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

BILINGOVÝ SYSTÉM A MONITOROVÁNÍ HOVORŮ PRO PBX ASTERISK

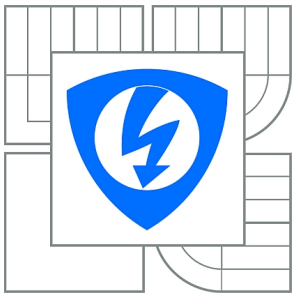
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR DEPIAK
AUTHOR

BRNO 2010

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

BILINGOVÝ SYSTÉM A MONITOROVÁNÍ HOVORŮ PRO PBX ASTERISK

BILLING SYSTEM AND CALL MONITORING FOR PBX ASTERISK

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

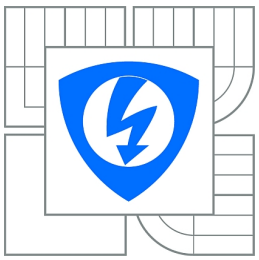
AUTOR PRÁCE — Bc. PETR DEPIAK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ KRAJSA

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Petr Depiak	***ID:***	78034
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Bilingový systém a monitorování hovorů pro PBX Asterisk

POKYNY PRO VYPRACOVÁNÍ:

Podrobně se seznamte s možnostmi multiplatformního programování pro Internet. Ve Vámi vybraném vývojovém nástroji navrhněte a realizujte bilingový systém pro softwarovou ústřednu Asterisk. Analyzujte také možnosti monitorování hovorů. Navrženou aplikaci zabezpečte proti vniknutí neautorizované osoby.

DOPORUČENÁ LITERATURA:

[1] KOSEK, Jiří. PHP :Tvorba interaktivních internetových aplikací. Podrobný průvodce /Praha :Grada Publishing,1998. 1. vyd. 490 s. ISBN 8071693731
[2] NARAMORE, Elizabeth. PHP5, MySQL, Apache: vytváříme webové aplikace / Vyd. 1. Brno : Computer Press, 2006. 813 s. ISBN 80-251-1073-7
[3] KOFLER, Michael; ÖGGL, Berndt .PHP 5 a MySQL 5 : průvodce webového programátora, Vyd. 1. Brno : Computer Press, 2007. 607 s., ISBN: 978-80-251-1813-9
[4] Meggelen, J.V, Smith, J., Madsen, L. Asterisk™ The Future of Telephony. Sevastopol: O'Reilly Media, Inc., 2005. ISBN 0-596-00962-3.

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Ondřej Krajsa

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato diplomová práce je zaměřena na tvorbu účtovacího systému s možností monitorování jednotlivých hovorů pro softwarovou ústřednu Asterisk. Účtování hovorů lze přizpůsobit pomocí skupin jednotlivých pravidel, skládající se z tarifikačních intervalů, číselné předvolby, použitého odchozího svazku a ceny za započtené jednotky. První část práce se zabývá vlastní instalací, konfigurací a přípravou jednotlivých komponent účtovacího systému. Je zde vysvětlena architektura účtovacího systému a osvětlen princip tvorby modelu databáze. Následně je zde rozebrán účel a principy jednotlivých funkcí systému včetně řešení. V poslední části je jednoduchý manuál k ovládání systému pomocí vytvořeného webového rozhranní.

KLÍČOVÁ SLOVA

Asterisk, Apache, MySQL, účtování, tarifikace, tarif, správa

ABSTRACT

This master's thesis is focused on developement of billing system with the options of monitoring individual calls for software exchange Asterisk. Billing of calls is adaptible with the help of group of individual rules, consisting of tariff impulses, numerical prefix, with help of outgoing trunk and cost of the billed unit. The first part of this work is focused on instalation, configuration and preparation of individual components of the billing system. In this work is explained the architecture of the billing system and highlighted the purpose of work of the model database. Next we focused on the purpose and the principal system invidual function of the system including solution. At last there is a simple manual to operate the system with the help of created web interface.

KEYWORDS

Asterisk, Apache, MySQL, billing, tariffication, tariff, administration

DEPIAK, Petr *Bilingový systém a monitorování hovorů pro PBX Asterisk*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2010. 63 s. Vedoucí práce byl Ing. Ondřej Krajsa,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Bilingový systém a monitorování hovorů pro PBX Asterisk“ 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.

Brno ....

(podpis autora)

Poděkování

Na tomto místě bych rád poděkoval vedoucímu diplomové práce Ing. Onřeji Krajsovi za zájem, připomínky a obětovaný čas, který věnoval této práci.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato práce se zabývá návrhem a realizací systému pro účtování a monitorování hovorů PBX Asterisk. Je zde zahrnuto seznámení s Asteriskem a konfigurací nutnou ke spolupráci s ostatními systémy, které společně tvoří funkční celek účtovacího systému.

Součástí je relační databáze MySQL. Databáze zde zastupuje nejen roli pro uchování dat a relací mezi nimi, ale i uchování vzájemné integrity. V databázi jsou uchovávány informace o uživatelských účtech, tarifech, hovorových datech, až po konfigurační data Asterisku. Značná část práce rozebírá kompletní návrh databázového modelu a metodu zabezpečení webové aplikace proti neautorizovanému přístupu do systému. Součásti návrhu je datové, procesní modelování diagramů entit a relací mezi nimi.

Jako prostředník pro interakci účtovacího systému s uživatelem slouží HTTP server Apache. Do něj jsou implementovány moduly pro podporu jazyka PHP a aplikační skripty. Společně generují HTML kód, který vizualizuje aplikační rozhraní.

PHP skripty budou součástí nejen webového aplikačního rozhraní, ale i součástí PBX Asterisk. K těmto skriptům bude přistupováno skrze rozhraní AGI vlastní ústředny. Tímto rozhraním lze přesměrovat dialplán Asterisku do externí aplikace, resp. skriptu. Ten bude rozhodovat o průběhu sestavení spojení na základě dat poskytnutých databází MySQL.

Celý účtovací systém je postaven na platformě linuxu Centos 5, který je nejčastěji používán jako vývojový OS. Verze Asterisku byla použita stabilní 1.4.29.1 .

1 ŘEŠENÍ STUDENTSKÉ PRÁCE

1.1 Úvod

Jak již bylo psáno dříve, jedná se o komplexní systém postavený z několika dílčích částí. Zejména se jedná o:

- OS Centos 5;
- PBX Asterisk + doplňkové moduly;
- databázový server MySQL + vytvořená databáze;
- HTTP server Apache + aplikace tvořena PHP skripty;
- PHP skripty pro AGI Asterisku.

1.2 PBX Asterisk

1.2.1 Co je Asterisk

„*Asterisk je software, který přemění běžný počítač na komunikační server.*“[1][8] Ačkoliv je tento výrok nadnesený, vystihuje účel Asterisku. Platí zde ovšem určité omezení výpočetní kapacity vlastního hardwaru serveru. Je logické, že nelze z „běžného“ počítače vytvořit komunikační server pro velký počet účastníků. Pro tento účel je využíván specializovaný serverový hardware.[2] Jako hostující platformu lze použít pouze OS na bázi Unix/Linux.

Asterisk je primárně navržen, aby zastal funkci pobočkové ústředny. Podporuje celou škálu komunikačních protokolů, pomocí kterých může být komunikace přenesena od analogové telefonní linky až po vysokorychlostní optickou síť.

Velká oblíbenost Asterisku vychází zejména z důvodu, že je vyvíjen pod GPL.[3] Tento systém díky své škálovatelnosti lze přizpůsobit různým účelům nasazení. Např. jako vlastní PBX ústřednu s možností hovorového záznamníku, jako bránu pro konverzi mezi komunikačními či signalizačními protokoly apod.[8][1]

1.2.2 Instalace

Instalace Asterisku může probíhat na různé platformy typu Unix nebo Linux. Nicméně primární podpora je Linuxových platforem. Pro Unixové platformy např. FreeBSD, Solaris nebo Mac OS X není příliš rozšířena podpora pro komunikační TDM periferie. Z tohoto důvodu se na tento typ systémů nasazení doporučuje za předpokladu, že

[1] Volně přeloženo z anglického textu.

[2] Pro velké ústředny je používán specializovaný hardware a software od komerčního výrobce.

[3] Zajišťuje nejen svobodné šíření tohoto softwaru, ale i jeho různých modifikací.

budou sloužit pouze pro VoIP komunikaci nebo ústředně bude předřazena telefonní brána.[1][9]

Při realizaci diplomové práce byla použita platforma OS *Centos 5* z původně plánovaného systému *OpenSuse 11.1*.[4] Důvodem změny bylo detailnější sblížení se se systémem během posledního semestru.

Instalaci Asterisku lze provádět více způsoby. Asterisk lze stáhnout např.:

- ve formě balíčků, jejichž popis a postup instalace jsou uvedeny v [10];
- ve formě zdrojových souborů dostupných z *www.asterisk.org*.

V práci byla využita druhá možnost. Kompletní dokumentace postupu instalace lze naleznout v literatuře [1] a [11].

Bohužel v každé dokumentaci se lze setkat s trochu odlišným postupem instalace. Ta se může mírně lišit v závislosti na použité platformě. Z tohoto důvodu je zde uveden otestovaný postup při samotném vývoji účtovacího systému.

1. z *www.asterisk.org* byla získána verze asterisku 1.4.29.1 spolu s *Asterisk-Addons 1.4.10* [5] poskytující potřebné rozšíření modulů pro ukládání záznamů CDR do databáze a funkci *Asterisk Realtime* viz. kap 1.2.4;
2. před vlastní kompilací rozbalených souborů musí OS obsahovat potřebné balíčky:
   - *gcc, glibc-kernheaders, cpp, binutils, glibc-headers, glibc-devel, bizon, libtermcap, libtermcap-devel, newt, newt-devel, ncurses, ncurses-devel*;[1][12]
3. v adresáři */usr/src/asterisk-1.4.29.1* je provedena konfigurace pomocí `./configure`;
4. výběr potřebných součástí asterisku příkazem `make menuconfig`. V tomto případě zůstalo výchozí nastavení;
5. kompilace a instalace pomocí `make && make install && make samples`.[1]

1.2.3 Instalace doplňků

Základní pilíř účtovacího systému je práce s databází. Ta zajišťuje uchování dat a udržení integrity mezi nimi. Toto je důvod, proč nelze využít výchozího ukládání CDR do textového souboru ve formě CSV. Data budou ukládána přímo do relační databáze. Aby Asterisk mohl využívat služby databázového serveru, je nezbytné nainstalovat doplňkové moduly a provést jejich konfiguraci.

CDR záznamy mají definovanou vlastní strukturu včetně atributů. Strukturu lze o některé atributy doplnit. Jedním z nich může být např. unikátní identifikátor

[4] Pro navrhovaný systém nezáleží, která ze zmiňovaných distribuci je použita.

[5] Různé verze Asterisk-addons, neboli doplňky nejsou kompatibilní se všemi verzemi Asterisku! Je nezbytné nalézt v dokumentaci, která verze „doplňků“ je vhodná k dané verzi Asterisku!

položek záznamů, který je generován samotným modulem. Pro přehlednější relaci bude vytvořen atribut, o který se modul Asterisku nebude zajímat a jedinečnost tohoto atributu bude hlídána databázovým serverem.

Pokud by bylo nutné doplnit atribut, jenž by byl generován Asteriskem, musela by být dopsána definice do zdrojového souboru modulu před kompilací. Podrobný postup je uveden v literatuře [7][13].

Před kompilací doplňků je nutné označit v nastavení `menuconfig` tyto moduly:

- app_addon_sql_mysql;
- cdr_addon_mysql;
- config_mysql.

V této fázi by měl mít Asterisk kompilované veškeré prostředky nutné ke spolupráci s databázovým serverem. Zbývá konfigurovat soubor */etc/Asterisk/cdr_mysql.conf*, ve kterém je nastaven:

- účet a heslo, pod kterým se Asterisk bude přihlašovat do databáze;
- adresa databázového serveru a jeho port či soket;
- název databáze v databázovém serveru, kde jsou předem vytvořeny struktury potřebných tabulek viz. [13].

1.2.4 Asterisk RealTime

Je název architektury, která umožňuje zahrnout do dialplánu příkazy resp. položky z databáze. Mezi položkami může být zahrnuta např. konfigurace účtů nebo příkazy dialplánu.

Výhodou takového řešení je zejména mechanismus, který okamžitě zahrne změny nastavení účtů bez nutnosti manuálního občerstvování konfiguračních souborů. Další výhodou může být např. centralizované umístění prostředků pro více Asteriskových ústředen a pružná změna jejich konfigurace. Ovšem značnou nevýhodou je zvýšení požadavků na databázový server. Na něj se Asterisk odkazuje při každém hovoru. Více informací o tomto tématu bude probráno dále v této práci. [13]

Existují dva způsoby načítání z databáze:

- v prvním případě se jedná o načítání konfigurace jednotlivých účtů a dialplánu, kdy se změny projeví ihned;[6]

[6]Každé technologii(SIP,IAX...) musí být vytvořena v databázi speciální entita. Dialplán má také předepsanou strukturu.

- druhý způsob lze v literatuře [19] nalézt pod označením „Asterisk RealTime Static“. Zde se jedná o doplňující nastavení k jednotlivým spojovacím technologiím. Jedná se o nastavení, které je většinou v návěští `[general]`. V práci je tato technika využita např. pro registraci trunků.

1.2.5 Rozhraní AGI

AGI (Asterisk Gateway Interface) je rozhraní PBX Asterisk, které ústřednu neomezují pouze na vlastní implementované zdroje. S využitím AGI se Asterisk stává komplexní řešení, které může být rozšířeno o externí aplikace. Ty mohou být napsány téměř v jakémkoliv skriptovacím nebo programovacím jazyce. Do těchto aplikací lze směrovat proud dat z jádra Asterisku. Data tak mohou být zpracovány nebo upraveny mimo Asterisk.

Výběr jazyka závisí na tom, k čemu je daný modul navržen.[21]

- Zkompilované binární soubory např. c,c++ jsou dobře optimalizované na konkrétní hardware, ale zabírají více místa a složitě se upravují nebo přenášejí.
- Jazyky Java nebo C# lze spouštět pod „virtálním strojem“. Výhoda oproti předchozímu je v přenositelnosti. Nevýhoda ve spotřebě paměti a úpravě programů.
- Interpretované jazyky mají delší odezvu spouštění, protože kód je nejdříve zkompilován. Výhoda je malá velikost zdrojového kódu a snadná modifikace. Z těchto jazyků je nejvíce používáno PHP.[2]

Rozhraní AGI poskytuje externí aplikaci pevně danou množinu dat a příkazů ve formě parametrů. Skrze ně probíhá komunikace. Bližší informace lze naleznout v lit.[2] a [21].

