

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ  
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ  
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION  
DEPARTMENT OF TELECOMMUNICATIONS

SERVER A KLIENT PRO SPRÁVU HOSTINGOVÝCH  
SLUŽEB S VYUŽITÍM FRAMEWORKU QT4 A LINUXU

DIPLOMOVÁ PRÁCE  
MASTER'S THESIS

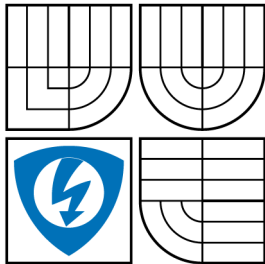
AUTOR PRÁCE  
AUTHOR

BC. JAKUB MATAS

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ  
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY  
A KOMUNIKAČNÍCH TECHNOLOGIÍ  
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND  
COMMUNICATION  
DEPARTMENT OF TELECOMMUNICATIONS

SERVER A KLIENT PRO SPRÁVU HOSTINGOVÝCH  
SLUŽEB S VYUŽITÍM FRAMEWORKU QT4 A LINUXU  
SERVER AND CLIENT FOR THE ADMINISTRATION OF THE HOSTING  
SERVICES USING THE QT FRAMEWORK AND LINUX

DIPLOMOVÁ PRÁCE  
MASTER'S THESIS

AUTOR PRÁCE  
AUTHOR

BC. JAKUB MATAS

VEDOUcí PRÁCE  
SUPERVISOR

ING. TOMÁŠ MATOCHA

BRNO 2011

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

## **ABSTRAKT**

Práce se zabývá návrhem a implementací klient/server aplikace pro správu hostingových služeb. Jsou uvedeny také ostatní řešení správy služeb a porovnány výhody a nevýhody oproti zadanému řešení. Je uveden popis jednotlivých hostingových služeb a jejich nastavení pro linuxovou distribuci Ubuntu. Pro zadaný typ správy je navržen komunikační protokol a datové úložiště sloužící k uložení všech klientských účtů a serverů. Ukázány jsou základní principy práce s C++ frameworkem Qt a jeho použití při implementaci serverové resp. klientské aplikace. Jsou zde také uvedeny základní nastavení serverové aplikace pro její spuštění na serveru jako služby. V poslední části je uveden popis práce s klientskou aplikací a správy klientských účtů.

## **KLÍČOVÁ SLOVA**

Webhosting, vzdálená správa, server, klient/server aplikace, komunikační protokol, Qt framework

## **ABSTRACT**

This thesis deals with the design and the implementation of a client/server application for the administration of hosting services. Other solutions of hosting services administration are listed as well and they are contrasted and compared with the assigned solution. A description of the particular hosting services and their setting for the Ubuntu Linux distribution are provided. A communication protocol and a data store serving to save all the client accounts and servers were designed. Basic principles of working with C++ framework Qt and its usage for the implementation of both the server and the client application are demonstrated. The basic settings of the server application enabling it to be launched on the server as a service are mentioned as well. In the last part a description is stated of working with the client applications and administration of the client accounts.

## **KEYWORDS**

Webhosting, remote administration, server, client/server application, communication protocol, Qt framework

MATAS J. *Fakulta elektrotechniky a komunikačních technologií*. Místo: Vysoké Učení Technické Brno. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2011. Počet stran s., Počet stran příloh s. příloh. Semestrální projekt. Vedoucí práce byl Ing. Matocha Tomáš.

## PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Server a klient pro správu hostingových služeb s využitím frameworku Qt4 a Linuxu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne .....

.....

(podpis autora)

## PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu práce Ing. Tomáši Matochovi, za odbornou pomoc, cenné rady a čas strávený při zpracování této diplomové práce.

# OBSAH

Úvod	12
<b>1 Internetové služby</b>	<b>13</b>
1.1 Hostigové služby . . . . .	14
1.1.1 Uživatelské účty . . . . .	14
1.1.2 Správa diskových kvót . . . . .	15
1.1.3 Domény . . . . .	15
1.1.4 Poštovní server . . . . .	16
1.1.5 Databázový server . . . . .	17
1.1.6 File Transfer Protocol (FTP) server . . . . .	18
1.1.7 Webový server . . . . .	18
1.1.8 Skriptovací jazyky . . . . .	19
1.2 Možná řešení správy . . . . .	20
1.2.1 Příkazová řádka . . . . .	20
1.2.2 Skript pro automatické vytvoření klientského účtu . . . . .	20
1.2.3 Aplikace s přístupem přes internetové rozhraní . . . . .	21
1.2.4 Aplikace typu klient/server . . . . .	22
<b>2 Návrh aplikace pro správu služeb</b>	<b>23</b>
2.1 Vybraný typ aplikace . . . . .	23
2.2 Služby obsluhované aplikací . . . . .	24
2.3 Nastavení služeb v implementované aplikaci . . . . .	25
2.3.1 Struktura klientských účtů na serveru . . . . .	25
2.3.2 Účty poštovních schránek . . . . .	25
2.3.3 Nastavení webového serveru Apache . . . . .	27
2.3.4 Nastavení databázového serveru MySQL . . . . .	29
2.3.5 Nastavení FTP serveru ProFTPD . . . . .	29
2.3.6 Návrh aplikace klient/server . . . . .	30

<b>3</b>	<b>Implementace aplikace</b>	<b>33</b>
3.1	Qt framework . . . . .	33
3.1.1	Kompilace projektů v Qt frameworku . . . . .	34
3.1.2	Nástroje pro práci v Qt frameworku . . . . .	35
3.2	Datového úložiště . . . . .	37
3.2.1	Jednotlivé tabulky úložiště . . . . .	37
3.2.2	Práce s datovým úložištěm . . . . .	39
3.2.3	Zabezpečení komunikace s datovým úložištěm . . . . .	40
3.3	Komunikační protokol . . . . .	41
3.3.1	Přenos příkazů a dat . . . . .	41
3.3.2	Komunikace pomocí protokolu . . . . .	42
3.3.3	Zabezpečení komunikace . . . . .	43
3.3.4	Příkazy protokolu . . . . .	44
3.4	Serverová část . . . . .	46
3.4.1	Obsluha komunikačního protokolu . . . . .	46
3.4.2	Nastavení serverové části . . . . .	47
3.4.3	Modulární systém . . . . .	48
3.4.4	Modul datového úložiště . . . . .	50
3.4.5	Modul obsluhy webového serveru Apache . . . . .	51
3.4.6	Modul pro obsluhu MySQL databáze . . . . .	52
3.4.7	Modul kvót pro MySQL . . . . .	53
3.4.8	Modul pro práci s poštovními účty . . . . .	53
3.5	Klientská část . . . . .	55
3.5.1	Určení použití . . . . .	56
3.5.2	Koncept ovládání . . . . .	57
3.5.3	Napojení na datové úložiště . . . . .	57
3.5.4	Implementace komunikačního protokolu . . . . .	58
3.5.5	Správa serverů . . . . .	58
3.5.6	Správa klientských účtů . . . . .	59
3.5.7	Rozšíření funkčnosti aplikace . . . . .	64

<b>4 Závěr</b>	<b>66</b>
<b>Reference</b>	<b>68</b>
<b>Seznam symbolů, veličin a zkratek</b>	<b>70</b>
<b>Seznam příloh</b>	<b>72</b>
<b>A Skript pro vytváření účtu</b>	<b>73</b>
<b>B Příklad souboru projektu pro Qt Framework</b>	<b>77</b>
<b>C Struktura datového úložiště</b>	<b>78</b>
<b>D Příkazy komunikačního protokolu</b>	<b>79</b>
D.1 Základní příkazy . . . . .	79
D.2 Příkazy pro práci s webovými účty . . . . .	80
D.3 Příkazy pro práci s databázovými účty . . . . .	81
D.4 Příkazy pro práci s e-mailovými účty . . . . .	82
D.4.1 Příkazy pro správu administrátorů schránek . . . . .	82
D.4.2 Příkazy pro správu schránek . . . . .	83
<b>E Soubory s nastavením aplikace</b>	<b>84</b>
E.1 Soubor s hlavním nastavením serverové aplikace . . . . .	84
E.2 Nastavení modul <i>storageunixhomedir</i> . . . . .	84
E.3 Nastavení modul <i>webserverapache</i> . . . . .	85
E.3.1 Příklad šablony virtuálního hosta . . . . .	86
E.4 Nastavení modul <i>dbservermysql</i> . . . . .	86
E.5 Nastavení modul <i>dbservermysqlquota</i> . . . . .	87
E.6 Nastavení modul <i>mailserverpostfixmysql</i> . . . . .	87
<b>F Konfigurační soubory služeb</b>	<b>88</b>
F.1 konfigurace virtuálního hosta webového serveru Apache . . . . .	88

# SEZNAM OBRÁZKŮ

1.1	Obrazovka administračního rozhraní Parallel Plesk Panel . . . . .	22
2.1	Schéma propojení částí aplikací a datového úložiště . . . . .	31
3.1	Uživatelské rozhraní QtCreatoru . . . . .	36
3.2	Příklad komunikace klienta se serverem při vytváření webového účtu .	43
3.3	Vnitřní struktura serverové aplikace . . . . .	48
3.4	Hlavní okno aplikace s otevřenými účty a seznamem serverů . . . . .	55
3.5	Nově vytvořený prvek pro měření síly hesla . . . . .	56
3.6	Nově vytvořený prvek pro nastavení kvóty . . . . .	56
3.7	Okno pro práci se servery . . . . .	59
3.8	Okno se seznamem klientských účtů . . . . .	60
3.9	Okno pro práci s klientským účtem . . . . .	60
3.10	Okno pro úpravu domény . . . . .	61
3.11	Okno pro přidání webového účtu . . . . .	62
3.12	Okno pro přidání databázového účtu . . . . .	63
3.13	Okno pro úpravu administrátorského účtu e-mailů . . . . .	64
C.1	Struktura tabulek účtů v databázi . . . . .	78

# SEZNAM TABULEK

1.1	Tabulka s řadami stavových kódů při zpracování zprávy . . . . .	17
1.2	Tabulka s řadami stavových kódů odpovědí při zpracování požadavku	19
2.1	Struktura domovského adresáře uživatele a popis jednotlivých adresářů	26
2.2	Některé důležité konfigurační volby webového serveru Apache pro hosta	28
2.3	Volby globálního nastavení ProFTPD serveru . . . . .	30
3.1	Výběr některých modulů frameworku Qt a jejich stručný popis . . . .	33
D.1	Základní příkazy protokolu . . . . .	79
D.2	Příkazy protokolu pro práci s webovými účty . . . . .	80
D.3	Příkazy protokolu pro databázové účty . . . . .	81
D.4	Příkazy protokolu pro účty administrátoru poštovních schránek . . .	82
D.5	Příkazy protokolu pro správu poštovních schránek . . . . .	83

# ÚVOD

Tato práce se věnuje návrhu a implementaci aplikace typu klient/server pro komplexní správu hostingových služeb, provozovaných na vzdálených serverech. Aplikace je primárně určena pro správce hostingových programů, ale její použití by nemělo být obtížné ani laiku. Pro pochopení textu jsou však nutné pokročilé znalosti operačního systému Linux, síťových služeb a programování v C++.

Serverová část aplikace bude navržena pro použití na linuxových serverech, a to s nainstalovanou linuxovou distribucí typu Ubuntu Server nebo Gentoo. Klientská část bude navržena tak, aby byla použitelná na jakékoliv platformě, podporující Qt framework (Linux, Windows, aj). V textu budou také popsána všechna potřebná nastavení aplikace a samotného serveru pro poskytování hostingových služeb.

V první části bude seznámení se samotnou podstatou hostingů, principem a využitím jednotlivých služeb. Součástí kapitoly bude také přehled možných typů řešení správy, od nejjednodušší správy pomocí příkazového řádku, až po komplexní aplikace určené pro správu velkého množství klientských účtů a účtů distributorů.

Druhá část textu bude popisovat důvody výběru klient/server dané aplikace. Dále budou popsány vybrané serverové aplikace implementující jednotlivé služby poskytované servery a jejich základní nastavení. Podle těchto nastavení bude navrhována serverová část aplikace.

Poslední část celé práce se již zabývá implementací daného řešení. Na začátku je zmíněno jak pracuje použitý C++ framework Qt a jak překládat projekty v něm napsané do aplikací. Uvedeno je také vývojové prostředí použité při vývoji aplikací.

Celá tato část je rozdělena do čtyř souvislých celků, tak aby čtenář postupně pochopil každou z nich a informace se nemísily dohromady. Nejprve jsou popsány základní části celého systému tj. datové úložiště a komunikační protokol. Na tyto celky již navazuje samotná implementace jak klientské tak serverové aplikace.

V poslední části implementace klientské aplikace jsou zmíněny některé výhody vytvořeného řešení a možnosti jejich využití při rozšiřování funkčnosti aplikace a zjednodušení přístupu ke klientským účtům a serverům.

# 1 INTERNETOVÉ SLUŽBY

Velká část uživatelů využívá internet ke komunikaci a k ukládání a čtení dat, data jsou umístěna na serverech, která jsou připojena do internetu pomocí síťového připojení o vysoké propustnosti a rychlé odezvě. Nejčastěji jsou servery připojeny přímo na páteřní síť jednotlivých firem, zajišťujících připojení samotných uživatelů k internetu. Tyto páteřní síť jednotlivých firem jsou vzájemně propojeny.

Aby uživatel mohl mít svá data umístěna na internetu, musí zaplatit finanční obnos vybrané společnosti zabývající se internetovými službami s možností pronájmu celého serveru nebo jeho části. Pronájem celého serveru je velmi nákladné, lze si tedy pronajmout pouze malou část serveru určenou ke zveřejnění své prezentace nebo dat. Společnost tak může mít na jednom serveru velké množství klientů. Pro jednodušší orientaci v poskytovaných službách se zavádějí tzv. hostingové programy, které určují jaké služby budou pro daného klienta přístupné a kolik diskového prostoru bude klientu vyhrazeno. Mezi nejčastější služby v hostingových programech je umístěn pro prezentaci dat webový server<sup>1</sup> se zapnutým některým skriptovacím jazykem, například Hypertext PreProcessor (PHP) nebo Active Server Pages (ASP), poštovní server pro správu elektronické pošty, popřípadě databázový server (nejčastěji *MySQL*).

Servery pro poskytování hostingových služeb jsou většinou postaveny na Linuxových nebo Berkeley Software Distribution (BSD) systémech. Pouze na malém procentu serverů je provozován systém firmy Microsoft a jeho platforma Internet Information Services (IIS), z důvodu nízkých finančních nákladů na software a vysoké stability serveru při použití Linuxové či BSD platformy.

Linuxové systémy instalované na těchto serverech nepotřebují ke své činnosti propracované grafické prostředí, ale většinou si vystačí pouze s příkazovou řádkou (ta je u linuxových systému velmi propracovaná a samotná umí zpracovávat jednoduchý skriptovací jazyk) nebo konzolovou aplikací. Vlivem nepoužívání grafického rozhraní systému je ušetřena část výpočetních prostředků ve prospěch spuštěných serverových

---

<sup>1</sup>Webový server je server, který zpracovává požadavky klientů (nejčastěji internetový prohlížeč) a odesílá statické nebo dynamicky generované stránky a soubory zpět ke klientu

aplikací.

Nejčastějšími aplikacemi spuštěnými na serverech, kde je nainstalován systém Linux nebo BSD, jsou webové servery, poštovní servery, proxy servery, databázové servery a další. Některé aplikace jsou spuštěny současně na jednom serveru a vzájemně se tak svou činností doplňují. Například poštovní server používá databázový server pro ověřování uživatelských účtů a webový server zpracovává požadavky klienta z rozhraní internetového prohlížeče pro přístup k elektronické poště.

Aplikace běžící na těchto systémech většinou nemají žádné grafické prostředí pro jejich konfiguraci a sledování stavu. Nastavení se provádí nejčastěji pomocí textových souborů, které jsou umístěny v konfiguračním adresáři systému (u systémů Linux se jedná o adresář */etc* umístěný v kořenu disku). Pouze další rozšiřující aplikace instalované současně se serverovou aplikací umožňují konfiguraci pomocí grafického prostředí.

Nedostatkem správy velkého množství klientů pronajímajících si část serverů může být správné nastavení serverových služeb co nejefektivněji tak, aby vyhovovalo požadavkům klienta.

## 1.1 Hostigové služby

### 1.1.1 Uživatelské účty

Každý uživatel využívající služby serveru má vytvořen přihlašovací účet, který může být reálný nebo virtuální. Reálným se rozumí, že uživatel má vytvořen vlastní domovský adresář a přihlašovací údaje jsou uloženy přímo v systému. Virtuální účty mohou být uloženy například v databázi či Lightweight Directory Access Protocol (LDAP), nemusí mít vlastní domovský adresář. Typickým příkladem využití virtuálních účtů je přihlašování k poštovním schránkám, kdy je vytvořen jeden reálný účet pro celou doménu, samotní uživatelé mají virtuální účty.

### 1.1.2 Správa diskových kvót

U každého uživatelského účtu v systému jsou určeny diskové kvóty, tj. kolik dat může mít uživatel uloženo na serveru. U virtuálních účtů jsou informace o kvótě uloženy spolu s informacemi o uživateli, samotný server obsluhující danou službu si kontroluje obsazení prostoru. U reálných účtů jsou kvóty poskytovány přímo souborovým systémem, který si sám kontroluje obsazený prostor. Samotný postup instalace kvót nalezneme například v dokumentaci [5] Gentoo linuxu.

Diskové kvóty jsou u hostingových systémů velmi důležité. Pokud by uživatel žádnou kvótu nastavenou neměl, mohlo by například při ukládání dat dojít k zaplnění celého disku a ostatní uživatelé by byli omezeni, v horším případě by mohli přijít o svá data.

U systému kvót poskytovaných souborovým systémem lze vytvářet skupiny. Do těchto skupin lze přidávat jednotlivé uživatele. Nastavení kvót se pak provádí pro celou skupinu a je aplikováno najednou na všechny uživatele ve skupině.

### 1.1.3 Domény

Doména označuje jednoznačné jméno počítače v síti a je provázána s IP adresou počítače. Podrobnější informace o struktuře domén a jejich principu odvozování nalezneme v dokumentu Request For Comments (RFC) 1034 [1]. Informace o doméně jsou uloženy na Domain Name System (DNS) serveru v souborech nebo v databázi a popisující danou doménu. Mezi hlavní parametry patří nastavení dvojice *doména - IP adresa*, IP adresa nebo název poštovního serveru, popřípadě adresa DNS serveru pro danou doménu.

