VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

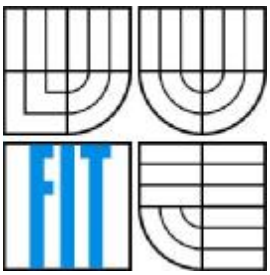
# PLÁNOVAČ S NADSTANDARDNÍMI FUNKCEMI

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MARTIN KRKAVEC
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# PLÁNOVAČ S NADSTANDARDNÍMI FUNKCEMI

REMINDER WITH SPECIAL FUNCTIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MARTIN KRKAVEC
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR WEISS
SUPERVISOR

BRNO 2007

# Abstrakt

Cílem této práce je navrhnout a na platformě Windows implementovat plánovač s nadstandardními funkcemi. Systém má obsahovat plánovací kalendář, budík, správu úkolů, centrální databázi dostupnou z internetu, synchronizaci v režimu offline, hierarchii uživatelů pro zadávání úkolů, běh aplikace na pozadí s vhodným způsobem upozorňování (např. popup okna) a upozorňování prostřednictvím e-mailů.



# Klíčová slova

plánovač, organizér, kalendář, budík, správa úkolů, databáze, Object Pascal, Delphi, MySQL, PHP



# Abstract

The goal of this BC is design and implementation of reminder with special functions on Windows platform. This system shall provide calendar, alarm, task management, central database accessible from internet network, synchronization in offline mode, user hierarchy for adding new tasks, application running in background with suitable alert system (i.e. window popup) and alerting with e-mail messages.



# Keywords

reminder, organizer, calendar, alarm, task management, database, Object Pascal, Delphi, MySQL, PHP



# Citace

Martin Krkavec: Plánovač s nadstandardními funkcemi, bakalářská práce, Brno, FIT VUT v Brně, 2007

# Plánovač s nadstandardními funkcemi

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Weisse. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Krkavec
14.5.2007

## Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Petru Weissovi za poskytnutou odbornou pomoc.

# Obsah

# Úvod

Organizéry, diáře, plánovací kalendáře a plánovače se díky velkému rozvoji výpočetní techniky a uspěchané době staly užitečnou součástí našeho života. Setkáváme se s nimi na pultech obchodů v různých provedeních jako jsou elektronické diáře, mobilní telefony nebo jako softwarové vybavení v počítačích. Jsou velkých přínosem nejen pro manažery nebo vedení středních a větších firem, ale i pro jejich zaměstnance, kterým není výpočetní technika cizí. Všem těmto uživatelům pomáhají při organizaci veškerých jejich časových plánů a rozvrhů.

V této bakalářské práci jsem se pokusil navrhnout a realizovat softwarový produkt – plánovač s nadstandardními funkcemi. Vycházel jsem přitom ze zadání neformální specifikace.

Pro tvorbu a provoz této aplikace jsem použil následující programové vybavení. Operační systém Microsoft Windows XP SP2, www server Apache verze 2.2.4 s modulem PHP 4.4.6, databázový server MySQL verze 3.23.49 a vývojové prostředí Delphi 7.

Plánovač jsem implementoval do dvou na sobě nezávislých klientů. V případě tlustého klienta se jedná o typickou aplikaci pro operační systém Microsoft Windows. V druhém případě je jedná o tenkého klienta v podobě webového rozhraní. Oba klienti využívají společnou centrální databázi a oba plní, až na několik rozšíření tlustého klienta, stejné hlavní funkce. Každý z těchto klientů má své přednosti i zápory.

V kapitole technologie popisuji problematiku plánovačů, existující řešení, použité jazyky (HTML, PHP, SQL), principy a architektury databází a databázová rozhraní v Delphi (BDE, ADO, dbExpress).

V následující pasáži podrobně popisuji funkce navržených klientů, návrh struktury centrální i lokální databáze. Nevyhýbám se ani diagnostice problémů, které se v průběhu vývoje aplikace vyskytly.

Implementační část již konkrétně řeší přístup k problematice v daném prostředí. V tomto případě se jedná o tlustého klienta. Je zde popsán způsob napojení na samotnou databázi. Dále jsou zde popsány použité třídy a komponenty, na kterých je aplikace postavena.

Na závěr se věnuji shrnutí průběhu vývoje a realizace mého návrhu. Také se zde zabývám úvahami o nových možnostech a rozšiřujících funkcích tohoto plánovače.

# 1 Technologie

## 1.1 Plánovač

Plánovačem se rozumí aplikace, která umožňuje jak naplánování úkolů a událostí na přesný den a na přesnou hodinu, tak jejich následné sledování a hlídání, spojené s vhodným způsobem upozornění v případě, že daný úkol či událost nastane.

### 1.1.1 Standardní funkce

Funkce bez kterých by plánovač nebyl plánovačem bezpodmínečně jsou:

- kalendář s možností listování
- budík
- správa úkolů
- upozorňování

### 1.1.2 Nadstandardní funkce

Určitým způsobem luxusní funkce, přidávající plánovači na univerzálnosti a všestrannosti použití mohou být například:

- vyhledávání a připomínání svátků, správa narozenin
- adresář kontaktů
- časované spouštění programů
- upozorňování prostřednictvím e-mailů či SMS
- centrální databáze úkolů dostupná z internetu, tedy bez nutnosti přenosu databázového souboru pro nezávislost na právě používané počítačové stanici
- hierarchie uživatelů pro zadávání úkolů

### 1.1.3 Existující řešení

Na českém internetu je dostupný poměrně široký výběr plánovačů, organizérů, diářů a různých kalendářů.

V podstatě všechny obsahují základní funkce opírající se o kalendář, hodiny a databáze různých událostí, narozenin, svátků a statistických veličin. Výjimkou nejsou ani rozsáhlé adresáře s možností tvorby hromadné korespondence.

Určitý problém nastává v okamžiku, kdy uživatel potřebuje přenést aplikaci včetně vlastních úkolů na jinou počítačovou stanici. Žádné totiž nedisponují centrální databází úkolů dostupnou z internetu, tím pádem ani hierarchií uživatelů pro zadávání úkolů.

Také absence upozorňování prostřednictvím mobilních e-mailů je velkým nedostatkem, protože vyžaduje přítomnost uživatele u počítače. Zvláště v dnešní době velkého rozvoje mobilních telefonů se informační zprávy SMS přímo nabízí.

## 1.2 Jazyk HTML a XHTML

Jazyky HTML[1] a XHTML[2] byly a jsou účelově vytvořené jazyky pro publikování stránek v síti WWW a definují, jak se má daný dokument zpracovat a zobrazit v klientském prohlížeči.

### 1.2.1 Historie

Vznik samotného jazyka HTML se datuje do roku 1990 ve Švýcarsku, kdy autoři Tim Berners-Lee a Robert Caillau vytvořili jednoduchý a přenosný formát pro tvorbu dokumentů. Postupem času byly oficiálně vydány verze 2.0 (rok 1994), 3.2 (květen 1996), 4.0 a 4.01 (prosinec 1997), kdy vyvíjení HTML končí. a začíná nový vývoj jazyka XHTML.

Jazyk HTML byl odvozen již z dříve vyvinutého univerzálního značkovacího jazyka SGML[3]. Standard jazyka XHTML vznikl transformací HTML, tak, aby vyhovoval tvorbě dokumentů XML[4] a byla zpětně zachována kompatibilita s HTML.

### 1.2.2 Základní koncepce

Dokumenty psané v těchto jazycích jsou charakteristické použitím značek (tagů) a případně jejich atributy. Názvy značek jsou ohraničeny ostrými úhlovými závorkami. Ty pak určují, jak se bude mezi ně vložený text na WWW stránce zobrazovat. Tato část dokumentu uzavřená mezi značkami tvoří

[1] HTML (Hypertext Markup Language) – hypertextový značkovací jazyk

[2] XHTML (Extensible Hypertext Markup Language) – rozšiřitelný značkovací jazyk

[3] SGML (Standard Generalized Markup Language) – univerzální značkovací jazyk

[4] XML (Extensible Markup Language) – rozšiřitelný značkovací jazyk

tzv. element (prvek) dokumentu. Atributy značky jsou informace, které dále upřesňují vlastnosti elementu.

Je potřeba dále rozlišit, jedná-li se o značku párovou či nepárovou. Příkladem párové značky je například značka pro vytváření odstavců <**p**>Odstavec</**p**>. Nepárová je například <**br**>, která ukončí aktuální řádek. U párových se rozlišuje, zdali je to značka počáteční (bez lomítka před názvem značky) například <**p**> nebo koncová (má lomítko před svým názvem) například </**p**>. Mezi některé lze vložit i další značky, které ovlivňují třeba formátování nebo vzhled stránky či textu.