1.2.6 Rozhraní AMI

AMI(Asterisk Manager Interface) umožňuje vzsálenou komunikaci s konzolí Asterisku prostřednictvím TCP/IP soketu. Pří komunikaci lze využívat množinu příkazů v rámci konzole.

Při zadávání příkazů je nutné dodržet jejich formální zápis, tzn. dodržet specifickou posloupnost a parametry. Jednotlivé parametry příkazu se oddělují odřádkováním[7]. Konec příkazu je pak označen znakem CR/LF za posledním odřádkovaným parametrem.

[7] Znak CR/LF.

Použitými parametry je určen typ paketu, který může být:

- Akce – požadavek na ústřednu (směr komunikace klient - ústředna);
- Odpověď – odezva na požadavek (směr komunikace ústředna - klient);
- Událost – informace generovné jádrem Asterisku (směr komunikace ústředna - klient).

AMI je v této práci využito pro webové rozhraní, které umožňuje vytáčení mezi vybranými pobočkami.

Konfigurace

Konfigurace AMI probíhá prostřednictvím souboru */etc/asterisk/manager.conf*.
V tomto souboru jsou vytvářeny pravidla a účty pro přístup k tomuto rozhraní. Každému účtu lze definovat oprávnění, které omezuji jeho působnost. Více informací lze nalézt v lit.[22]

V této práci je využit AGI skript *tarif_check.inc.php* pro řízení odchozích hovorů. Řízení probíhá na základě dotazu do databáze a zjištění, zda je pro volající účet povoleno, resp. stanoveno tarifikační pravidlo.

1.3 Internetové aplikace

Poměrně často je pod tímto pojmem představa nějaké internetové stránky či formuláře. Ovšem to není úplně přesné. Stránku[8] lze prohlásit za část aplikace. A to jen tehdy, splňuje-li určitou logickou návaznost mezi zpracovávanými daty. V podstatě aplikace je určitá míra abstrakce zahrnující předem definované procesy běžící na serveru nebo serverech či u klienta. HTML stránky jsou dnes běžně používány pro vizualizaci rozhraní mezi uživatelem a aplikací.

Za multiplatformní aplikaci je považována aplikace, které nezáleží na jakém OS je provozována. Stejně tak z jakého OS je k této aplikaci přistupováno. Jestliže je použito univerzálnosti webových stránek společně s interaktivností jazyků pro jejich sestavení, jako např. PHP, vzniká silný základ pro implementaci multiplatformní aplikace.

Výhoda spočívá také v dobré přenositelnosti. Aplikace je tvořena kódem některého z interpretovaných jazyků, které mají vlastní syntaxi a sémantiku. Ve spolupráci s jejich interpretem nezáleží např. ani na použitém HTTP serveru[9].

[8] myšlen HTML dokument

[9] Webový server musí podporovat CGI modul interpreta jazyka, ve kterém je aplikace napsána.

1.3.1 Interpret kódu dynamických aplikací

Internetové aplikace postavené na dynamicky se generujících HTML dokumentech nelze provozovat bez interpreta na webovém serveru.

Interpreta lze chápat jako prostředníka mezi zdrojovým kódem psané v některém programovacím jazyce a webovým serverem. Samotný server zapouzdřuje a odesílá data ve formátu, kterému musí rozumět klient. Dále může být kód interpreta součástí dokumentu HTML. Interpret je pak součástí prohlížeče a vykoná kód dříve, než vizualizuje dokument uživateli. Interprety lze tedy rozdělit podle působnosti a to:

1. Na straně klienta – zdrojový kód je společně se značkovacím odeslán klientovi. Klient zdrojový kód, nejčastěji JavaSkriptu, předá interpretovi[10]. Společně pak reprezentují formu stránky uživateli.
2. Na straně serveru – zdrojový kód je nejdříve HTTP serverem předán interpretovi. Interpret sestaví výslednou podobu HTML na základě vnitřních stavových proměnných aktuální relace a argumentů odeslaných od klienta. Výslednou podobu označkovaného dokumentu předá zpět HTTP serveru. Server odešle dokument klientovi.

Při vytváření aplikace je nutné si uvědomit, jakým účelům bude obsah sloužit. Na určité aplikace se hodí použít zpracování kódu na straně klienta a na jiné zase zpracování na straně serveru.

U klienta se nejčastěji zpracovávají různé animační prvky stránky – nabídky menu, animace tlačítek a obrázků... Zpracovávají se takové psané kódy, které následně nemají interakci se serverem a nedokáží narušit bezpečnost aplikace, dat v databázi nebo samotného serveru. Důležité je pamatovat i na to, ačkoliv se klientovi standardně viditelně nezobrazí veškerá data skriptu, uživatel může prohlížet samotný zdrojový kód odeslaný serverem. Na toto je třeba pamatovat zejména při porovnání citlivých dat(hesel a jiných informací), které by mohly být odhaleny.

Na straně serveru se vykonávají takové procesy, u kterých je požadována utajenost nebo jejich zabezpečení proti modifikaci zvenčí. Ke klientovi přicházejí pouze výsledná data. Např. při autentizaci do aplikace server požádá uživatele o autentizační údaje. Klient odešle data na server a ten je ve formě argumentů předá do aplikace. Aplikace tyto data porovná s databází a zjistí-li shodu, sestaví rozšířený obsah HTML dokumentu, který předá HTML serveru. Ten jej zašle zpět klientovi.

S tímto souvisí uchování informací k jednotlivým relacím a zajištění jejich jedinečnosti. Další informace k práci s relacemi jsou obsaženy v kap.1.3.3.

[10]Interpreti na straně klienta jsou nejčastěji součástí prohlížeče, případně aplikace je ošetřena tak, že požádá o jejich doinstalování.

PHP

PHP je interpretovaný jazyk navržen pro vývoj internetových aplikací. Jeho funkce není závislá pouze na HTML serverech, ale programy lze provozovat jako samostatnou GUI nebo konzolovou aplikaci.

V případě interpretovaných skriptů rozlišujeme, jakým způsobem jsou spouštěny. A to:

- jako CLI skripty spouštěné pomocí příkazové řádky;
- jako CGI skripty spouštěné web serverem.

CGI zpracovávají standardní vstup STDIN mírně odlišným způsobem a vyčítání proudu proměnných je problematičtější. Např. u PHP AGI skriptů je vhodnější zachytávání proudu dat z Asterisku pomocí PHP-CLI. [1][2]

Vlastní kód PHP lze jednoduše vložit mezi značky „<?php“ a „?>“ v HTML jazyce nebo jako doplněk např. v jiném interpretovaném jazyce. PHP je proto velmi oblíbený k tvorbě nejen dynamických HTML dokumentů.

V projektu je PHP využito k vytvoření externí CLI, respektive AGI skriptu. Tímto skriptem bude explicitně ovlivňováno chování dialplánu Asterisku či monitorování hovorů.

Pomocí předem definovaných funkcí lze jednoduše aplikaci připojit k relační databázi MySQL, Oracle nebo jiné. Použití a koncepce je jednodušší než u většiny konkurenčních jazyků. Od verze PHP3 je tento jazyk schopen objektového programování. Soubory potřebné k funkci aplikací v PHP kódu lze volně stáhnout na Internetu. Aktuální řada je PHP5.[3]

Perl

Perl patří stejně jako PHP mezi interoretované programovací jazyky. Nejčastější využití je pro tvorvu HTML nebo skriptování v OS při zpracování souborů. Pomocí Perl lze psát i složité programy nebo aplikace, které podporují uživatelské grafické rozhraní. Perl vznikl na platformě Unix, ale protože se jedna o interpretační jazyk, byl rychle rozšířen na většinu platforem (Windows, Novell, MS DOS...).[4]

Pyhon

Pyhon je další jazyk z rodiny interpretovaných jazyků. Při jeho vývoji byl kladen důraz na jednoduchost a co největší efektivitu. Obecně se tento jazyk doporučuje pro začátečníky díky jeho přehlednosti.

Koncepčně je tento jazyk více pragmatický. To umožňuje psát nejen objektově orientované, ale i procedurální paradigma.[3]

ASP.NET

ASP.NET je profesionální technologie pro tvorbu webových aplikací z dílny Microsoft. Je součástí nadstavby operačního systému, kterou je .NET Framework, pomocí něhož se aplikace spojují s databázemi, grafikou, práci se soubory a dalšími systémovými objekty. Výhodou technologie ASP.NET není omezení na jediný programovací jazyk. .NET Framework počítá s použitím programovacích jazyků, které byly určeny zejména pro objektové programování „okenních“ aplikací. Programátor může pracovat v jazycích jako je Visual Basic.NET, C#NET, ale v Perl či Pascalu. Multijazyčnost je umožněna pomocí kompilace na řízený kód MSIL. Ten je následně zpracováván prostředím Common Language Runtime obsaženého v .NET Framework.

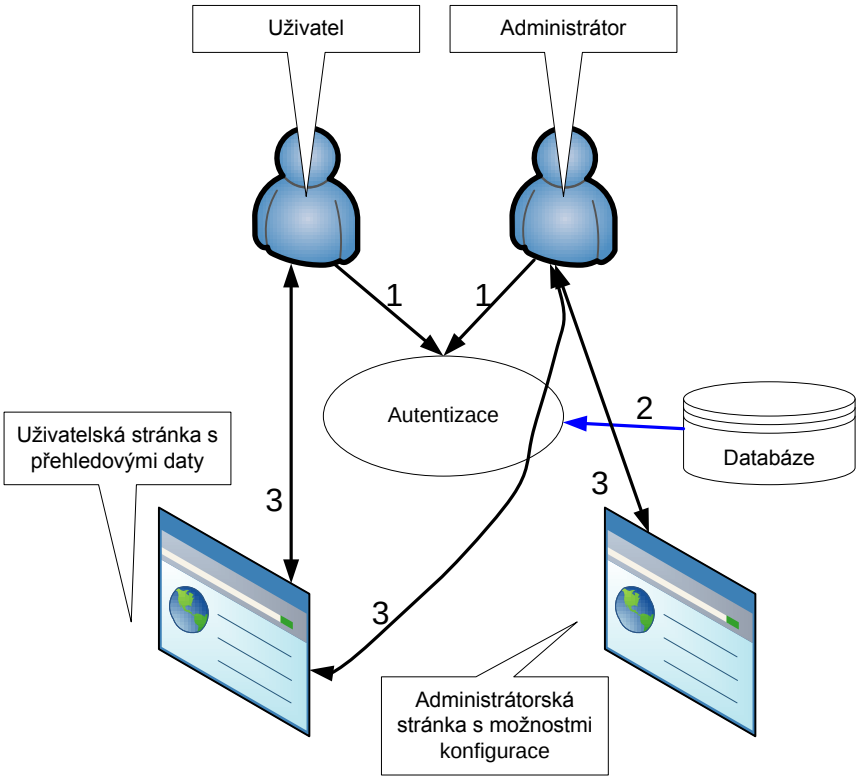
1.3.2 Uživatelské rozhraní účtovacího systému

Uživatelské rozhraní je důležitou součástí účtovacího systému. Pro uživatele tím odpadá nutná znalost nastavení dialplánu nebo jednotlivých poboček PBX konfiguračními soubory. Tímto rozhraním lze nastavit většinu potřebných částí Asterisku, které logicky souvisí s účtovacím systémem.

Jednou z hlavních priorit uživatelského rozhraní je poskytnout dohled nad vlastním účtováním hovorů. Je zde možné vytvářet a upravovat pravidla, kterými jsou jednotlivé hovory klasifikovány a následně účtovány. Dle potřeb uživatelů lze aplikovat pravidla pro účtování individuálně. Architektura tarifikace je založena na sloučení „elementárně“ popsaných specifik. Těmi jsou tarifní skupina a použitý prefix čísla.

Rozhraní poskytuje nejen funkce pro administrátory, ale poskytuje i přehledová data uživatelům, využívající volání přes Asterisk s tímto systémem. Pro uživatele to jsou zejména informace, které slouží k dohledu nad zřízenými telefonními účty, zejména:

- Počet účtů s kreditním tarifem – Prepaid;
- počet účtů s fakturovaným tarifem – Postpaid;
- hodnota ve zbývajících kreditech;
- průběžná hodnota k fakturaci za aktuální období;
- celková neuhrazená hodnota k fakturaci;
- provolaná doba celkově i jednotlivých účtů za aktuální období;
- výpis volání k jednotlivým účtům
  - cílová volba;
  - čas uskutečnění hovoru;
  - doba hovoru;
  - cena hovoru;



Obr. 1.1: Znázornění přístupu ke kořenovým HTML stránkám.

– identifikátor účtovací položky.

Účtovací systém je tvořen dvěma základními částmi a to *uživatelskou* a *administrační*. Obě tyto části jsou na sobě nezávislé a lze je např. vzhledově úplně odlišit. Uživatel s administrátorskými oprávněními může pracovat v administrativní i uživatelské rovině, protože i na administrátora je současně nahlíženo jako na uživatele viz. obr.1.1.

Systém je rozdělen na dvě roviny, ale lze ho přizpůsobit i detailněji. Je to umožněno zejména navrženou architekturou databáze. Uživatelské rozhraní je složeno z jednotlivých modulů, resp. skriptů, které vykonávají nad systémem určitou funkci. Jednotlivé moduly může využívat jen uživatel, který má potřebná oprávnění. Aby nevznikla složitá situace s nastavením oprávněním každého uživatele, tak je nejprve vytvořena celá skupina[11], k níž se definují systémová oprávnění. Následně je uživatel přidán do dané skupiny.

[11] Ve výchozím nastavení administrátoři a uživatele.

Architektura kořenových HTML

Systém je rozvržen na dva separované kořenové zdrojové skripty, generující HTML dokumenty – pro uživatele a administrátory. Principiálně jsou oba skripty totožné. Do těchto skriptů jsou podle požadavků implementovány další zdrojové kódy umožňující rozšíření programové výbavy pro potřebnou operaci. Např. je-li žádost k vytvoření nového uživatele, tarifu. . . , systém rozšíří kořenový skript o potřebný modul.

Výhoda tohoto řešení spočívá v možnosti tvořit přehledné účelové skripty zaměřené na konkrétní operaci. Dále lze systém rozšiřovat o další funkce implementací nových skriptů.

Implementace skriptů

Implementace spočívá v:

- nahrání zdrojového souboru s kódem skriptu na místo, ke kterému je oprávněn HTML server přistupovat;
- registrování nahraného souboru v databázi – pro tento účel vytvořena entita *functions* viz. tab.1.20 v kap.1.7;
- definice oprávnění uživatelských skupin pro přístup k tomuto skriptu;
- určení, za jakých okolností lze skript načíst – určit tzv. životní cyklus[12].