Informace o doménách jsou uloženy na primárním a sekundárním serveru. Primární server je hlavním serverem a data na něm uložená jsou majoritní. Sekundární servery pouze kopírují nastavení primárních serverů, snižují tak zátěž serverů primárních nebo obsluhují požadavky klientů při výpadku primárního serveru.

Nejčastějším zástupcem DNS serverů je server Berkeley Internet Name Domain (BIND).

### 1.1.4 Poštovní server

Tento server se stará o přijímání, odesílání a filtrování pošty a poskytuje rozhraní pro přístup k poštovním schránkám jednotlivých uživatelů pomocí protokolu Internet Message Access Protocol (IMAP) nebo Post Office Protocol version 3 (POP3).

Server se skládá ze tří základních částí, Mail Submission Agent (MSA) – agent zodpovědný a odeslání zprávy, MTA – agent zodpovědný a přenos zprávy a Mail Delivery Agent (MDA) – agent zodpovědný a doručení zprávy do schránky. Tyto části jsou na sobě závislé a navzájem se podporují. Na většině serverů je Mail Transfer Agent (MTA) rozšířen ještě o další agenty, které rozšiřují schopnost MTA o filtrování pošty (např. označení nevyžádané pošty).

Prvním článkem řetězce je agent MSA. Přijímá zprávu od uživatele (např. poštovní klient KMail, MS Outlook aj.) a předává ji určenému MTA. Zodpovídá za správné přenesení zprávy v síti a ověření klienta pomocí některých ověřovacích metod (SMTP-AUTH, Remote Authentication Dial In User Service (RADIUS)). Pro přenos zpráv od klienta používá protokol Simple Mail Transfer Protocol (SMTP) a to buď se zabezpečeným nebo nezabezpečeným ověřováním uživatele.

MTA přebírá zprávu od MSA nebo od jiného MTA. Samotné MTA zjišťuje co nejvíce informací o zprávě a jak ji má doručit. Nejprve agent zjistí pro jakou doménu je zpráva určena. Když se jedná o jinou doménu než jakou MTA spravuje, předá ji poštovnímu serveru vedenému v DNS pro danou doménu. Pokud se jedná o doménu, kterou MTA spravuje, zprávu přepoše přes rozšiřující agenty (viz dále) a předá ji MDA. Samotné MTA kontroluje několik parametrů zprávy (např. správně zadanou doménu, správný formát hlaviček, atd.) a pokud některému parametru zpráva nevyhoví, je odmítnuta nebo zahozena. Odesílatel zprávy je o jejím stavu vždy informován pomocí stavového kódu. Rozdělení stavových kódů nalezneme v tabulce 1.1.

MDA přebírá zprávu od MTA a zodpovídá za doručení zprávy do schránky místnímu uživateli. Přijaté zprávy může ještě filtrovat pomocí globálních nebo uživatelských filtrů a doručit ji do správné složky, ale nijak neupravuje obsah zprávy.

Rozšíření MTA pomocí dalších agentů umožňuje kontrolu dalších parametrů. Například rozbalení archivu a jeho antivirovou kontrolu nebo porovnání adresy

Řada kódu	Popis
2xx	vše proběhlo v pořádku
4xx	označují dočasný problém, odesílatel se může pokusit zprávu doručit později
5xx	trvalý problém (např. neexistující schránka či uživatel)

Tabulka 1.1: Tabulka s řadami stavových kódů při zpracování zprávy

odesílatele se seznamem adres SPAMu<sup>2</sup>.

Nejčastějšími agenty MTA bývá aplikace POSTFIX, Sendmail, Microsoft Exchange Server a jiné. Většina z výše jmenovaných serverů lze také použít jako MDA, ale pro využití pokročilejších vlastností doručování se používá např. aplikace mail-drop. Kompletní informace o serveru POSTFIX nalezneme v [4].

K přístupu ke schránkám se používá IMAP nebo POP3 server. Tyto servery ověřují uživatele a zpřístupňují mu obsah jeho poštovní schránky. Tyto servery se používají také při ověřování uživatelů komunikujících přes SMTP protokol. Odstraní se tak duplicitní záznamy o uživatelských účtech (každý účet je definován pouze jednou).

### 1.1.5 Databázový server

Databáze je označením pro datové úložiště, podobné dřívější kartotéce, ale v elektronické podobě. Existuje několik druhů databází. Nejčastěji používané jsou relační databáze, které data shromažďují do tabulek. Relační databáze je jedním ze základních úložišť většiny internetových aplikací. U relačních databází se všechna data ukládají do tabulek. Tento typ uložení má však některé nevýhody, proto se začínají vyvíjet databáze objektové, ve kterých jsou data uložena jako objekty a dá se k nim lépe přistupovat (např. stromová hierarchie).

Pro přístup k databázi se používá databázový server, obsluhuje jednotlivé dotazy na databázi a jeho výstupem mohou být požadovaná data. Zodpovídá také za optimalizaci uložení databáze na paměťovém médiu a konzistenci dat v databázi. Mezi

<sup>2</sup>SPAM je termín pro hromadné zprávy s nevyžádaným nebo komerčním obsahem

nejznámější databázové servery patří Oracle, Microsoft SQL server, MySQL a jiné. Pro internetové stránky se většinou používá databáze MySQL nebo méně rozšířená PostgreSQL. Je to z důvodu svobodné licence a nízkých nákladů na pořízení.

Komunikace mezi aplikací a serverem probíhá a použití Structured Query Language – strukturovaný dotazovací jazyk databází (SQL) jazyka. Je to jazyk, který se používá k popisu požadavku na databázi. Nevýhodou tohoto jazyka jsou drobné rozdíly v implementaci u každého serveru, což může velmi ovlivnit přenosnost aplikace mezi databázemi. Příklad implementace SQL jazyka v *MySQL* nalezneme v dokumentaci k *MySQL* serveru [6].

### 1.1.6 FTP server

Pro zpřístupnění souborů, uložených na serveru se nejčastěji využívá FTP server. Tento server po přihlášení uživatele zpřístupní obsah jeho domovského adresáře nebo pokud přihlášení není vyžadováno nebo je anonymní, zpřístupní veřejné úložiště. Uživatel tak může vzdáleně přenášet a pracovat se soubory uloženými na serveru.

Pro komunikaci mezi klientem a serverem se používá protokol FTP nebo zabezpečenou verzi Secure File Transfer Protocol (SFTP). Samotná komunikace mezi serverem a klientem probíhá pomocí jednoduchých příkazů. Uživatel je po provedení příkazu informován o jeho stavu pomocí stavových kódů.

### 1.1.7 Webový server

Jednou z nejdůležitějších částí hostingových služeb je webový server. Obsluhuje požadavky klientů, kteří s ním komunikují prostřednictvím některého z internetových prohlížečů. Po vyřízení požadavku server vrací zpracovaná data zpět ke klientu. Nejčastěji soubory HyperText Markup Language (HTML), obrázky a jiné. Server také klientu zasílá v odpovědi stavový kód, označující jak byl požadavek zpracován. Seznam řad stavových kódů nalezneme v tabulce 1.2. Podrobnější informace o jednotlivých stavových kódech nalezneme v dokumentu RFC 2616 [2] popisujícího HyperText Transfer Protocol (HTTP) verze 1.1.

Řada kódu	Popis
1xx	informační kódy (např. o přesměrování na jiný protokol)
2xx	požadavek byl v pořádku zpracován
3xx	přesměrování požadavků (např. při přesměrování na jinou adresu)
4xx	chyby při vyřizování požadavku (např. kód 401 - stránka nebyla nalezena)
5xx	chyby zpracování požadavku na webovém serveru (interní chyby, atd.)

Tabulka 1.2: Tabulka s řadami stavových kódů odpovědí při zpracování požadavku

Pokud jsou soubory uloženy na serveru v podobě, v jaké se zobrazují v internetovém prohlížeči klienta, jedná se o statické stránky. Server však může zpřístupňovat některý ze skriptovacích jazyků (např. PHP, ASP), obsah souborů pomocí nich generovat a výsledek odesílat zpět ke klientovi. Tyto soubory však reálně na serveru nemusí existovat. Metoda se označuje jako dynamické vytváření obsahu a umožňuje vytvářet obsah podle požadavků klienta.

Webový server lze také nastavit tak, aby přístup k některým adresářům a souborům byl chráněn heslem, popřípadě úplně zakázán. Pomocí zvláštního konfiguračního souboru *.htaccess* lze také nastavit přepis adresy (modulu rewrite u serveru apache) do čitelnějšího formátu.

Asi nejznámějšími zástupci webových serverů je Apache, nginx, Microsoft IIS a další.

### 1.1.8 Skriptovací jazyky

Na většině webových serverů je instalován alespoň jeden ze skriptovacích jazyků. Tyto jazyky jsou instalovány jako rozšiřující moduly pro daný webový server.

Skriptovací, nebo také interpretační jazyk se používá pro tvorbu dynamického obsahu internetových stránek, pro přístup k databázi, pro práci se soubory na serveru a další. Nejčastěji je využívají redakční systémy pro přímou úpravu internetové

prezentace.

Mezi nejznámější jazyky patří například PHP, ASP, PERL a také stále populárnější Python či Ruby On Rails.

## 1.2 Možná řešení správy

### 1.2.1 Příkazová řádka

Nejjednodušším řešením nastavení serverových aplikací je správa pomocí příkazové řádky a konfiguračních souborů. Avšak nevýhodou této správy je, že při nasazení na velké množství klientů je nutné pro každého klienta upravovat parametry jednotlivých služeb zvlášť a vytvoření nebo úprava nového klienta je tak velmi časově náročná. Další nevýhodou je nemožnost upravit některé parametry u všech klientů najednou, což způsobuje nedostatečnou flexibilitu a časovou náročnost řešení.

### 1.2.2 Skript pro automatické vytvoření klientského účtu

Pokročilejší možností správy služeb a uživatelů je vytvořením skriptu (např. ve skriptovacím jazyku *bash*, *pythonu* aj.), který pomocí předaných parametrů, dotazů nebo předaného souboru jednotlivé služby nastaví a spustí. Výhodou je rychlé vytvoření klienta a nastavení jeho služeb, nevýhodou řešení je opět nemožnost hromadné úpravy parametrů několika klientů současně a s tím i stejná časová náročnost jako u předchozího řešení.

Příklad takového skriptu ve skriptovacím jazyce *bash* nalezneme v příloze A. Jeho použití je následující:

```
server ~ # ./script.sh domena heslo
```

Skript vytvoří uživatelský účet a vytvoří databázi. Zapiše nastavení pro doménu do konfigurace webového serveru (zde Apache2) a obnoví jeho konfiguraci. Na závěr se skript zeptá na vytvoření odkazů na aplikace *phpMyAdmin* pro správu databáze,

*postfixadmin* pro správu schránek elektronické pošty a volby rozhraní pro čtení elektronické pošty. Daný skript je velmi jednoduchý, neumožňuje však pokročilejší nastavení, jako je například přidělení diskové kvóty nebo vytvoření primárního účtu k administraci schránek elektronické pošty.

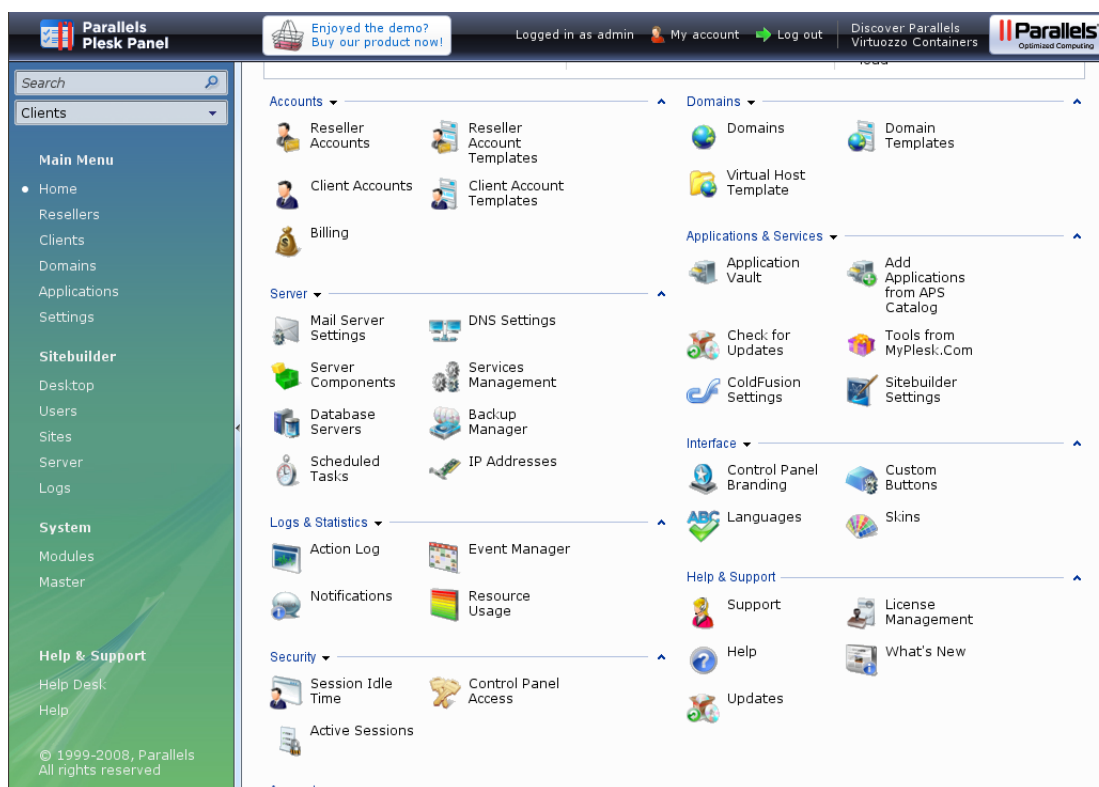
### 1.2.3 Aplikace s přístupem přes internetové rozhraní

Další možností správy služeb a účtů je využití některé z pokročilých aplikací jako jsou Parallel Plesk Panel, VHCS, WebMin, cPanel atd., určených pro komplexní správu serveru, klientských účtů a poskytovaných služeb. Tyto aplikace obsluhují jednotlivé spuštěné služby serveru. Pokud by daná aplikace neumožňovala správu některé části serveru, je možné její funkčnost rozšířit přidáním potřebného zásuvného modulu dodávaného výrobcem nebo vlastní implementací požadované funkce. Obsahují velké množství konfiguračních voleb pro nastavení serverových aplikací a samotného serveru, což umožňuje použití i pro administrátora, který má o problematice hostingových služeb pouze základní znalosti.

Tyto aplikace běží na pozadí serveru a lze k nim přistupovat přes některý z internetových prohlížečů. Pro připojení k aplikaci stačí znát pouze IP adresu nebo doménový název serveru, který chceme spravovat a port aplikace, na kterém je povolena komunikace.

Některé z pokročilejších aplikací tohoto typu umožňují správu několika serverů najednou. Přes uživatelské rozhraní se lze přepnout na podřízený server a pracovat přímo s ním. Tyto aplikace tak umožňují například rozprostřít zátěž mezi několika serverů nebo převod klientského účtu mezi servery. Mezi nejznámější aplikace tohoto typu patří například Parallels Plesk Panel, který využívají některé české IT společnosti poskytující hostingové služby (např. Active 24). Rozhraní této aplikace můžeme vidět na obrázku 1.1.

Výhodou těchto aplikací je soustředění různých nastavení serveru a služeb na něm provozovaných do jediné aplikace. Velkou výhodou je také možnost vytváření šablon a fakturaci služeb. Nevýhodou pro klienta může být velké množství voleb a tedy složitější nastavení.



Obrázek 1.1: Obrazovka administračního rozhraní Parallels Plesk Panel

Firmy vyvíjející tyto aplikace poskytují pro hostingové firmy plnou servisní podporu a jejich produkty jsou tak často velmi drahé. Existuje také několik aplikací, které jsou uvolněny pod svobodnou licencí a jejich užívání je bez poplatků. Uživatel se pak může rozhodnout, jestli mu vyhovuje platit k produktu podporu, nebo zda požaduje produkt bez jakékoliv podpory.

### 1.2.4 Aplikace typu klient/server

Tyto aplikace jsou rozděleny do dvou částí. Serverová část je spuštěna na pozadí na serveru a klientská část k ní vzdáleně přistupuje. Obě části aplikace spolu komunikují pomocí daného protokolu, nejčastěji po zabezpečeném kanále.

Výhodou je jednotné prostředí pro více serverů a možnost přihlášení na více serverů najednou. Nevýhoda je nutnost instalace klientské části na počítači administrátora.

## 2 NÁVRH APLIKACE PRO SPRÁVU SLUŽEB

### 2.1 Vybraný typ aplikace

Dle zadání práce byla zvolen systém aplikace pro správu hostingových služeb typu klient / server. Takto navržená aplikace má pro obsluhu hostingových služeb několik výhod i nevýhod.

Výhody:

- Připojení k různým serverům nezávisle na klientovi - jednotlivá připojení k serverům se budou realizovat podle typu služby a serveru, který danou službu poskytuje.
- Navigace mezi účty klientů pomocí modálních oken. Uživatel tak může lehce přepínat mezi jednotlivými účty nebo si zobrazit nastavení z několika účtů najednou.
- Rychlost odezvy aplikace - přenášena jsou pouze data nutná pro řízení serverové části, nepřenáší se žádná část grafického rozhraní.
- Zabezpečení správného nastavení služby, tj. konfigurační soubor služby nastavuje serverová část aplikace, ne klientská část.
- Stejně prostředí klientské části pro různé druhy serverů.

Nevýhody:

- Spuštění klientské části na počítači klienta - klientská část aplikace musí být spuštěna na počítači či zařízení uživatele, který bude správu provádět, což není vždy možné, protože uživatel nemusí mít aplikaci k dispozici.
- Nutnost zabezpečení - typ správy umožňuje úpravu účtů na serveru a tedy i změnu hesel a jejich samotný přenos přes síť. Proto je nutná zabezpečená komunikace mezi jednotlivými částmi (klient - server - datové úložiště).

Protože jsou jednotlivé části odděleny a mimo strukturu komunikačního protokolu na sobě nezávislé, je možné jednotlivé části implementovat v různých programovacích jazycích a pro různá zařízení a platformy (např. mobilní terminály).

Navrhovaná aplikace bude obsahovat základní správu účtů a hostingových služeb, provozovaných na daném serveru. Aplikace musí také zabezpečit připojení klientské části k serverové, a to šifrováním komunikace mezi jednotlivými částmi.

