. 11: Administrace novinek

Datová velikost headeru je kolem 30 kB (pro obrázky) a 14 kB pro flash animaci mobilního telefonu a 9 kB pro flash animaci textového sloganu. Velikost html kódu úvodní stránky se pohybuje kolem 18 kB.

7.2.3. Jádro stránek a funkčnosti

Vlastní stránky, zobrazené v prohlížeči, jsou napsané v jazyce *html*, normou *XHTML 1.0 Transitional*[11] a jsou podle této normy validní. Kód stránek je napsán přehledně, pro pozdější snadné zásahy do něj.

Jednotlivé části stránek jsou rozděleny do několika základních souborů a fungují jako šablona, načítaná v php. Tzn., že máme jeden soubor (pojmenovaný jako *index.php*) do kterého se postupně načítají soubory, celkově tvořící vzhled stránek. Do těchto souborů jsou pak inkludovány texty a obsah (v php funkcí *include*, případně funkcí *require*), podle zvoleného odkazu z menu stránek.

Stránky jsou testovány a laděny ve třech nejpoužívanějších internetových prohlížečích současné doby: Internet Explorer, Mozilla Firefox a Opera. V těchto prohlížečích vypadají stránky téměř identicky a zobrazují se správně.

7.2.4. Katalog výrobků

Součástí www prezentace je i internetový katalog výrobků. Tento katalog je řešen pomocí php a databáze *MySQL*. Má vlastní webovou administraci výrobků a stromově strukturované rozbalovací menu jednotlivých kategorií. Každá kategorie může mít libovolný počet podkategorií. V databázové tabulce je toho docíleno tak, že každá kategorie má definováno, kromě svého *id* i id nadřazené kategorie ze stejné tabulky.

Každý výrobek může mít libovolný počet obrázků a obrázky jsou po vložení automaticky zmenšeny do třech velikostí (php funkcí *imagecopyresampled*). Po kliknutí na obrázek se otevře nové okno (vertikálně i horizontálně centrované) s obrázkem, které je pomocí javascriptu automaticky zmenšeno na velikost obrázku a lze jej zavřít kliknutím na obrázek.

Další a podrobnější popis internetového katalogu by již přesáhl rozsah a téma této bakalářské práce.

[11] dostupné z: http://www.w3.org/TR/xhtml1/

8. Publikace firemních dat

Existuje několik možností jak propojit firemní databázi, nebo jen některá data z ní, s internetovou prezentací. Některá řešení jsou implementačně a cenově náročná, jiná nejsou bezpečná pro firemní data a některá řešení jsou složitá na obsluhu. Vyvstávají nám zde základní otázky:

- Opravdu je nutné něco z firemní databáze zveřejnit?
- Kolik jsme ochotni investovat do zveřejňování dat?
- Jak moc mají být data na internetu aktuální?
- Jak uživatelsky složitá bude aktualizace dat?

Pokud si na první otázku odpovíme kladně a číslo u druhé otázky bude malé, je docela možné, že nám na třetí otázku nezbude nic jiného než odpovědět: Moc aktuální nebudou, protože aktualizace dat bude hodně manuální.

Co se týče bezpečnosti, asi žádná firma nemá zájem na tom, aby citlivá firemní data bylo možné nějakým způsobem přečíst, zcizit nebo smazat.

Musíme se také ohlížet na to, jak jsou firemní data uložená. Jaký software je použit a je-li firma, která software vytvořila, ochotna spolupracovat. Ideální by bylo, kdyby se tvůrce software, použitého ve firmě, chopil celkového řešení publikace některých dat sám. Toto se však týká jen firem, které používají software, pracující s databází, udělaný jim na míru nebo jiný software pracující s nějakým typem databáze uložené na lokálním disku ve firmě. Jako příklad uvedu Autorizovaný Nokia servis Hradec Králové (dále zadavatel), který používá software, vytvořený na zakázku, pro správu oprav mobilních telefonů, tisk opravenek, faktur apod.

8.1. Konkrétní zadání

Zadavatel si přeje, aby jeho zákazník měl možnost, na internetových stránkách, zjistit stav opravy svého mobilního telefonu. Tato data (informace o zákazníkovi, mobilním telefonu, stavu opravy a závadě) jsou uložena v databázi na lokálním disku PC v servisu u zadavatele, včetně dalších citlivých dat.

Zákazník bude mít, na internetových stránkách, možnost, podle čísla opravy a IMEI svého mobilního telefonu, zjistit, jaký je stav jeho opravy. Číslo opravy a IMEI svého telefonu mu je známo z opravenky vystavené v servisu. Proč zadávat číslo opravy i IMEI? Číslo opravy proto, že jeden telefon (jedno IMEI) může mít i více oprav. IMEI telefonu proto, že kdybychom zadávali jen číslo opravy, mohl by si každý zjišťovat např. to, kolik je v servisu oprav apod. (postupným zadáváním jednoho čísla opravy za druhým – číslo opravy se s každou další opravou zvyšuje o jedničku).