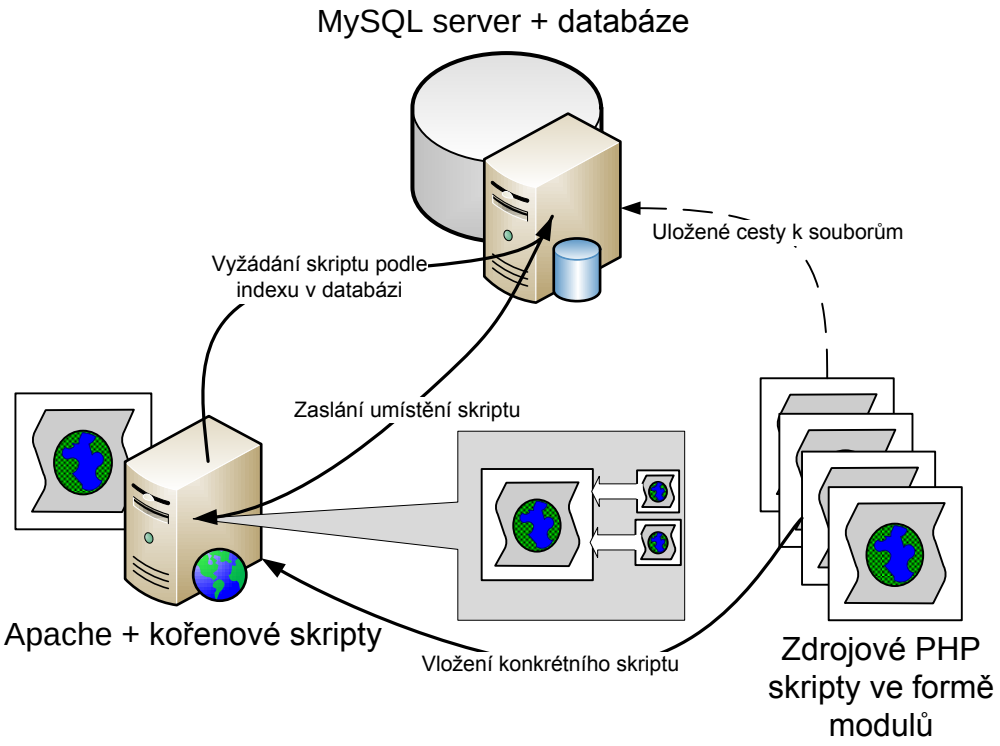
1.3.3 Zabezpečení uživatelského rozhraní systému

Zabezpečení přístupu do webového rozhraní systému je zajištěno uživatelským účtem. Účet je definován loginem a heslem. Aby nedošlo k odhalení hesla během autentizace, heslo je před odesláním převedeno jednosměrnou funkcí na haš o pevné délce. Tento řetězec je ověřen serverem, který jej porovnává s otiskem hesla uloženého v databázi.

Po úspěšné autentizaci si server i klient uchovávají informace o jejich vzájemné relaci. Po odhlášení ze systému server i klient data o relaci zničí. Při zahájení nové relace je opět nutné projít procesem autentizace. Odhlášení ze systému je možné způsoby:

- odhlášení pomocí odkazu vytvořeného pro tento účel;
- zavřením okna prohlížeče;
- nečinností po určitou dobu – doba je stanovena v konfiguračním souboru */etc/php.ini* v direktivě `session.cache_expire`,

[12] Podle definice životního cyklu je zobrazován odkaz na skript v menu.



Obr. 1.2: Princip načítání PHP modulů do kořenového skriptu.

1.4 HTTP server

HTTP server je program, který na základě požadavků klienta[13] formulovaný protokolem HTTP/HTTPS zasílá požadovaný obsah dat nebo HTML dokumentu. Data musí být uložena v místě, které je serveru přístupné.

Rozlišují se dva základní typy WWW dokumentů, které jsou zasílány klientovi:

1. *Statické dokumenty* – označkovaný obsah je předem pevně daný;
2. *Dynamické dokumenty* – obsah označkovaného dokumentu je generovaný na základě různých podmínek jazyka, který dokument sestavuje. Těmito jazyky mohou být např. PHP, ASP, Perl, Python apod.

1.4.1 Apache

Apache je program zajišťující HTTP server. Je vyvíjen pod značkou *The Apache Software Fundation*. Jedná se o komunitu vývojářů i uživatelů. Apache je aktuálně

[13]Klientem se rozumí webový/HTML prohlížeč např. Internet Explorer, Mozilla Firefox, Opera...

vyvíjen pod licencí *Apache License, Version 2.0.*[14]

Apache patří mezi nejvíce používané HTTP servery. Je dostupný pro platformy Windows či platformy Unixového/Linuxového typu. Nabízí spoustu možností. např.:

- podporu zabezpečeného spojení HTTPS;
- virtuální hosting – několik domén na jednom serveru;
- možnost autentizace uživatele;
- funkci reverzního Proxy – využívané na serverových farmách pro tzv. load balancing;
- podporu skriptovacích jazyků apod.

Instalace Apache

Instalační soubory serveru Apache je možné získat ze stránek komunity *www.apache.org* nebo stáhnout instalační balíček prostřednictvím OS. Při vývoji účtovacího systému byl Apache instalován společně s OS *Centos 5*. Instalace je poměrně jednoduchá a tato práce se jí nebude dále zabývat. Dokumentaci lze naleznout v lit.[14].

Konfigurace Apache

Po instalaci Apache je vytvořen jeden nebo několik konfiguračních souborů.[15] Hlavním je */etc/httpd/conf/httpd.conf*, ve kterém je provedeno nastavení. Direktivou `DocumentRoot` je určena výchozí adresa serveru do */var/www/html*. V tomto umístění jsou zdrojové soubory vytvořené webové aplikace k účtovacímu systému.

Aplikace je složená z množství PHP skriptů a dalších souborů. Z hlediska dodržení bezpečnostního návrhu aplikace je požadováno, aby uživatel mohl prostřednictvím webového serveru přistupovat pouze k určitým skriptům. Při nedodržení této podmínky by mohl útočník se znalostí konkrétních jmen souborů se skripty obejít bezpečnostní model aplikace a modifikovat data v databázi.

Požadovaného efektu bylo dosaženo tak, že souborům, které mají být přístupné uživatelům byla ponechána přípona *.php* a ostatním souborům s proveditelným kódem byla nastavena přípona *.inc.php*. Dále je ve stejné složce, kde jsou umístěny soubory se skripty, tj. */var/www/html* vytvořen soubor *.htacces*, ve kterém jsou specifikována pravidla pro server Apache k dané složce. V těchto pravidlech je zakázáno listování ve složce pomocí prohlížeče a dále je zakázáno přistupovat k souborům s koncovkou *.inc.php*.[14][15]

Výpis souboru *.htaccess*:

[14] Jedná se podobně jako GPL o open-source licenci.[14]

[15] Závisí na distribuci.

```
Otpions -Indexes
<Files ~ "\.inc">
  Order Allow,Deny
  Deny From All
</Files>
```

1.5 Návrh databázového modelu

Před vlastním návrhem řešení, jak bude databáze koncipována, je nezbytné osvojit si některé pojmy a zásady. Jestliže je databáze dobře navržena, ušetří spoustu řádků programování.

Databáze je schopna zajistit spoustu událostí ve vlastní režii a tato schopnost ušetří spoustu dat mezi dotazy aplikace na databázový server. Aplikace tudíž nemusí ošetřovat některé výjimky nebo obsahovat duplicitně implementované funkce.[6][5]

1.5.1 Databázový systém

Databázový systém je software zajišťující implementaci databáze od výrobce, kterými jsou např. MS SQL Server, Oracle, MySLQ[16]...

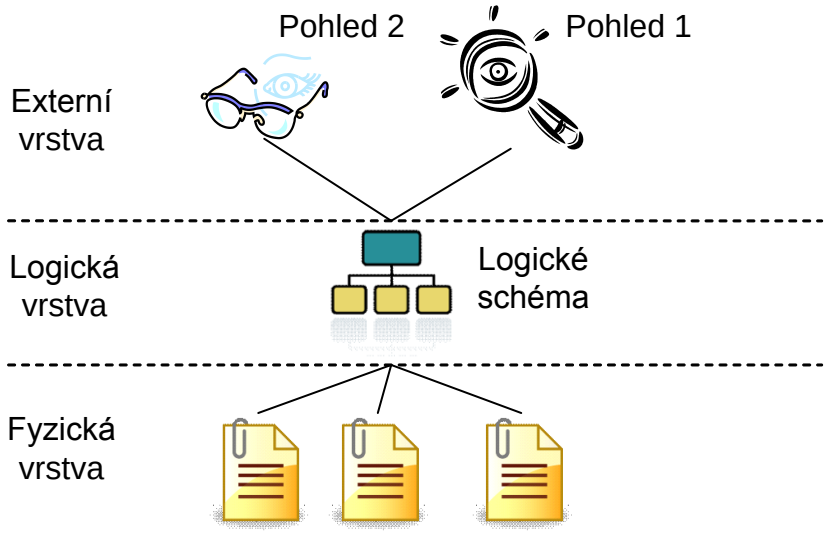
Existuje celá řada modelů databází. Nejvhodnější a nejpoužívanější jsou databáze pracující na tzv. relačním modelu.[5]

Databázové systémy zajišťují uživatelům určitou datovou abstrakci. Co si pod tímto představit? Data, která jsou v databázi uložena v jediné kopii lze zobrazovat různými pohledy uživatelům. Tzn. jsou-li v databázi uložena vzájemně logicky související data, různými pohledy je lze reprezentovat ve vhodné formě konkrétnímu uživateli, oddělení nebo aplikaci, která si o ně požádá. Databázový systém se dále stará o tyto úkoly:

- Zajišťuje správu několikanásobných přístupů uživatelů k datům a ošetřuje vzájemné konflikty mezi aktualizacemi provedené odlišnými instancemi přístupu resp. jinými uživateli;
- Implementuje transakce, které zajišťují integritu mezi daty při několika současných změnách logicky svázaných dat. Při selhání nebo zrušení transakce vrátí všechna zasažená data do stavu před transakcí. Po úspěšné transakci jsou zasažená data změněna a transakce je úspěšná jen tehdy, jestliže všechny požadované změny byly úspěšné.
- Zajišťuje dotazové rozhraní dotazového jazyku – SQL.
- Zajišťuje další mechanismy např. pro zotavení, zálohu, bezpečnost...

[16]Byla využita při řešení této práce.

K zajištění určité abstrakce je databázový systém rozdělen do několika vrstev viz obr.1.3. Nejnižší neboli *fyzická vrstva* zajišťuje reprezentaci databáze v podobě souborů na datová úložiště. *Logická vrstva* se stará o logickou návaznost mezi daty, která jsou reprezentována databázovými objekty. *Externí vrstva* poskytuje tzv. pohledy. Ty umožňují vyjádřit konkrétní data potřebná jako vstupní informace do konkrétní aplikace.[6][5]



Obr. 1.3: Rozdělení vrstev abstrakce v databázovém systému.

1.5.2 Základy databáze

Při návrhu databáze se využívají pojmy *entita*, *atribut* a *relace*. Jsou to základní jednotky při návrhu logického schéma databáze.[6][5]

Entity

Pomocí entit při návrhu modelu jsou popisovány určité objekty, o kterých se shromažďují informace. Jedná se o třídu, která definuje určité jedinečné vlastnosti. Entity jsou navrhovány tak, aby 2 entity popisující různé věci neobsahovaly duplicitně vlastnosti, které mají společné.

Atributy

Reprezentují jednotku skutečnosti. Entity zahrnují atributy, pomocí jejichž složení jsou popsány charakteristické vlastnosti. Např. bude-li entita vyjadřovat automobil, tak zjednodušeně může obsahovat atributy:

- značka;
- model;
- rok výroby;
- barva...

Relace

Relace vyjadřují vztahy mezi souvisejícími entitami. Relace mezi entitami mohou být omezeny kardinalitou. Ta určuje maximální počet instancí entit jednotlivých stran dané relace. Relace mohou být typu [5] :

- jedna k jedné–(1:1);
- jedna k více–(1:N);
- více k více–(N:M);
- rekurzivní relace.

Datové typy

Datovými typy jsou blíže specifikovány atributy. Pomocí DT lze omezit určitou množinu informací na znaky, které jsou pro konkrétní atributy smysluplné. To napomáhá databázovému systému, jakým způsobem bude data zpracovávat a uchovávat. Např. je-li použitý 2-krát DT Integer, databázový systém ví, jak s těmito daty naložit, pokud bude požadováno jejich odečtení, násobení,... Při jiném DT by se mohl výsledek operace lišit od očekávaného. DT používané v databázích jsou do jisté míry standardizované, ale každý výrobce používá i mnoho vlastních DT. Kolekce datových typů jsou předem stanoveny konkrétním databázovým systémem.

Omezení

Omezení definují aplikační pravidla nad konkrétním objektem. Každé omezení má vlastní jméno, na které databázový systém odkazuje v chybové hlášce, např. je-li požadována operace, jež není dovolená.

1. *Omezení primárního klíče* – každá entita(tabulka) musí obsahovat unikátní identifikátor. Ten může být složen i z několika atributů. Společně ale musí být jedinečné pro každý záznam.

2. *Referenční omezení* – zajišťuje platnost relací. Relace jsou svázány pomocí cizích klíčů, které se vkládají do dceřiných tabulek. Tímto jsou ošetřeny stavy, kdyby měl být vytvořen záznam v dceřiné tabulce, ale neodkazoval by na žádný záznam v rodičovské tabulce. Stejně tak nemůže být odstraněn záznam v rodičovské tabulce bez odstranění dceřiného záznamu.
3. *Check omezení* – lze jím ověřovat platnost vstupních dat. Pravidlo vyhodnocuje, zda vstupní data splňují podmínku pro zápis či nikoliv.
4. *Triggery* – neboli spouště fungují jako rozšířené *check*. Samotné *check* dokáží hlídat pouze atributy v jedné tabulce. To ale nestačí při ověřování informací z více tabulek. Triggery se přiřadí k určitým hlídaným atributům a spustí se při pokusu zápisu. Nejdříve jsou informace získané funkcí triggeru a po vyhodnocení vrací *True* nebo *False* jako check.

1.5.3 Pohledy

Patří k dalším usnadněním databázových systémů. Do pohledu jsou ukládány předem vytvořené dotazy, které mohou spojovat i několik tabulek a vybírat k zobrazení jen určité atributy[17] a dokonce i určité položky[18]. Výhoda je zejména v tom, že v programu aplikace použijeme jen příkaz pro získání pohledu. Databáze si zajistí spojení tabulek dle pravidel pohledu a následný výstup do aplikace ve vlastní režii. Pohled se jeví jako běžná tabulka, ale existuje jen virtuálně.

Dále pohledy nabízejí určitou nezávislost na aplikaci. Při případné změně logiky databáze jsou jen poupraveny pohledy, ale pro aplikaci se změny nevyskytnou.