## 2.2 Služby obsluhované aplikací

Serverová část aplikace bude obsluhovat nastavení jednotlivých služeb provozovaných na serveru. Seznam služeb a vybraného zástupce poskytující danou službu:

- Účty jednotlivých klientů využívající hostingové služby.
- Zvolen webový server Apache. Stručný popis webového serveru nalezneme v kapitole 1.1.7. Popis samotného webového serveru Apache nalezneme v dokumentaci [7]. Popis použitého nastavení serveru nalezneme v 2.3.3.
- Databázový server MySQL. Popis databázového serveru nalezneme v kapitole 1.1.5 a jeho nastavení v 2.3.4.
- Poštovní server bude složen z několika následujících částí. Doručování pošty bude zajišťovat server Postfix. Pro přístup ke schránkám a k ověřování uživatelů bude instalován server Dovecot, který bude doplněn filtry AmavisNew pro antivirovou kontrolu a SpamAssasin pro označení a filtraci nevyžádané pošty. Popis funkce poštovního serveru nalezneme v kapitole 1.1.4, nastavení poštovních schránek nalezneme v kapitole 2.3.2.
- Pro přístup k domovskému adresáři klientů bude spuštěn FTP server s názvem ProFTPD, umožňující jednoduchou konfiguraci a propojení se systémem přihlašování v systému.
- Dále budou instalovány skriptovací jazyky PHP, popřípadě Python, pro tvorbu dynamického obsahu. Podrobnější popis služby nalezneme v kapitole 1.1.8.

Podle požadavků klienta však bude možné přidávat některé další služby. Ty se budou instalovat a nastavovat použitím konzole s připojením k danému serveru.

## 2.3 Nastavení služeb v implementované aplikaci

### 2.3.1 Struktura klientských účtů na serveru

Na serveru bude využito základní systémové úložiště uživatelských účtů. V něm budou uloženi jak systémoví uživatelé (*root*<sup>1</sup>, atd.), tak klientské účty. Klientské účty budou využívat standardní linuxové uživatelské účty s jejich domovskými adresáři, ale bez přístupu především k příkazové řádce a ostatním částem systému.

#### Struktura domovského adresáře účtu

Každý hostingový účet bude mít vytvořen vlastní domovský adresář, k němu budou přidělena všechna oprávnění daného uživatele. Přidány budou také oprávnění webového serveru. Samotná struktura domovského adresáře bude vypadat následovně:

```
/home/{doména}/
```

Kde za doménu se doplní primární doména, přiřazená k účtu, bez speciálních znaků (např. tečky). Název účtu tedy bude ve většině případů odpovídat názvu domény. Samotný obsah domovského adresáře a popis jednotlivých adresářů je uveden v tabulce 2.1.

### 2.3.2 Účty poštovních schránek

Účty poštovních schránek budou uloženy v databázi a budou nezávislé na hostingových účtech. Jeden hostingový účet tak bude moci mít několik schránek elektronické pošty a účtů k nim. Pomocí těchto účtů bude také prováděno ověření uživatele při přihlášení přes SMTP protokol.

---

<sup>1</sup>Hlavní uživatel v systému Linux, který má přiřazena veškerá práva a může tedy provádět v systému cokoliv.

Podadresář	Popis
htdocs	Adresář s internetovou prezentací. Pouze tento adresář je viditelný všem připojeným.
cgi-bin	Adresář pro Common Gateway Interface (CGI) skripty externích aplikací.
logs	Adresář se soubory, obsahující chybové hlášky webového serveru při zpracování požadavku klienta. U produkčních serverů je chybový výstup do stránek potlačen a chyby se ukládají do souborů v tomto adresáři.
error	Adresář s vlastními chybovými stránkami. Stránka se určuje podle čísla stavového kódu odpovědi serveru při zpracování požadavku. Popis stavových kódů nalezneme v tabulce 1.2. Struktura stránek je definována podle použitého webového serveru (v daném případě webový server Apache). Jejich strukturu nalezneme v dokumentaci [7].
icons	Adresář s ikonami použitými při výpisu obsahu adresáře pomocí webového serveru.

Tabulka 2.1: Struktura domovského adresáře uživatele a popis jednotlivých adresářů

Každé doméně bude přiřazen administrátorský účet. Pomocí tohoto administrátorského účtu si klienti budou moci schránky sami upravovat a měnit jejich parametry (disková kvóta, heslo, ...).

### Adresář s poštovními schránkami

Samotné schránky budou uloženy v adresáři `/var/mail/` a přístup k nim bude povolen pouze uživateli `root` a poštovnímu serveru. Pokud by totiž byly schránky umístěny v domovském adresáři, mohl by uživatel přihlášený pomocí protokolu FTP přistupovat do kterékoliv poštovní schránky a číst nebo upravovat její obsah. Adresářová struktura schránky bude následující:

```
/var/mail/{domena}/{schranka}/
```

K samotnému vytváření schránek elektronické pošty bude využita internetová aplikace PostfixAdmin nebo dále navrhovaná aplikace. Pro přihlášení administrátora schránek dané domény budou využity přihlašovací údaje přidělené k hostingovému účtu.

### 2.3.3 Nastavení webového serveru Apache

Základní instalace a nastavení webového serveru Apache nalezneme v [9]. Pokud budeme chtít k serveru přistupovat také pomocí protokolu *https* (zabezpečený protokol *http*), musí být správně nainstalovány a nastaveny potřebné certifikáty. Pokyny k instalaci certifikátů nalezneme v dokumentaci k dané distribuci nebo např. v [10].

Aby byl certifikát důvěryhodnější, lze jej ověřit a podepsat autoritou. Pro ověření certifikátu se vytvoří požadavek na certifikát Certificate Signing Request (CSR) a předá se certifikační autoritě (např. CACert). Autorita po ověření totožnosti uživatele požadavek na certifikát podepíše a vytvoří platný certifikát. Platný certifikát je poté instalován do webového serveru.

#### Obsluha více domén

Webový server může obsluhovat několik domén najednou i když bude mít pouze jednu veřejnou<sup>2</sup> IP adresu. Pro rozlišení se využívá služby virtuálních hostů v systému. Systém pracuje tak, že klient odešle požadavek na server, server z požadavku zjistí na jakého hosta chce klient přistupovat a požadavek zpracuje. Tento postup však není možný při použití zabezpečovacího protokolu Secure Sockets Layer (SSL) verze 1 a 2. U tohoto druhu spojení se nejprve vymění informace o zabezpečení a teprve poté se zpracovává požadavek. Server tak nemůže zjistit k jakému hostu chce klient přistupovat. V roce 1996 byla vydána nová verze protokolu SSL verze 3, která obsahovala novou vlastnost tzv. Server Name Indication (SNI) – Identifikátor názvu serveru, podle níž může server při sestavování zabezpečeného spojení zjistit informaci o hostu (doméně), se kterou chce klient komunikovat. Podle toho je vybrán správný host a také certifikát. Nástupce protokolu SSL se v roce 1999

---

<sup>2</sup>Veřejnou IP adresou se rozumí adresa, která je unikátní v celé síti internet a všem viditelná

stal protokol Transport Layer Security (TLS) ve verzi 1. Definici tohoto protokolu nalezneme v RFC 2246 [3]. Komplexnější popis těchto protokolů nalezneme v [11]. Jediný problém způsobují zastaralé verze prohlížečů (Internet Explorer 6), které protokol SSL od verze 2 výše nepodporují a neodesílají SNI. Server jim poté poskytne pouze prvního zabezpečeného hosta.

## Nastavení virtuálních hostů

Samotné nastavení jednotlivých virtuálních hostů je uloženo v souborech v adresáři `/etc/apache2/vhosts.d/`

kde pro každého hosta je vymezen jeden soubor. Strukturu souboru nalezneme v příloze F.1. Nejdůležitější nastavení v souboru jsou uvedeny v tabulce 2.2, podrobnější přehled nastavení nalezneme v dokumentaci webovému serveru Apache [7].

Volba	Popis
ServerName	doménový název hosta, podle něj server určuje, který host se bude zpracovávat (k jeho rozšíření slouží volba <i>ServerAlias</i> )
DocumentRoot	adresář ke kořenu webu
Directory	nastavení chování jednotlivých adresářů
Log	nastavení určující strukturu a uložení záznamů při vyřizování požadavku (chyby, výjimky, přístupy, atd.)
SSLEngine	zapnutí nebo vypnutí zabezpečeného spojení
SSLProtocol	minimální verze protokolu, kterým se bude komunikovat

Tabulka 2.2: Některé důležité konfigurační volby webového serveru Apache pro hosta

Každý virtuální host bude uložen jako `z` na domovský adresář s daty účtu. Samotné odkazy budou uloženy v adresáři `/var/www/` webového serveru. `z` na adresář bude odpovídat adresáři hosta, tedy `/var/www/ucet` bude směřovat na adresář `ucet` v uživatelských účtech.

Pomocí navrhované aplikace se bude pracovat s jednotlivými virtuálními hosty a jejich nastavením. Serverová část aplikace bude také zodpovědná a vytváření a odstraňování odkazů na domovské adresáře v adresáři webového serveru.

### 2.3.4 Nastavení databázového serveru MySQL

Databázový server bude instalován a nastaven do výchozího stavu.

Navrhovaná aplikace bude vytvářet databáze pro jednotlivé uživatelské účty. Databáze budou uloženy ve výchozím datovém úložišti databáze. V případě MySQL databáze se jedná o adresář `/var/lib/mysql`. K tomuto adresáři musí mít přístup pouze uživatel `root` a samotný databázový server. V opačném případě by si mohl, kterýkoliv uživatel prohlížet databáze ostatních uživatelů.

#### Kvóty pro databáze

Protože databáze MySQL nenabízí systém nastavení kvót pro databáze nebo uživatele, bude instalován také program, který dané kvóty zprostředkuje. Tento program bude spuštěn jako služba na pozadí serveru, a bude v určitých intervalech kontrolovat databáze. Interval kontroly bude nastaven optimálně tak aby zbytečně nezatěžoval server (např.: 15 minut). V tomto intervalu budou klienti moci daný limit přesáhnout, ale po opětovné kontrole již budou databáze blokovány.

### 2.3.5 Nastavení FTP serveru ProFTPD

Server bude sloužit uživatelům pro přístup k domovským adresářům a jejich správě pomocí FTP protokolu. Samotný server ProFTPD bude instalován a nastaven dle [12]. V nastavení je potřeba některé volby upravit podle potřeby.

První úprava nastavení metody ověřování uživatelů ze systému. Změna se provede volbou `AuthPAM` do stavu `on`. Tato volba způsobí, že se přes protokol FTP budou moci připojit všichni uživatelé v systému daného serveru. Přidají se také některé konfigurační volby pro globální nastavení:

```
<Global>
```

```
AllowRetrieveRestart on
```

```

    AllowStoreRestart on
    DefaultRoot ~
    ListOptions "-a"
    RootLogin off
</Global>

```

Popis voleb nalezneme v tabulce 2.3. Výpis všech konfiguračních voleb a nastavení serveru nalezneme v dokumentaci [13].

Volba	Popis
AllowRetrieveRestart	povolení obnovení přerušného stahování souboru
AllowStoreRestart	povolení obnovení přerušného nahrávání souboru
DefaultRoot	nastaví kořenový adresář na domovský adresář uživatele
ListOptions	zapnutí zobrazování linuxových skrytých souborů
RootLogin	povolení nebo zakázání přihlášení root uživatele

Tabulka 2.3: Volby globálního nastavení ProFTPD serveru

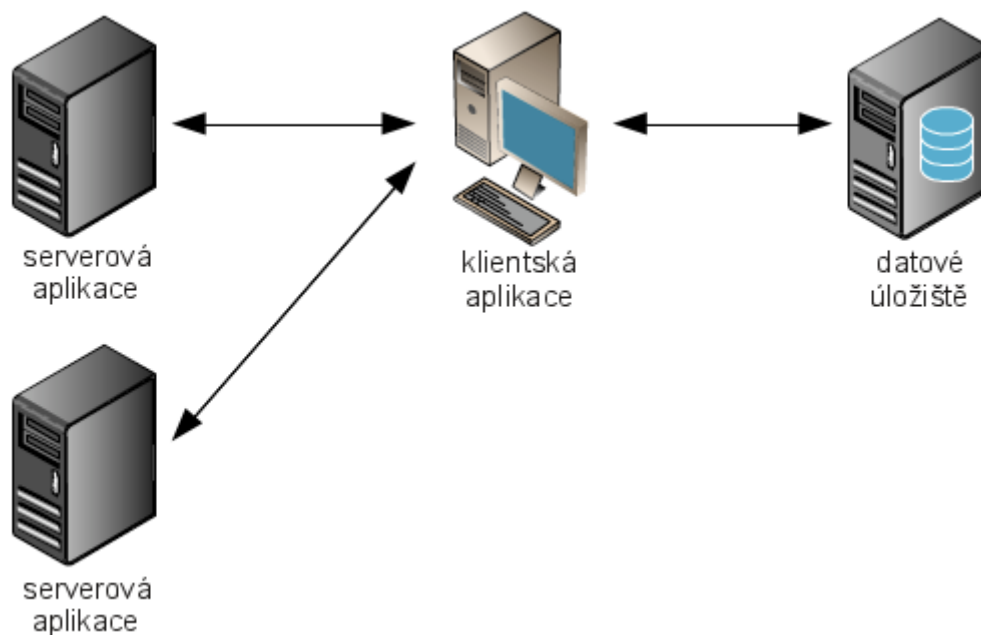
Protože FTP server bude navázán na systém účtů na serveru, nebude tedy nutné již žádné další nastavení a ani samotná serverová část aplikace nebude muset s FTP serverem komunikovat a měnit jeho vlastnosti.

### 2.3.6 Návrh aplikace klient/server

Podle požadavku na možnost obsluhy různých serverů z několika aplikací, bude systém rozdělen na tři základní části. Každá část bude obsluhovat pouze tu část, ke které je určena a nebude zasahovat do ostatních. Schéma propojení jednotlivých částí lze nalézt na obrázku 2.1.

Jednotlivé části systému:

1. Serverová aplikace
2. Klientská aplikace
3. Datové úložiště



Obrázek 2.1: Schéma propojení částí aplikací a datového úložiště

### **Serverová aplikace**

Serverová část bude na serveru obsluhovat jednotlivé služby běžící na daném serveru. Bude také komunikovat s klientskou částí pomocí předem definovaného protokolu. Popis protokolu nalezneme v kapitole 3.3. Serverová část bude obsluhovat pouze služby, které jsou na daném serveru v provozu.

### **Klientská aplikace**

Klientská část bude spuštěna na straně klienta jako aplikace s grafickým uživatelským rozhraním. Pomocí ní bude uživatel přes serverovou část pracovat s účty na samotných serverech. Tato aplikace bude přistupovat jak k serverové části pomocí protokolu, tak k datovému úložišti pomocí SQL dotazů.

### **Datové úložiště**

Datové úložiště bude sloužit k ukládání všech dat v celém systému. Bude navrženo jako databáze, která bude uložena na serveru v interní síti. K této části bude přistupovat pouze klientská aplikace, která zde bude ukládat strukturu klientských účtů, databází, poštovních schránek a jiné. Databáze bude také obsahovat podrobnější

informace o klientovi (adresa, aj.) a seznam všech serverů a přiřazených typů služeb, spuštěných na těchto serverech.

Pomocí struktury datového úložiště, bude možné analyzovat strukturu účtů na daném serveru nebo jejich rozložení mezi servery. V datové struktuře se nebudou ukládat hesla k účtům a to z bezpečnostních důvodů. Pokud by tak došlo ke zcizení databáze, nedošlo by k ohrožení klientů.

## 3 IMPLEMENTACE APLIKACE

Implementace serverové i klientské aplikace bude provedena v jazyce C++ a pomocí frameworku Qt.

### 3.1 Qt framework

Qt framework je Open Source (otevřený zdrojový kód) C++ framework, který vyvíjí firma Nokia. Jedná se o platformě nezávislý aplikační framework s možností využití Graphical User Interface (GUI). Pro grafická rozhraní využívá návrhový vzor Model-View-Controller (MVC), podle kterého jsou data oddělena od jejich zpracování a pohledu na ně. Tento framework lze použít jak pro vývoj aplikací pro příkazové řádky, tak pro vývoj aplikací s grafickým rozhraním.

Základní částí celého frameworku je část *QtCore*. Ta obsahuje hlavní třídy pro tvorbu aplikace (především třídu *QObject*<sup>1</sup>) a třídy pro lokalizaci aplikace. Zbytek frameworku je rozdělen do modulů. Tyto moduly rozšiřují základní funkčnost o další možnosti.

Modul	Popis
<i>QtCore</i>	Základní jádro celého frameworku.
<i>QtGui</i>	Základní modul pro práci s uživatelským rozhraním aplikací.
<i>QtNetwork</i>	Modul pro práci se síťovou komunikací. Obsahuje implementaci jak klientské tak serverové části.
<i>QtSql</i>	Modul pro práci s databázemi. Umožňuje přístup do databází pomocí dotazů v jazyce SQL.
<i>QtOpenGL</i>	Modul pro práci s OpenGL grafikou.
<i>QtXml</i>	Modul pro práci s eXtensible Markup Language (XML) dokumenty.

Tabulka 3.1: Výběr některých modulů frameworku Qt a jejich stručný popis

<sup>1</sup>Základní třída Qt frameworku. Tuto třídu dědí všechny ostatní třídy.

Výhodou použití modulárního rozdělení je snížení rozsahu závislostí aplikace. Některé moduly mohou být závislé na knihovnách v systému a musely by být také instalovány, i když je aplikace nebude potřebovat. Typickým příkladem může být modul *QtSQL*, u kterého musí být pro správnou funkčnost instalována alespoň jedna knihovna s přístupem k SQL databázi (typicky SQLite). V tabulce 3.1 jsou uvedeny některé moduly, dodávané přímo s frameworkem Qt.

Více informací a struktury Qt frameworku lze nalézt v oficiální dokumentaci [14].

### 3.1.1 Kompilace projektů v Qt frameworku

Základním souborem všech projektů vytvořených v Qt frameworku je soubor s informacemi o projektu. Jedná se o soubor s příponou *.pro*. V tomto souboru je celý projekt popsán. Jsou zde především uvedeny všechny zdrojové soubory, připojené knihovny, šablony, atd. Příklad obsahu souboru projektu pro serverovou aplikaci nalezneme v příloze B.

Je zde především uvedeno:

- Druh projektu - jestli se jedná o aplikaci, sdílenou knihovnu, plugin pro rozšíření Qt frameworku nebo sjednocení více projektů.
- Typ aplikace/knihovny - o jaký druh aplikace se jedná (aplikace s GUI nebo konzolová aplikace).
- Moduly - které moduly se mají k projektu připojit a které se naopak k projektu připojovat nesmí. U konzolové aplikace tak lze odstranit závislost na grafickém rozhraní.
- Knihovny - další knihovny, které se mají k projektu připojit, pomocí této volby se připojují externí knihovny.
- Zdroje - seznam všech souborů se zdrojovými kódy.
- Hlavičky - seznam všech hlaviček zdrojových souborů.
- Formuláře - seznam všech formulářů připojených k projektu.