8.2. Možnosti řešení

8.2.1. Propojení programu s online databází

Nabízí se nám tu myšlenka, že stávající software by byl upraven tak, aby po zadání nebo změně opravy, vložil konkrétní data do MySQL databáze, která je přístupná prezentaci na serveru poskytovatele webhostingu. Pro zobrazení dat by to bylo nejjednodušší řešení. Ale jak jsem již podotknul v kapitole 2.2 (Nevýhody offline aplikace), pokud nevlastníme vlastní server, je to téměř neproveditelné.

8.2.2. Propojení online prezentace s lokální databází

Toto řešení je již uskutečnitelné, ale implementačně náročnější. Ve chvíli, kdy se potřebujeme z jiného PC dostat na to naše, musíme znát jeho IP adresu. Zadavatel má veřejnou IP adresu, tak by to problém nebyl. Chceme-li mít data přístupná v kteroukoliv dobu, musí nám ve firmě běžet stále zapnutý počítač, který je přístupný z venku.

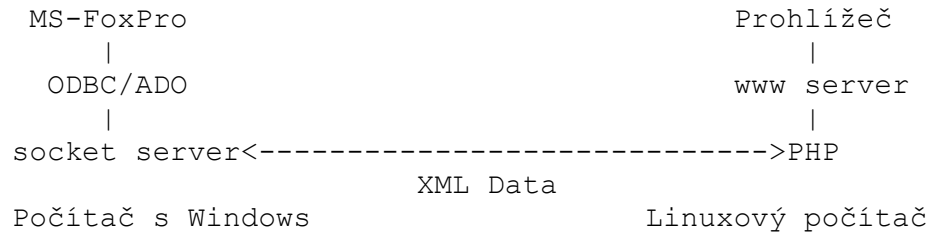
Při konzultaci s programátorem, který firemní software programoval, mi bylo sděleno, že software používá databázi Microsoft Visual FoxPro. Programátor mi poskytl i jméno databáze, tabulky a sloupců, které jsou nutné pro uskutečnění řešení.

Pro připojení z jakéhokoliv počítače do databáze na jiném počítači použijeme *ODBC Driver* (Open Database Connectivity) *Microsoft Visual FoxPro Driver*[12]. Tento ovladač je nutné stáhnout ze stránek Microsoftu. Po

[12] dostupné z: http://msdn2.microsoft.com/en-us/vfoxpro/bb190233.aspx

nainstalování jej bude možné použít. Nastavení ODBC najdeme (ve Windows XP) v Ovládacích panelech pod Nástroji pro správu, položka Datové zdroje. Po otevření se musíme přepnout na záložku Systémové DSN. Systémové DSN jsou datové zdroje přístupné pro kteréhokoliv uživatele, odkudkoliv (teoreticky). Tam je nutné FoxPro Driver přidat a nastavit mu požadované parametry a datový soubor, který bude přístupný.

Takto dosáhneme přístupu do datového souboru FoxPro z kterékoliv aplikace na lokálním PC, přes DSN připojení do ODBC. Pro naše potřeby to však je nedostačující. Aby nám byl systémový DSN zdroj přístupný i z jiného počítače, musíme nějakým způsobem připojit ODBC na port počítače a naslouchat připojení. Pro účely PHP nám postačí open source *ODBC Socket Server*[13], který nám zpřístupní ODBC datové zdroje přes TCP/IP rozhraní, na bázi XML. Tento program je, po stažení a rozbalení, nutné spustit jako službu (v příkazovém řádku: `...cesta\ODBCSocketServer.exe /service`) a poté ve správci služeb nastavit její spouštění na hodnotu *automaticky*. Program obsahuje i export do registrů s jeho nastavením. V registrech si pak (podle manuálu dostupného z pdf[14]) můžeme nastavit port, přes který bude ODBC přístupné, a také IP adresy, kterým bude přístup umožněn. To bude z bezpečnostního hlediska nutné nastavit podle IP serveru, na kterém jsou www stránky zadavatele umístěny.



Připojení z PHP se realizuje pomocí třídy *ODBCSocketServer*, kterou je možné stáhnout ze stránek *PHPBuilder.com*[15]. Pro připojení nám stačí

[13] dostupné z: http://odbcsock.sourceforge.net/
[14] dostupné z: http://odbcsock.sourceforge.net/release/odbcmanual.pdf
[15] dostupné z: http://www.phpbuilder.com/columns/timuckun20001207.php3

znát IP adresu ODBC počítače a port, na který se připojíme. Spojení z PHP je provedeno funkcí *fsockopen* a data získaná SQL dotazem jsou parsována z XML.

Pro ideální bezpečnost firemního počítače by bylo dobré použít dobrý softwarový firewall, ve kterém bychom umožnily přístup pouze z jedné IP adresy a na jeden port.