V jazyce XHTML oproti HTML je nutné ukončovat i nepárové značky, aby byl dokument validní. Například pro ukončení řádku je nutné zapsat místo <**br**> v HTML, <**br/**> v XHTML. Další změnou v XHTML je pojmenovávaní značek a atributů. Ty musí být v XHTML zapsány malými znaky. Například následující zápis:

```
<body onLoad=”funkce();”>
```

lze zapsat v HTML korektně, avšak v XHTML ne, protože onLoad obsahuje v názvu atributu velké písmeno. Základní struktura dokumentu v XHTML vypadá následovně:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Titulek stránky</title>
    </head>
    <body> </body>
</html>
```

Klíčové slovo **!DOCTYPE** definuje, jaké DTD[5] bude použito při psaní dokumentu. DTD uvádí, které značky a jejich atributy, lze při psaní dokumentu použít. Kořenový element <**html xmlns="http://www.w3.org/1999/xhtml"></html**> určuje začátek a konec textu ve formátu XHTML. V elementu <**head**> lze definovat například titulek dokumentu, kódování, styl zobrazení, klíčová slova či jazyk. Tělo dokumentu je vždy vymezeno párovou značkou <**body**>.

## 1.3 Jazyk PHP

PHP (rekurzivní zkratka PHP: Hypertext Preprocessor, „PHP: Hypertextový preprocesor“, původně Personal Home Page) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML,

[5] DTD (Document Type Description) – jazyk pro popis struktury XML případně SGML dokumentu, viz [7]

XHTML či WML[6], což je velmi výhodné pro tvorbu webových aplikací. PHP lze ovšem také použít i k tvorbě konzolových a desktopových aplikací.

### 1.3.1 Historie

Od roku 1994 je PHP jedním z nejpoužívanějších způsobů tvorby dynamicky generovaných WWW stránek. Jeho tvůrce (Rasmus Lerdorf) jej vytvořil pro svou osobní potřebu přepsáním z Perlu[7] do jazyka C. Sada skriptů byla vydána ještě v témže roce pod názvem Personal Home Page Tools, zkráceně PHP.

V polovině roku se systém PHP spojil s programem Form Interpreter stejného autora. Tak vzniklo PHP/FI 2.0. Zeev Suraski a Andi Gutmans v roce 1997 přepsali parser a zformovali tak základ PHP3. Současně byl název změněn na dnešní podobu PHP hypertext procesor. PHP3 vyšlo v roce 1998, bylo rychlejší, obsahovalo více funkcí. Také běželo i pod operačním systémem Windows.

V roce 2000 vychází PHP verze 4, o čtyři roky později pak verze 5 s vylepšeným objektovým přístupem, podobným jazyku Java.

### 1.3.2 Princip

PHP skripty jsou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě, skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (mj. MySQL[8], ODBC[9], Oracle, …), podporu celé řady internetových protokolů (HTTP, SMTP[10], SNMP[11], FTP[12], IMAP[13], POP3[14], …).

---

[6] WML (Wireless Markup Language) – značkovací jazyk založený na jazyce XML umožňující tvorbu online dokumentů pro mobilní zařízení

[7] Perl je interpretovaný programovací jazyk, populární nástroj pro tvorbu CGI skriptů

[8] MySQL – multiplatformní databáze, komunikace s ní probíhá pomocí jazyka SQL

[9] Microsoft ODBC (Open Database Connectivity) – standardizované softwarové API (rozhraní pro programování aplikací) pro přístup k databázovým systémům

[10] SMTP (Simple Mail Transfer Protocol) – protokol určený pro přenos zpráv elektronické pošty

[11] SNMP (Simple Network Management Protocol) – protokol sloužící k potřebám správy sítí

[12] FTP (File Transfer Protokol) – protokol aplikační vrstvy z rodiny TCP/IP určen pro přenos souborů

[13] IMAP (Internet Message Access Protocol) – protokol pro přístup k e-mailovým schránkám

[14] POP3 (Post Office Protocol ver. 3) – protokol používající se pro stahování emailů ze serveru na klienta

PHP se stalo velmi oblíbeným především díky jednoduchosti použití a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi. V kombinaci s databázovým serverem (především s MySQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Díky velmi častému nasazení na serverech se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP nebo Perl.

### 1.3.3 Základ vytváření skriptů

PHP skripty se vkládají přímo do zdrojového kódu HTML (XHTML). Jsou ohraničeny značkami <**?php** a **?**>. Kód uvnitř těchto značek je při zpracovávání stránky předán interpretu jazyka PHP. Skript lze také vytvořit jako samostatný soubor s příponami například **.php**, **.php3** nebo **.phtml**. Ten je pak možné pomocí příkazu **require** vložit opět do HTML stránky. Následující příklad znázorňuje tyto dvě možné varianty.

Obsah souboru **script.php**:

```
<?php
    $string2 = "světe";
    echo $string2;
?>
```

Obsah HTML souboru:

```
<!DOCTYPE …>
<html>
    <head> … </head>
    <body>
        <?php
            $string1 = "Ahoj";
            echo $string1;
            require("script.php");
        ?>
    </body>
</html>
```

Při spuštění takto vytvořené HTML stránky v internetovém prohlížeči, dojde ke zobrazení dvou slov: „Ahoj světe“, kde „Ahoj“ je výsledek zpracování skriptu přímo v HTML kódu a „světe“ je výsledek zpracování souboru script.php interpretem PHP.

### 1.3.4 Session

Session slouží k trvalejšímu uchování hodnot. Díky nim tak může být proměnné uložena hodnota, bez toho, aniž by bylo potřeba její hodnotu přenášet. Session je standardně uložena na WWW serveru a není, z důvodu bezpečnosti, přístupná klientovi. Lze je využít například při autentizaci uživatele, tvorbě elektronického obchodu a dalších aplikací.

## 1.4 Databáze

Databáze je organizovaná skupina dat, skládající se z tabulek. Ta může být představována jediným fyzickým souborem, jako je tomu například v případě Microsoft Access, může jich však být i více, jako je to možné u serveru Oracle.

Někdy je možné setkat se s pojmem databáze jako s pohledem na skupinu dat, kterou představuje pouze jediná lokální tabulka.

### 1.4.1 Princip

Nejběžnější formou, jak jsou v současné době data uložena, je model databáze relační[15].

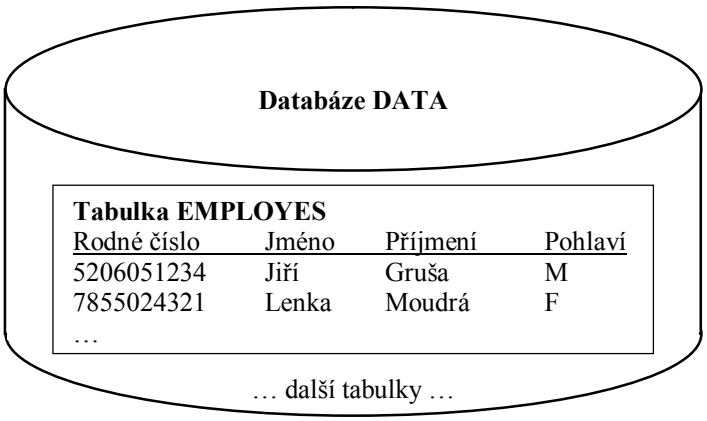
Vlastní data se pak člení do subcelků, které se nazývají tabulky. Každá tabulka pak může obsahovat v podstatě libovolné množství záznamů – řádků (databázový systém vždy velikost omezuje, v případě větších systémů je to však velmi vysoké číslo). Každý záznam (řádek) si pak lze představit jako jedinou logickou jednotku, která sestává ze sloupců, jež v konečné fázi popisují jednotlivé informace daného záznamu. Sloupce postihují záznam podobně jako atributy vlastnosti objektu. Mohou být taktéž různých typů: řetězec, číslo. datum apod.

Nejlépe lze vše ilustrovat na následujícím příkladu:

Na stanici STATION existuje relační databáze DATA, která obsahuje určité množství tabulek (například 10), z nichž jedna nese informace o zaměstnancích podniku. Název této tabulky je EMPLOYES.

Jak je vidět, zmíněná tabulka EMPLOYES, která zde slouží za příklad, obsahuje pro každý záznam (řádek) data jediného pracovníka (struktura se obecně určuje při zakládání tabulky). Tato data jsou určena sloupci pojmenovanými Rodné číslo, Jméno, Příjmení a Pohlaví, která každého zaměstnance (zde velmi stručně) identifikují. Podobných tabulek je v databázi samozřejmě více.

[15] Relační databáze je databázový systém, který je založen na relačním modelu dat a relační algebře




**Příklad databáze – Databáze DATA uložená například v souboru DATA.x**

Vedle tohoto modelu existují rovněž již zmíněné lokální tabulky, které nejsou uloženy v žádné databázi. Rozdílem oproti předchozímu modelu je to, že tabulky nejsou „obaleny“ databází, nýbrž vystupují zcela samostatně jako soubory na disku – lze je například kopírovat. Nutno říci, že tento model je v současné době používán spíše výjimečně, u složitých projektů téměř nikdy. Lze jej však velmi dobře využít pro jednodušší aplikace a demonstrační účely.

#### 1.4.1.1 Primární klíč

Při práci s databázovými tabulkami je výhodné mít alespoň jeden sloupec, jehož hodnota bude jednoznačně identifikovat záznam v tabulce. Pokud tato hodnota není příliš velká (například v počtu bajtů), zvolíme ji za tzv. primární klíč. Primárním klíčem může být pole nebo kombinace polí, jednoznačně identifikující každý záznam v databázové tabulce a žádné pole které je součástí nesmí obsahovat hodnotu NULL. Vezmeme-li v úvahu předchozí příklad, lze za primární klíč označit sloupec Rodné číslo, protože neexistují dva občané, kteří by měli rodné číslo shodné, tedy pro žádné dva řádky v této tabulce nemůže nastat situace, že by hodnota primárního klíče byla totožná.

#### 1.4.1.2 Index

Obsahuje-li tabulka velké množství řádků, ve kterých dochází k častému vyhledávání, je účelné vytvořit index. Ten popisuje pozici specifických vět a nesrovnatelně urychlí proces hledání. Cenou za to je však spotřeba systémových prostředků (výkon i místo na disku), které musí systém vynaložit na jeho neustálé udržování (přidání věty, smazání věty je totiž doprovázeno úpravou tvaru indexu).

U klasických databází je index skrytý, programátor jej pouze připraví většinou již při deklaraci struktury tabulky (obecně jej však lze přidat i následně); v případě lokálních tabulek se vyskytuje ve formě dalšího fyzického souboru, který je odlišen pouze jinou koncovkou. Ten musí být rovněž kopírován při případném přemístění tabulky – jeho nepřítomnost způsobí totiž běžně chybu.

#### 1.4.1.3 Entita, atributy

Databázi si lze představit jako nějaký soubor dat, který popisuje část reálného světa (například kartotéka, kniha jízd apod.). Entita je pak prvek reálného světa, který je popsán svými vlastnostmi. Tyto vlastnosti se označují jako atributy. U člověka by to bylo například: jméno, příjmení, pohlaví, věk, bydliště atd.

#### 1.4.1.4 Relace

Velmi důležitým pojmem je relace neboli vazba mezi entitami. Jednotlivé entity odpovídají prvkům reálného světa, které mezi sebou mají určitý vztah. Tento vztah musí být zachován i v databázovém systému. Například každý člověk má svůj právě jeden občanský průkaz. Zde mluvíme o vazbě 1:1.

Druhou vazbou je vazba 1:N, která odpovídá situaci, kde jeden člověk může vlastnit více aut, ale jedno auto má pouze jednoho vlastníka.

Poslední vazba je typu N:M. V tomto případě není kardinalita vztahu nijak omezena. Tato situace je například u registrace předmětů studenty. Jeden student si může zapsat několik předmětů a daný předmět může si může zapsat několik studentů.

### 1.4.2 Architektury

Postupem času začaly vzrůstat požadavky na množství uložených dat a počet připojených klientů. Toto vedlo ke změně přístupu a postavení některých prvků v architektuře.

#### 1.4.2.1 Jednovrstvá

Jedná se o architekturu nejstarší, kde veškerá komunikace probíhá na jedné vrstvě. Nezřetelné a nepředvídatelné interakce mezi různými částmi aplikace v této architektuře způsobují, že vývojář každé části musí znát i zbytek celé aplikace. Důsledkem toho je při rozšiřování aplikací exponenciálně vzrůstající doba vývoje, nákladů a zvyšování rizika. Jejich výhodou je však stabilita, spolehlivost a bezpečnost.

#### 1.4.2.2 Dvouvrstvá – klient/server

Tato architektura je nejčastěji používaná systémy s centrálním úložištěm dat. Veškerá data jsou uložena na konkrétním centrálním počítači. Jednotliví klienti, pracující na svých lokálních počítačích různě umístěných v síti, kteří se pak připojí k serveru. Ten zpracuje jejich požadavky a vrátí výsledek. Požadavky narozdíl od jednovrstvé architektury mají velmi malou velikost.

Takto vytvořený přístup je velmi rychlý a efektivní. Výhodou je, že server dokáže obsloužit až tisíce požadavků najednou. Toto však razantně zvyšuje požadavek na výpočetní výkon serveru. Nevýhodou u této architektury je, že při výpadku serveru nejsou klienti schopni nijak získávat data.

#### 1.4.2.3 Třívrstvá

Třívrstvá architektura představuje rozšíření modelu klient/server. Rozděluje komponenty aplikační logiky a datového přístupu pomocí segmentace aplikace na aplikační a datové servery. Na klientech pak zůstává presentační vrstva popř. část aplikační logiky. Komponenty přístupu na data mohou být rovněž umístěny na aplikačních serverech. Vzniká tak oproti dvouvrstvé architektuře navíc prostřední vrstva. Ta je složena z komponent funkcionality a zajišťujících sdílené služby tzv. middleware.

Samotná aplikace běží na aplikačním serveru, nikoliv na databázovém. Databázový požadavek je odeslán z počítače klienta přes aplikační server na databázový. Ten dotaz zpracuje a informace jsou odeslány klientovi prostřednictvím aplikačního serveru. Výkon lze ještě zvýšit tím, že se omezí počet přenosů (dotazů) mezi databázovým serverem a klientem.

### 1.4.3 Jazyk SQL

SQL (Structured Query Language, strukturovaný dotazovací jazyk) je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích..

Pro práci s jazykem SQL a dotazy na databázi je nutné se připojit na databázový server. Po připojení je klientovi nabídnuta příkazová řádka, kde může provádět dotazy. Tento jazyk je deklarativní, tedy zajímá nás pouze výsledek. Ne to, jak je tento dotaz proveden, i když v některých případech je potřeba znát systém, jak se daný konkrétní dotaz provede z hlediska efektivity.

Jazyk SQL se dělí do několika částí. První z nich je DDL[16]. Jedná se o jazyk pro vytváření databázových schémat a katalogů. Částí druhou je SDL[17], definující, jak a jakým způsobem se budou tabulky do databáze ukládat. Další část jazyka SQL je VDL[18], který určuje, jakým způsobem se budou vytvářet pohledy. Poslední a zároveň nejdůležitější je část jazyka SQL zvaná DML[19]. Ten definuje základní příkazy jako jsou SELECT, DELETE, UPDATE a INSERT, kterými lze manipulovat s daty. Například pro vybrání všech informací o mužských zaměstnancích z tabulky zaměstnanců EMPLOYES lze napsat :