1.5.4 Normalizace

Normalizace je takový postup, kdy se dekomponují vzájemné souvislosti mezi daty. Data jsou roztříděna do jednotlivých entit pospojovaných relacemi. Základní normalizace se rozdělují na 3 normální formy:

1. Normální forma – pro splnění musí být odstraněny atributy s násobnými hodnotami;
2. Normální forma – do 2. NF patří relace, která má všechny neklíčové atributy závislé pouze na celém PK[19].
3. Normální forma – je splněna, pokud jsou odstraněny tranzitivní závislosti. Tzn. neklíčové atributy relace závisejí pouze na vlastním PK.

[17] Atributy – na úrovni sloupců.

[18] Položky – na úrovni řádků.

[19] Primární klíč může být složen z více atributů, které jsou společně v rámci relace jedinečným identifikátorem.

Existuje i 4. a 5. „normální forma“. Ty se již v běžné praxi často nenasazují.

1.5.5 MySQL server

MySQL je relační databázový systém. Za jeho vznikem stála firma MySQL AB, která nyní patří do portfolia Sun Microsystems. MySQL je multiplatformní databázový server, který je možný provozovat na různých OS. Nejčastěji je však nasazován v sadě s dalším nekomerčním softwarem v tzv. kombinaci LAMP[20].[16] Toto byl i důvod nasazení při realizace účtovacího systému v této práci.

Instalace

V této práci instalace nebude rozebírána. Návod lze nalézt v lit.[20].

1.6 Účtování hovorů

Tato kapitola se zabývá, na základě jakých parametrů a jakým způsobem bude systém účtovat cenu hovorů realizovaných přes PBX Asterisk. Základní princip spočívá ve vytvoření tarifů neboli tarifních skupin, pomocí kterých bude účtována cena jednotlivým klientům.

1.6.1 Tarif

Při účtování nastává situace, která bude muset řešit správné přiřazení tarifů k jednotlivým spojením. Parametry, na jejichž základě bude přiřazen odpovídající tarif mohou být:

1. kdo volá, resp. z jakého typu[21] účtu je požadováno spojení;
2. jaká tarifní skupina je přiřazena volajícímu;
3. jaká je předvolba k dosažení volaného (poskytovatel může stanovit různé ceny na základě předvolby – do zahraničí, k jinému operátorovi či poskytovateli...

Všechny potřebné informace zajišťují CDR záznamy poskytované Asteriskem. V této fázi jsou určeny atributy, kterými bude identifikován tarif určující cenu.

[20] Linux, Apache, MySQL, PHP.

[21] SIP,H323,IAX... V práci je implementována technologie SIP.

Tarif – tarif

Entita *tarif* sdružuje informace o cenové mapě. Trendem dnešní doby je stanovovat nelineární tarifikaci. Cena resp. počet tarifikačních impulsů se na základě délky uplynulého hovoru mohou měnit. Často je operátorem stanovena vyšší cena v prvních minutách hovoru.

Pro dodržení normalizace je tato entita spojena s dalšími entitami viz. dále. Jedna z relací typu „jedna k více“ je mezi tabulkou *tarif* a tabulkou účtovacích intervalů *billing_periods*.

Tab. 1.1: Entita udržující základní popis položky – tarif

Atribut	**Datový typ**	**Index**
idTARIF	INT	PK,AI
Name	Varchar(32)	NN
idCENAJEDNOTKY	INT	FK,NN
description	Varchar(160)	
validity	Bool	NN
last_change	Timestamp	NN

- *idTARIF* – slouží jako umělý PK k jednoznačné identifikaci konkrétní položky;
- *Name* – Název tarifní položky, indexován jako unikátní;
- *idCENAJEDNOTKY* – cizí klíč k určení jednotkové ceny;
- *description* – popis tarifní položky;
- *validity* – určuje platnost záznamu;
- *last_change* – časové razítko poslední změny.

Účtovací intervaly – billing_period

V této entitě jsou uchovávány záznamy dvojic atributů *uptotime* a *unionnum*. První atribut určuje, od jaké doby hovoru bude každou sekundu účtován počet tarifikačních impulsů. Ty jsou určeny druhým atributem.

Výslednou cenu takového účtování lze zjistit pomocí stanovené ceny za jednotku. Pro lepší názornost účtovací intervaly mohou vypadat např. jako v tab.1.2

Tab.1.2 obsahuje příklad, podle kterého se v první sekundě zaznamená 30 tarifikačních pulsů. V 1/2 minuty hovoru bude započteno dalších 30 pulsů a od 60 sek. bude za každou započatou sek. přičten 1 puls.

Sumou všech pulsů za hovor trvající 1:20 minut je podle tab.1.2 výsledná hodnota 80 pulsů. Tato hodnota po vynásobení s hodnotou jednotkové ceny *cost* např. 0,05 Kč tvoří výslednou cenu 4 Kč. Cena je připojena k entitě *tarif* cizím klíčem

Tab. 1.2: Příklad účtovacích intervalů.

uptotime	**unionnum**
0	30
1	0
30	30
31	0
60	1

idCENAJEDNOTKY z entity *union_cost*. Sloupec *unionnum* obsahuje hodnoty koeficientů, které na základě přepočtu s jednotkovou sazbou reprezentují výslednou cenu započatých časových intervalů.

Tab.1.2 ve skutečnosti obsahuje pouze řádky všech hodnot vložených do systému. Posloupnost každé tarifní položky vzniká až spojením k tabulce *Tarif*. K jediné položce entity *tarif* musí být umožněno připojit více položek účtovacích intervalů. Zároveň pro dodržení 1.NF musí být použitelný jakýkoliv řádek pro více tarifních položek. Cena se pak může odlišovat pouze jednotkovou cenou v tarifních položkách. Z výše uvedeného vyplývá, relace mezi tabulkou *tarif* a tabulkou *billing_period*, obsahující položky účtovacích intervalů, vztah M:N. Pro jejich vzájemné svázání je použita vazební neboli spojovací tabulka. Tato tabulka je v databázi pod označením *spec_billing_period*. Z jejího účelu je zřejmé, že primární klíč je dvojice klíčů entit *tarif* a *billing_period*

Tab. 1.3: Entita položek účtovací mapy – billing_period

Atribut	**Datový typ**	**Index**
idBILLINGPERIOD	INT	PK,AI
uptotime	INT	UQ,NN
unionnum	INT	
validity	Bool	NN
last_change	Timestamp	NN

- *idBILLING_PERIOD* – umělý PK pro jednoznačnou identifikaci záznamu v tab. *billing_period*;
- *uptotime* – viz. levý sloupec tab.1.2;
- *cost* – viz. pravý sloupec tab.1.2

Tab. 1.4: Spojovací entita účtovací mapy – spec_billing_period

Atribut	Datový typ	Index
idTARIF	INT	PK,FK
idBILLINGPERIOD	INT	
validity	Bool	NN
last_change	Timestamp	NN

1.6.2 Tarifní skupina

Tarifní skupina stanovuje pravidla, podle kterých lze k hovoru jednoznačně určit cenu. Aby nenastala kolize mezi dvěma či více pravidly je vytvořena entita *spec_tarif_mix*. Tato entita určuje jednoznačnost mezi:

- tarifní skupinou;
- prefixem čísla;
- odchozím trunkem;

Hodnoty prefixu čísla a tarifní skupiny musí být vždy unikátní. Systém podle těchto klíčů určuje nastavení ve vytáčecím plánu, jestli jde o pokus přímého nebo vzdálené spojení prostřednictvím trunku.

Tarifní skupina – tarif_mix, spec_tarif_mix

Entita uchovává údaje o názvu skupiny pro identifikaci při konfiguraci systému administrátorem. Dále obsahuje časový údaj začátku platnosti skupiny.

Tab. 1.5: Entita tarifní skupiny – tarif_mix

Atribut	Datový typ	Index
idTARIFMIX	INT	PK,AI
Name	Varchar(45)	UQ,NN
start_date	Datetime	NN
description	Varchar(160)	
validity	Bool	NN
last_change	Timestamp	NN

- *idTARIFMIX* – umělý PK nebo i FK pro jednoznačnou identifikaci záznamu;
- *Name* – jedinečný název tarifní skupiny;
- *start_date* – datum a čas, od kdy začne záznam nabývat platnost;
- *idTRUNK* – FK pro udržení relace s entitou *trunk*

Tab. 1.6: Entita specifikace tarifní skupiny – spec_tarif_mix

Atribut	**Datový typ**	**Index**
idNUMBER	INT	PK,FK
idTARIFMIX		
idTRUNK	INT	FK
idTARIF		
validity	Bool	NN
last_change	Timestamp	NN

- *idTARIF* – FK pro udržení relace s entitou *tarif*

Způsob platby – pay_type, spec_pay_type

Rozšířením skrze tarifní skupinu je možnost definice typu platby:

- Postpaid – platba zpětně za určité časové období;
- Prepaid – platba předem. Z této částky je postupně odečítána sazba za provedené volání.

Tab. 1.7: Vazební entita definující typ platby k tarifní skupině – spec_pay_type

Atribut	**Datový typ**	**Index**
idTARIFMIX	INT	PK,FK
idPAYTYPE	INT	FK
validity	Bool	NN

Tab. 1.8: Entita definující název platby – pay_type

Atribut	**Datový typ**	**Index**
idPAYTYPE	INT	PK,AI
name	Varchar(15)	NN,UQ
validity	Bool	NN

- *idTARIFMIX* – PK,FK připojení relace k tab.1.5;
- *idPAYTYPE* – identifikátor způsobu platby, který určuje i její název;

Přiřazení tarifní skupiny účtu – assign_tarif_mix

Každý volací účet může mít přidruženo několik tarifních skupin. Pro daný okamžik účtování je na základě data *start_date* algoritmem vybrána jediná skupina. Lze tak

z dlouhodobého hlediska naplánovat více tarifních skupin, které mohou postupně nabývat platnosti.

Tab. 1.9: Entita připojující k účtu tarifní skupinu – assign_tarif_mix

Atribut	**Datový typ**	**Index**
idTARIFMIX	INT	PK,FK
id		
idTECHNOLOGY		
validity	Bool	NN
last_change	Timestamp	NN

- *idTARIFMIX* – PK,FK připojení relace k tab.1.5;
- *id* – PK,FK jeden z klíčů relace k tab.1.10 ;
- *idTECHNOLOGY* – PK,FK další z klíčů relace k tab.1.10

1.6.3 Identifikace účtů

V kap.1.6.1 a 1.6.2 se jednalo o informace týkající se vytváření popisu účtovacích pravidel. Tato kapitola popisuje způsoby identifikace účtu k uživateli a k účtovacím pravidlům.

Asterisk poskytuje různé technologie[22] spojení. Ačkoliv databáze je navržena pojmout různé technologie, podpůrné prostředky[23] jsou z důvodu rozsáhlosti po domluvě s vedoucím práce zaměřeny jen na SIP.

Pro ostatní způsoby komunikace by bylo nutné v databázi vytvořit entitu specifickou pro každou technologii. Název nové entity a popis vložit do tabulky *technology*. V aplikačním rozhraní a v podpůrných skriptech by bylo nutno vytvořit složitou logiku, protože každá technologie využívá jiné parametry nastavení. Např. skript pro vkládání účtů by musel být přizpůsoben jednotlivým množinám vkládaných parametrů.

Jednoznačná identifikace záznamů v databázi je určena entitou *user_account* viz. tab.1.10. Entita sdružuje informace o relacích:

- vlastníka účtu;
- technologie účtu;
- tarifní skupiny;
- plateb.

[22] SIP, IAX...
[23] Určité skripty a aplikační rozhraní.

Tab. 1.10: Entita identifikující účet – user_account

Atribut	**Datový typ**	**Index**
id	INT	PK,FK
idTECHNOLOGY		
idUSER	INT	FK
idBILL	INT	FK
validity	Bool	NN
last_change	Timestamp	NN

Tab. 1.11: Entita identifikující technologii spojení – technology

Atribut	**Datový typ**	**Index**
idTECHNOLOGY	INT	PK,FK
technology	Varchar(45)	UQ,NN
table_name	Varchar(45)	UQ,NN

- *id* – PK,FK připojení relace k tab.1.12;
- *idTECHNOLOGY* – PK,FK klíč určuje relace k tab.1.11 ;
- *technology* – obsahuje název, např. „SIP“;
- *table_name* – entita, ze které je požadován klíč *id*. Může se vyskytovat např. tabulka *iax_buddies*, které obsahuje *id* překrývající se s tabulkou *sip_buddies*.[24]

SIP účet Asterisku – sip_buddies

Pro konfiguraci účtu SIP jsou v Asterisku možné dva způsoby – pomocí „plochého“ souboru */etc/asterisk/sip.conf* nebo s využitím databáze, tzv. funkce „Asterisk realtime“. Ve druhém případě není nutné občerstvovat Asterisk při vytvoření nebo změně účtu. Změna se projeví ihned.

Informace o SIP klientech musí být opatřeny klíčovými atributy a jejich názvy, pomocí kterých ovladač Asterisku získává informace.[25][13] Tato entita viz.tab.1.12 (nejčastěji *sip_buddies*) patří mezi systémové tabulky Asterisku.

- *id* – PK k jednoznačné identifikaci položky v rámci tabulky;
- *name* – obsahuje login, kterým se SIP klient k ústředně přihlašuje;
- *callerid* – obsahuje informace, která by se měla zobrazit volané straně, zpravidla stejná hodnota jako v *name*

[24] Ostatní části účtovacího systému jsou zaměřeny pouze na SIP.
[25] Součástí doplňku k Asterisku.

Tab. 1.12: Vybrané atributy entity účtů SIP – sip_buddies

Atribut	**Datový typ**	**Index**
id	INT	PK
name	Varchar(80)	NN
callerid	Varchar(80)	
canreinvite	CHAR(3)	
context	Varchar(80)	
host	Varchar(31)	NN
language	CHAR(2)	
md5secret	Varchar(80)	
type	Varchar(6)	NN
disallow	Varchar(100)	
allow	Varchar(100)	
nat	Varchar(5)	NN

- *canreinvite* – atribut určuje, bude-li po zahájení spojení možné směrovat datový tok přímo mezi klienty. Existuje zde několik možností nastavení(viz.[17]), ale v této práci je *canreinvite* striktně zakázáno. Hodnota nového záznamu bude nastavena na *„no“*.
- *context* – určuje v jakém kontextu dialplánu v případě příchozího spojení je Asteriskem vyhledáváno spojovací pravidlo. Účtovací systém využívá dva kontexty viz.kap. 2.2.
- *host* – atribut určuje jakým způsobem bude určena klientova adresa. V případě hodnoty *dynamic* bude informace upřesněna až po nezbytné registraci klienta. Při požadavku ke spojení s určitým klientem má Asterisk informaci, na jakou adresu vyšle požadavek INVITE.
- *language* – atribut specifikuje, jaká jazyková zvuková sada bude použita při přehrávání hlášek např. pro IVR automat;
- *md5secret* – atribut obsahuje heslo vzniklé např. MD5 hašem řetězce složeného z <*name*>:<realm>:<heslo>, kde hodnota *realm* je v účtovacím systému nastavena „asterisk“;
- *type* – *friend,peer,user* omezuje možnosti klienta volat i přijímat, pouze volat nebo pouze přijímat hovory;
- *disallow* – obsahuje seznam zakázaných kodeků(nejčastěji „*all*“ a atributem *allow* se kodeky povolují selektivně);
- *nat* – nastavení chování Asterisku viz.[18]. Nastaveno na hodnotu „*yes*“.

