VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vytvoření webové stránky pro využívání webových služeb

Create Web Pages with Web Services

2014 Tomáš Byrtus

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Byrtus**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vytvoření webové stránky pro využívání webových služeb
Create Web Pages with Web Services

Zásady pro vypracování:

Cílem bakalářské práce je naprogramování webové aplikace, která pomocí webových služeb bude komunikovat s telefonní ústřednou Asterisk. Webová aplikace umožní vytvářet spojení mezi uživateli a možnost zasílání DTMF kódů.
1. Úvod do problematiky webových služeb.
2. Popis JAVA API rozhraní v telefonní ústředně Asterisk.
3. Naprogramování webové aplikace pro komunikaci s Asteriskem.
4. Ověření funkčnosti v laboratorních podmínkách.

Seznam doporučené odborné literatury:

Madsen L., Meggelen J.V., Bryant R. *Asterisk: The Definitive Guide* O'Reilly Media 2011
Kalin M. *Java Web Services: Up and Running* O'Reilly Media 2009
Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Nevlud**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. května 2014

.....

Poděkování

Děkuji mému vedoucímu práce panu Ing. Nevludovi za užitečné rady a motivaci během psaní této práce. Poděkování také patří všem mým příbuzným, ale i kolegům ze společnosti Třinecké železárny a.s. za podporu během mých studií.

Abstrakt

Ústředna Asterisk 12 nabízí širokou škálu možností řízení telefonních hovorů. Tématem této práce je otestovat, jaké jsou možnosti zasílání DTMF kódů s využitím webových služeb. Webová služba je stavěna na základě architektury REST s využitím protokolu HTTP a programovacího jazyka PHP. Uživatelské rozhraní včetně provázání s PHP je zajištěno pomocí technologií javascript a HTML. V práci je také zmíněna technologie java, pro zjištění možností knihovny asterisk-java, která se hojně využívá při budování aplikací k rozšíření funkčnosti ústředny.

Klíčová slova

Asterisk, webové služby, REST, SOAP, java, XML, asterisk-java

Abstract

Asterisk 12 PBX provides a wide range of options for controlling phone calls. The theme of this thesis is to test how is it possible to send DTMF codes with using web services. Web service is built on REST architecture using protocol http and programming language PHP. A user interface including binds to PHP is secured by javascript and HTML technology. The thesis will show also java technology for discovering posibilities of asterisk-java library wich is often used to build applications to extend functionality of pbx.

Keywords

Asterisk, web services, REST, SOAP, java, XML, asterisk-java

Seznam zkratek

- CD – Compact disc
- CDR – Call Detail Records
- CEL – Channel Event Logging
- CSS – Cascading Style Sheets
- CLI – Command Line Interface
- DNS – Domain Name Systém
- DTD – Document Type Definition
- DTMF – Dual Tone Multi-Frequency
- DHCP – Dynamic Host Configuration Protocol
- FTP – File Transfer Protocol
- HDD – Hard Disk Drive
- HTTP – Hypertext transfer protokol
- JSON – JavaScript Object Notation
- MIME – Multipurpose Internet Mail Extensions
- ODBC – Open Database Connectivity
- OS – Operační systém
- PHP – Hypertext PreProcessor, původně Personal Home Page
- RAM – Random Access Memory
- REST – Representational State Transfer
- RPC – Remote Procedure Call
- SIP – Session Initiation Protocol
- SMTP – Simple Mail Transfer Protocol
- SOAP – Simple Object Access Protocol
- TCP/IP – Transmission Control Protocol/Internet Protocol
- UDDI – Universal Description, Discovery and Integration
- URI – Uniform Resource Identifier
- URL – Uniform Resource Locator
- VoIP – Voice over Internet Protocol
- W3C – World Wide Web Consortium
- WSDL – Web Services Description Language
- WWW – World Wide Web
- XML – eXtensible markup language