- Adresář sílové aplikace - adresář, kam se má uložit přeložený kód.
- Obrázkové zdroje - seznam souborů s cestami k obrázkům použitých v projektu (nejčastěji ikony).
- Překlady - seznam překladů pro různé jazykové mutace.

Samotný projekt se kompiluje do zdrojové podoby pomocí příkazu *qmake*. Tento příkaz projde soubor projektu, připojí dané knihovny a vytvoří soubor *MakeFile*<sup>2</sup> s postupem kompilace. Po zpracování projektu pomocí *qmake* již lze projekt přeložit a popřípadě nainstalovat pomocí standardního příkazu *make*.

Postup kompilace projektu v příkazech, předpokládá se uložení projektu ve složce *source*:

```
sh ~ # qmake ./source/project.pro
sh ~ # make
sh ~ # make install #instalace projektu
```

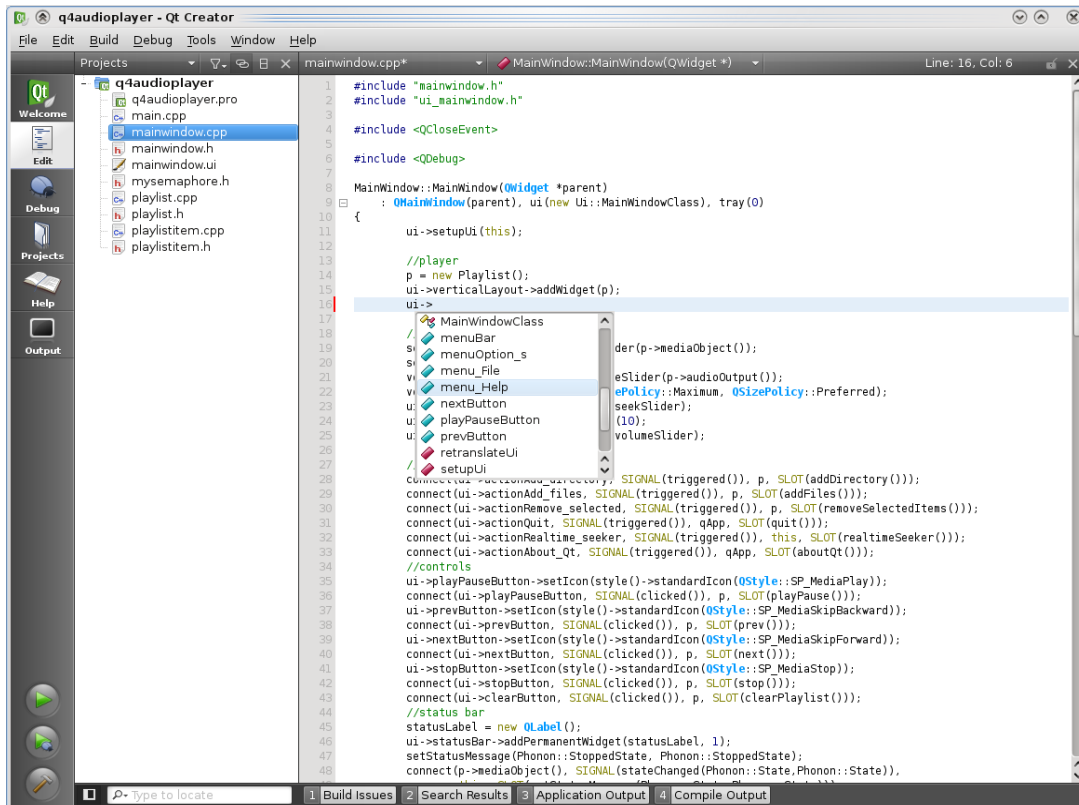
### 3.1.2 Nástroje pro práci v Qt frameworku

Asi nejznámějším nástrojem pro tvorbu aplikací v Qt frameworku je QtCreator. Jedná se o kompletní vývojové prostředí pro vývoj projektů v Qt frameworku, které je vytvořené přímo vývojářským týmem Qt frameworku a soustředí se tedy především na tento framework. Ukázku rozhraní lze nalézt na obrázku 3.1, kde je otevřen zdrojový kód. Samotné vývojové prostředí si s sebou nese všechny potřebné knihovny Qt frameworku a to i kompletní dokumentaci. Po instalaci je ihned možné vytváření aplikací. V QtCreatoru lze také aplikace ladit pomocí integrovaného ladícího nástroje (debugeru) a sledovat tak obsahy proměnných v bodech přerušení (breakpoints).

QtCreator obsahuje tyto části:

---

<sup>2</sup>Soubor *MakeFile* určuje postup kompilátoru při kompilaci zdrojových kódů a řeší závislosti mezi zdroji



Obrázek 3.1: Uživatelské rozhraní QtCreatoru

- Integrated Development Environment (IDE) pro psaní zdrojových kódů (doplnuje zvýraznění syntaxe a doplňování kódu jak z knihoven tak z projektu).
- Návrhář pro tvorbu formulářů a grafických prvků.
- Režim pro ladění aplikace
- Dokumentaci k frameworku.
- Správu projektů.

Vývojáři Qt frameworku také poskytují rozšíření pro vývojové prostředí Microsoft Visual Studio. Toto rozšíření přidává podporu Qt frameworku a návrháře formulářů přímo do Visual Studia, takže aplikace lze vyvíjet stejně jako pro platformu dotNet (.NET). Projekty v Qt frameworku lze samozřejmě psát v jakémkoliv editoru či vývojovém prostředí, které podporuje programování v C++.

## 3.2 Datového úložiště

Datové úložiště je jednou z hlavních částí celého projektu. Jsou v něm soustředěny všechny informace o klientských účtech a serverech. Samotné datové úložiště je realizováno pomocí databáze s několika tabulkami uloženými v databázovém serveru MySQL.

Do datového úložiště se ukládají informace, které popisují strukturu účtů nebo serverů. Například u databázového účtu je zde uvedeno pouze jméno databáze, adresa serveru, kde je databáze uložena a kvóta maximálního místa. Z bezpečnostních důvodů se zde neukládají hesla k jednotlivým účtům, protože při zcizení databáze by byly prozrazeny. Aby byly hesla ukládány, musel by být implementován systém šifrování nebo znemožněna krádež databáze (například omezení přístupu pouze vybraným IP adresám).

### 3.2.1 Jednotlivé tabulky úložiště

Datové úložiště obsahuje několik tabulek. Každá z tabulek má určitý význam a obsahuje pouze relevantní data k dané části.

#### **Tabulka *accounts***

Jedná se o hlavní tabulku celého datového úložiště, obsahuje seznam klientských účtů a jejich základní informace (např. adresa, poznámka, datum vytvoření, atd.). Tato tabulka také obsahuje sloupec pro zamknutí účtu, viz práce s datovým úložištěm 3.2.2.

#### **Tabulka *domains***

Tabulka obsahující všechny spravované domény, je navázána relací typu 1:N k tabulce s uživatelskými účty pomocí cizího klíče. Uchovávají se v ní pouze informace o doméně, její název a expirace.

V další verzi systému je plánováno rozšíření této tabulky pro propojení se systémem firmy *Ignum s.r.o.* pro správu domén. V tabulce by tak přibýly další sloupce s podrobnostmi o doméně (vlastník, fakturace, asociace s IP adresou podle služby).

Propojení systémů je velmi komplikovaný krok, který by byl na míru této práce a v dosavadní implementaci se s ním nebudeme zabývat.

### **Tabulka *servers\_types***

Tato tabulka obsahuje pouze typy serverů, nebo-li typy služeb, pro které lze přiřadit samotné servery.

### **Tabulka *servers***

Tabulka se seznamem serverů a jejich adresami. Z této tabulky se vybírají servery, přiřazené k jednotlivým službám. Podle typu serveru (viz kapitola 3.2.1 tabulka *servers\_types*) se data filtrují.

### **Tabulka *web\_accounts***

Tabulka, která obsahuje všechny účty pro webové servery. Tato tabulka je propojena relací 1:N s tabulkou *accounts* a relací 1:1 s tabulkou *servers*. Obsahuje informace o daném hostingovém účtu, tedy jeho název, expiraci, doménu přiřazenou k tomuto účtu, nastavení diskové kvóty, je-li je u konfigurace serveru použito výchozí nastavení a status účtu.

Podle nastavené domény se na serveru nastaví konfigurační soubor serveru. Domény se načítají ze seznamu domén pro tento účet. Nejsou však předávány pomocí cizího klíče domény, ale pomocí názvu samotné domény a to z důvodu nezávislosti na seznamu domén. Při zrušení domény by se tak odstranil i celý účet a uživatel by ztratil veškerá svá data.

Status účtu umožňuje jeho vypnutí, zapnutí či přesměrování z důvodu údržby. Při změně statusu se účet nemaže, resp. nevytváří, ale pouze se daný účet odpojí, resp. připojí.

### **Tabulka *db\_accounts***

Tabulka obsahuje všechny databázové účty. Tato tabulka je opět navázána relací 1:N k tabulce s uživatelskými účty a relací 1:1 k tabulce se servery. Opět obsahuje pouze základní informace o databázovém úložišti a to jméno databáze, přidělenou

kvótu, datum vytvoření a datum expirace. V této tabulce nejsou uloženy informace o typu serveru (MySQL, PostgreSQL, ...) a účet je tedy nezávislý na použitém databázovém serveru.

### **Tabulka *mail\_adm\_accounts***

Tato tabulka obsahuje seznam všech administrátorských účtů k poštovním schránkám k dané doméně. Je navázána relací 1:N k tabulce s klientskými účty a relací 1:1 k tabulce se servery. Jsou v ní uvedeny pouze potřebné údaje popisující strukturu účtů a jejich nastavení. A to název účtu (bez domény), doména spravovaná pomocí vybraného účtu, datum vytvoření a expirace, velikost diskové kvóty pro všechny schránky v dané doméně, maximální počet aliasů a schránek pro danou doménu.

Doména k účtu není navázána na seznam domén patřících ke klientskému účtu, ale je zkopírována. opět z důvodů stejných jako pro tabulku *web\_accounts*.

Kompletní strukturu tabulek i s datovými typy lze nalézt v příloze C. Ve schématu, uvedeném v příloze C, jsou znázorněny také vazby mezi jednotlivými tabulkami a jejich klíči.

## **3.2.2 Práce s datovým úložištěm**

Přístup k datovému úložišti využívá pouze klientská aplikace, která zde ukládá potřebná data do předem zmíněných tabulek. Serverová část aplikace s datovým úložištěm nebude vůbec pracovat viz obrázek 2.1.

Samotný přístup z klientské aplikace je prováděn pomocí tříd *QSqlTable*, *QSqlTableRelation* a *QSqlRecord* Qt frameworku. Qt framework využívá k přístupu k datům v databázi návrhový vzor *Active Record*. Tím vytváří databázově nezávislou vrstvu mezi položkami v databázi a jejich reprezentací v aplikačním kódu. Datové úložiště je tak nezávislé na použité databázi a se záznamy tak lze pracovat bez omezení, jako například nepodporované transakce v MySQL u databázového úložiště typu My Indexed Sequential Access Method (MyISAM).

K přístupu k jednotlivým datům je použita třída *QSqlRecord*, která reprezentuje jeden řádek z tabulky. Podrobnější informace o použití tříd obsluhujících databázi

lze nalézt v dokumentaci k Qt framework [14] v modulu pro práci s databázemi *QtSQL*.

### 3.2.3 Zabezpečení komunikace s datovým úložištěm

V implementované aplikaci je komunikace mezi klientskou aplikací a databází nešifrována. Pokud by však komunikace byla prováděna přes nezabezpečenou síť, je dobré komunikaci šifrovat. Pro zabezpečení jsou přenášena data neobsahující hesla uživatelských účtů a tedy odposlechnutí komunikace nepředstavuje velký bezpečnostní problém. Útočník se dozví jak je účet nastaven, na kterém serveru má data, ale k samotným datům se již nedostane.

Větší nebezpečí představuje samotné přihlášení klientské aplikace k databázi. V ní při přihlašování je přeneseno nešifrované heslo k databázi, které může útočník odposlechnout. Po zjištění hesla si bude moci upravovat účty, zjišťovat e-mailové adresy a jiné.

Databázový server typu MySQL šifrování komunikace podporuje, a to pomocí SSL protokolu. Pro zavedení komunikace je tak nutné vytvořit klíče (privátní a veřejný), pomocí těchto klíčů pak budou informace šifrovány.

V implementované aplikaci bude komunikace mezi datovým úložištěm a aplikací nešifrovaná. Předpokládá se uložení na interním databázovém serveru. Po úspěšném odladění aplikace bude databáze přesunuta na jeden z databázových serverů a tam udržována. Pro komunikaci pak bude využito šifrování.

## 3.3 Komunikační protokol

Komunikační protokol je další důležitou součástí celého systému. Definuje pravidla, jak spolu budou klientská a serverová aplikace komunikovat. Především určuje strukturu příkazů a přenášených dat. Určuje také, jak se budou celá data přenášet přes síť.

Celý protokol je tvořen jako dynamická knihovna, která se zavádí při spuštění aplikace. V linuxových systémech se jedná o soubory s příponou *so* (MS Windows používá *dll*). Dynamická knihovna umožní použití stejných zdrojových kódů jak pro klientskou tak pro serverovou aplikaci. Knihovnu stačí přeložit pro danou platformu a nakopírovat do všech ostatních aplikací. Samotná aplikace ji načte při startu. Není tedy nutné překládat celou aplikaci znovu, při menších změnách komunikačního protokolu.

Knihovna s implementovaným protokolem slouží k zapouzdření jednotlivých příkazů a jejich dat a přenos pomocí Transmission Control Protocol (TCP) protokolu, v aplikaci se chová jako objekt. Při vytvoření objektu mu jsou předány parametry komunikace, buď IP adresa a port počítače, ke kterému se má připojit, nebo již vytvořený soket<sup>3</sup>. Parametr s vytvořeným soketem je využit v serverové aplikaci. Předávání parametrů je využito u klientské aplikace, kde se aplikace připojuje k různým počítačům.

### 3.3.1 Přenos příkazů a dat

Přenos dat v protokolu se provádí na transportní vrstvě protokolu TCP. Tato vrstva je zvolena z důvodu zaručení správného doručování datagramů i doručování datagramů ve správném pořadí.

V knihovně je k implementaci komunikace použitý modul Qt frameworku *QtNetwork*. Obsahuje třídu *QTcpSocket*, která implementuje komunikaci na transportní vrstvě TCP protokolu.

Celé příkazy jsou tvořeny jako textový řetězec, který se dále dělí na části, oddělené oddělovačem. V implementovaném protokolu se jedná o American Standard

---

<sup>3</sup>Soketem rozumíme komunikační kanál, ve kterém se přenáší data.

Code for Information Interchange (ASCII) k  $1E_{HEX}$  - *record separator* (oddělovač záznamů). k  $1E_{HEX}$  je zvolen z důvodu bezpečnosti, v běžných textech se nevyskytuje a nehrozí tak kolize s neúmyslným vložením. Pro kontrolu jsou ještě oddělovače záznamů z přenášených dat odstraněny. Pro tvorbu komplikovanějších struktur lze v přenášených datech použít k  $1D_{HEX}$  "group separator". V implementovaném protokolu však zatím není využit, bude se využívat při přenosu podrobnějších seznamů (např. seznam e-mailů).

Jako první se v celém bloku nachází příkaz, podle kterého server zjistí jakou funkci má provést. Tento příkaz je pevně daný protokolem a nelze jej měnit. Je to z důvodu zpětného rozložení a určení, co se má podle příkazu provést. a samotným příkazem následuje oddělovač záznamů  $1E_{HEX}$ , po něm uživatelská data. Pokud je dat více, jsou také odděleny oddělovačem záznamů  $1E_{HEX}$ . Popis příkazů lze nalézt v kapitole 3.3.4. Podrobný popis všech příkazů a jejich parametrů pak nalezneme v příloze D.

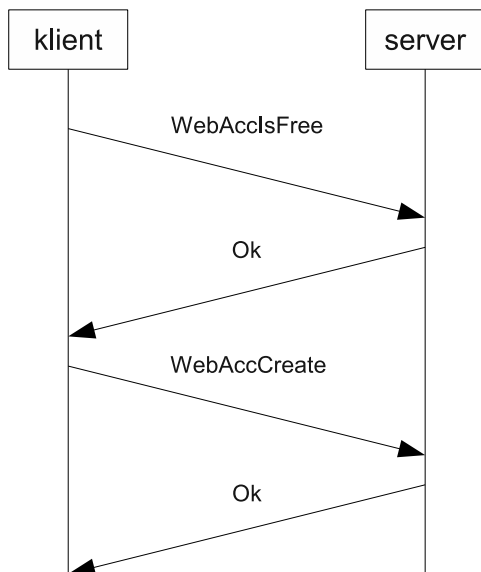
Struktura příkazu protokolu:

```
COMMAND 1E DATA
```

```
COMMAND 1E DATA 1E DATA 1E DATA
```

### 3.3.2 Komunikace pomocí protokolu

Samotnou komunikaci protokolem mezi serverovou a klientskou aplikací navazuje vždy klientská aplikace, jako první se pokusí připojit k vybranému serveru. Pokud připojení selže, je klient informován o neschopnosti serveru odpovídat a celá operace je přerušena. Pokud server na připojení odpoví, je komunikace zahájena a lze přenášet příkazy s daty. Po odeslání každého příkazu klientská aplikace čeká na jeho zpracování. Po zpracování příkazu server vrací odpověď o stavu úlohy. Pokud akce proběhne správně, je přenesen příkaz *Ok* zpět ke klientovi. Pokud ale akce selže, je klient informován chybovým příkazem *Error*. V chybové hlášce může být také přenesena další data, například chybová zpráva. Až po pozitivní odpovědi od serveru pošle klient další příkaz ke zpracování. Zjednodušený příklad komunikace pro vytvoření



Obrázek 3.2: Příklad komunikace klienta se serverem při vytváření webového účtu

webového účtu nalezneme na obrázku 3.2. Pro uložení upravených dat do datového úložiště je nutné, aby všechny příkazy proběhly korektně.