1.6.4 Nalezení tarifu

Pro přiřazení tarifní položky k hovoru je potřebné doplnit data, s jejichž pomocí bude hovor zaúčtován. V kap.1.6.1 byly zmíněny informace, podle kterých lze párovat hovory k tarifním položkám. K tomuto účelu je vytvořena struktura v databázi.

Klíčové informace jsou poskytovány záznamy CDR ukládaných do tabulky *cdr*, které jsou vytvářeny po ukončení hovoru. Po uložení záznamu je v databázi spuštěn trigger, který zpracuje některé informace a vytvoří záznam v tabulce *call_records*. Vývojový diagram triggeru lze nalézt v příloze B.

Informace z těchto záznamů jsou zpracovávány skriptem uloženým mezi skripty uživatelského rozhraní */build/cdr_count.inc.php*[26]. Tento skript musí být periodicky spouštěn. Perioda je nastavena pomocí programu *cron* po uplynutí stanoveného časového intervalu. Obsah tabulky *crontab* může vypadat následovně:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * \
/var/www/html/build/cdr_count.inc.php
```

Záznam popisuje pravidelné spouštění skriptu každých 5 minut. Tato doba určuje maximální dobu od uskutečněného volání, než dojde k aktualizaci informací v účtovacím systému. Interval lze zvolit i delší – 5 minut bylo zvoleno pro odladňování systému. Obecný vývojový diagram zpracování záznamů je zobrazen v příloze D

Předvolby

Každé tarifní skupině může být stanovena sada předvoleb. Předvolby jsou ukládány v tabulce *dest_number*. Uložené záznamy předvoleb lze definovat k jednotlivým tarifním skupinám *tarif_mix* pomocí entity *spec_tarif_mix* viz. tab.1.6. Ze závislosti PK se nesmí vyskytovat stejné předvolby v jedné tarifní skupině.

Tab. 1.13: Entita pro uchování řetězce předvoleb(prefixů) v systému – dest_number

Atribut	**Datový typ**	**Index**
idNUMBER	INT	PK,AI
number	Varchar(6)	UQ,NN
last_change	Timestamp	NN
validity	Bool	NN

[26]Relativní cesta od souboru *index.php*. Místo uložení má pouze vliv na nastavení tabulky *crontab* programu *cron* a definici cesty ke konfiguračnímu souboru *db_connect.inc.php*,kde jsou definovány informace pro připojení k databázovému serveru.

- *idNUMBER* – umělý PK;
- *number* – uchovává řetězec předvolby, který musí být unikátní v rámci tabulky;

Trunky

Trunk neboli přenašeč určuje komunikační kanál mezi ústřednami. Pomocí nich lze docílit dosažení volaného v rámci jednoho autonomního systému[27] nebo mezi různými autonomními celky.

Trunky mezi cizími systémy mohou být zpoplatněny jako služba. U protokolu SIP je využíváno principu, kdy se místní ústředna pro cizí ústřednu tváří jako běžný uživatel[28]. Z tohoto důvodu musí být stanoven účet, kterým se trunk k dané ústředně registruje. Tyto účty jsou monitorovány stejně jako ostatní, proto lze stanovit i náklady na provoz jednotlivých trunků.

Tab. 1.14: Entita s informacemi o trunku – *trunk*

Atribut	**Datový typ**	**Index**
idTrunk	INT	PK,AI
Name	Varchar(45)	UQ,NN
HOST	Varchar(45)	NN
description	Varchar(160)	
last_change	Timestamp	NN
validity	Bool	NN
idPROVIDER	INT	FK,NN

- *idTRUNK* – umělý PK;
- *Name* – název, který je indexován jako jedinečný;
- *HOST* – určuje adresu cílového konce trunku;
- *idPROVIDER* – přiřazuje jméno providera koncové ústředny.

V systému se pro trunk stanovují účty uložené v entitě *sup_buddies*. Během přiřazení konkrétního účtu, který je typu *friend* je u tohoto účtu přepsán atribut *context* a *host*. Kontext je přepsán na hodnotu „incomming-calls“, aby bylo možné oddělit pravidla příchozích spojení.[29] Host je přepsán hodnotou zadanou při vytváření

[27] takový systém se skládá z většího počtu ústředen spravující přidružené uživatele...

[28] Mezi ústřednami se tvoří i pevné spoje. V takovém případě Asterisku se jedná o účty typu „peer“

[29] V koniguračním souboru asterisku – *extensions.conf* lze nastavit vlastní politiku příchozích spojení. V práci je konfigurováno vytáčení SIP klapky podle předané hodnoty.

trunku. Dále pak při změně použitého SIP účtu je u nového účtu přepsána hodnotou *host* z entity *trunk*.

K přepisování položky *host* dochází z důvodu, aby měl Asterisk informaci, kam odesílat hovory směrované na daný trunk. Aktualizace databáze je vykonávána v transakčním režimu. V případě chyby v některém kroku jsou všechny zasažené hodnoty navráceny.

Po konfiguraci trunků je nutné jejich přihlášení – registrace k nadřazené ústředně. To je řešeno systémovou entitou *ast_config*, do které jsou zapisovány globální nastavení nejen pro technologii SIP. Toto nastavení nelze uskutečnit pomocí dříve popisované entity *sip_buddies*, která slouží výhradně jednotlivým účtům.

U entity *ast_config* je stejně jako u *sip_buddies* pevně stanovena struktura viz.[19]. Na rozdíl od předchozí entity je nutné pro aplikování změn restartovat Asterisk nebo občerstvit nastavení např. pomocí SSH appletu ve webovém rozhraní.

1.6.5 Záznamy o volání – CDR

Záznamy o volání, neboli „call data records“ jsou záznamy automaticky poskytovány Asteriskem po každém hovoru. Standardně jsou zapisovány do souboru typu CSV. V této práci byla provedena modifikace nastavení Asterisku, jež s pomocí dodatečných doplňkových modulů způsobilo ukládání CDR do databáze. V tomto případě do MySQL.

Před vlastním ukládáním záznamů je nezbytné, aby databáze obsahovala cílovou entitu *cdr*, do níž budou záznamy vkládány. Formát entity je zobrazen v tab.1.15.[7]

- *id* – PK k identifikaci konkrétní položky;
- *calldate* – hodnota obsahuje časový údaj pokusu spojení;
- *clid* – hodnota obsahuje identifikační hodnotu volajícího, která může být modifikována;
- *src* – účet volajícího;
- *dst* – účet volaného;
- *dcontext* – kontext volaného;
- *channel* – obsahuje informace o použitém příchozím kanále. Hodnota je zpracovávána při účtování hovoru. Obsahuje informace o použité technologii a účtu volajícího. Při příchozím volání lze identifikovat, jestli se jedná o externí volání.
- *dstchannel* – obsahuje informace o použitém odchozím kanále. Společně s hodnotou atributu *dst* lze rozeznat přímé spojení nebo použití trunku.
- *lastapp* – poslední příkaz Asterisku před ukončením hovoru;
- *duration* – celkový využitý čas Asterisku. Započítáván od začátku do konce komunikace s Asteriskem.

Tab. 1.15: Entita pro záznamy o hovorech – cdr

Atribut	**Datový typ**	**Index**
id	INT	PK,AI
calldate	Datetime	NN
did	Varchar(80)	NN
src	Varchar(80)	NN
dst	Varchar(80)	NN
dcontext	Varchar(80)	NN
channel	Varchar(80)	NN
dstchannel	Varchar(80)	NN
lastapp	Varchar(80)	NN
lastdata	Varchar(80)	NN
duration	INT	NN
billsec	INT	NN
disposition	Varchar(45)	NN
amaflags	INT	NN
accountcode	Varchar(20)	NN
uniqueid	Varchar(32)	NN
userfield	Varchar(255)	NN

- *billsec* – čas spojení klienta s požadovaným cílem resp. účtovaný čas;
- *disposition* – hodnota příznaku z *ANSWERED, NO ANSWER, BUSY, FAILED*;
- *amaflags* – příznaky nastavované Asteriskem podle konfigurace;
- *accountcode* – nastavení doplňujících informací k účtu;
- *uniqueid* – Asterisk je po úpravě doplňků schopen generovat jedinečné číselné řetězce;[7]
- *userfield* – volitelné pole.

K jednotlivým záznamům přidává účtovací systém vlastní informace. Vkládané informace slouží pro režijní účely a optimalizaci některých procesů. Určité informace jsou zpracovány z aktuálně vkládaného záznamu do entity *cdr*. Zpracují se informace o volajícím, směru volání – při použití příchozího volání přes trunk se zapisuje odlišná hodnota. Algoritmus zpracování CDR záznamů je zobrazen v příloze B.

V další fázi jsou hodnoty zpracovány a aktualizovány skriptem *cdr_count.inc.php*, jehož princip je znázorněn v příloze D. Skript je popsán v kap.1.6.4. Skript aktualizuje atributy ceny *price*, způsob aktuální platby tarifní skupiny (předplaceno nebo

fakturováno) a odkaz na položku entity *tarif*, kde je uložena cenová mapa. Redundance tohoto záznamu je z důvodu zpětného dohledání záznamů, kterými byl hovor fakturován. Předpokládá se pravděpodobná modifikace tarifní skupiny.

Tab. 1.16: Režijní informace k záznamům entity *cdr – call_records*

Atribut	**Datový typ**	**Index**
idCALL_RECORDS	INT	PK,AI
idCDR	INT	FK
id		
idTECHNOLOGY		
validity	Bool	NN
last_change	Timestamp	NN
price	FLOAT	
checked	Bool	NN
idTARIF	INT	
idDIRECTION	INT	FK
idPAYTYPE	INT	

- *idCALL_RECORDS* – PK určující jedinečnost;
- *idCDR* – FK tvořící relaci do tab.1.15;
- *id,idTECHNOLOGY* – PK tvořící relaci do tab.1.10
- *price* – obsahuje hodnotu vypočtené ceny hovoru;
- *checked* – určuje, zda byl u záznamu proveden výpočet ceny;
- *idTARIF* – zde může být vložen identifikátor účtovacího tarifu.
- *idDIRECTION* – identifikátor odchozího volání nebo příchozího hovoru skrz trunk;
- *idPAYTYPE* – obsahuje typ platby, pod kterou patřila tarifní skupina účtující hovor.

1.7 Uživatelská oprávnění

V aplikaci se vychází z možnosti dvou uživatelských rolí – *administrátor* a *uživatel*. Pro každou roli je udělen přístup k vybraným funkcím. Běžní uživatelé mohou mít např. přístup jen k omezeným funkcím.

Systém poskytuje uživateli funkce vyjmenované v kap.1.3.2. Aby toho bylo možné docílit, je v databázi vytvořená logická struktura.

Struktura je složena z entit :

- *user_group*;
- *spec_user_group*;
- *spec_group_rights*;
- *functions*;
- *functions_sequences*;

Webové rozhraní je schopno uživateli poskytnout povolené funkce a vygenerovat i menu závislé na stavové informaci. Stavová informace uchovává hodnotu indexu aktuálního otevřeného formuláře[30].

Tab. 1.17: Entita s informacemi o skupině – *user_group*

Atribut	**Datový typ**	**Index**
idUSERGROUP	INT	PK,AI
Name	Varchar(45)	UQ,NN
description	Varchar(255)	
last_change	Timestamp	NN
validity	Bool	NN

Tab. 1.18: Entita propojující uživatele se skupinou – *spec_user_group*

Atribut	**Datový typ**	**Index**
idUSER	INT	PK,FK
idUSERGROUP		
validity	Bool	NN
last_change	Timestamp	NN

Tab. 1.19: Entita propojující skupinu s funkcemi systému – *spec_group_rights*

Atribut	**Datový typ**	**Index**
idUSERGROUP	INT	PK,FK
idFUNCTIONS		
validity	Bool	NN
last_change	Timestamp	NN

[30]V této souvislosti je zamýšlen ekvivalent k PHP „modulu“ či „funkci“ systému.

Tab. 1.20: Entita s informacemi funkcí systému – *functions*

Atribut	**Datový typ**	**Index**
idFunctions	INT	PK,AI
Name	Varchar(32)	NN,UQ
description	Varchar (255)	
validity	Bool	NN
last_change	Timestamp	NN
scriptname	Varchar(80)	NN

Tab. 1.21: Entita s informacemi návaznosti mezi funkcemi – *functions_sequences*

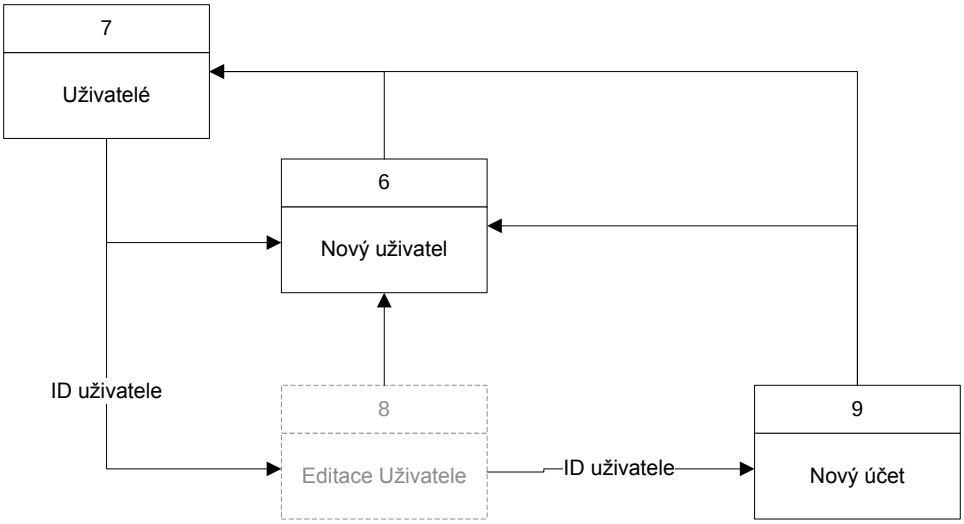
Atribut	**Datový typ**	**Index**
idFunctions	INT	PK,FK
idnextFUNCTIONS		
validity	Bool	NN
last_change	Timestamp	NN
submenu	TINYINT(2)	NN

- *idUSERGROUP* – tvoří PK či FK k určení skupiny;
- *Name* – obsahuje název zobrazovaný v systému. Název tarifní skupiny je zobrazován při určení skupiny uživatele a název funkce je zobrazován v menu webového rozhraní systému.
- *idUSER* – je FK k entitě s informacemi o uživateli;
- *idFUNCTIONS* – obsahuje PK klíč entity *functions* nebo FK entity *functions_sequences*;