```
SELECT * FROM employes WHERE pohlavi = 'M';
```

[16] DDL (Data Definition Language)

[17] SDL (Storage Definition Language)

[18] VDL (View Definition Language)

[19] DML (Data Manipulation Language)

### 1.4.4 MySQL

Produkt MySQL je bezplatný systém správy relačních databází. Pro svou širokou využitelnost na platformě Linux či Unix si získal tento produkt výsadní postavení díky své pružnosti, rychlosti a spolehlivosti. Komunikace s touto databází probíhá na pomocí jazyka SQL. Podobně jako u ostatních databází je tento dialekt jazyka SQL s některými rozšířeními.

MySQL umožňuje práci více uživatelů najednou s dobrými možnostmi zabezpečení. Jednotlivé tabulky lze mezi sebou různě provázat. Toto je možné docílit pomocí více vláken. Každé vlákno označuje separátnost jednoho každého procesu a dotazu, zpracovávaného serverem.

## 1.5 Databázová rozhraní v Delphi

Rozlišujeme 2 typy přístupu do databází:

- UniDirectional – záznamy lze procházet pouze dopředu, nelze například přistoupit k záznamu číslo 15 nebo se ze záznamu 28 vrátit na 27. Příkladem tohoto přístupu je dbExpress. Návrh takovýchto aplikací je mírně složitější (problémy například s automatickým ukládáním změn). Je nutno: omezit se pouze na procházení dopředu, uložit do nějaké paměti (k tomu lze použít například komponentu ClientDataSet)
- Directional – je nabízen přímý přístup k záznamům. Tato vlastnost existuje například v lokálních databázích, například BDE. Návrh je jednodušší.

### 1.5.1 Databázová architektura v Delphi

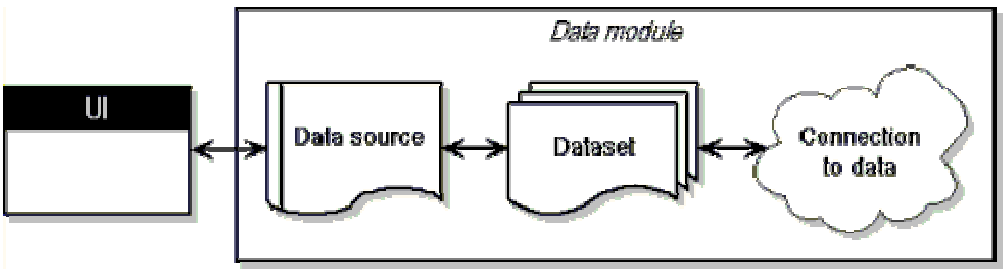



**Schéma databázové architektury v Delphi**

#### 1.5.1.1 Data module

Datový modul v Delphi vytváříme jako místo pro uložení nevizuálních datových komponent, poté mohou být všechny takto vložené komponenty přístupné ze všech formulářů (= jednotek) pomocí klasické tečkové notace název_modulu.název_nevizuální_komponenty.

#### 1.5.1.2 Třída TDataSet

Třída TDataSet je základní třídou (předkem) všech komponent, která zprostředkovává získávání dat z databáze. TDataSet se otevírá metodou Open, zavírá Close. Dále jsou k dispozici metody:

- pro přímý přístup k záznamu (řádku tabulky) – Locate (nepodporují UniDirectional komponenty)
- pro přímý přístup k atributu (sloupci aktuálního řádku) – Metoda FieldByName vrací instanci třídy TField (například TIntegerField, TBlobField – v závislosti na doméně atributu) pro textové vyjádření lze použít virtuální metody třídy TField – AsString; obdobně existuje například AsInteger. Příklad: Query.FieldByName('movie').AsString:='terminator';
- pro procházení záznamy: First, Last, Next, Prior
- pro editaci záznamů: Post, Edit, Append, Delete
- pro práci s bookmarky: například GetBookmark, GotoBookmark

#### 1.5.1.3 Potomci TDataSetu

Od třídy TDataSet jsou odvozeny tyto třídy:

- **dotazovací třídy** – třídy umožňující pokládat dotazy do databáze, název této třídy se liší podle přístupu - například pro ADO existuje třída TADOQuery. Dotazy se vkládají do vlastnosti SQL.
- **tabulkové třídy** – třídy zprostředkovávající procházení celé tabulky, například pro ADO existuje třída TADOTable.
- **uložené procedury** – třídy umožňující spouštění uložených procedur v databázi a procházení jejich výsledků, například pro BDE existuje třída TStoredProc, pro dbExpress existuje třída TSQLStoredProc.

### 1.5.2 BDE

BDE (Borland Database Engine) byl navržen především pro práci s lokálními databázemi (množina tabulek uložených v souborech na lokálním nebo síťovém disku).

Přestože se již řadu let vede diskuse o kvalitách a výkonu tohoto databázového stroje, stále má v Delphi poměrně neochvějnou pozici. Důvodem je asi především fakt, že BDE obsahuje snad nejširší paletu podpůrných nástrojů a funkcí pro práci s nejrůznějšími databázemi. Také jeho používání je relativně jednoduché, ovšem na druhou stranu je trochu obtížnější šíření a instalace aplikací postavených na BDE.

### 1.5.3 ADO

Další možný přístup k datům se skrývá pod názvem ADO (ActiveX Data Objects). Funguje na principu sady objektů COM, které přistupují do databáze prostřednictvím rozhraní OLEDB (případně také ODBC). Také v případě ADO existuje celá řada ovladačů pro nejrůznější databázové platformy. Chceme-li úspěšně provozovat aplikaci postavenou na principu ADO, je zapotřebí Microsoft ADO 2.1 (ADO je technologií společnosti Microsoft), rozhraní OLEDB (příp. ovladač ODBC), pro databázové servery postavené na SQL příslušné klientské vybavení, a samozřejmě především databázi.

### 1.5.4 dbExpress

Jednou z hlavních charakteristik a zároveň výhod této nové databázové knihovny je její nezávislost na platformě a její nekomplikovanost. Je k dispozici jak pro Windows, tak pro Linux. Společnost Borland ji zavedla spolu s vývojem projektu Kylix[20], právě z výše uvedených důvodů. Další z jejích výhod je ta, že není potřeba (například na rozdíl od BDE) instalovat na klientských počítačích žádný další software potřebný pro její běh a jakkoli tyto počítače konfigurovat.

Knihovna dbExpress podporuje řadu databázových serverů, a to Interbase od společnosti Borland, MySQL, Oracle, Informix, DB2 od IBM a SQL Server od Microsoft. Ovladače si může napsat každý vlastní tak, aby mu přesně vyhovovaly. Nutno podotknout, že to není nijak zvlášť obtížné.

V porovnání s „konkurenčními“ architekturami pro přístup k datům, jako je například BDE, ADO, má knihovna dbExpress lépe vytvořenou architekturu, z čehož vyplývá, že máme dostatek možností, co s daty dělat a nejsme ničím omezováni a vázáni.

Zároveň se díky komponentám, které zajišťují uchovávání dat v lokální paměti, stává architektura dbExpress velmi silným nástrojem pro přístup k datům a umožňuje nám snadnou manipulaci s nimi.

[20] Kylix – integrované vývojové prostředí firmy Borland, určené pro vývoj aplikací v operačním systému GNU/Linux, podobný produktům Delphi a C++Builder téže firmy, určeným k vývoji ve Windows

# 2 Popis řešení

## 2.1 Funkce aplikace

Hlavním cílem bylo vytvořit plánovač se správou úkolů, hierarchií uživatelů pro jejich zadávání a s centrální databází úkolů dostupnou z internetu. To vše spolu s tenkým i tlustým klientem s bohatými možnostmi upozorňování. Pro tlustého klienta navíc ještě kalendář s možností listování, budík a režim offline s možností synchronizace.

## 2.2 Návrh a struktura databáze

Jádrem celého systému je právě MySQL databáze, která obsahuje veškeré informace o uživatelích a jejich úkolech. Struktura databáze není nijak složitá, jedná se o dvě tabulky – **uzivatele** a **ukoly**.

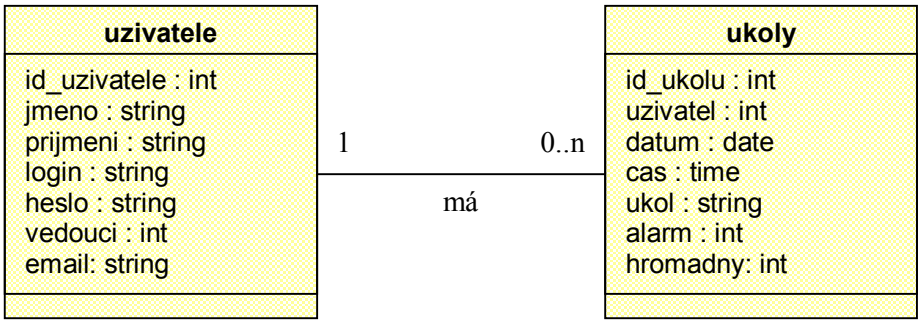
### 2.2.1 Integritní omezení

Referenční integritu je v MySQL možné zajistit využitím typů tabulek InnoDB, ve kterých lze definovat integritní omezení například kaskádní mazání hodnot, kaskádní úprava hodnot, zamezení smazání hodnot, které jsou využívány, apod.

Toto podstatně ulehčuje práci, jelikož tyto situace nemusí programátor ošetřovat ručně. Existují však i další omezení která bylo třeba ošetřit programově. To se týká například kontroly dat vkládaných do databáze apod.