Obsah

Seznam obrázků

Seznam výpisů kódu

1 Úvod

Tato práce vznikla za účelem otestovat možnosti ústředny Asterisk 12 pro zasílání DTMF kódu. DTMF kódy budou zasílány pomocí webových služeb, s využitím REST architektury. Druhá varianta webových služeb SOAP bude popsána spíše teoreticky, se záměrem seznámit se s nimi. Díky architektuře REST, se nabízí jednoduchá možnost použít klasický programovací jazyk PHP v kombinaci s javascriptem, HTML a CSS pro vytvoření komunikace s ústřednou. PHP volím právě pro jeho jednoduchost a čitelnost. V práci je uvedeno, jak by se mohl kód implementovat s využitím dostupných knihoven v jazyce JAVA, ale pouze pro seznámení čtenáře se syntaxí, knihovna je velmi dobře pochopitelná. PHP používám ve verzi 5.3.27 spolu s Apachem 2.2. Vše bude fungovat na jednom hardwarovém stroji, s virtualizovaným unixovým serverem Ubuntu server 12.04, na kterém poběží ústředna a na operačním systému Windows 7 bude umístěn HTTP server Apache. Všechny mnou vytvořené zdrojové kódy, jsou dostupné na přiloženém CD.

HW specifikace:

Procesor:	i5-3470T (2,9GHz)
OS:	Windows 7 Professional 64b
RAM:	4GB DDR3
HDD:	128GB SSD
Grafická karta:	Intel HD Graphics 2500

2 Základní pojmy

V této kapitole se pokusím čtenáře seznámit se základními pojmy používané v bakalářské práci. Pojmy jsou popsány stručně a pro případný zájem jsou uvedeny odkazy na zdroj informací pro detailnější popis.

2.1 Internet

Internetem [4] je označována celosvětově propojená počítačová síť. Počítače mezi sebou komunikují pomocí rodiny protokolů TCP/IP. Nejznámější využívanou službou je WWW, tzv. internetové stránky, které slouží pro distribuci multimédií pomocí hypertextových odkazů. WWW využívá pro přenos komunikace protokolu HTTP.

Pro další popis definujme pojmy klient a server. Za klienta budeme považovat internetový prohlížeč, který funguje na jakémkoliv operačním systému s aktivním připojením k počítačové síti. Server bude jakýkoliv stroj, opět s připojením k počítačové síti, který poskytuje internetové stránky.

2.2 HTTP [5]

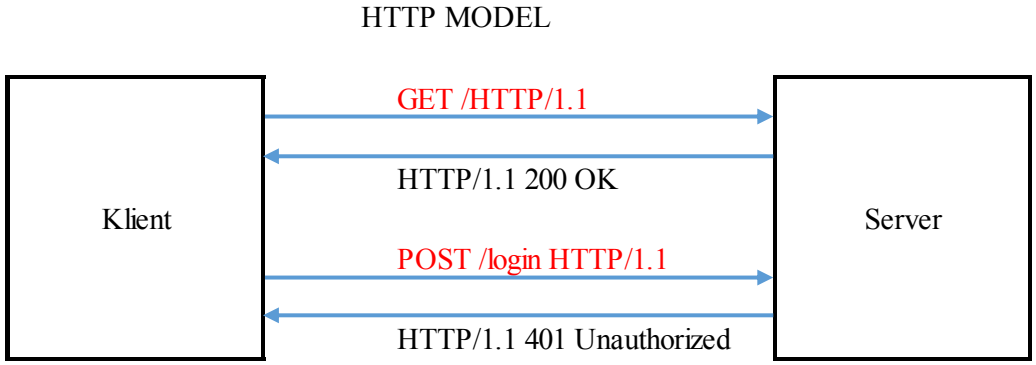
Tento protokol přenáší data nejčastěji ve formátu HTML za pomocí hypertextových odkazů. Takový odkaz zpravidla začíná klíčovým slovem http:// nebo https://. Rozdíl mezi http a https je v tom, že https umí komunikaci zakódovat, aby se nedala zneužít třetími stranami. HTTP je velmi podrobně specifikován jako standard v dokumentu s názvem RFC 2616 [6].

2.2.1 Základní parametry protokolu HTTP

- Je přístupný na portu TCP/80.
- Nejpoužívanější verze je HTTP/1.1
- Identifikace adresy je zajištěna pomocí URL [7]

2.2.2 Komunikace mezi klientem a serverem

Komunikace probíhá formou požadavek-odpověď.