- *idnextFUNCTIONS* – zastává stejnou funkci jako *idFUNCTIONS* v entitě *functions_sequences*;
- *scriptname* – obsahuje relativní cestu[31] ke skriptu provádějící určitou operaci;
- *submenu* – je atribut, který určuje hierarchii v menu systému. Může nabývat hodnot 0-2 viz. kap.1.7.1.

1.7.1 Tvorba a generování menu

Generování menu je závislé na vazbách entit z předchozí kap.1.3.2. Při tvorbě menu hraje důležitou roli entita *functions_sequences*. Popisuje vazby mezi funkcemi systému.

[31] vztaženou ke zdrojovému dokumentu *index.php* resp. *admin_page.php*



Obr. 1.4: Diagram datových toků pro prácí s uživatelskými účty.

Položka *submenu* obsahuje informaci, zda nebo v jaké úrovni bude zobrazena položka menu. Může se jednat o tři případy:

1. jedná se o položku menu zobrazovanou v nejvyšší hierarchii (hodnota 0) – zahrnuje množinu jednotlivých funkcí s logickou souvislostí;
2. položky dceřinné (hodnota 2) – jsou zobrazeny po kliknutí na mateřskou položku;
3. položky zobrazené při splnění podmínky vstupu přes nastavenou funkci (hodnota 1) – tyto položky na vstupu požadují určitou vstupní hodnotu;

Obrázek 1.4 zobrazuje diagram datových toků uspořádaných procesů pro práci s uživatelským nastavením. Obsah v entitě *functions_sequences* zobrazuje tab.1.22.

- 1. a 3.řádek – položka 6 bude zobrazena z položky 6 i 7 jako podřízená položka;
- 2., 4., 6., a 8.řádek – z těchto položek bude položka 7 zobrazena v hlavních položkách menu v 1. úrovni;
- 3.řádek – položka 6 je podřazenou položkou položky 7;
- 5., 7. a 9.řádek – určují, že položka 9 bude zobrazena ve 2.úrovni menu, jestliže se uživatel bude nacházet ve funkcích 8 nebo 9 a současně se cílová položka 9 nachází v položkách dostupných ze 7 položky 1.úrovně.

Tab. 1.22: Příklad tabulky definující přechody mezi funkcemi systému

řádek	**idFUNCTIONS**	**idnextFUNCTIONS**	**submenu**
1.	6	6	2
2.	6	7	0
3.	7	6	2
4.	7	7	0
5.	7	9	1
6.	8	7	0
7.	8	9	1
8.	9	7	0
9.	9	9	1

Tímto způsobem je vytvořené celé menu systému. Výhoda je ve vysoké přizpůsobitelnosti, ovšem za cenu vyšší složitosti. Princip mechanismu spočívá v rekurzivní relaci entity *functions* přes *function_sequences*.

2 DOPŇUJÍCÍ INFORMACE K ÚČTOVACÍMU SYSTÉMU

2.1 Popis ovládání a funkce rozhraní systému

Tato kapitola slouží jako uživatelský manuál k ovládání a popisů funkcí webového rozhraní systému. Některé hodnoty např. loginů a hesel jsou vztaženy k testovacímu pracovišti na ústavě telekomunikací v laboratoři č.427. Adresa virtuálního serveru je 192.168.10.9.

Systém nejprve vyzve uživatele k přihlášení. Lze použít např. login „xdepia00“ a heslo „asdf“. Tento účet patří k administrátorské skupině. Po přihlášení se lze v systému přepnout do rozhraní pro běžné uživatele „Uživatel“, které slouží pro přehled aktuálně přihlášeného uživatele. Další možností je zvolit administrátorský režim „Administrátor“, ve kterém lze provádět změny konfigurace. Do vytvořeného účtu s uživatelským oprávněním se lze přihlásit pomocí „xkoblich“ „asdf“. Ve většině případů mají klapky stejné heslo jako jejich název. Zejména řada 6xx.

Pro odhlášení ze systému je vytvořen odkaz se zobrazením loginu aktuálního uživatele.

2.1.1 Uživatelé

Tato položka menu zastřešuje funkce pro práci s uživatelskými účty do systému. Po jejím výběru je zobrazen filtr pro hledání uživatele. Současně se nabídka menu rozšíří o položku *Nový uživatel*, kde lze založit nový účet pro přihlášení do systému.

Nový uživatel

Pod touto položkou je zobrazen formulář pro sběr základních informací o uživateli. Systém kontroluje jejich hodnotu. Položkou „Uživatelská skupina“ lze nastavit oprávnění účtu.

Editace uživatele

Tento formulář je zobrazen po volbě konkrétního uživatele z filtru uživatelů. Zde je možné editovat parametry zadané při vytváření účtu.

Současně je zobrazen seznam SIP účtů přiřazených k vybranému uživateli. Přes odkaz vybraného účtu lze editovat jeho parametry.

Nový účet

Položka menu je zobrazena, je-li z filtru zvolen konkrétní uživatel, ke kterému bude účet vytvořen a přidělen. Při vytváření SIP účtu jsou zvoleny jen základní parametry z mnoha dostupných(viz.lit.[13]). Lze zvolit typ účtu, ale v tomto systému je doporučeno využívat pouze „friend“. V nabídce „Tarifní skupina“ musí být zvolena položka, kterou bude účet tarifikován. „IP/Název klienta“ je doporučeno zatrhnout „dynamická“, pokud administrátor nemá v úmyslu speciální využití tohoto účtu. Hodnota „Kontext“, pokud nebyly provedeny změny konfiguračního souboru */etc/asterisk/extensions.conf*[1] by měla být nastavena na „default“.[2]

2.1.2 Tarify

V této skupině funkcí je možné vytvářet a upravovat tarifikační pravidla. Filtr lze přepnout pro vyhledávání tarifních položek nebo tarifní skupiny. Význam těchto položek je popsán v kap.1.6. Po vstupu do těchto položek je možné editovat jejich parametry podobně jako u uživatelů.

Tarifní intervaly

Pod touto položkou je možné do systému zadávat tarifikační hodnoty, ze kterých je možno vybírat při vytváření nebo editaci tarifní položky.

Nová položka

Specifikuje účtovací mapu pro pravidla tarifní skupiny.

Nová skupina

Vytváří pravidla, podle kterých bude vybírána tarifní mapa pro specifikované směry hovorů a uživatele – viz. *Nový účet*. Na laboratorním pracovišti je vytvořena skupina, která směruje hovory začínající na „01“, např.„01805“ k nadřazené ústředně. Té je předán parametr zkrácený o prefix pravidla skupiny. Tzn., že u nadřazené ústředny bude požadavek na číslo „805“.

Předvolby

Obdobně jako *Tarifní intervaly* jsou pomocí předvoleb nastavovány hodnoty, které se uplatní při výběru pravidel tarifní skupiny.

[1] Konkrétní umístění záleží na distribuci OS

[2] V kontextu „default“ je nastaveno předávání řízení vytáčení do AGI skriptu.

2.1.3 Trunky

Tato sekce obsahuje nástroje pro správu trunků. Je zde přehled současně vytvořených trunků, kde se nalezá vždy alespoň účet Local. Tento účet je virtuální a je využit v jiných částech systému pro vytváření pravidel a účtování místních[3] hovorů. Hodnoty parametrů trunku „Local“ mohou být jakkoliv vyplněny.

U nachystaného pracoviště je vytvořen i trunk „SIP_TRN_0“, který vytváří spojení s nadřazenou ústřednou přes definovaný účet. Při editaci trunku, která vede k jakékoliv změně, je vždy požadováno heslo zvoleného účtu. Heslo slouží pro kontrolu a zabraňuje zneužití jiných účtu pro směrování hovorů, ke kterým heslo není známo. Zvolený účet je přepnut do kontextu „incomming-calls“ pro příjem hovorů. Současně je změněna hodnota *host* podle nastavení trunku.

Nový trunk

Slouží k vytvoření nového trunku. Při uložení jsou u vybraného účtu změněny hodnoty *context* a *host* stejně jako při editaci. Dále je vytvořen záznam v tabulce *ast_config*. Tento záznam při nejbližším restartu nebo aktualizaci nastavení Asterisku odesílá požadavek registrace k ústředně s adresou zadanou v „HOST“.

2.1.4 Poskytovatelé

V této sekci jsou vytvářena jména poskytovatelů, která slouží pro informaci k jednotlivým trunkům.

2.1.5 Účty asterisk

Zde jsou zobrazeny veškeré zřízené SIP účty Asterisku. Po výběru zvoleného účtu je zobrazeno konkrétní nastavení s možností editace. Jsou zde zobrazeny přiřazené tarifní skupiny, z nichž je barevně odlišena aktuálně platná. Dále je možné účet přiřadit konkrétnímu uživateli nebo zrušením při volbě „vyberte“. V tomto případě je účet přiřazen účtu „System“.

2.1.6 Faktury

Tato položka skrývá údaje k fakturaci jednotlivých účtu s přiřazenou tarifní skupinou. Po zvolení jsou faktury zobrazené v rámci časových období stanovených na 1 měsíc. Po zvolení jednotlivých položek je k těmto obdobím zobrazen výpis hovorů.

[3] V rámci této ústředny.

U hovorů, u kterých proběhlo spojení,resp. byl vytvořen tarif popisující proběhlý požadavek, bude vypočtena cena hovoru.[4]

Ve výpisu volání je možné zobrazit příchozí nebo odchozí volání pro zvolenou klapku. Pomocí zatrhávacích polí lze zobrazit hovory s nulovou délkou. Dále je možné zrušit omezení dříve vybraným obdobím.[5]

V případě že se jedná o fakturovaný tarif resp. v daném období bylo alespoň jedno takové volání, v administrátorském výpisu je zobrazena možnost zaplacení. Zaplacené faktury jsou v položce „Faktury“ zobrazeny jen v případě zrušení volby „Jen nezaplacené“.

Dobití kreditu

Pomocí této funkce lze dobíjet kreditní hodnotu k jednotlivým účtům. Standardně i účet s nastavenou tarifní skupinou se způsobem platby „fakturou“ má k dispozici kreditní účet. Ten je využit např. při změně způsobu platby v tarifní skupině.

2.1.7 SSH

Na tomto místě je Java applet SSH terminálu pro, připojení a konfiguraci systému. Pro jeho funkci je nezbytné na klientské stanici instalovat Java Runtime Environment. Bez něj prohlížeč nebude schopen spustit applet. Pomoci appletu se lze na testovací pracoviště přihlásit pomocí jména „root“ a hesla „aaa“.

2.1.8 Účty

Tato funkce v režimu „Uživatel“ poskytuje přihlášené osobě informace o jednotlivých přiřazených účtech. Po výběru konkrétního účtu je zobrazen výpis volání.

2.1.9 Přehled statistik

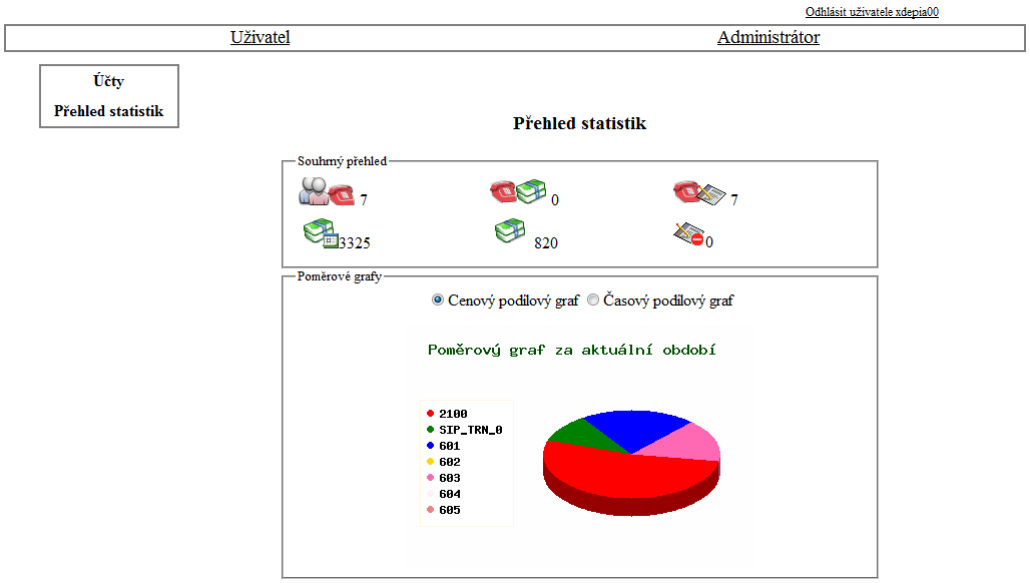
Tato položka obsahuje souhrnné informace o přidělených účtech viz. obr.2.1.10

2.1.10 Vytočit účet

Tato funkce slouží k inicializaci spojení pomocí webového rozhraní. Přihlášeným uživatelem je zvolen jeden z vlastních účtů a zadáno číslo volaného účtu. Výběr volaného lze pomocí zadáním čísla nebo výběrem z účtů obsažených v tomto systému a přidělených konkrétnímu uživateli.

[4]Maximálně do doby pravidelného spouštění skriptu pro výpočet ceny.

[5]Pro vybraný účet je zobrazená kompletní historie.



Obr. 2.1: Ukázka přehledu statistik uživatele.

2.2 Souhrnný přehled instalace systému

Systém je možné provozovat za předpokladu, že jsou instalovány a nastaveny veškeré potřebné komponenty.

2.2.1 Příprava asterisku

Hlavní komponentou je program Asterisk, který tvoří PBX ústřednu. Popis instalace je v kap.1.2.2 a 1.2.3. Problém se může vyskytnout při kompilaci zvukových kodeků, proto jsou doporučeny instalovat základní kodeky *Alaw a gsm*. Tyto kodeky jsou předdefinované novým SIP uživatelům.

Doplňující moduly v podobě *Asterisk-addons* jsou instalovány pro podporu databázového serveru. Existuje několik typů databázových ovladačů, ale systém je tvořen pro ovladač k MySQL. Ovladači je nutné definovat přihlašovací pravidla k databázi MySQL serveru. Autentizační údaje k databázi pro zapisování CDR záznamů jsou uloženy v souboru */etc/asterisk/cdr_mysql.conf*, kde jsou položky:

```
[global]
hostname=localhost    ;adresa databázového serveru
dbname=mydb           ;jméno databáze v databázovém serveru
table=cdr             ;název tabulky pro ukládání záznamů,
                      ;struktura musí odpovídat tab. 1.14
password=<heslo>      ;heslo účtu na MySQL serveru s právem
                      ;zápisu do databáze 'mydb'
```