### 3.3.3 Zabezpečení komunikace

Komunikace pomocí vytvořeného protokolu je nezabezpečená, protože umožňuje odposlech a odhalení hesel uživatelských účtů, která se přenášejí v datech. Proto lze v implementaci protokolu vyměnit komunikaci pomocí *QTcpSocket* a třídu *QSSL-Socket*, která vytváří soket, kterým lze šifrovat komunikaci pomocí SSL klíčů. V implementaci šifrování použito není, z důvodu ladění protokolu aplikací Wireshark, který zachytává a analyzuje veškerou síťovou komunikaci.

Ve finální implementaci aplikace však bude zavedeno šifrování přenosů, a to tak, že serverová aplikace bude disponovat veřejným klíčem a klientská aplikace pak bude obsahovat privátní klíč, pomocí kterého se bude přihlašovat k serverové aplikaci.

Implementace nebude obsahovat žádný systém přihlašování k serveru pod uživatelským jménem nebo heslem, z toho důvodu, že se jedná o aplikace určené administrátory, a samotné ověření aplikace se provede pomocí klíčů. Nevýhodou využití

šifrování pomocí klíčů je možnost jejich ukradení. V takovém případě bude nutné vytvořit nové klíče a předat je ke všem serverovým i klientským aplikacím. Tomuto scénáři by šlo předejít využitím automatické distribuce klíčů ke všem serverovým a klientským aplikacím, systémem přihlášení administrátorů k datovému úložišti. Datové úložiště by však muselo obsahovat systém pro ověření uživatelů.

### 3.3.4 Příkazy protokolu

Implementovaný protokol obsahuje velké množství příkazů, tyto příkazy jsou rozděleny do čtyř skupin, kde každá skupina je určena ke správě dané služby. Vyjimku tvoří poslední skupina obecných příkazů, která je nejčastěji využívána pro odpovědi. Samotná struktura příkazu je uvedena v kapitole 3.3.1.

Seznam skupin:

- GroupMain - hlavní skupina s příkazy. Jsou zde obsaženy příkazy pro zajištění komunikace.
- GroupWebAcc - skupina pro práci s webovými účty.
- GroupDbAcc - skupina pro práci s databázovými účty.
- GroupMailAcc - skupina pro práci s e-mailovými účty.

#### Příkazy pro webové účty

Příkazy pro webové účty patří do skupiny *GroupWebAcc* a jsou určeny ke správě webových účtů na serveru. Pomocí těchto příkazů se účty vytvářejí, nastavují, odstraňují aj. Příkazy z této skupiny začínají řetězcem *WebAcc*. Po tomto úvodním řetězci je již určitý příkaz (např.: *Create* pro vytvoření účtu).

#### Příkazy pro e-mailové účty

Příkazy e-mailové účty patří do skupiny *GroupMailAcc* a jsou určeny ke správě e-mailových účtů na serveru. Tato skupina se dělí ještě na dvě podskupiny, první skupina je určena pro práci s administrátorskými účty poštovních schránek, z kapitola 2.3.2. Příkazy z této skupiny začínají řetězcem *MailAdmAcc*. Po řetězci jsou

již příkazy pro samotnou práci. Druhá skupina je určena pro práci se samotnými schránkami uživatelů. Příkazy pro tuto skupinu začínají řetězcem *MailAccMail*, poté samotným příkazem.

### **Příkazy pro databázové účty**

Příkazy pro databázové účty se řadí do skupiny *GroupDbAcc*. Jedná se o skupinu s nejmenším počtem příkazů, protože práce s databázovým účtem nepotřebuje velké množství informací. Převážně si vystačí s příkazy pro vytvoření a smazání databázového účtu.

### **Ostatní příkazy**

Ostatní příkazy se řadí do skupiny *GroupMain* a nejsou určeny ke speciálním účelům. Pomocí nich probíhá potvrzování operací a základní kontrola protokolu. Patří zde například příkazy *Ok* nebo *Error*.

## 3.4 Serverová část

Další hlavní součástí systému je serverová aplikace, je instalována na každém serveru, který má být aplikací obsluhován. Je spuštěna jako služba na pozadí, kde přijímá a zpracovává požadavky klientů.

Na serveru lze spustit tuto aplikaci na pozadí (jako daemon) pomocí příkazu:

```
start-stop-daemon --start --user whrc:whrc /usr/local/bin/whrc_server
```

Příkaz spustí aplikaci *whrc\_server* pod uživatelem *whrc* a skupinou *whrc*. Po spuštění již aplikace běží na pozadí a lze k ní přistupovat z klientských aplikací.

Samotný příkaz *start-stop-daemon* vytváří pro daný proces soubor, který identifikuje jestli je proces spuštěn nebo zastaven. Nejčastěji se tyto soubory ukládají do adresáře */var/run* a název souboru se odvíjí od názvu aplikace. Pro předchozí příkaz by byl vytvořen soubor *whrc\_server.pid*. Po zastavení procesu aplikace je soubor s identifikací automaticky odstraněn.

Pro zastavení spuštěné aplikace lze použít následující příkaz:

```
start-stop-daemon --stop /usr/local/bin/whrc_server
```

Uživatel, pod kterým je aplikace spuštěna, musí mít podle dané konfigurace nastavena potřebná přístupová práva. Například při použití modulu pro domovské adresáře (*storageunixhomedir*), musí mít uživatel oprávnění vytvářet, resp. mazat uživatelské účty.

### 3.4.1 Obsluha komunikačního protokolu

V aplikaci je použita implementace komunikačního protokolu zmíněná v kapitole 3.3. Protokol je k aplikaci připojen jako knihovna (viz kapitola 3.3), což umožňuje výměnu knihovny bez nutnosti nového přeložení celé aplikace.

Samotná knihovna je k aplikaci přilinkována<sup>4</sup> při jejím překladu do strojového kódu, což zajistí její přímé načtení při spuštění aplikace a nemožnost spuštění, pokud knihovna v systému není instalovaná.

Po spuštění aplikace je vytvořen komunikační soket pomocí vytvoření objektu typu *QTcpServer*. Tento objekt je pomocí signálu/slotu<sup>5</sup> připojen k metodě, která se stará o zpracování příkazů. V této metodě je vytvořen nový objekt protokolu, kterému je při vytváření předán ukazatel na objekt nově otevřeného soketu. Pomocí tohoto soketu aplikace komunikuje s klientskou aplikací.

Nastavení objektu *QTcpServer* pro komunikace se provádí přes nastavení aplikace, která je popsána v kapitole 3.4.2. Z tohoto nastavení se pro objekt komunikace vybere pouze adresa, na které má aplikace naslouchat<sup>6</sup> a port.

Aplikace je postavena jako neblokující, tedy schopna obsloužit několik klientů najednou. Tento systém je vytvořen vnitřní logikou objektu *QTcpServer*. Objekt při vytvoření obsahuje smyčku, která kontroluje příchozí připojení. Při detekci připojení je tímto objektem vyslán signál, který je připojen na výše zmíněnou metodu pro zpracování a která daný signál přijme. Hlavní smyčka objektu *QTcpServer* může běžet dále a čekat na další připojení.

Po ukončení komunikace je soket odstraněn a uvolněna použitá paměť, vyhrazená pro vytvořené spojení. Ostatní připojení klienti jsou obsluhováni zvlášť a nejsou tedy odpojením nijak ovlivněni.

### 3.4.2 Nastavení serverové části

Pro spuštění aplikace je nutné vytvořit její konfigurační soubor, který obsahuje nastavení, podle kterého se aplikace bude chovat.

V konfiguračním souboru je především uvedena IP adresa a port, na kterých má aplikace naslouchat na příchozí připojení od klientů. Obsahuje také informace o modulech, které se mají po spuštění načíst.

---

<sup>4</sup>Do aplikace jsou jen vloženy odkazy na potřebné funkce v knihovně.

<sup>5</sup>Qt Framework využívá pro interakci objektů signály a sloty. Více o signálech a slotech lze nalézt v dokumentaci ke Qt Frameworku [14]

<sup>6</sup>Pokud se nastaví adresa na *0.0.0.0*, bude aplikace naslouchat na všech adresách dostupných v systému

jádro serveru					
knihovna protokolu	zpracování požadavku				
	Zpracování požadavku webového účtu		Zpracování požadavku databázového účtu		Zpracování požadavku poštovního účtu
	Modul uložení účtu	Modul webového serveru	Modul DB serveru	Modul DB kvóty	Modul poštovního serveru

Obrázek 3.3: Vnitřní struktura serverové aplikace

Syntaxe souboru je intuitivní, a lze ji provádět v jakémkoliv textovém editoru. Spuštěná aplikace musí mít samozřejmě práva pro čtení tohoto souboru, ale z bezpečnostních důvodů by neměla mít povolen zápis. Příklad konfiguračního souboru lze nalézt v příloze E.1. V tomto souboru je nastaveno naslouchání na adrese *0.0.0.0* a portu 2266.

### 3.4.3 Modulární systém

Samotná serverová část je napsána tak, aby byla lehce rozšiřitelná a umožňovala různé druhy konfigurací serverů služeb. Proto byl zvolen modulární systém z obrázek 3.3, který umožňuje jednoduchou změnu chování systému, aniž by bylo nutné přepisovat zdrojové kódy či složitě upravovat nastavení. Každý modul systému má na starost pouze o jednotlivou část, kterou má obsluhovat.

Podle načtených modulů serverová aplikace určuje, jaké služby je schopna poskytovat a jaké druhy účtů spravovat. Služby, ke kterým nemá načtené moduly obslužit odmítne s chybou o nepodporované službě.

Použití modulárního systému také umožňuje jednodušší úpravy celého systému. Například při změně webového serveru *Apache* a *nginx* lze pouze v nastavení aplikace změnit modul obsluhující webový server a aplikaci znovu restartovat. Samotný modul se již postará o ostatní věci týkající se daného serveru.

Při zpracování příkazu uvnitř aplikace si aplikace nejprve zjistí, o jaký typ

příkazu se jedná a předá jej příslušnému modulu či modulům. Modul resp. moduly se poté postarají o vykonání akce definované přijatým příkazem.

Každý z modulů také může obsahovat vlastní nastavení. Pomocí těchto nastavení lze moduly dále konfigurovat, nezávisle na nastavení aplikace. Například databázový modul bude obsahovat uživatelské jméno a heslo pro přístup do databáze, kde budou vytvořeny účty uživatelů. Samotné nastavení jednotlivých modulů se v souborech ukládá do složky */etc/whrc/modules*. Jestli modul potřebuje vlastní nastavení závisí pouze na jeho implementaci.

## Rozdělení skupin modulů

Systém využívá pro moduly pět skupin, kde každý modul z dané skupiny obsluhuje pouze svou část systému. Například modul ze skupiny *WebAcc* obsluhuje pouze nastavení účtů u webového serveru, ale neobsluhuje diskovou kvótu přidělenou k tomuto účtu.

Základní skupiny modulů:

- *Storage* - skupina starající se o samotné uložení uživatelských účtů na disku, nastavení kvót pro tyto účty, atd. (k těmto účtům se přistupuje přes FTP protokol).
- *WebAcc* - skupina pro nastavení webového serveru přiřazeného k danému webovému účtu.
- *DbAcc* - skupina obstarávající databázové účty.
- *DbAccQuota* - skupina pro práci s kvótami databázových účtů.
- *MailAcc* - skupina obstarávající poštovní účty (administrátory schránek a samotné schránky).

## Definice rozhraní

Samotné moduly jsou v serverové aplikaci definovány jako rozhraní. Tyto rozhraní implementují moduly z dané skupiny. Rozhraní musí být v modulu kompletně implementované, jinak by se daný modul nepodařilo načíst.

V serverové aplikaci jsou vytvořena tato rozhraní:

- *accstorageinterface* - Rozhraní pro tvorbu modulů pro práci s webovými účty uživatelů.
- *dbserverinterface* - Rozhraní pro implementaci správy databázových účtů.
- *dbquotainterface* - Rozhraní pro tvorbu modulů obsluhujících kvóty databázových účtů.
- *mailserverinterface* - Rozhraní modulu pro obsluhu účtů na poštovním serveru.
- *webserverinterface* - Rozhraní pro tvorbu modulů pro obsluhu webových serverů.

Tato rozhraní jsou definována pouze jako hlavičkové soubory, ve kterých jsou uvedeny metody, které implementující modul musí implementovat. Pomocí těchto předdefinovaných metod serverová aplikace komunikuje s modulem.

### Nahrávání modulů

Samotná serverová aplikace si po spuštění a po načtení jejího nastavení načte potřebné moduly z daných skupin a uloží si je do paměti jako objekty, které využívá až v okamžiku komunikace s klientskou částí.

Nahrávání modulů se provádí pomocí objektu *QPluginLoader*. Objektu se při vytváření předá parametrem adresářová cesta k souboru s modulem (například */usr/lib/whrc/libmodule.so*). Po úspěšném nahrání modulu se testuje, jestli daný modul implementuje rozhraní pro daný modul. Pokud v modulu není implementováno dané rozhraní, aplikace vyvolá chybový návratový kód a ukončí se s chybovou hláškou o chybném modulu.

### 3.4.4 Modul datového úložiště

V implementované aplikaci je implementován jako modul *storageunixhomedir*. Již z názvu vyplývá že daný modul je určen pro uložení účtů v domovských adresářích na Linux systémech.

Především slouží pro obsluhu datového úložiště pro webové hosty. Umožňuje jejich vytváření, mazání, přejmenování. Upravuje také nastavení kvót pro jednotlivé účty. Samotná kontrola kvót je přenechána na souborovém systému, který při překročení zamezí zápisu do adresáře, pro který je kvóta určena.

### Implementace příkazů systému

K práci s účty je využito standardních linuxových příkazů pro práci s uživateli (*useradd*, *usermod*, ...), pro nastavení kvóty v souborovém systému (*edquota*) a některé další příkazy např. ke zjištění velikost adresáře příkaz *du*.

Samotné příkazy jsou spuštěny pomocí objektu *QProcess*, který spouští daný příkaz. Po provedení je vrácen návratový kód příkazu, podle kterého lze určit jak proběhl. Pokud příkaz proběhl správně, je vždy vrácen návratový kód 0. Při chybném návratovém kódu je také vrácena chybová zpráva příkazu, která se přes výstup modulu vrací zpět do serverové aplikace a je odeslána klientu.

### Nastavení modulu

Pomocí nastavení tohoto modulu se nastavují základní parametry příkazu *useradd*, především:

- Název primární skupiny, do které bude uživatel patřit.
- Seznam ostatních skupin uživatele.
- Komentář uživatele.
- Příkazový interpret (ve výchozím stavu je použito */bin/false* pro zakázání příkazového řádku).

V nastavení lze také definovat další adresáře, které se mají vytvořit spolu s domovským adresářem. Kompletní nastavení modulu lze nalézt v příloze E.2.

### 3.4.5 Modul obsluhy webového serveru Apache

V aplikaci je implementován jako modul s názvem *webserverapache*. Je určen pro obsluhu webového serveru Apache. Modul pracuje s konfiguračními soubory serveru

i samotným serverem, protože po úpravě konfiguračních souborů je vždy nutné tomto serveru aktualizovat konfiguraci. Modul nepracuje s uživatelskými daty, pouze dostane od modulu *storage* cestu k adresáři, kde jsou data uložena (nutné pro nastavení konfiguračního souboru webového serveru).

### **Struktura konfiguračních souborů Apache**

Základním adresářem pro tento modul je */etc/apache2/whrc-sites*. V adresáři modulu jsou uloženy všechny konfigurační soubory serveru Apache pro jednotlivé účty<sup>7</sup>. Na základním adresáři je závislý adresář */etc/apache2/whrc-sites-enabled*, který obsahuje pouze odkazy na všechny aktivní účty.

Posledním adresářem je samotný datový adresář. V něm jsou uloženy data uživatelských účtů nebo odkazy na jejich adresáře. Ve výchozím stavu se jedná o adresář */var/www*, ale pomocí nastavení modulu jej lze změnit.

### **Šablona pro vytváření virtuálního hosta, její zpracování**

Modul při vytváření nového uživatelského účtu používá buď nastavení přenesené z klientské aplikace, nebo předem připravenou šablonu. Při ukládání nastavení jsou nejprve přepsány některé řetězce a údaje o účtu (například doména) a poté je obsah uložen jako nový konfigurační soubor webového serveru Apache do adresáře zmíněného v kapitole 2.3.3. Samotný příklad šablony lze nalézt v příloze E.3.1.

### **Nastavení modulu**

Modul obsahuje nastavení, jak pracovat s webovým serverem Apache (příkaz pro načtení konfigurace) a názvy adresářů použitých v adresářové struktuře účtu. Kompletní nastavení toho modulu lze nalézt v příloze E.3.

## **3.4.6 Modul pro obsluhu MySQL databáze**

Modul je určen ke správě databázových účtů v databázi. Pracuje jak s databázemi, tak s uživateli, vytvořenými k této databázi a majícími k ní plný přístup. Protože

---

<sup>7</sup>Jedná se soubory s definicí virtuálních hostů.

většina databází nemá přímou podporu diskových kvót, modul proto neobsahuje systém pro jejich přidělování.

V implementovaném řešení je modul nazván *dbservermysql*, a je určen ke správě databází uložených na serveru MySQL. Uvnitř modul využívá pro komunikaci s databází jazyk SQL. SQL dotazy jsou tvořeny obecným zápisem a lze je použít i pro některé jiné typy databázových serverů (např. PostgreSQL).

Nastavení modulu pro přístup k databázi je uveden v příloze E.4.

### **Napojení na databázi**

Na databázi je modul napojen pomocí třídy *QSqlRecord* a k databázi se připojuje pomocí uživatele, který má přístupová práva ke všem databázím. Pomocí tohoto uživatele jsou spravovány jak uživatelé tak databáze.

### **3.4.7 Modul kvót pro MySQL**

Modul je implementován jako obsluha externí aplikace *MySQL Quota Daemon* a její databáze. Modul přistupuje do databáze této aplikace a přímo upravuje záznamy v tabulce o kvótách. Aplikace si poté sama načte nastavení kvót pro dané databáze. Nastavení modulu nalezneme v příloze E.5.

### **Princip udělování kvót**

Aplikace běží na pozadí v systému a v pravidelných intervalech kontroluje určené databáze na obsazené místo. Pokud zjistí, že velikost databáze překračuje kvótu, odebere uživateli databáze právo pro vložení nových záznamů do této databáze. Po odebrání přebytečných záznamů je po pravidelném intervalu zase právo pro vkládání nových záznamů přiděleno.