Přístup do firemní databáze přes webovou prezentaci byl zatím spuštěn jen pro testování. Zde, v této práci, zveřejňuji jen jedno IMEI telefonu: 357605006421015 a jeho opravenku číslo: ZL-07-7646. Telefon byl již opraven, jak se dozvíte z testovacích stránek[16].

8.2.3. Manuální propojení s online prezentací

Jsou dva způsoby manuálního propojení. Jedním je duplicitní zadávání dat – do firemního software a přes webové rozhraní do online databáze. Druhým způsobem je použití exportu z firemního software a následně jeho nakopírování na FTP server. Export je prováděn do datových souborů XML, které by bylo možné ve www prezentaci parsovat a případně vkládat do online databáze. Druhý způsob je mnohem náročnější na implementaci než ten první, navíc exporty všech opravných listů jsou poněkud rozsáhlé, tudíž jejich online parsování na www serveru by mohlo být časově a výkonnostně náročné.

Opět bychom mohli data získávat i přes ODBC. Pro tento případ, kdybychom nechtěli umožnit z www serveru přístup na náš počítač, by bylo možné napsat spustitelnou aplikaci, která by se připojila na lokální ODBC rozhraní, získala všechna potřebná data z databáze a nahrála je na FTP server ve zjednodušené podobě, pro snadnější parsování.

[16] dostupné z: http://test.sirucek.eu/

9. Závěr

Touto prací bylo demonstrováno alternativní řešení pro administraci některé z částí internetové prezentace. Možnosti softwarových produktů na bázi aplikací pro operační systémy jsou rozsáhlé. Ale jejich vlastní vývoj je leckdy mnohem složitější než vývoj administračního rozhraní přes webový prohlížeč, a klientů, které by byli ochotni zaplatit vývoj hodně funkční aplikace, je příliš málo. Vše má své pro i proti. Je jen na každém z nás, pro kterou možnost se rozhodneme.

Propojení s firemní databází bylo realizováno freeware softwarem ODBC Socket Server. Ale existují i placená řešení, např. *UniverSQL*[17] a *SequeLink Server*[18]. Veškerá konfigurace a nastavení probíhala přímo v Autorizovaném Nokia servisu v Hradci Králové, s podporou pana Jaroslava Vosáhla (majitele firmy).

Aplikace vytvořená při této bakalářské práci je volně ke stažení na internetové adrese *http://test.sirucek.eu/download/web-admin-empty.zip* (verze bez testovacího nastavení a souborů) a na adrese *http://test.sirucek.eu/download/web-admin-testovaci.zip* (verze s testovacím nastavením i soubory). Dále je dostupná (i se zdrojovými soubory) na CD, které je přílohou této bakalářské práce.

[17] dostupné z: http://www.sidespace.com/products/universql/
[18] dostupné z: http://www.datadirect.com/products/sequelink/slsocket54/index.ssp

ÚDAJE PRO KNIHOVNICKOU DATABÁZI

Název práce	Aplikace pro správu internetových stránek a propojení firemní databáze s internetovou prezentací
Autor práce	Lukáš Sirůček
Obor	Informační technologie
Rok obhajoby	2007
Vedoucí práce	RNDr. Rak Josef
Anotace	Tato práce se zabývá problematikou softwarového řešení pro malé společnosti, které si přejí měnit obsah své internetové prezentace pomocí aplikace pro prostředí operačního systému Windows. Vzorová aplikace je naprogramována ve vývojovém prostředí Delphi 2005. V této práci se zabývám i možnostmi publikace některých firemních dat z interního softwarového řešení, vytvořeného na zakázku jinou společností. Tato práce se konkrétně dotýká autorizovaného Nokia servisu v Hradci Králové a její součástí jsou i internetové stránky s katalogem produktů a zobrazením stavu konkrétní opravy pro zákazníka.
Klíčová slova	Delphi, aplikace, www, internetová prezentace, ODBC, databáze, FTP, správa internetových stránek

Seznam použité literatury

1. KADLEC, V. *Delphi: hotová řešení.* 1. vyd. Brno : Computer Press, 2003. ISBN 80-251-0017-0.

2. LACKO, L. *Web a databáze.* 1. vyd. Praha : Computer Press, 2001. ISBN 80-7226-555-5.

3. SVOBODA, L. *1001 tipů a triků pro Delphi.* 2. aktualiz. vyd. Brno : Computer Press, 2003. ISBN 80-7226-488-5.

4. UCKUN, Tim. *ODBC Socket Server* [online]. c2000 [cit. 2007-04-29]. Dostupný z WWW: http://www.phpbuilder.com/columns/timuckun20001207.php3

5. *Wikipedie: Otevřená encyklopedie: Borland Delphi* [online]. c2007 [cit. 2007-04-20]. Dostupný z WWW: http://en.wikipedia.org/w/index.php?title=Borland_Delphi&oldid=130300934