Naneštěstí úložiště dat typu InnoDB se mi nepodařilo zprovoznit (více v kapitole 2.4 Problémy během řešení). Proto je i kaskádní mazání hodnot řešeno programově. Jedná se vlastně jen o případ, kdy se při odstranění uživatele odstraní i všechny jeho úkoly.

### 2.2.2 Diagram tříd



### 2.2.3 Návrh a popis tabulek

Tabulky jsou navrženy tak, aby nevznikala redundantnost, tedy duplicitní hodnoty a data byla konzistentní, pokud je to možné. Datové typy jednotlivých sloupců jsou voleny tak, aby vyhovovaly aktuální potřebě a byly částečně naddimenzované.

Tabulka **uzivatele** – informace o jednotlivých uživatelích systému:

| id_uzivatele | SMALLINT | PRIMARY KEY, UNSIGNED,<br>NOT NULL, AUTO_INCREMENT |
|---|---|---|
| jmeno | VARCHAR(15) | NOT NULL |
| prijmeni | VARCHAR(20) | NOT NULL |
| login | VARCHAR(5) | NOT NULL |
| heslo | VARCHAR(10) | NOT NULL |
| vedouci | TINYINT | NOT NULL |
| email | VARCHAR(50) | NOT NULL |

Tabulka **ukoly** – informace o jednotlivých úkolech uživatelů systému :

| id_ukolu | INT | PRIMARY KEY, UNSIGNED,<br>NOT NULL, AUTO_INCREMENT |
|---|---|---|
| uživatel | SMALLINT | FOREIGN KEY, UNSIGNED, NOT NULL |
| datum | DATE | NOT NULL |
| cas | TIME | NOT NULL |
| ukol | VARCHAR(30) | NOT NULL |
| alarm | TINYINT | NOT NULL |
| hromadny | TINYINT | NOT NULL |

## 2.3 Návrh aplikace

Celý systém sestává z tlustého a tenkého klienta, vzájemně nezávislých, ale závislých právě na vzdálené databázi MySQL.

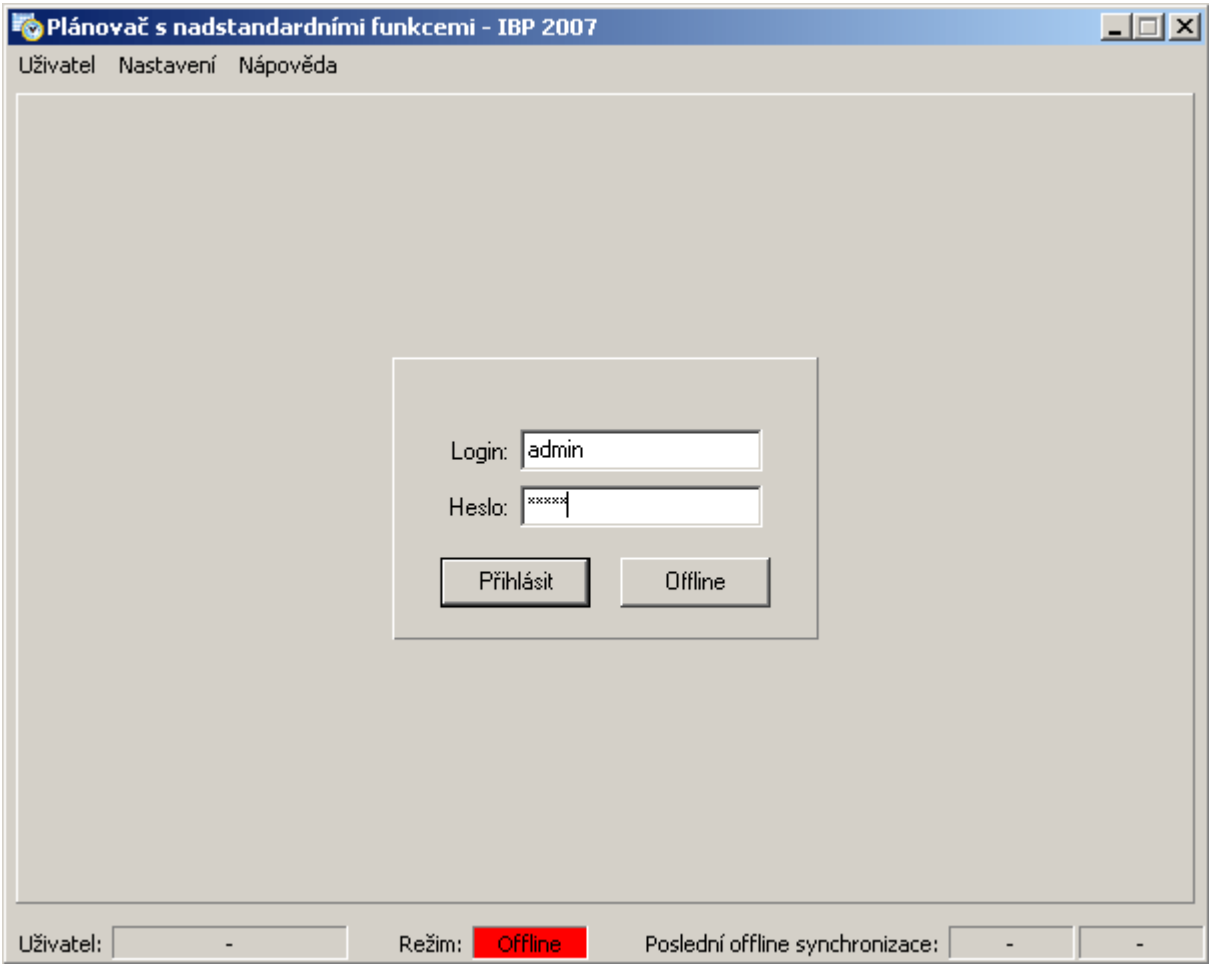
### 2.3.1 Tlustý klient

Tlustým klientem je aplikace běžící pod OS Windows. Tato aplikace je výsledným produktem kompilace zdrojového kódu vytvořeného v jazyce Object Pascal v implementačním prostředí Delphi 7.

Hlavní rámec okna je tvořen uživatelským menu v horní části, které umožňuje větvení programu a stavovým řádkem v části spodní, který obsahující všeobecné informace. Celou aplikaci je dle nastavení možné minimalizovat na pozadí do tray lišty, takže nepřekáží v panelu úloh přičemž je plně funkční. Dále podrobně popíši tři hlavní části aplikace.

#### 2.3.1.1 Přihlašovací okno

Zobrazí se po spuštění programu.

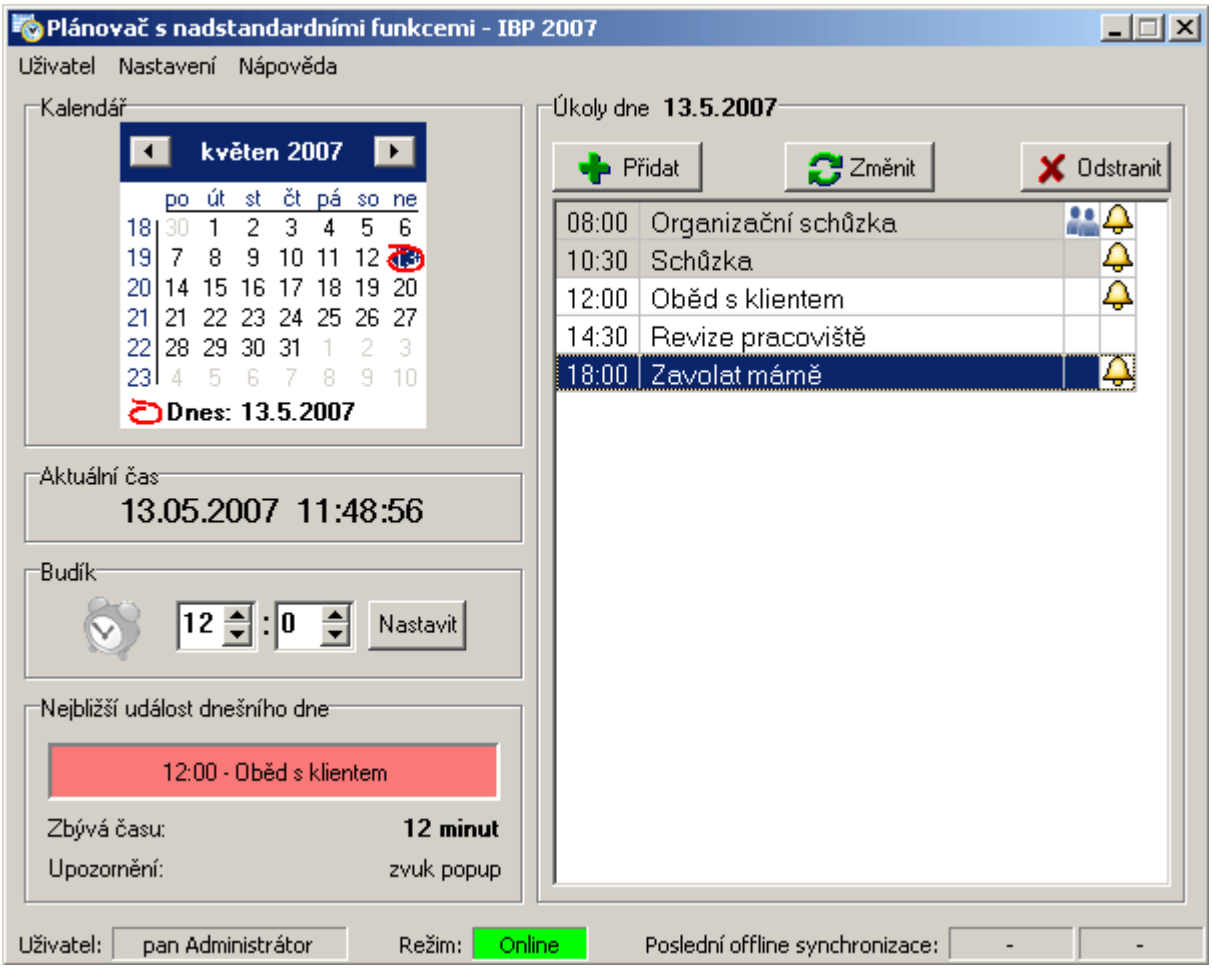



**Tlustý klient – Přihlašovací okno**

Jeho hlavní funkcí je porovnání přihlašovacích údajů zadaných uživatelem se záznamy ve vzdálené databázi v tabulce *uzivatele* a v případě shody údajů se po stisknutí tlačítka **Přihlásit** uskuteční připojení v režimu **Online**. Současně se v adresáři s aplikací vytvoří soubor *user.dat*, do kterého se uloží údaje o přihlášeném uživateli včetně hesla v zakódované formě, pro pozdější možnost připojení v režimu **Offline**.

#### 2.3.1.2 Hlavní okno

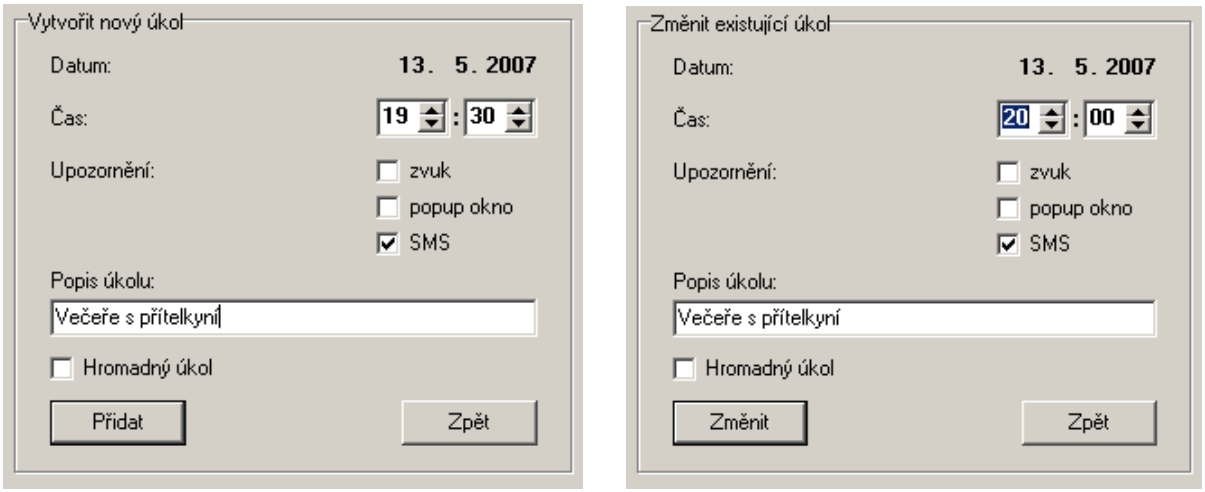
Je nejdůležitější částí aplikace.




**Tlustý klient – Hlavní okno v režimu Online**

Jádrem hlavního okna je kalendář, hodiny a seznam úkolů dne. Z nich jsou odvozeny funkce:

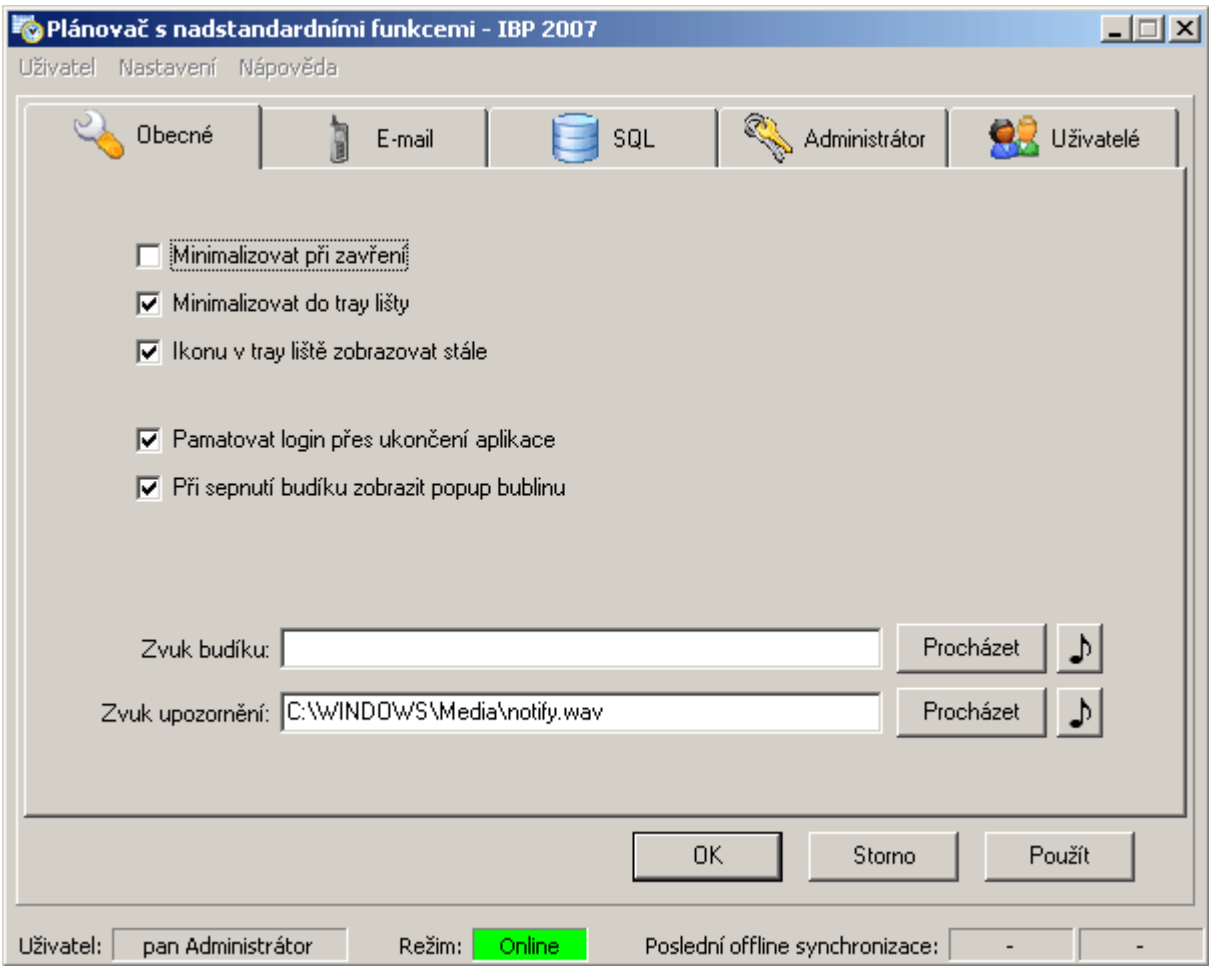
- **Aktuální čas** – odvozen od systémového času, aktualizující se každých 1000 ms
- **Budík** – lze nastavit s přesností na minuty
- **Nejbližší událost dnešního dne** – aktualizována každou minutu
- **Úkoly dne** – aktualizovány každou minutu daty ze vzdálené databáze (režim Online), správu těchto úkolů umožňují tlačítka **Přidat**, **Změnit** a **Odstranit**.




**Tlustý klient – Správa úkolů**

### 2.3.1.3 Možnosti nastavení

Zobrazí se po vybrání položky **Možnosti** z menu **Nastavení**.




**Tlustý klient – Nastavení aplikace**

Pomocí záložek v záhlaví tohoto okna se může uživatel přepínat mezi následujícími kategoriemi:

- **Obecné** – nastavení chování okna v rámci operačního systému, zvuků apod.
- **E-mail** – přímá změna emailové adresy přihlášeného uživatele v databázi
- **SQL** – nastavení parametrů pro připojení k databázi MySQL (host, jméno databáze, uživatelské jméno a heslo)

Do dalších kategorií má přístup pouze administrátorský účet:

- **Administrátor** – nastavení parametrů pro připojení k SMTP serveru, úklid serveru
- **Uživatelé** – správa uživatelských účtů

#### 2.3.1.4 Funkce synchronizace v režimu Offline

Funkce je dostupná z uživatelského menu v režimu Offline.

Uživatel ji použije v případě, kdy nemůže být trvale připojen k databázi. Její spuštění nahraje úkoly týkající se přihlášeného uživatele ze vzdáleného serveru do lokálního databázového souboru, kde s nimi uživatel pracuje, jakoby byl připojen ke vzdálené databázi MySQL.

Pokud uživatel vytvoří v režimu Offline nový úkol, je třeba jej opět funkcí synchronizace nahrát do vzdálené databáze. Do té doby je tento nový úkol v seznamu zvýrazněn.

#### 2.3.1.5 Hierarchie uživatelů

V systému existují tři typy uživatelů:

- **administrátor**
- **vedoucí**
- **běžný uživatel**

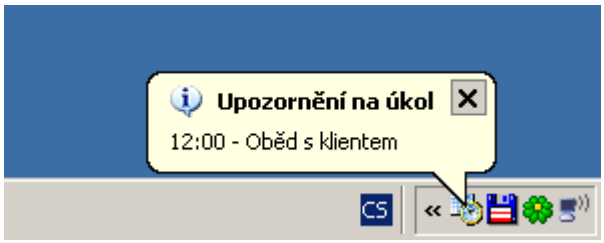
Administrátorský účet je vytvořen již při tvorbě databáze a programově jej nelze odstranit. Aplikace přihlášená k tomuto účtu je určena pro nepřetržitý provoz. To z toho důvodu, že administrátorská aplikace hlídá všechny takové úkoly ve vzdálené databázi, na které je třeba upozornit odesláním zprávy na mobilní e-mail a tyto zprávy také odesílá. Administrátor jako jediný může měnit nastavení parametrů pro připojení k SMTP serveru a spravovat uživatelské účty.

Vedoucí se od běžného uživatele liší pouze skutečností, že může zadávat, měnit a mazat hromadné úkoly.

#### 2.3.1.6 Upozorňování

Aplikace podporuje tři druhy upozorňování:

- **zvukové** – v možnostech nastavení možno zvolit libovolný wave soubor, pokud není žádný zvolen dochází s přehrávání výchozího
- **popup bublina** – uživatel využije především při běhu aplikace na pozadí
- **SMS** – upozornění odesílané na mobilní e-mail z administrátorské aplikace (viz 2.3.1.5 Hierarchie uživatelů)



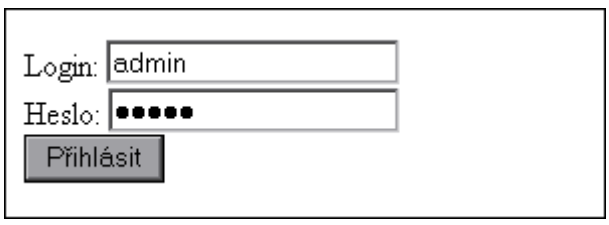

**Tlustý klient – Upozornění popup bublinou**

## 2.3.2 Tenký klient

Tenkým klientem je v tomto případě dynamická webová stránka PHP. Je speciálně optimalizovaná pro zobrazování na mobilních zařízeních, tedy i objem přenášených dat mezi mobilním zařízením a internetovou sítí je minimální.

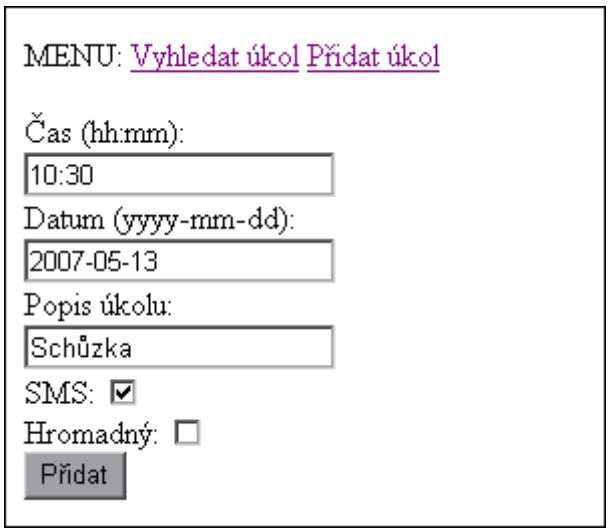
Celá dynamická webová stránka sestává z pěti částí:

- **index.php** je hlavní část a volá v případě potřeby části ostatní
- **connect.php** obsahuje údaje potřebné pro připojení k databázi MySQL a zprostředkovává připojení
- **login.php** obsahuje formulář k zadávání přihlašovacích údajů uživatele a jejich ověření



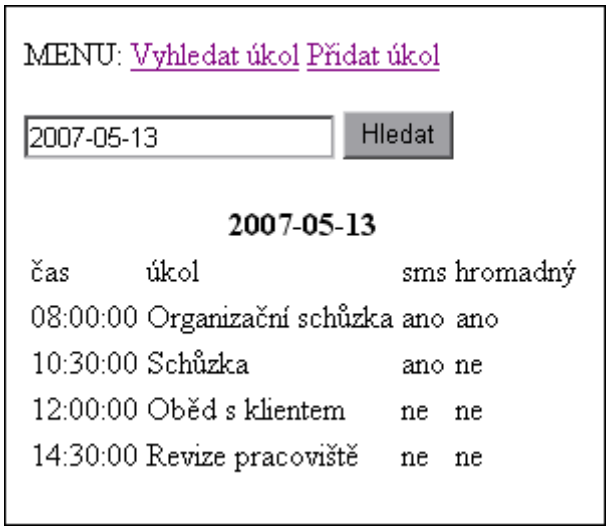

**Tenký klient – Úvodní přihlašovací formulář**

- **ukol.add.php** obsahuje formulář pro přidávání nových úkolů




**Tenký klient – Formulář pro přidání nového úkolu**

- **ukol.search.php** obsahuje formulář pro vyhledávání úkolů a zprostředkovává jejich výpis k zadanému datu




**Tenký klient – Formulář pro vyhledání úkolů**

Pro používání tenkého klienta stačí pouze všechny tyto části dynamické webové stránky umístit na jakýkoliv webový server s podporou PHP skriptů a na něj se následně připojit například z mobilního zařízení. Tenký klient lze použít také v případě, že se nacházíte u počítače, kde je přístup na internet omezen pouze na prohlížení webových stránek a tedy se z tlustého klienta nemůžete připojit na databázi MySQL.