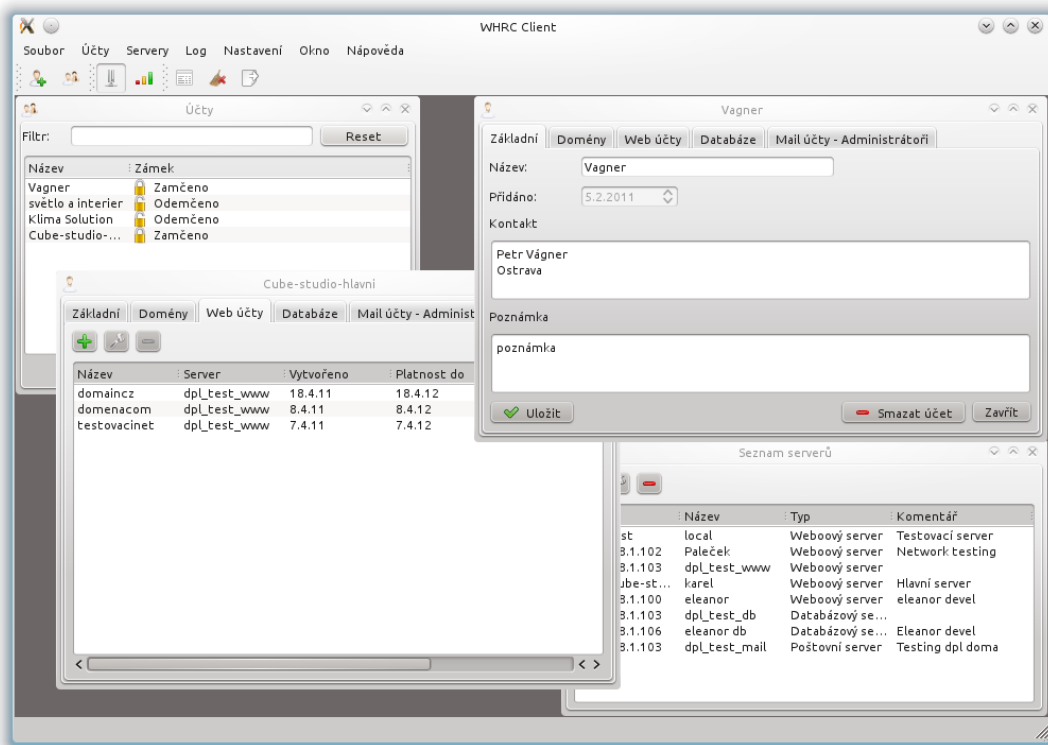
### **3.4.8 Modul pro práci s poštovními účty**

Modul je určen pro správu účtů administrátorů poštovních schránek a samotných poštovních schránek. V implementované aplikaci je modul vytvořen pro systém, kdy poštovní server *postfix* spolu se systémem *dovecot* ukládá účty v databázi. K této

databázi přistupuje také externí webová aplikace *postfixadmin*, která slouží administrátorům ke správě schránek v přidělených doménách. Jako název modulu byl zvolen *mailserverpostfixmysql*. Kompletní ukázkou nastavení modulu nalezneme v příloze E.6.

### **Napojení na databázi**

K databázi modul přistupuje opět pomocí parametrů, předaných v nastavení modulu. Jsou zde uvedeny všechny informace o tabulkách s uživateli, doménami atd. Je zde také vybráno šifrování hesel v databázi. Podporováno je v této verzi zatím pouze nešifrované heslo a otisk hesla v MD5.

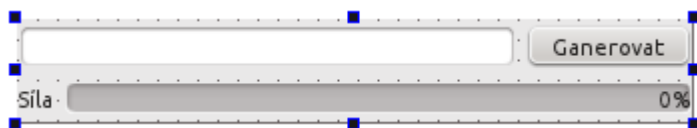


Obrázek 3.4: Hlavní okno aplikace s otevřenými účty a seznamem serverů

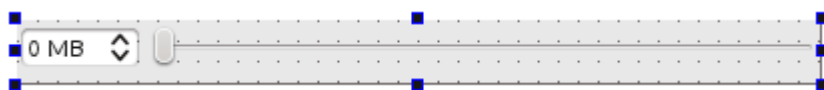
### 3.5 Klientská část

Poslední implementovanou částí celého systému je klientská aplikace. Slouží ke správě všech účtů a správě serveru a propojuje všechny hlavní části celého systému uvedených výše v kapitole 2.3.6, viz obrázek 2.1.

Jako jediná aplikace využívá knihovnu Qt frameworku pro grafické uživatelské rozhraní (GUI) *QtGui*, která přidává podporu pro tvorbu oken, formulářů, formulářových prvků, volby vzhledu a práci s nimi. Použití toho modulu výrazně zjednoduší návrh a implementaci formulářů celé aplikace. V aplikaci je implementováno také několik nových prvků pro formuláře, které Qt framework neobsahuje. Jejich implementací se vytvořily objekty, které jsou v aplikaci vloženy na více místech, což ulehčilo vývoj některých dialogových oken (především pro úpravu služeb) a omezilo tvorbu duplicitního kódu. Samotné prvky se primárně vytvářejí vlastní implementací metod objektu *QWidget*. Jako základní třídu lze použít také prvky, které jsou již implementovány a rozšířit tak funkčnost těchto prvků.



Obrázek 3.5: Nově vytvořený prvek pro měření síly hesla



Obrázek 3.6: Nově vytvořený prvek pro nastavení kvóty

Prvním nově vytvořeným prvkem je prvek pro měření síly hesla. Zobrazuje sílu zadaného hesla podle bodů připsaných na základě splněné podmínky. Například při délce hesla osm znaků a více je přičteno deset bodů a pokud obsahuje čísla i písmena, je přičteno dalších deset bodů, atd. Obsahuje také tlačítko pro vytvoření náhodného hesla. Samotný prvek je zobrazen na obrázku 3.5. Tento prvek je využitý ve všech dialogových oknech při zadávání hesla.

Dalším vytvořeným prvkem je ovládání kvóty účtu. Obsahuje spojení několika prvků pro nastavení velikosti (jezdec a textové pole) a vzájemně je propojuje. Jedná se o nejjednodušší vytvořený prvek a lze jej nalézt na obrázku 3.6.

Je také vytvořen prvek pro vytváření reálného uživatelského jména, který rozšiřuje standardní textové pole. Při zadání textu v něm proběhne posloupnost příkazů, které odstraní všechny znaky kromě písmen a číslic. Tento prvek je využitý vždy při práci s názvy účtů služeb.

Posledním prvkem je prvek pro tvorbu emailových adres, spolupracuje se dvěma vstupními proměnnými a to s názvem a doménou. Pomocí nich na výstupu skládá a zobrazuje validní e-mailovou adresu.

### 3.5.1 Určení použití

Aplikace je určena zatím pouze pro správce serverů, protože neobsahuje žádný uživatelský systém, který by zprostředkoval přihlášení do systému a ani žádný systém přidělování práv, podle kterého by aplikace určovala, k čemu bude moci přihlášený uživatel přistupovat.

K úpravě účtů klienty jsou spíše určeny internetové aplikace, které i klient může spustit v prohlížeči a upravit i potřebné parametry služeb. Příkladem může být aplikace *postfixadmin*, která je instalována spolu s poštovním serverem a umožňuje správu poštovních schránek pro danou doménu resp. domény. Služby převážně spravuje administrátor, protože nezkušený uživatel si neumí upravit nastavení webového serveru, které by navíc mohlo ovlivnit chování celého webového serveru.

### 3.5.2 Koncept ovládání

Aplikace je navržena tak, aby úprava účtů byla co nejjednodušší a logická. Tedy například při otevření seznamu účtů a vybrání požadovaného účtu se uživateli otevře okno s detailem účtu a všemi službami daného účtu. V seznamech se zobrazují nejdůležitější informace o dané službě (jméno, expirace).

#### Práce s MDI

Pro rozložení oken účtů v aplikaci je využito systému Multiple document interface (MDI) – režim s více otevřenými okny zároveň, což umožňuje otevření několika klientských účtů současně a pohodlné přepínání mezi nimi. Zobrazení oken v aplikaci lze nalézt na obrázku 3.4, kde jsou otevřeny detaily některých účtů, seznam účtů a seznam serverů.

Výjimku z tohoto systému uspořádání oken aplikace jako dokumentů tvoří dialogová okna úprav, které jsou zobrazeny jako modální dialogy. Nelze je tedy schovat a při jejich otevření se nelze přepnout na jiné otevřené okno. Mezi tyto dialogová okna patří především úpravy služeb, účtů a serverů.

### 3.5.3 Napojení na datové úložiště

Aplikace je napojena také na datové úložiště. Samotná implementace je provedena pomocí tříd z modulu Qt frameworku *QtSQL*. Při aktualizaci záznamů využívá návrhového vzoru *Active record* a potřebných tříd z modulu *QtSQL*. Je tak umožněna změna datového úložiště na jiný druh databáze (konstrukci SQL dotazů zajišťuje modul *QtSQL*).

Pro připojení k datovému úložišti je nutné v aplikaci nastavit požadované informace, především uživatelské jméno, heslo, název databáze a adresu serveru. Nastavení se provádí přímo v aplikaci v dialogu pro nastavení.

Aplikace využívá také zámku klientských účtů. Je-li při otevírání účet zamknut, aplikace ohlásí, že se jedná o zamknutý účet a zeptá se, má-li pokračovat. Pokud je i přesto účet otevřen, dojde k jeho opětovnému zamknutí. V seznamu účtů jsou zamknuté účty indikovány obrázkem zámku u záznamu. Po ukončení úpravy účtu aplikace automaticky daný účet odemkne.

### 3.5.4 Implementace komunikačního protokolu

Klientská aplikace stejně jako serverová obsahuje stejnou implementaci protokolu popsaného výše v kapitole 3.3. Opět je protokol přiložen jako sdílená knihovna.

Aplikace má odděleny části pro zpracování záznamů z datového úložiště a pro odesílání příkazů na server. Odesílání dat provádí speciální objekt aktualizace serveru (např. *ServerDBUpdate*). Teprve tento objekt je zodpovědný za přenos dat pomocí implementovaného protokolu. Vytvořenému objektu pro aktualizaci se předají potřebné parametry (jméno, kvóta, server) a podle nich objekt při započetí aktualizace určí, jaká data se budou odesílat na server. Pokud přenos selže, je na výstupu metody pro kontrolu přenosu vráceno *false* a metodou *getErrorMessage* vrácena chybová hláška ze serveru.

Při odesílání dat na server objekt sám vytvoří objekt indikátoru průběhu přenosu a aktualizuje jej. Po ukončení přenosu je indikátor automaticky uzavřen.

### 3.5.5 Správa serverů

Pomocí klientské aplikace je možné pracovat se všemi servery. Seznam těchto serverů je uložen v datovém úložišti, ke kterému aplikace přistupuje. V aplikaci lze k seznamu přistupovat pomocí hlavní nabídky *Servery* ⇒ *Seznam serverů* nebo pomocí ikony v panelu nástrojů.

Po vybrání je zobrazeno okno se seznamem a s nástroji pro práci s jednotlivými servery. Celé okno se seznamem lze najít na obrázku 3.7.

Adresa	Název	Typ	Komentář
localhost	local	Webový server	Testovací server
192.168.1.102	Paleček	Webový server	Network testing
192.168.1.103	dpl_test_www	Webový server	
karel.cube-st...	karel	Webový server	Hlavní server
192.168.1.100	eleanor	Webový server	eleanor devel
192.168.1.103	dpl_test_db	Databázový se...	
192.168.1.106	eleanor db	Databázový se...	Eleanor devel
192.168.1.103	dpl_test_mail	Poštovní server	Testing dpl doma

Obrázek 3.7: Okno pro práci se servery

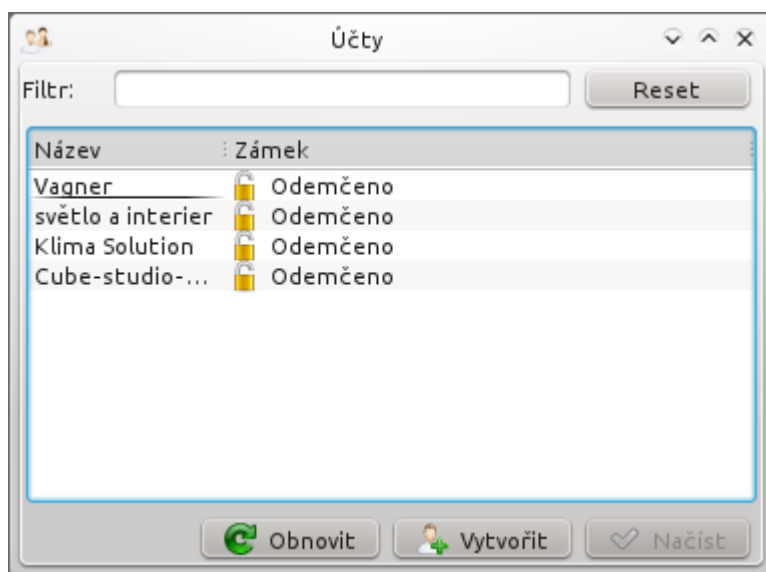
Při otevření vybraného serveru nebo při vytvoření nového serveru je otevřeno okno s parametry serveru. K úspěšnému vytvoření záznamu o serveru je nutné zadat pouze dvě položky a to jméno serveru, tj. jméno, pod kterým bude server pojmenován a adresu serveru. Adresa může být zadána ve formátu IP adresy nebo doménového jména (systém mezi nimi nerozlišuje). Po úspěšném uložení záznamu o serveru lze již se serverem pracovat ve všech účtech.

### 3.5.6 Správa klientských účtů

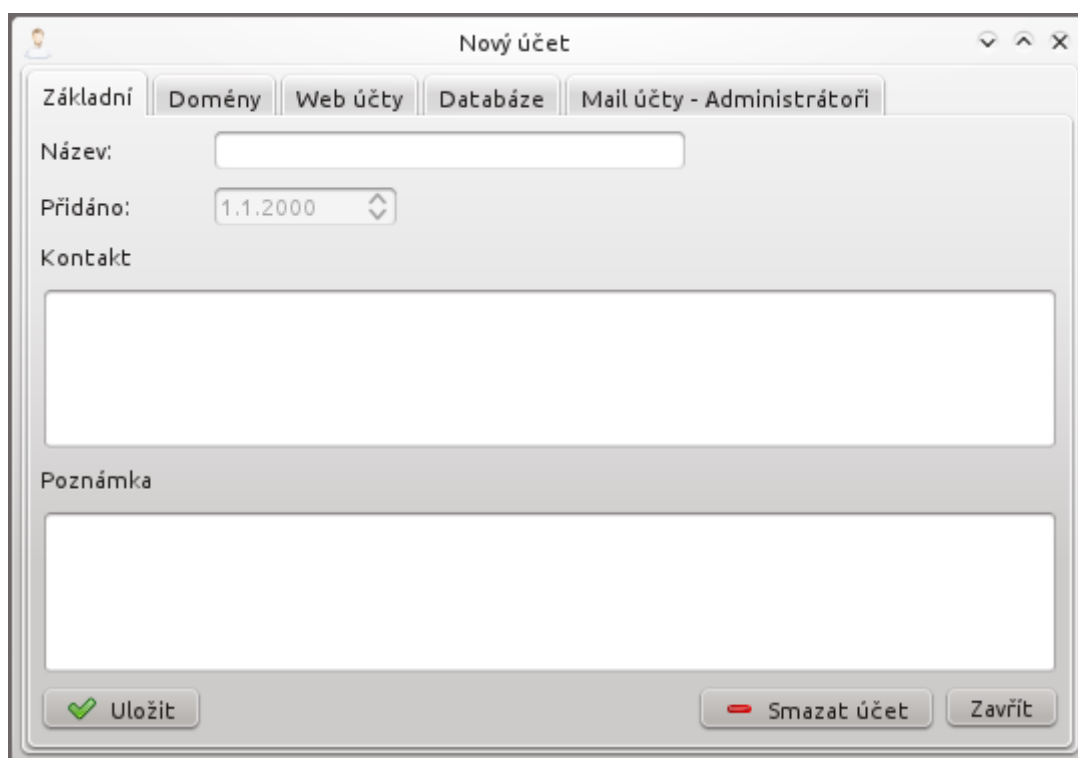
Správa účtů je rozdělena do dvou typů modálních oken. První typ slouží k zobrazení seznamu všech účtů v systému. Pomocí okna prvního typu uživatel vybírá, který účet chce otevřít, popřípadě jestli chce vytvořit nový účet. Náhled okna se seznamem je zobrazen na obrázku 3.8. Z obrázku je také patrné, který účet je odemčen, resp. zamčen. Podle zámku aplikace pozná, že se již s vybraným účtem pracuje. Okno obsahuje také jednoduchý filtr pro vyhledávání v účtech.

Při otevření nebo vytvoření nového účtu je již zobrazeno hlavní okno pro vybraný účet. Náhled okna je zobrazen na obrázku 3.9. Samotné okno je rozděleno do několika souvisejících částí pomocí záložek, mezi kterými uživatel může přepínat.

V první záložce okna jsou základní informace o účtu. Je zde uvedeno jméno účtu, pod kterým se zobrazuje ve výše zmíněném seznamu účtů. Je to jediný povinný



Obrázek 3.8: Okno se seznamem klientských účtů



Obrázek 3.9: Okno pro práci s klientským účtem

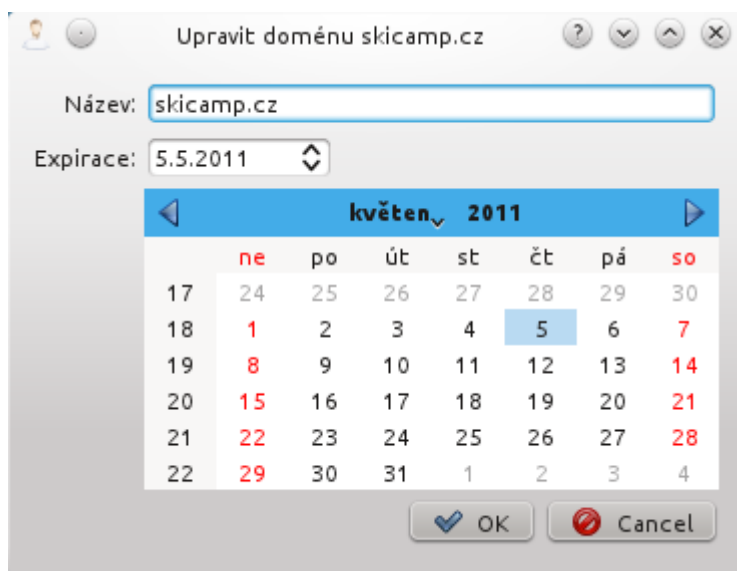
parametr pro založení nového účtu. Ostatní parametry slouží k upřesnění informací o daném účtu. V dalších verzích tohoto systému by měly být doplňující informace účtu využity k dalším účelům, především k tvorbě faktur na základě kontaktních

informací, odesílání e-mailových zpráv o založení, změně nebo zrušení účtu.