```
user=<login>          ;login účtu s právem zápisu do databáze
;port=3306            ;použit v případě, není-li
                      ;MySQL součástí stejného hardware
sock=/var/lib/mysql/mysql.sock   ;místní soket využitý
                      ;pro komunikace s MySQL
```

Obdobné nastavení obsahuje konfigurační soubor pro správu klientů SIP z databáze */etc/asterisk/res_mysql.conf*. Rozdíl je v nastavení `table=sip_buddies`. Následně je nutné upravit soubor */etc/asterisk/extconfig.conf*. Musí obsahovat řádky:

```
[setting]
sip.conf => mysql,mydb,ast_config   ;ast_config je systémová tabulka
                                    ;obsažena ve vytvořené databázi
                                    ;pro nastavení obecných parametrů
                                    ;konfiguračních souborů
sipuser => mysql,mydb,sip_buddies
sippeers => mysql,mydb,sip_buddies  ;přepne na vyhledávání účtů z db
```

Pro funkci hlídání odchozích hovorů a odečítání kreditu je nutné zkopírovat skript *tarif_check.php* a složku s knihovnami PHPAGI *include* do adresáře */var/lib/asterisk/agi-bin/*. Následně soubor */etc/asterisk/extensions.conf* musí obsahovat:

```
[default]
exten => _X.,1,AGI(tarif_check.php)    ;předá vytáčecí pravidla
                                       ;AGI skriptu
exten => _X.,n,Goto(S-${DIALSTATUS},1)
exten => S-NOANSWER,1,Playback(vm-noobodyavaiil);
exten => S-CHANUNAVAIL,1,Playback(pbx-invalid);
exten => S-CONGESTION,1,Playback(vm-noobodyavaiil);

[incomming-calls]
exten => _X.,1,Dial(SIP/${EXTEN})      ;pravidlo pro příchozí hovory,
                                       ;lze přizpůsobit podle potřeby
```

Použité názvy zvukových souborů k přehrávání závisí na nainstalované sadě.

V dalším kroku je potřeba oživit rozhraní AMI pro vytáčení z webové aplikace. V konfiguračním souboru */etc/asterisk/manager.conf* je obsaženo:

```
[general]
enabled = yes                          ;aktivuje AMI
```

```
port = 5038                                ;port na kterém Asterisk bude
bindaddr = 127.0.0.1                       ;adresa, na které bude naslouchat
[billing]                                  ;definice účtu
sescret=billing                            ;definice hesla
deny=0.0.0.0/0.0.0.0                       ;zakáže komunikaci na všech
permit=127.0.0.1/255.255.255              ;povolí místní komunikace
                                           ;localhost
read=system,call,log,verbose,command,agent,user,all
write=system,call,log,verbose,command,agend,user,all
                                           ;definice oprávnění
```

Jestliže bude požadován jiný účet a jiné heslo, je nezbytné tyto změny upravit i v souboru */var/www/html/click_to_call.inc.php*[6] na řádcích 158 a 159.

2.2.2 Příprava MySQL

Po nainstalování MySQL serveru je nutné vytvořit uživatelský účet k oprávnění vytvoření nové databáze. Možností je vytvořit nejprve pod dostatečně privilegovaným účtem databázi „mydb“ a následně vytvořit účet s možností čtení a zápisu do této databáze.

Privilegovaný účet je použit pro vytvoření struktury databáze a pro přístup Asterisku k vytvořené databázi. Následně vytvořit databázi pomocí přiloženého souboru. Zdrojový soubor databáze lze vytvořit po načtení modelu a jeho exportováním programem MySQL Workbench. Vložení databáze je možné přímo z konsolové řádky Linuxu, jestliže je instalovaný klientský software.

Pro přístup do databáze testovací pracoviště se lze připojit pomocí jména „root“ a hesla „root“.

2.2.3 Příprava Apache

Potřebné informace k instalaci HTTP serveru Apache jsou uvedeny v kap.1.4.1. Do složky */var/www/html* je nutné zkopírovat strukturu souborů webového rozhraní aplikace. Pro přístup k databázi je nutné nakonfigurovat údaje účtu v souboru */var/www/html/build/db_connect.inc.php*, pod nímž se systém přihlašuje do databáze.

[6] tato cesta byla použita při realizaci.

3 ZÁVĚR

Cílem práce bylo vytvořit účtovací a monitorovací systém. Nejprve bylo nutné nastudovat danou problematiku. Ta zahrnovala vlastní zprovoznění systému. Jako OS byl zvolen Centos 5. Na tento systém byl instalován software PBX Asterisk. Verze zvolená pří začátku zpracování této práce byla 1.4.29.1. V této době se jednalo o stabilní verzi oproti přicházejícím verzím 1.6.

Po instalaci Asterisku byli postupně nasazovány o zkoušeny funkce potřebné pro účtování hovorů. Zejména se jednalo o konfiguraci, která nastavila ukládání záznamů o volání(CDR) do databáze MySQL. Další doplňující architekturou byla funkce „RealTime“, která umožňuje načítání konfiguračních dat Asterisku z databáze MySQL.

Aby byla možná interakce systému s uživatelem, bylo vytvořeno webové rozhraní. Pro funkci tohoto rozhraní byla nutná implementace HTTP serveru Apache včetně nastavení pro podporu skriptovacího jazyka PHP. Současně s Asteriskem a serverem Apache bylo nutné pro správnou funkci doplnit i databázový server MySQL.

Značná část diplomové práce se zabývá návrhem a tvorbou databáze. Na základě databáze a nastavení Asterisku byly vytvářeny PHP skripty webového rozhraní, PHP skript pro účtování hovorů a PHPAGI skript pro řízení dialplánu Asterisku. Pomocí webového rozhraní jsou nastavována tarifikační pravidla, vč. možnosti konfigurace některých částí Asterisku, např zřizování účtů, konfigurace trunků....
Na základě těchto poskytnutých dat je aplikován PHP skript pro stanovení ceny hovorů, který je opakovaně spouštěn OS. Součásti účtování je i řídící skript PHPAGI, který zabraňuje spojení účtům, které nemají vytvořena pravidla pro zaúčtování daného požadavku. Ve webovém rozhraní je implementován Java applet pro možnost připojení k hostujícímu OS a jeho konfiguraci.

Další z možností uživatelského rozhraní je zobrazit klientům, se zřízeným účtem v tomto systému, statistiky a výpisy hovorů. Doplňující službou je možnost vytáčet klapky pomocí implementované funkce systému a nastaveného AMI rozhraní Aterisku.

Veškeré funkce byli postupně aplikovány na testovací pracoviště v laboratoři č.427 a konzultovány s vedoucím práce.

LITERATURA

[1] V. MEGGELEN, Jim, SMITH, Jared, MADSEN, Leif. *Asterisk : The Future of Telephony.* Mike Loikides; Colleen Gorman. 1st edition. Sebastopol : O'Rreilly Media, 2005. 376 s. Dostupné z URL: <http://www.cyberciti.biz/tips/download-asterisk-howto-tutorial-book.html>. ISBN 0-596-00962-3.

[2] SIMIONOVICH, Nir. *Asterisk Gateway Interface 1.4 and 1.6 Programming : Design and develop Asterisk-based VoIP telephony platforms and services using PHP and PHPAGI.* Edited by Rashmi Phandis, Dhiraj Chandiramani, Gaurav Datar; Sumathi Shridhar. 1st edition. Birmingham : Packt Publishing, 2009. 200 s. ISBN 987-1-847194-46-6.

[3] KADLEC, Josef. *Obecné pojednání o programovacích jazycích.* Linuxsoft [online]. 2004, č. 1 [cit. 2009-10-10]. Dostupný z URL: <http://www.linuxsoft.cz/article.php?id_article=268>

[4] ADAMEC, L., et al. *Perl referenční příručka ke skriptovacímu jazyku : Charakterisktika Perlu* [online]. 2007 [cit. 2008-05-26]. Dostupný z URL: <http://perl.lukada.cz/charakteristika.html>

[5] OPPEL, Andrew. *Databáze bez předchozích znalostí : průvodce pro samouky.* Jiří Matoušek; David Krásenský. 1. vyd. Brno : Computer Press, 2006. 320 s. ISBN 80-251-1199-7.

[6] ŠEDA, Miloš. *Databázové systémy : doplňující text ke konzultacím v 3. ročníku kombinovaného bakalářského studia oboru Aplikovaná informatika a řízení.* [s.l] : [s.n.], 2002. 76 s. Dostupný z URL: <http://www.uai.fme.vutbr.cz/~mseda/DBS02_BS>

[7] IMADSEN. *Voip-Info.org : Asterisk cdr mysql* [online]. 2009, Last modification 20th November 2009 [cit. 2009-12-05]. Dostupný z URL: <http://www.voipinfo.org/wiki/view/Asterisk+cdr+mysql>

[8] Digium. *THE OPEN SOURCE TELEPHONZ PROJECT* [online]. 2009 [cit. 2009-12-05]. Dostupný z URL: <http://www.asterisk.org/>

[9] OEJ, VOIPON. *Voip-Info.org : VoIPgateways* [online]. 2009, Last modification 30th November 2009 [cit. 2009-05-06]. Dostupný z URL: <http://www.voipinfo.org/wiki/view/VoIP+Gateways>

[10] KOUBEK, Martin. *Zracování a monitoring tarifikace pro systém Asterisk.* Praha, 2007. 80 s. ČVUT v Praze. Fakulta elektrotechnická. Vedoucí doplomové práce Ing. David Jandera. Dostupný z URL: <http://koubek.ic.cz/asterisk_billing.pdf>

[11] Digium. *Asteriskguru.com : Installation on SuSE* [online]. c2006 [cit. 2009-12-06]. Dostupný z URL: <http://www.asteriskguru.com/tutorials/asterisk_installation_compilation_suse.html>

[12] WILIE, JOHNLANGE. *Voip-Info.org : Asterisk Linux SuSE* [online]. 2008 [cit. 2009-12-06]. Dostupný z URL: <http://www.voip-info.org/wiki/view/Asterisk+Linux+SuSE>

[13] SADEESH. *Iterasi : Asterisk Realtime with MySQL.* [online]. 2006 [cit. 2009-11-25]. Dostupný z URL: <http://www.iterasi.net/openviewer.aspx?sqrlitid=yw2mi6aaw0sgerjel6-yga>

[14] *Licences : The Apache Software Foundation* [online]. The Apache Software Foundation, c2010 [cit. 2010-05-25]. Licences. Dostupný z URL: <http://www.apache.org/licenses/>

[15] LOPAŠOVSKÝ, Tomáš. *ROOT.CZ* [online]. 13.3.2001 [cit. 2010-05-25]. Malý průvodce konfigurací Apache. Dostupný z URL: <http://www.root.cz/clanky/maly-pruvodce-konfiguraci-apache-2/>

[16] SUCHÝ, Jakub. *LOGiOS : Informační technologie* [online]. 2009 [cit. 2010-05-25]. Koncept LAMP. Dostupný z URL: <http://linux-meeting.logios.cz/dl/webserver-na-linuxu.pdf>

[17] OEJ, *Voip-Info.org : Asterisk sip canreinvite* [online]. 2010 Last modification on Fri 26 of Mar 2010 [cit. 2009-12-06]. Dostupný z URL: <http://www.voip-info.org/wiki/view/Asterisk+sip+canreinvite>

[18] OEJ, JUSTRUMOURS *Voip-Info.org : Asterisk sip nat* [online]. 2009 Last modification on Tue 14 of Jul 2009 [cit. 2010-04-05]. Dostupný z URL: <http://www.voip-info.org/wiki/view/Asterisk+sip+nat>

[19] UTDRMAC, *Voip-Info.org : Asterisk RealTime Static* [online]. 2009 Last modification on Mon 26 of Oct, 2009 [cit. 2010-05-06]. Dostupný z URL: <http://www.voip-info.org/wiki/view/Asterisk+RealTime+Static>

[20] JAKEL, Milan. *Interval.cz* [online]. 2001 [cit. 2010-04-25]. Instalujeme MySQL na Linux. Dostupný z URL: <http://interval.cz/clanky/instalujeme-mysql-na-linux/>

[21] OEJ, *Voip-Info.org : Asterisk AGI* [online]. 2009 [cit. 2009-12-06]. Dostupný z URL: <http://www.voipinfo.org/wiki/view/Asterisk+AGI>

[22] OEJ, CERVAJS *Voip-Info.org : Asterisk AGI* [online]. 2010 [cit. 2010-05-03]. Dostupný z URL: <http://www.voip-info.org/wiki/view/Asterisk+manager+API>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AGI rozhraní Asterisku – Asterisk Gateway Interface

AMI Asterisk Manager Interface

ASP Active Server Pages

CDR záznam o hovoru vytvořený Asteriskem – Call Data Records

CGI protokol pro propojení externí aplikace s web serverem – Common Gateway Interface

CLI příkazová konzole – Command Line Interface

CLR Common Language Runtime

CSV hodnoty oddělené čárkami – Comma-Separated Values

DT Datové typy – Data Type

ER Entitně relační diagram

FK Cizí klíč – Foreign-Key

GPL všeobecná veřejná licence – General Public License

GUI grafické uživatelské rozhraní – Graphical User Interface

HTML značkovací jazyk pro hypertext – HyperText Markup Language

HTTP původně protokol pro výměnu HTML dokumentů – Hypertext Transfer Protocol

HTTPS nadstavba HTTP se zabezpečeným přenosem pomocí SSL nebo TLS – Hypertext Transfer Protocol Secure

IL Intermediate Language

IVR interaktivní hlasový automat – Interactive Voice response

LAMP Linux, Apache, MySQL, PHP

MD5 Algoritmus pro vytváření jedinečného otisku zprávy pevné délky – Message-Digest algorithm 5

MS Microsoft

MSIL Microsoft Intermediate Language

NF Normální forma

NET Network

OS operační systém – Operation System

PBX pobočková ústředna – Private Branch Exchange

PHP hypertextový preprocesor – Hypertext Preprocessor

PK Primární klíč – Primary Key

SIP Protokol pro inicializaci relací – Session Initiation Protocol

SQL strukturovaný dotazovací jazyk – Structured Query Language

SSL vrstva pro zabezpečení socketů – Secure Sockets Layer

STDIN standardní vstup – Standard Input

TDM Časový multiplex – Time Division Multiplex

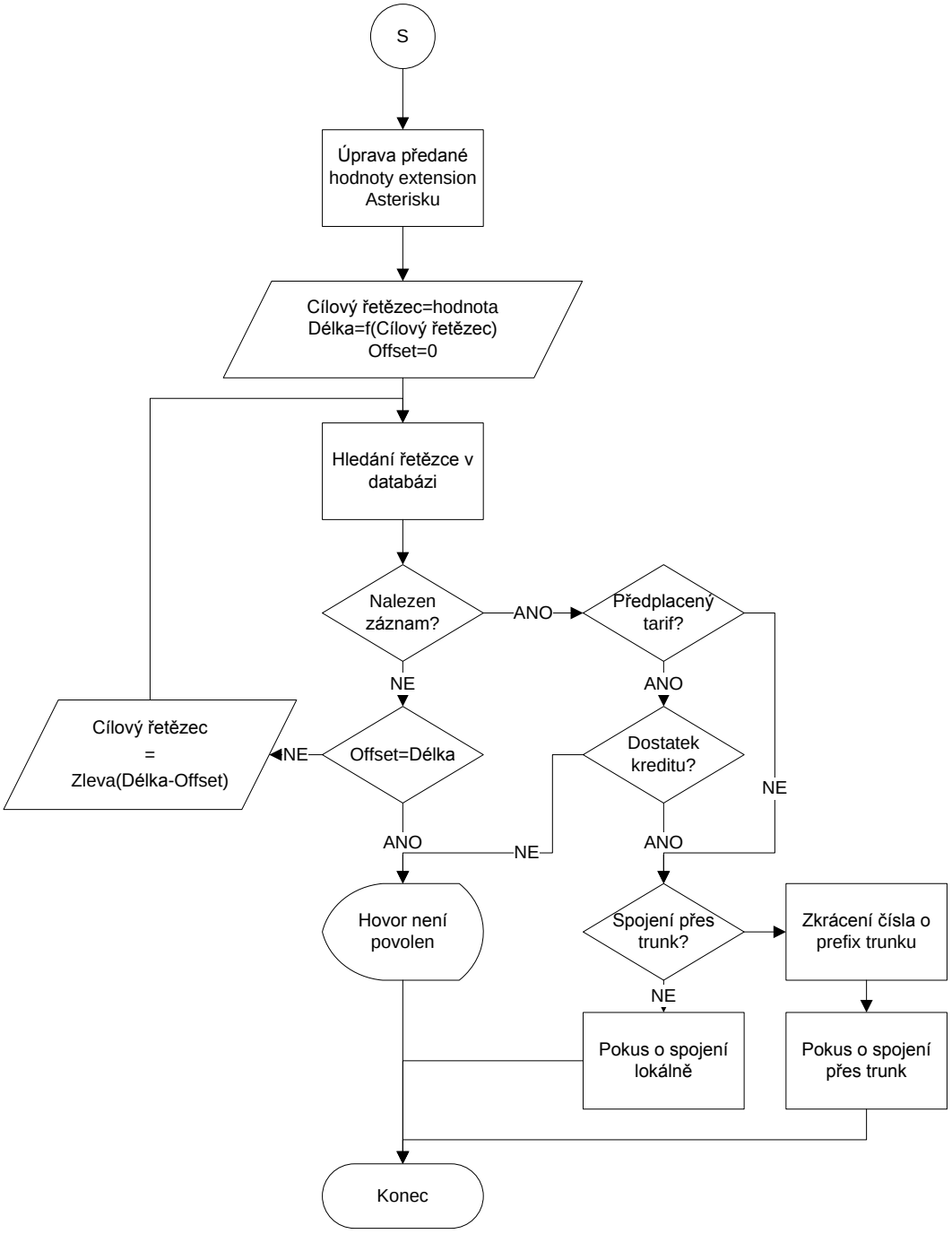
TLS protokol pro zabezpečení dat mezi aplikacemi – Transport Layer Security

VoIP Voice over IP

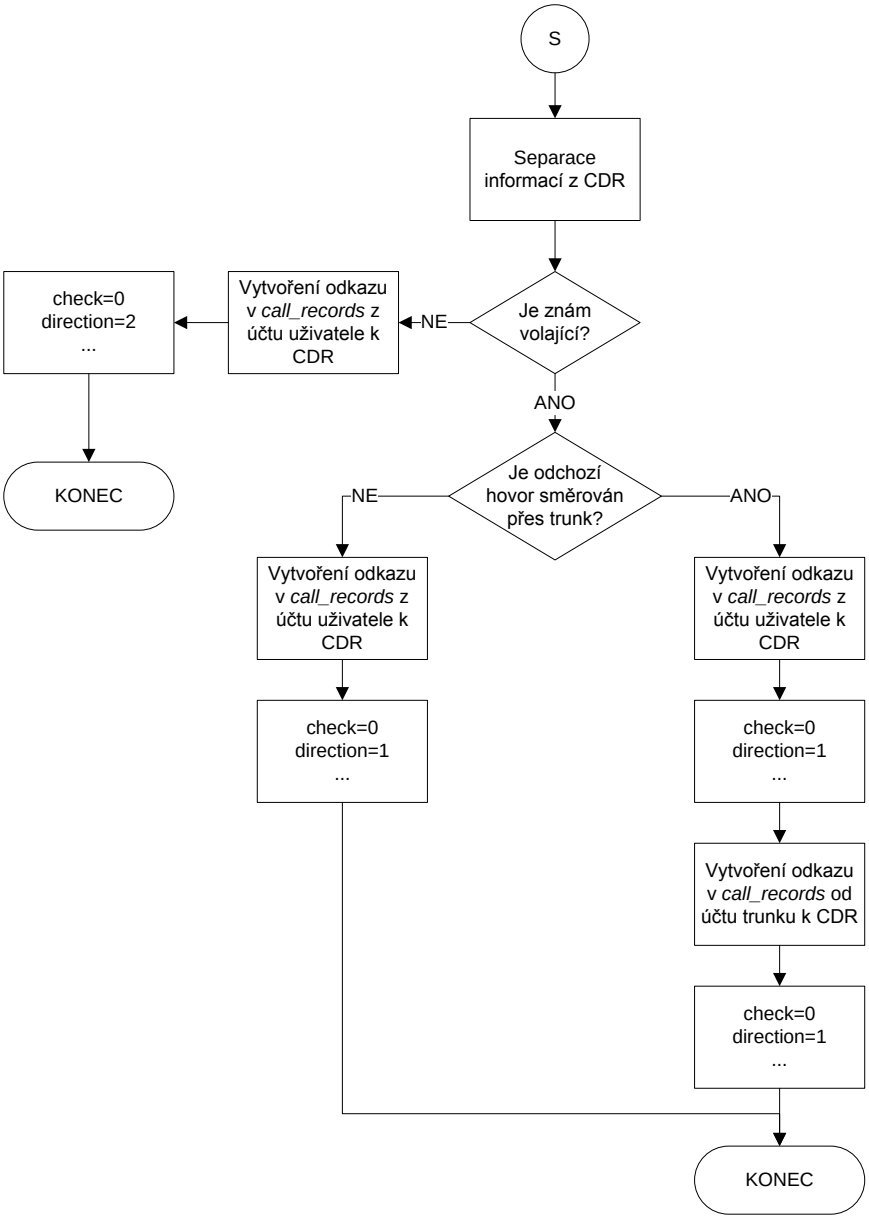
WWW distribuovaný soubor dokumentů v Internetu – World Wide Web

SEZNAM PŘÍLOH

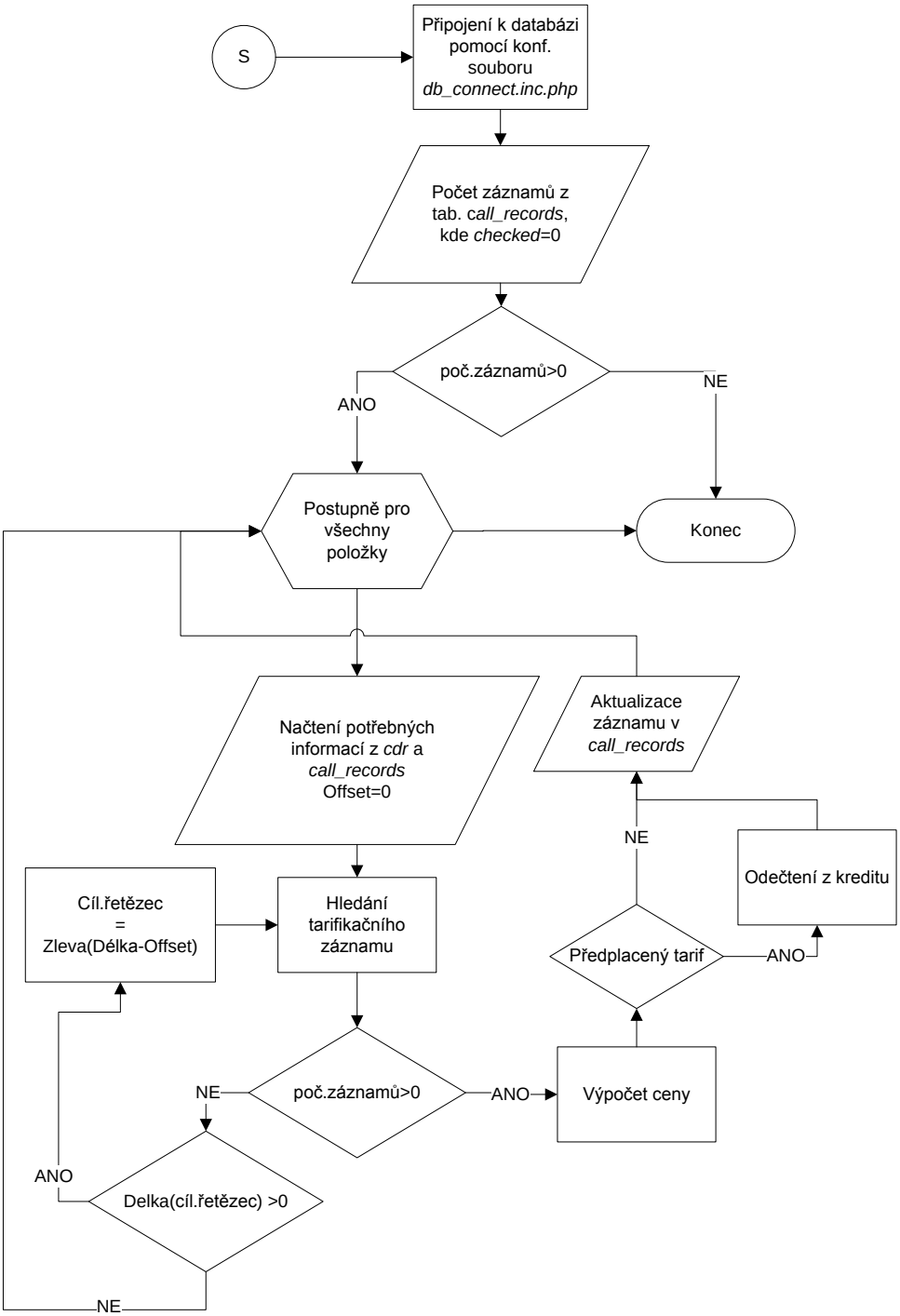
A PRINCIP FUNKCE AGI SKRIPTU V DIAL-PLÁNU ASTERISKU



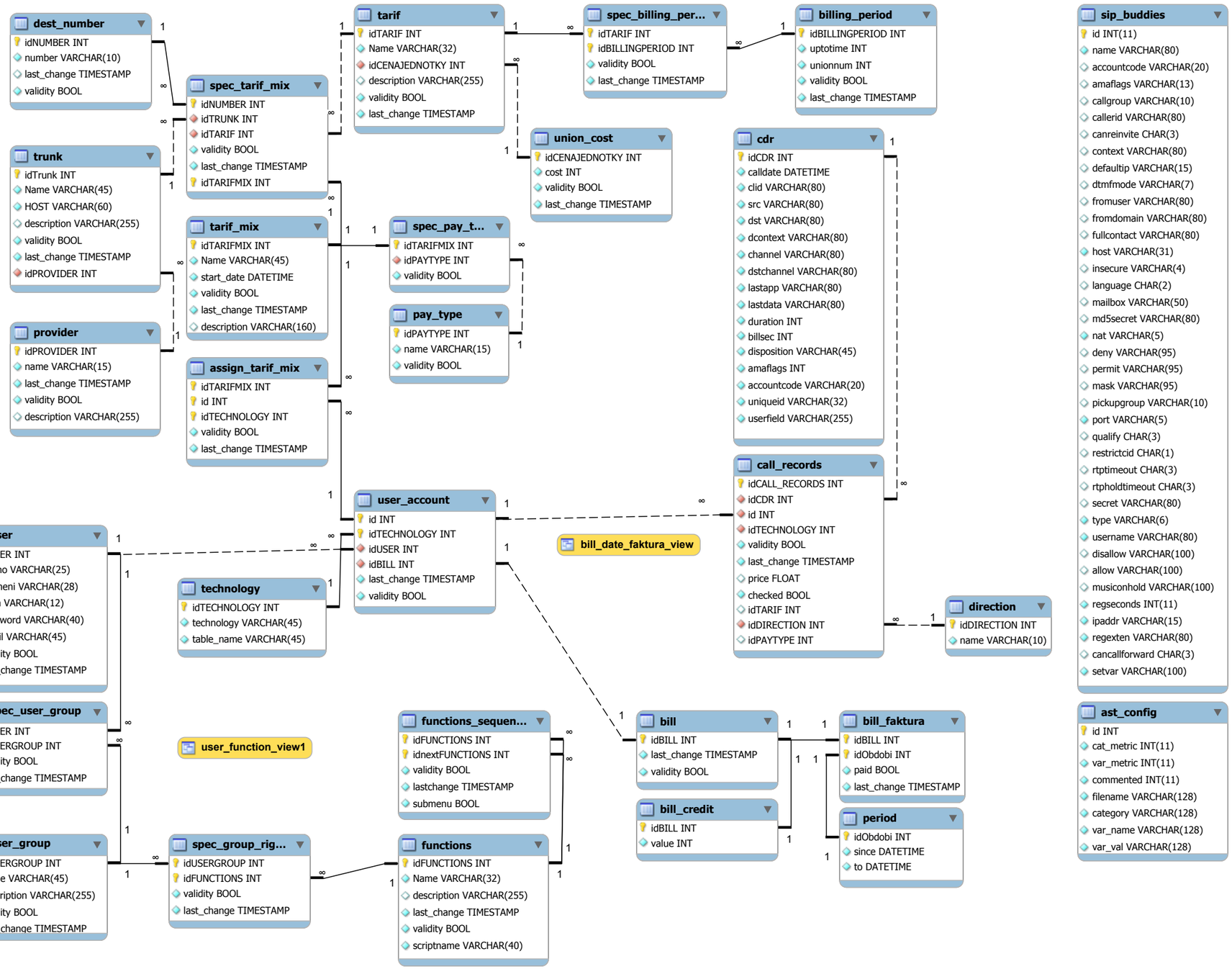
B DATABÁZOVÝ TRIGGER SPOUŠTĚNÝ VLOŽENÍM ZÁZNAMU CDR DAT



C PRINCIP ALGORITMU ÚČTOVÁNÍ HOVORŮ



D ER MODEL DATABÁZE



E Diagram datových toků webového rozhranní