## 2.4 Problémy během řešení

Během řešení jsem narazil na množství technických i logických problémů. Tyto zřejmě patří k vývoji a implementaci každé aplikace a snad ani nestojí za zmínku. Postupně se mi je pomocí podrobnějšího studování implementačního prostředí a drobných přepracování dílčích částí návrhu podařilo poměrně rychle překonat.

Ovšem časově nejnáročnější problém jsem byl nucen řešit hned v úvodu implementace. Jednalo se o nekompatibilitu implementačního prostředí Delphi 7, které mám k dispozici, s oficiálně dostupnými verzemi databázového serveru MySQL, tedy verze 4.1 a vyšší. Po dlouhém hledání jsem zjistil, že Delphi 7 je kompatibilní pouze s MySQL knihovnami verze 3.23. Protože tuto verzi už není možné na oficiálních stránkách stáhnout, musel sem hledat v peer-to-peer sítích, kde ji naštěstí ještě několik uživatelů vlastnilo a s Delphi 7 bylo dle předpokladů plně funkční.

V zápětí jsem zase zjistil, že verze 3.23 má velmi chatrně zpracovanou správu tabulek InnoDB a ani přesným postup podle manuálu [3], kde jsem dokonce objevil několik podstatných nepřesností, se mi nepodařilo nastavit tabulkám integritní omezení. Proto jsem se rozhodl nahradit integritní omezení ON DELETE CASCADE programovým ošetřením.

# 3 Vlastní implementace

## 3.1 Připojení a komunikace s MySQL

### 3.1.1 TSQLConnection

Komponenta umožňující připojení na vzdálený SQL server - např. mySQL, MSSQL. Jednou z nejdůležitějších vlastností objektu je vlastnost **Params** obsahující parametry spojení (login/databáze/heslo/izolace transakcí) ve formě seznamu. Dále je nutno nastavit **DriverName** (typ SQL serveru. Nastavení vlastnosti Connected v object-inspectoru proběhne okamžité připojení k serveru (již v době návrhu formuláře). Nastavením vlastnosti **LoginPrompt** na false lze vypnout přihlašovací dialog.

### 3.1.2 TSimpleDataSet

Zapouzdřuje komponenty DataSetProvider, ClientDataSet a SQLDataSet (obsahuje interně vlastnost **DataSet**). Nutno nastavit vlastnost **Connection** a v interním DataSetu nastavit také nejdůležitější vlastnost **CommandText**, tedy SQL příkaz, který se má provést. Umí rovněž pracovat přímo s SQL soubory.

### 3.1.3 TDataSource

Je rozhraním mezi vizuálními komponentami ze záložky **Data Controls** a instancí třídy TDataSet (což lze chápat jako obraz databáze – paměť) - vlastnost **DataSet**. Jako DataSet lze použít například výsledek nějakého SQL dotazu.

### 3.1.4 TDBGrid

je vizuální komponentou a zobrazuje údaje z DataSource ve formě tabulky na obrazovce. Důležité je, že údaje jsou „živé“, že je lze editovat a že při správné konstelaci vlastností připojených komponent je tak možné změny přenášet přímo do fyzické databáze.

Tuto komponentu je velmi vhodné kombinovat s komponentou **DBNavigator**. Nejdůležitější vlastností je **DataSource**, ve které musíme přiřadit komponentě DBGrid příslušný zdroj dat.

## 3.2 Hlavní komponenty

### 3.2.1 TTimer

Jednou z nejužitečnějších komponent je komponenta Timer na paletě Systém. Jak z názvu vyplývá, tato komponenta slouží k časování. Je-li vlastností **Enabled** zapnut, způsobuje pravidelné volání události **OnTimer** v určitém časovém cyklu, nastaveného vlastností **Interval**.

### 3.2.2 TMonthCalendar

Komponenta s bohatými možnostmi práce s datem. Disponuje velmi elegantním designem. Nejdůležitější vlastnost **Date** obsahuje datum právě vybraného dne v kalendáři.

### 3.2.3 TMainMenu

Pro práci s hlavní nabídkou se využívá komponenta MainMenu. Ikonku objektu lze umístit kdekoliv do formuláře, neboť po spuštění programu není objekt vidět. Jen v horní části okna se objeví jednotlivé nabídky, které jsme v hlavním menu vytvořili.

### 3.2.4 TStatusBarPro

Jedná se o stavový řádek umístěný ve spodní části formuláře. Aplikace vypadá mnohem lépe, je-li rozšířena o stavový pruh, díky kterému je uživatel informován o aktuálně platných volbách, případně o tom, co se právě děje.

TStatusBarPro je rozšířená verze komponenty TStatusBar. Každý panel může obsahovat obrázek, bublinkovou nápovědu, událost na kliknutí nebo dvojklik. Umožňuje vkládat na sebe další prvky.

### 3.2.5 TCoolTrayIcon

Komponenta, které umožňuje velmi snadnou práci s ikonou aplikace na hlavním panelu vedle hodin (lišta tray). Tato ikonka, která obvykle zastupuje programy, jež neustále běží na pozadí, může velmi snadno urychlit přístup k některým funkcím aplikace a zároveň podávat užitečné informace.

Tato komponenta nabízí kromě celkem obvyklých funkcí, které podobné komponenty mají, ještě několik drobných vylepšení navíc.

## 3.3 Odesílání e-mailů

### 3.3.1 TIdSMTP

Slouží k odeslání pošty. Najdeme ji na záložce Indy Clients. Samotné odeslání zprávy je provedeno pomocí metody TIdSMTP.Send(), kdy je jako parametr přejímán již připravený objekt TIdMessage.

Je třeba nastavit vlastnosti připojení k SMTP serveru **Host**, **Port**, případně také **AuthenticationType**, **Username** a **Password**.

### 3.3.2 TIdMessage

Slouží jako objekt pro zprávu jako takovou, tedy pro uložení těla zprávy, všech potřebných parametrů, příloh a podobně. Najdeme ji na záložce Indy Misc. Tento objekt je pak předáván jako parametr komponentě IdSMTP.

Hlavní vlastnosti jsou tedy **From**, **Recipients**, **Subject**, **Body** a **Attachments**.

# Závěr

Cílem této bakalářské práce bylo navrhnout plánovač s nadstandardními funkcemi a tento implementovat na platformě Microsoft Windows.

Průběh vzniku této práce je možné rozdělit na dvě části – teoretickou a praktickou. Teoretická část probíhala od průzkumu základních informací, analýzou požadavků až po návrh funkčnosti. Praktickou část pak tvoří implementace nejprve tlustého klienta, poté klienta tenkého.

Jelikož byla aplikace vyvíjena postupně, každou její nově implementovanou část jsem následně testoval v provozu. Během tohoto procesu jsem postupně objevoval další možnosti možných vylepšení, které jsem se snažil vzápětí implementovat.

V případě tlustého klienta jsem při návrhu grafického designu kladl důraz na přiměřenou jednoduchost a intuitivní a snadnou orientaci. V případě tenkého potom na maximální optimalizaci kvůli zobrazení a práci s ním na mobilních zařízeních.

Musím říci, že s mojí volbou implementačního prostředí jsem nakonec spokojen, i když jsem si tím nebyl zezačátku zcela jistý. A to i přes četné problémy, které spíše vznikaly z mé neznalosti, než pro nedokonalost vývojového prostředí. Delphi je velice mocný nástroj, obzvláště v oblasti práce s databázemi.

Implementační část tlustého klienta zahrnuje asi 2500 řádků okomentovaného zdrojového kódu jazyka Object Pascal a implementační část tenkého klienta asi 200 řádků v pěti zdrojových souborech jazyka PHP.

Aplikaci jsem včetně konfigurace databázového serveru zprovoznil na svém vlastním počítači s operačním systém Windows XP. Tenkého klienta jsem zpřístupnil na internetové adrese:

```
http://raven.d2.cz/planovac
```

Jak jsem během testování aplikace zjistil, možnosti dalšího vývoje jsou poměrně široké. Z hlediska důležitosti a pořadí implementace bych se zaměřil především na upozornění určitou dobu před samotnou událostí, opakované úkoly a možnost vedoucího zadávat úkoly jednotlivcům. Další funkce, které by zpříjemnily a urychlily práci s aplikací, mohou být například správa delších textových informací – poznámek, zapamatování nastavení budíku i přes ukončení aplikace, nebo automatické vygenerování loginu a hesla při přidávání nového uživatele do systému. Určitě by také bylo užitečné zajistit, aby při zadávání hromadného úkolu, byli o této skutečnosti všichni uživatelé systému automaticky informováni odesláním zprávy na mobilní e-mail. Přestože v původním návrhu jsem neuvažoval o složitějším zabezpečení, během tvorby aplikace jsem došel k přesvědčení, že zakódováním lokálních souborů s nastavením by se předešlo úmyslným negativním zásahům do systému.

Díky tomuto projektu jsem se velmi podrobně seznámil s některými mnou doposud neprozkoumanými technologiemi, a to převážně s širokými možnostmi práce s databázemi ve vývojovém prostředí Delphi. Při vývoji jsem využil i mnoho získaných znalostí, týkajících se návrhu a implementace software.

Cíle, které jsou vytyčeny v neformální specifikaci, jsem v zásadě splnil a program funguje podle mých představ.

# Literatura


[1] KENNEDY, Bill, MUSCIANO, Chuck. HTML & XHTML: The Definitive Guide. 5th edition. [s.l.] : O'Reilly, 2002. 670 s. ISBN 0-596-00382-X.

[2] The PHP Documentation Group. PHP: Manuál PHP [online]. The PHP Documentation Group, c1997-2004 , Last updated: Mon, 24 Apr 2006 [cit. 2006-04-25]. Text v češtině. Dostupný z WWW: <http://php.ftp.cvut.cz/manual/cs/>.

[3] MySQL AB. MySQL 5.0 Reference Manual [online]. MySQL AB, c1995-2006 , 2006-04-24 (revision: 1910) [cit. 2006-04-25]. Text v angličtině. Dostupný z WWW: <http://dev.mysql.com/doc/refman/5.0/en/index.html>.

[4] PIRKL, Josef. *Komponenty v Delphi*. 1. vyd. Praha : Computer Press, c2002. xviii, 438 s. ISBN 80-7226-746-9.

[5] SEDLÁČEK, Jiří, SLABA, Jiří. *Delphi v kostce*. 1. vyd. Praha : BEN, 1997. xx, 425 s. ISBN 80-86056-12-0.

[6] CANTÚ, Marco. *Mistrovství v Delphi 2 : pro Windows 95/NT*. 1. vyd. Praha : Computer Press, c1996. xivi, 975 s. ISBN 80-85896-75-3.

[7] W3C. XHTML™ Modularization 1.1: XHTML DTD Module Implementations [online]. W3C, c2006 , 13 February 2006 [cit. 2006-04-25]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/TR/xhtml-modularization/dtd_module_defs.html>.

# Seznam příloh

**Příloha A**

Obsah přiloženého CD

**Příloha B**

SQL dotazy pro vytvoření struktury databáze

# Příloha A – Obsah přiloženého CD

Součástí této bakalářské práce je i přiložený CD obsahující text práce, zdrojové soubory, přídavné komponenty neobsažené v Delphi a použité v aplikaci, potřebné knihovny i zkompilovanou aplikaci připravenou ke spuštění.

Také obsahuje instalační balík serverové aplikace MySQL v3.23.49.

CD má následující adresářovou strukturu:

- ./**components** – komponenty Delphi nutné pro kompilaci zdrojového kódu
- ./**dll** – knihovny nutné pro překlad, případně spuštění aplikace
- ./**docs** – technická zpráva ve formátech .doc a .pdf
- ./**executable** – zkompilovaná aplikace se všemi náležitostmi ke spuštění
- ./**mysql** – instalační balík MySQL v3.23.49
- ./**source**
    - ./source/**delphi** – zdrojové kódy tlustého klienta pro Delphi 7
    - ./source/**php** – zdrojové soubory .php tenkého klienta
- ./**sql** – SQL skript pro vytvoření struktury databáze

# Příloha B – SQL dotazy pro vytvoření struktury databáze

Vytvoření databáze **planovac**:

```
DROP DATABASE IF EXISTS planovac;
CREATE DATABASE IF NOT EXISTS planovac;
USE planovac;
```

Vytvoření tabulky **uzivatele**:

```
DROP TABLE IF EXISTS uzivatele;
CREATE TABLE IF NOT EXISTS uzivatele (
  id_uzivatele SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  jmeno VARCHAR(15) NOT NULL ,
  prijmeni VARCHAR(20) NOT NULL ,
  login VARCHAR(5) NOT NULL ,
  heslo VARCHAR(10) NOT NULL ,
  vedouci TINYINT NOT NULL ,
  email VARCHAR(50) NOT NULL ,
  UNIQUE (login)
);
```

Vytvoření tabulky **ukoly**:

```
DROP TABLE IF EXISTS ukoly;
CREATE TABLE IF NOT EXISTS ukoly (
  id_ukolu INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  uzivatel SMALLINT UNSIGNED NOT NULL ,
  datum DATE NOT NULL ,
  cas TIME NOT NULL ,
  ukol VARCHAR(30) NOT NULL ,
  alarm TINYINT NOT NULL ,
  hromadny TINYINT NOT NULL ,
  INDEX (uzivatel) ,
  FOREIGN KEY (uzivatel) REFERENCES uzivatele ON DELETE CASCADE
);
```

Vytvoření administrátorského účtu **admin**:

```
INSERT INTO uzivatele
  VALUES('','admin','admin','admin','admin',1,'');
```

Závěrečné potvrzení transakce:

```
commit;
```