Ostatní záložky jsou již určeny ke správě jednotlivých služeb vybraného účtu. Po přepnutí na danou záložku je zobrazen seznam účtů dané služby. U každého seznamu jsou k dispozici i nástroje pro práci s účty služeb.

## Správa domén

Záložka s názvem *Domény* obsahuje všechny domény přiřazené k vybranému klient-skému účtu. Je zde uveden především její název a doba vypršení platnosti domény. Pomocí nástrojů umístěných nad seznamem lze s doménami pracovat (přidání, změna a mazání domén). Dialogové okno pro úpravu domény je zobrazeno na obrázku 3.10.



Obrázek 3.10: Okno pro úpravu domény

Domény ze seznamu se podle potřeby automaticky načítají do dialogů, kde jsou propojeny s daným účtem (např. přiřazení domény k webovému účtu).

## Správa Webových účtů

Záložka *Web účty* je určena ke správě webových účtů. Její rozložení je stejné jako u záložky pro správu domén. Seznam účtů ale obsahuje více informací, především

název účtu, (pomocí tohoto názvu se uživatel přihlašuje protokolem FTP), server kde je účet vytvořen, dobu vypršení, status účtu, přiřazenou doménu a jiné.

Při zakládání nového webového účtu nebo úpravě existujícího účtu je otevřeno okno pro úpravu, zobrazené na obrázku 3.11. Okno obsahuje velké množství voleb pro kompletní nastavení účtu. Pomocí okna lze také u vybraného účtu upravovat samotnou konfiguraci webového serveru. Jedná se o konfigurační soubor popsany v serverové modulu pro práci s webovým serverem v kapitole 3.4.5. Uživatel si může načíst také vlastní nastavení z lokálního disku.

Úprava Webového účtu domaincz

Název účtu: domaincz

Skutečný název: domaincz

Heslo:  Generovat

Síla 0%

Server: dpl\_test\_www

Doména: cube-studio.cz

Disková quota: 1031 MB

Obsazeno: 0 MB z 1031 MB Načíst

Nastavení web serveru Načíst šablonu

Použít výchozí nastavení serveru Načíst nastavení

Proměnné pro přepis v konfiguraci (stačí vybrat, na serveru se přepíše):

- Název účtu
- Název serveru (např. domena.com)
- Aliasy serveru (např. www.domena.com net.domena.com)
- Doplň adresář k www

Vytvořeno: 18.4.2011

Expirace: 18.4.2012

Status: Údržba OK Cancel

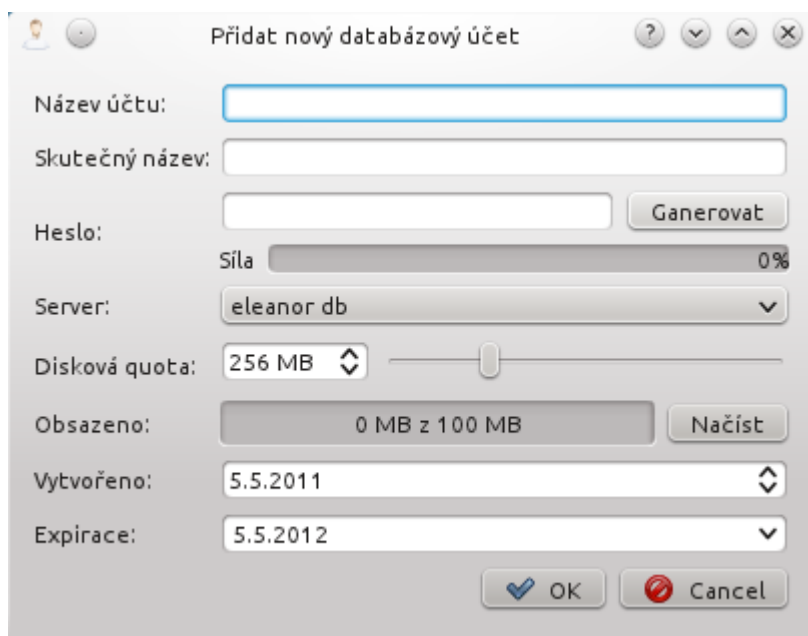
Obrázek 3.11: Okno pro přidání webového účtu

V okně pro úpravu resp. přidání záznamu jsou použity nově vytvořené prvky z kapitoly 3.5 a slouží k jednodušší úpravě celého formuláře.

## Správa Databází

Záložka *Databáze* v detailu účtu je opět téměř shodná se záložkou *Domény*. Obsahuje navíc jen některé další údaje o databázi přiřazené k účtu, např. údaje o serveru, na kterém je databáze uložena.

Dialogové okno pro přidání, resp. úpravu databázového účtu, je zobrazeno na obrázku 3.12. Okno opět využívá některé nově vytvořené formulářové prvky (viz kapitola 3.5). Ve formuláři jsou pak uvedeny všechny důležité údaje pro vytvoření databáze. Při potvrzení formuláře aplikace kontroluje všechny zadané hodnoty a pokud je některá chybně zadána, uživatel je na tuto skutečnost upozorněn chybovou hláškou.

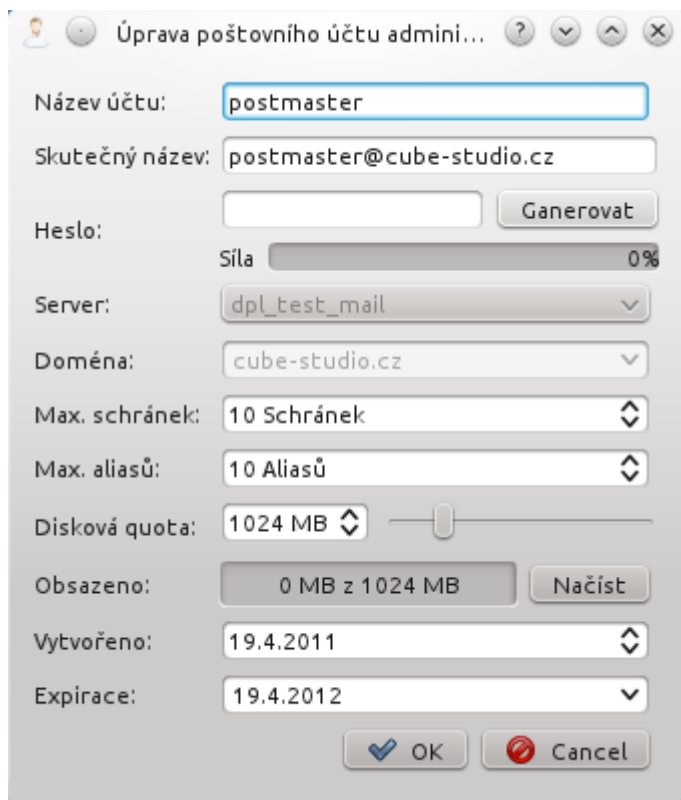


Obrázek 3.12: Okno pro přidání databázového účtu

## Správa Mail účtů

V implementované aplikaci je vytvořena pouze záložka *Mail účty - Administrátoři* pro správu administrátorských účtů k e-mailovým schránkám, správa samotných e-mailových účtů implementována není, protože administrátoři domén si mohou schránky již spravovat přes rozhraní v internetovém prohlížeči. Rozložení prvků v záložce je opět shodné s předešlými záložkami.

Při přidávání, resp. úpravě administrátorského účtu e-mailů je otevřeno dialogové okno zobrazené na obrázku 3.13. Pomocí formuláře lze nastavit všechny potřebné parametry účtu a jsou v něm opět použity některé nově vytvořené formulářové prvky zmíněné výše.



Úprava poštovního účtu administrátorského

Název účtu:

Skutečný název:

Heslo:   Síla  0%

Server:  ▼

Doména:  ▼

Max. schránek:  ▼

Max. aliasů:  ▼

Disková quota:

Obsazeno:

Vytvořeno:  ▼

Expirace:  ▼

Obrázek 3.13: Okno pro úpravu administrátorského účtu e-mailů

### 3.5.7 Rozšíření funkčnosti aplikace

Díky využití rozložení oken aplikace typu MDI a zapouzdření některých částí, je možné lehce implementovat další prvky aplikace, které zjednoduší její užívání nebo rozšíří její funkčnost.

#### Možná implementace průvodců

V implementované aplikaci lze také vytvořit průvodce pro přidání účtu, tak aby zjednodušil celý proces vytváření klientských účtů. Díky použití objektů pro *update* serveru by tak byla implementace velmi jednoduchá, protože všechny potřebné

funkce již systém obsahuje. K samotné implementaci v aplikaci stačí využít třídu *QWizard*, která je součástí Qt frameworku a slouží k tvorbě průvodců.

### **Kontrola stavu serverů**

Aplikace obsahuje již vytvořené, ale nevyužité okno, pomocí kterého lze přidat podporu kontroly stavu jednotlivých serverů v celém systému. Toto okno lze přichytit k hlavnímu oknu aplikace tak, aby bylo vždy viditelné. Samotná implementace kontroly by mohla být provedena pomocí vláken tak, aby neovlivňovala hlavní prostředí aplikace a samotnou komunikaci. Samotná kontrola by pak probíhala buď pomocí dotazů *ping* nebo kontrolou připojení na předem zadaný port. V okně by byl server při selhání označen jako nedostupný.

### **Statistiky účtů**

Pomocí známého seznamu serverů a účtů lze také vytvářet statistiky využití jednotlivých serverů. Systém by pak mohl zobrazovat servery, na kterých je největší množství účtů, či servery na kterých může dojít místo na disku. V aplikaci zatím žádné okno ani část, která by tuto funkčnost přidávala implementována není, ale její přidání by díky práci s modely (popisy tabulek v datovém úložišti) nemělo být obtížné. Jedinou překážkou by tak zůstávala nutnost upravit komunikační protokol a přidat do něj další příkazy, např. pro zjištění volného diskového prostoru.

## 4 ZÁVĚR

Aplikace typu klient/server pro správu hostingových služeb přináší nový pohled na samotnou správu vzdálených systémů. Její rozdělení umožňuje využití jednoho klienta pro správu několika serverů, resp. práci několika klientů na jednom serveru současně.

Vytváří tak jednotné rozhraní pro všechny poskytované služby (webhosting, databáze, e-mailly) a tvorbu účtů k nim, nezávisle na druhu služby spuštěné na serveru. Výhodou je možnost implementace migrace klientských účtů i s daty mezi servery a rozložení zátěže při nadměrném vytižení služeb u klientského účtu.

Pro pochopení problematiky byl uveden základní princip hostingových služeb. Uveden byl také popis jednotlivých služeb spuštěných na serveru spolu s jejich základním nastavením, vysvětlen princip uložení jednotlivých uživatelských účtů na serveru a také účtů pro elektronickou poštu v databázi. Popsány byly pouze služby, které navrhovaný systém aplikací bude obsluhovat, a to v jejich základním nastavení.

Samotné navržení systému správy pomocí aplikace klient/server a její implementace probíhala ve dvou částech. V první části jsou uvedeny všechny potřebné požadavky na služby a aplikace, a navrženo rozdělení struktury systému do čtyř na sobě závislých částí.

V druhé části práce již probíhala implementace jednotlivých navržených částí systému. Uveden je především popis jejich principu, nastavení a samotná implementace pomocí frameworku Qt.

První implementovanou částí bylo datové úložiště klientských účtů, bylo navrženo a je implementováno jako sada tabulek v databázi, ke které přistupuje pouze klientská aplikace.

Další důležitou částí byl návrh a implementace komunikačního protokolu. Protokol byl implementován jako sdílená knihovna, která je součástí jak serverové, tak klientské aplikace. Vyvíjená verze zatím neobsahuje šifrování komunikace z důvodu testování. Tento nedostatek lze, jak bylo zmíněno v kapitole 3.3.3, lehce odstranit před vydáním verze určené pro reálné nasazení.

U implementace serverové části je popsán systém napojení na komunikační protokol a především systém zásuvných modulů, pomocí kterých lze aplikaci přenastavit na požadovanou službu, či omezit její funkčnost, bez nutnosti jejího opětovného překladu.

V poslední části je uvedena implementace a obsluha klientské části aplikace. V této části je především zdůrazněno využití frameworku Qt, jeho modulu pro tvorbu grafického uživatelského prostředí a modulu pro přístup k databázi a tedy i k datovému úložišti. Na konci kapitoly jsou uvedeny některé možnosti rozšíření aplikace.

Současný stav aplikace zatím neobsahuje pouze kompletní implementaci e-mailových služeb, ale pouze jejich hlavní části, sloužící ke správě administrátorských účtů e-mailů. Samotná správa schránek nebyla implementována z důvodů jejího nevyužití, jak bylo uvedeno v kapitole 3.5.6.

V celé práci byla popsána kompletní implementace zadané aplikace klient/server pro správu hostingových služeb a uvedeny všechny kroky vedoucí ke správnému návrhu celého systému a samotné implementace klientské, resp. serverové aplikace za použití frameworku Qt. Navržená aplikace splňuje všechny požadavky zadání práce pro správu na různých úrovních. Systém správy po přidání zabezpečení je připraven pro nasazení na produkční servery a správu klientských účtů hostingových služeb.

## REFERENCE

- [1] Network Working Group *DOMAIN NAMES - CONCEPTS AND FACILITIES* [online]. 1987 [cit. 5.11.2009].  
Dostupné z URL: <<http://www.ietf.org/rfc/rfc1034.txt>>.
- [2] Network Working Group *Hypertext Transfer Protocol - HTTP/1.1* [online], poslední revize červen 1999 [cit. 10.11.2009].  
Dostupné z URL: <<http://www.w3c.org/Protocols/rfc2616/rfc2616.html>>.
- [3] Network Working Group *The TLS Protocol Version 1.0* [online], poslední revize leden 1999 [cit. 10.11.2009].  
Dostupné z URL: <<http://www.ietf.org/rfc/rfc1034.txt>>.
- [4] Kyle D. Dent *POSTFIX kompletní průvodce*. O'REILLY, GRADA, 2005. 252 s. ISBN: 8024710293
- [5] Kim Nielsen *User/Group Limitations* [online], poslední revize 8.6.2007 [cit. 12.11.2009]. Dostupné z URL:  
<<http://www.gentoo.org/doc/en/security/security-handbook.xml?part=1&chap=5>>.
- [6] Oracle corporation *MySQL 5.1 Reference Manual* [online], 10.12.2010, [cit. 14.12.2010]. Dostupné z URL:  
<<http://dev.mysql.com/doc/>>.
- [7] The Apache Software Foundation *Apache HTTP Server Version 2.2 Documentation* [online]. 2009 [cit. 21.11.2009].  
Dostupné z URL: <<http://httpd.apache.org/docs/2.2/>>.
- [8] Boleslav Bobčík *PAM - správa autentizačních mechanismů* [online], poslední revize 19. 9. 2000 [cit. 21.11.2009]. Dostupné z URL:  
<<http://www.root.cz/clanky/pam-sprava-autentizacnich-mechanismu/>>.
- [9] Gentoo Wiki *Apache2* [online], poslední revize 2. 11. 2009, [cit. 25.11.2009].  
Dostupné z URL: <<http://en.gentoo-wiki.com/wiki/Apache2>>.

- [10] Gentoo Wiki *Apache2/SSL Certificates* [online], poslední revize 3. 8. 2009, [cit. 25.11.2009]. Dostupné z URL:  
<[http://en.gentoo-wiki.com/wiki/Apache2/SSL\\_Certificates](http://en.gentoo-wiki.com/wiki/Apache2/SSL_Certificates)>.
- [11] Wikipedia *Transport Layer Security* [online], poslední revize 11. 11. 2009, [cit. 25.11.2009].  
Dostupné z URL: <[http://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security)>.
- [12] Gentoo Wiki *ProFTPD* [online], poslední revize 27. 8. 2009, [cit. 1.12.2009].  
Dostupné z URL: <<http://en.gentoo-wiki.com/wiki/ProFTPD>>.
- [13] Mark Lowes *ProFTPD A User's Guide* [online], 2001, [cit. 1.12.2009]. Dostupné z URL:  
<<http://www.proftpd.org/localsite/Userguide/linked/userguide.html>>.
- [14] Nokia corp. *Qt Documentation* [online], 2010, [cit. 22.4.2011]. Dostupné z URL:  
<<http://doc.qt.nokia.com/4.7/index.html>>.
- [15] Remigiusz Modrzejewski *MySQL Quota Daemon* [online], 22.11.2009, [cit. 14.3.2011]. Dostupné z URL:  
<<http://lrem.net/software/mysql-quota-daemon.xhtml>>.

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

.NET dotNet

ASCII American Standard Code for Information Interchange

ASP Active Server Pages

BIND Berkeley Internet Name Domain

BSD Berkeley Software Distribution

CGI Common Gateway Interface

CSR Certificate Signing Request

DNS Domain Name System

FTP File Transfer Protocol

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IDE Integrated Development Environment

IIS Internet Information Services

IMAP Internet Message Access Protocol

IP Internet Protocol

LDAP Lightweight Directory Access Protocol

MD5 Message-Digest algorithm

MDA Mail Delivery Agent

MDI Multiple document interface

MSA Mail Submission Agent

MTA Mail Transfer Agent

MVC Model-View-Controller

MyISAM My Indexed Sequential Access Method

PAM Pluggable Authentication Modules

PHP Hypertext PreProcessor

POP3 Post Office Protocol version 3

RADIUS Remote Authentication Dial In User Service

RFC Request For Comments

SFTP Secure File Transfer Protocol

SMTP Simple Mail Transfer Protocol

SNI Server Name Indication

SQL Structured Query Language – strukturovaný dotazovací jazyk databáží

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

VHCS Virtual Hosting Control System

XML eXtensible Markup Language

# SEZNAM PŘÍLOH

<b>A</b>	<b>Skript pro vytváření účtu</b>	<b>73</b>
<b>B</b>	<b>Příklad souboru projektu pro Qt Framework</b>	<b>77</b>
<b>C</b>	<b>Struktura datového úložiště</b>	<b>78</b>
<b>D</b>	<b>Příkazy komunikačního protokolu</b>	<b>79</b>
D.1	Základní příkazy . . . . .	79
D.2	Příkazy pro práci s webovými účty . . . . .	80
D.3	Příkazy pro práci s databázovými účty . . . . .	81
D.4	Příkazy pro práci s e-mailovými účty . . . . .	82
D.4.1	Příkazy pro správu administrátorů schránek . . . . .	82
D.4.2	Příkazy pro správu schránek . . . . .	83
<b>E</b>	<b>Soubory s nastavením aplikace</b>	<b>84</b>
E.1	Soubor s hlavním nastavením serverové aplikace . . . . .	84
E.2	Nastavení modul <i>storageunixhomedir</i> . . . . .	84
E.3	Nastavení modul <i>webserverapache</i> . . . . .	85
E.3.1	Příklad šablony virtuálního hosta . . . . .	86
E.4	Nastavení modul <i>dbservermysql</i> . . . . .	86
E.5	Nastavení modul <i>dbservermysqlquota</i> . . . . .	87
E.6	Nastavení modul <i>mailserverpostfixmysql</i> . . . . .	87
<b>F</b>	<b>Konfigurační soubory služeb</b>	<b>88</b>
F.1	konfigurace virtuálního hosta webového serveru Apache . . . . .	88

## A SKRIPT PRO VYTVÁŘENÍ ÚČTU

```
#!/bin/bash
# webhosting - Copyright (C) 2008 Jakub Matas <jakubmatas@gmail.com>
#promene
phpmyadmin_ver="3.2.2.1";
db_root_user="xxx";
db_root_pass="xxx";
webhostname=$1;
webhostpass=$2;
webmail_clients="[1]squirrelmail [2]roundcube";

if [ $1 ]; then
    echo "Webhosting: ${webhostname}";
    webhostdb='echo "${webhostname}" | sed 's/[.-_]//g' ';
    if [ ${#webhostdb} -gt 15 ]; then
        echo "Zkracuji DB name";
        webhostdb=${webhostdb:0:15}
    fi
    echo "Webhosting DB name: ${webhost_db}";
else
    echo "chyba v zadani jmena hostingu nebo hesla";
    exit 1;
fi

# ===== uživatel
adduser -m -g users -G ftp,mail -c "uživatel webhostingu"
-s /bin/false ${webhostname};
'echo ${webhostname}:${webhostpass} | chpasswd';
# ===== domovske adresář a složky
#vytvoreni adresare pro hosting
mkdir -p /home/${webhostname}/{htdocs,cgi-bin,logs,error,icons};
```

```

touch /home/${webhostname}/htdocs/favicon.ico;
cp /var/www/localhost/htdocs/index.html
/home/${webhostname}/htdocs/index.html;
chown -R ${webhostname}:users /home/${webhostname}/.;
chmod -R 755 /home/${webhostname}/.;
chown -R apache:apache /home/${webhostname}/logs
chmod -R 755 /home/${webhostname}/logs
#vytvoreni zastupce v adresari s www
ln -sf /home/${webhostname}/ /var/www/${webhostname};
# ===== apache conf
echo -e "\
<VirtualHost *:80>\n
    ServerName www.${webhostname}\n
    ServerAlias ${webhostname} *.${webhostname}\n
    DocumentRoot /var/www/${webhostname}/htdocs/\n
    <Directory \"/var/www/${webhostname}/htdocs\">\n
        AllowOverride FileInfo Options AuthConfig\n
        Allow from all\n
    </Directory>\n
    LogFormat \"%h %l %u %t \\\"%r\\\"\" %>s %b\" common\n
    CustomLog /var/www/${webhostname}/logs/access_log common\n
    ErrorLog /var/www/${webhostname}/logs/error_log\n
    CustomLog /var/www/${webhostname}/logs/referer_log \
    \"%{Referer}i -> %U\"\n
    CustomLog /var/www/${webhostname}/logs/agent_log \
    \"%{User-agent}i\"\n
</VirtualHost>\n" >> /etc/apache2/vhosts.d/01_${webhostname}.conf;
#reload apache
/etc/init.d/apache2 reload;
# ===== databáze
mysql -u root --password=${db_root_pass} -e "CREATE USER

```

```

'${webhostdb}'@'localhost' IDENTIFIED BY '${webhostpass}';
mysql -u root --password=${db_root_pass} -e "GRANT USAGE
ON * . * TO '${webhostdb}'@'localhost' IDENTIFIED BY '${webhostpass}'
WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 10";
#vytvoreni databaze
mysql -u root --password=${db_root_pass} -e "CREATE DATABASE
IF NOT EXISTS ${webhostdb}";
mysql -u root --password=${db_root_pass} -e "GRANT ALL PRIVILEGES
ON ${webhostdb} . * TO '${webhostdb}'@'localhost'";

# ===== subaplikace
# phpMyAdmin
echo -n "Instalovat phpmyadmin?[a/n]";
read install_phpmyadmin;
if [ ${install_phpmyadmin} == a ]; then
    ln -sf /var/www/phpmyadmin/htdocs
        /home/${webhostname}/htdocs/phpmyadmin;
fi;

# webmail
echo -n "Instalova webmail klienta?";
read install_mails;
if [ ${install_mails} == a ]; then
    crypt_pass='./md5crypt.php ${webhostpass}';
    echo -n "Kterého webmail klienta instalovat?";
    for client in ${webmail_clients}; do
        echo ${client};
    done;
    read install_client;
    case ${install_client} in

```

```

1)ln -sf /var/www/squirrelmail/htdocs
    /home/${webhostname}/htdocs/webmail;
;;
*)ln -sf /var/www/roundcube/htdocs
    /home/${webhostname}/htdocs/webmail
esac
fi;

# postfixadmin
echo -n "Instalovat postfixadmin?";
read install_postfixadmin;
if [ ${install_postfixadmin} == a ]; then
    ln -sf /var/www/postfixadmin/htdocs
        /home/${webhostname}/htdocs/mailadmin;
fi;

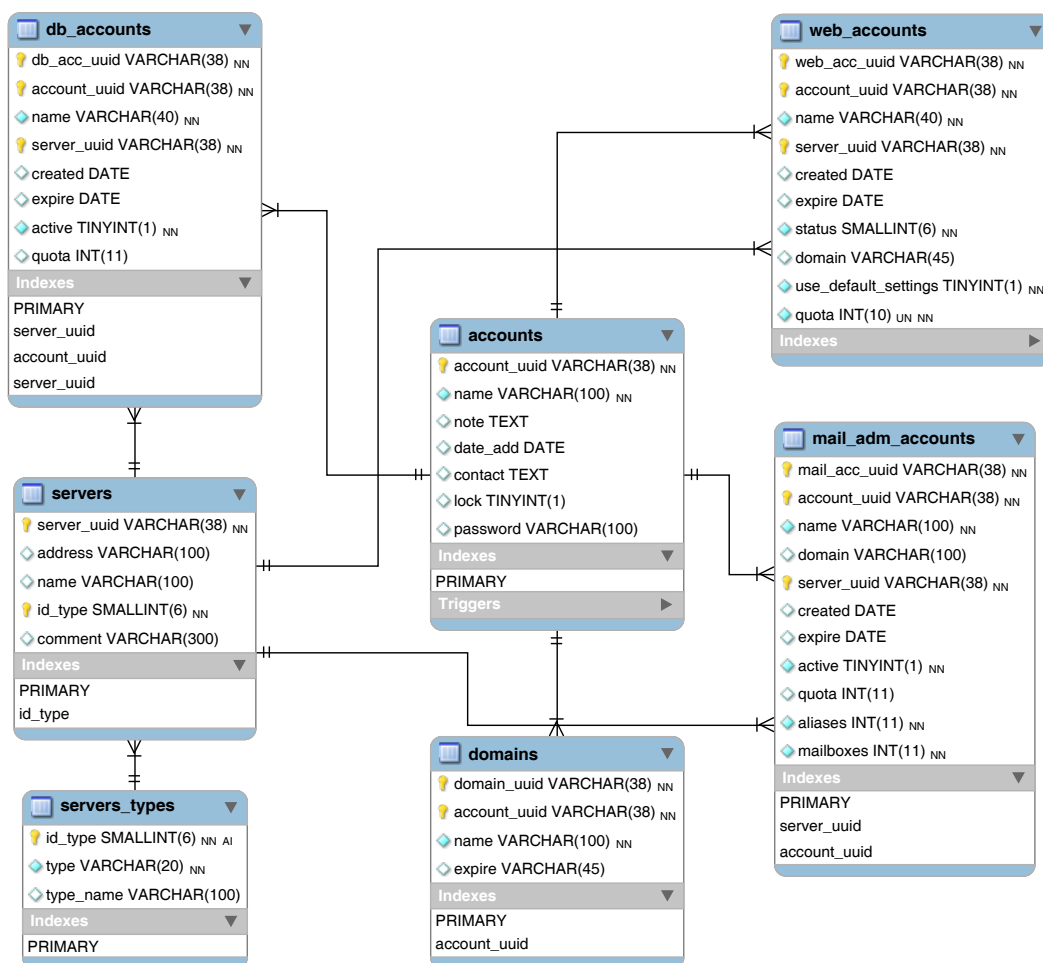
# ===== konec
echo "NAME: "\${webhostname};
echo "PASS: "\${webhostpass};
echo "DB_NAME: "\${webhostdb};
history -c;
exit 0;

```

## B PŘÍKLAD SOUBORU PROJEKTU PRO QT FRAMEWORK

```
# -----  
# Project created by QtCreator 2010-04-24T11:33:28  
# -----  
QT += network \  
    sql  
QT -= gui  
TARGET = ../app/whrc_server  
LIBS += -L/lib/ -lwhrccp  
INCLUDEPATH += ../lib/  
  
CONFIG += console  
CONFIG -= app_bundle  
TEMPLATE = app  
SOURCES += main.cpp \  
    server.cpp  
HEADERS += server.h \  
    webserverinterface.h \  
    dbserverinterface.h \  
    accstorageinterface.h \  
    dbquotainteface.h \  
    mailserverinterface.h
```

## C STRUKTURA DATOVÉHO ÚLOŽIŠTĚ



Obrázek C.1: Struktura tabulek účtů v databázi

# D PŘÍKAZY KOMUNIKAČNÍHO PROTOKOLU

Parametry v hranatých závorkách nejsou povinné.

## D.1 Základní příkazy

Příkaz	Popis <i>Parametry</i>
Ok	kladná odpověď [ <i>výsledek dotazu</i> ], [ <i>výsledek dotazu</i> ], ...
Error	chybná odpověď [ <i>text chybové zprávy</i> ]
Check	kontrola serveru nebo protokolu - není využito
Incorrect	špatně přenesený nebo nepodporovaný příkaz
Protocol	přenesení kontroly protokolu - není využito <i>verze protokolu</i>

Tabulka D.1: Základní příkazy protokolu

## D.2 Příkazy pro práci s webovými účty

<b>Příkaz</b>	<b>Popis</b> <i>Parametry</i>
WebAccIsFree	kontrola existence účtu <i>název účtu</i>
WebAccSetStatus	nastavení statusu účtu <i>název účtu, status kód</i>
WebAccCreate	vytvoření nového účtu <i>název účtu</i>
WebAccSetPassword	nastavení hesla účtu <i>název účtu, heslo</i>
WebAccSetDomain	nastavení domény účtu - není využito <i>název účtu, doména, [doména], [doména], ...</i>
WebAccSetQuota	nastavení kvóty k zadanému účtu <i>název účtu, kvóta v MB</i>
WebAccGetQuota	vrací velikost kvóty k zadanému účtu <i>název účtu</i>
WebAccGetSize	vrací velikost zadaného účtu v MB <i>název účtu</i>
WebAccRemove	odstraní zadaný účet <i>název účtu</i>
WebAccRename	přejmenuje zadaný účet <i>název účtu, nový název účtu</i>
WebAccGetSettings	vrací nastavení serveru pro daný účet <i>název účtu</i>
WebAccSetSettings	nastaví server pro zadaný účet <i>název účtu, nastavení</i>

Tabulka D.2: Příkazy protokolu pro práci s webovými účty

## D.3 Příkazy pro práci s databázovými účty

<b>Příkaz</b>	<b>Popis</b> <i>Parametry</i>
DbAccIsFree	kontrola existence účtu <i>název účtu</i>
DbAccCreate	vytvoření nového účtu <i>název účtu</i>
DbAccSetPassword	nastavení hesla k zadanému účtu <i>název účtu, heslo</i>
DbAccSetQuota	nastavení kvóty pro zadaný účet <i>název účtu, kvóta v MB</i>
DbAccGetQuota	vrací velikost kvóty zadaného účtu <i>název účtu</i>
DbAccGetSize	vrací velikost zadaného účtu <i>název účtu</i>
DbAccRemove	odstraní zadaný účet <i>název účtu</i>
DbAccRename	přejmenuje zadaný účet <i>název účtu, nový název</i>

Tabulka D.3: Příkazy protokolu pro databázové účty

## D.4 Příkazy pro práci s e-mailovými účty

### D.4.1 Příkazy pro správu administrátorů schránek

<b>Příkaz</b>	<b>Popis</b> <i>Parametry</i>
MailAdmAccIsFree	kontrola existence administrátora domény <i>název, doména</i>
MailAdmAccCreate	vytvoření účtu administrátora domény <i>název, doména</i>
MailAdmAccSetQuota	nastaví kvótu pro zadanou doménu <i>doména, kvóta v MB, max počet schránek, max. počet aliasů</i>
MailAdmAccGetQuota	vrátí kvóty pro zadanou doménu <i>doména</i>
MailAdmAccGetSize	vrací velikost všech schránek z dané domény <i>doména</i>
MailAdmAccSetPassword	nastaví nové heslo pro admin. účet <i>název účtu, doména, heslo</i>
MailAdmAccRemove	odstraní předaný admin. účet <i>název účtu, doména</i>
MailAdmAccRename	přejmenuje daný admin. účet <i>název účtu, nový název, doména</i>

Tabulka D.4: Příkazy protokolu pro účty administrátoru poštovních schránek

## D.4.2 Příkazy pro správu schránek

Příkazy pro správu poštovních schránek, které jsou v protokolu sice navrženy, ale zatím je klientská ani serverová část nevyužívá.

<b>Příkaz</b>	<b>Popis</b> <i>Parametry</i>
MailAccMailIsFree	kontroluje jestli daná schránka je volná <i>název, doména</i>
MailAccMailList	vrací seznam schránek ze zadané domény <i>doména</i>
MailAccMailCreate	vytvoří novou schránku v zadané doméně <i>název, doména</i>
MailAccMailSetPassword	nastaví heslo pro danou schránku v dané doméně <i>název, doména, heslo</i>
MailAccMailRemove	odstraní zadanou schránku z dané domény <i>název, doména</i>
MailAccMailRename	přejmenuje zadanou schránku z dané domén <i>název, nový název, doména</i>

Tabulka D.5: Příkazy protokolu pro správu poštovních schránek

## E SOUBORY S NASTAVENÍM APLIKACE

### E.1 Soubor s hlavním nastavením serverové aplikace

Obsah souboru *whrc.ini*.

```
[main]
# communication port
port=2266
# bind address
address=0.0.0.0

# enabled modules for disable set module to text disable
# e.g.: storage=disabled

[modules]
# storage module
storage=unixhomedir
# webserver module
webserver=apache
# db server module
dbserver=mysql
# db quota module
dbquota=mysqlquotadaemon
# mail module
mailserver=postfixmysql
```

### E.2 Nastavení modul *storageunixhomedir*

Obsah souboru *unixhomedir.ini*.

```
# Base settings for module UnixHomeDir
```

```

[user]
# base user group
basegroup=users
# otehr user groups separed by ","
othergroups=
# base user shell
shell=/bin/false
# comment for user
comment="Uzivatel webhostingu"
# post create script
[post]
post-create-script=
# configuration of dirs
[dirs]
# create this dirs in homedir
createdirs=cgi,var

```

### E.3 Nastavení modul *webserverapache*

Obsah souboru *apache.ini*.

```

# apache module config file
[apache]
reload-conf=/etc/init.d/apache2 reload
site-tpl=/etc/whrc/modules/site_tpl.conf

[dirs]
sites-enabled=/etc/apache2/whrc-sites-enabled/
sites=/etc/apache2/whrc-sites/
var-www=/var/www/
log=logs
www=www

```

```
tmp=tmp
```

### E.3.1 Příklad šablony virtuálního hosta

Obsah souboru *site\_tpl.conf*.

```
<VirtualHost *:80>
    ServerName {SERVERNAME}
    ServerAlias {SERVERALIASES}
    DocumentRoot {WWWDIR}/
    <Directory "{WWWDIR}">
        AllowOverride FileInfo Options AuthConfig Indexes
        Allow from all
    </Directory>
    LogFormat "%h %l %u %t \"%r\" %>s %b" common
    CustomLog {LOGSDIR}/www_access.log common
    ErrorLog {LOGSDIR}/www_error.log
    CustomLog {LOGSDIR}/www_referer.log "%{Referer}i -> %U"
    CustomLog {LOGSDIR}/www_agent.log "%{User-agent}i"
</VirtualHost>
```

## E.4 Nastavení modul *dbservermysql*

Obsah souboru *mysql.ini*.

```
[mysql]
host=localhost
user=whrc
password=whrc

[user]
access-host=%
```

## E.5 Nastavení modul *dbservermysqlquota*

Obsah souboru *mysqlquotadaemon.ini*.

```
[mysql]
host=localhost
user=whrc
password=whrc
database=whrc
table=quota
```

## E.6 Nastavení modul *mailserverpostfixmysql*

Obsah souboru *postfixmysql.ini*.

```
[db]
host=localhost
user=whrc
password=whrc
database=postfix

[tables]
admins=admin
domains=domain
domain-admins=domain_admins
mailboxes=mailbox
aliases=alias

[accounts]
# password crypt function. Possible values is: clear, md5, md5-crypt.
password-crypt=md5
maildir=/var/mail/
```

## F KONFIGURAČNÍ SOUBORY SLUŽEB

### F.1 konfigurace virtuálního hosta webového serveru Apache

```
<VirtualHost *:80>
    ServerName www.domena.cz
    ServerAlias domena.cz *.domena.cz
    DocumentRoot /var/www/domena.cz/htdocs/
    <Directory "/var/www/domena.cz/htdocs">
        AllowOverride FileInfo Options AuthConfig Indexes
        Allow from all
    </Directory>
    LogFormat "%h %l %u %t \"%r\" %>s %b" common
    CustomLog /var/www/domena.cz/logs/access.log common
    ErrorLog /var/www/domena.cz/logs/error.log
</VirtualHost>
```

```
<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /etc/ssl/apache2/ssl.crt
    SSLCertificateKeyFile /etc/ssl/apache2/private.key

    ServerName www.domena.cz
    ServerAlias domena.cz *.domena.cz
    SSLOptions StrictRequire
    SSLProtocol all -SSLv2

    DocumentRoot /var/www/domena.cz/htdocs/
    <Directory /var/www/domena.cz/htdocs/>
        SSLRequireSSL
```

```
    Order Deny,Allow
    Allow from All
</Directory>
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /var/www/domena.cz/logs/https_access_log common
ErrorLog /var/www/domena.cz/logs/https_error_log
</VirtualHost>
```