Obrázek 2.1 Požadavek odpověď model

Uživatel do svého klienta zadal nějakou adresu URL. Např. http://www.seznam.cz. Takovou adresu prohlížeč zpracuje do strukturovaného požadavku a odešle jej s žádostí jako na předchozím obrázku GET /HTTP1.1. Ve skutečnosti se neodesílá přímo adresa URL ale její překlad v podobě IP adresy dotazovaného síťového místa či obsahu. O tento překlad se stará DNS [8]. Serverem je navrácena odpověď HTTP/1.1 200 OK, která symbolizuje úspěch a tedy zobrazení samotné webové stránky. Obdobou je metoda POST, která na obrázku ukazuje žádost o provedení /login s odpovědí HTTP/1.1 401, která znamená zamítnutí odeslaných údajů k ověření (třeba pro přihlášení k emailu, kdy uživatel odesílá informace o své identifikaci a heslu).

2.2.3 Požadavek

Požadavek je informace, která se zasílá cílenému serveru s žádostí o provedení nějaké akce.

2.2.3.1 Příklad požadavku

```
GET / HTTP/1.1
Host: edison.sso.vsb.cz
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101
Firefox/28.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: xyz
Connection: keep-alive
```

Figure 2.1 Požadavek

2.2.3.2 Příklad odpovědi

```
HTTP/1.0 200 Connection established
```

Figure 2.2 Odpověď

Odpovědi mají svůj stavový kód. Podle začátku kódu můžeme určit, jak proběhla komunikace:

1. 1xx jsou informační zprávy
2. 2xx indikuje úspěch
3. 3xx znamená přesměrování
4. 4xx jsou chybové hlášky
5. 5xx chyby na straně serveru

3 Webové služby

3.1 Co jsou to webové služby

Webové služby jsou distribuované aplikace, které mohou být nasazeny a spuštěny z různých zařízení. Jako příklad může být služba pro zjištění akcií nějaké společnosti. Ta se skládá z několika části kódu, každá část může být umístěna na samostatném serveru a služba lze volat z PC, kapesního PC (tablety apod.), mobilního telefonu atd. Prostředkem pro přenos komunikace je protokol HTTP. Služby se dělí na dva základní druhy SOAP a REST.

3.2 Proč využít webové služby

V dnešní době lidé využívají spousty informačních systémů pro evidenci dat. Ať už jde o data týkající se třeba personalistiky, mezd, výroby, chemického složení stavu ovzduší nebo třeba informací o vykonávání své záliby. Tyto informační systémy nemusí být a také nejsou programovány jenom v jednom jazyce, ale mohou být naprogramovány v libovolném jazyce. Co když budeme potřebovat mezi těmito systémy komunikovat a jeden bude naprogramován v jazyce PHP a druhým bude třeba JAVA? Právě zde mají místo webové služby, protože dokážou zprostředkovat komunikaci mezi různými systémy.

3.3 Klíčové prvky webových distribuovaných systémů

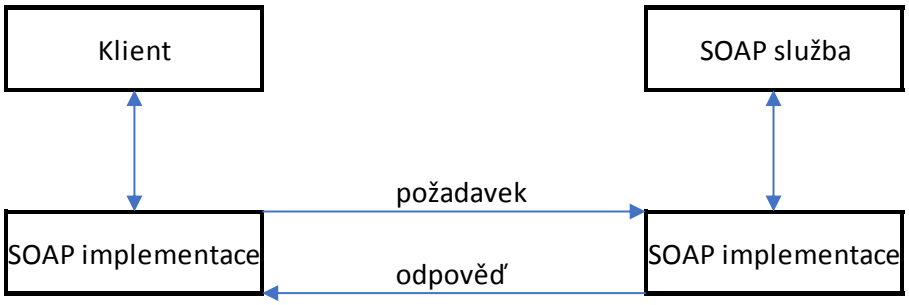
Mezi významné prvky webových distribuovaných systémů patří:

1. Otevřená infrastruktura – služby jsou nasazovány za pomoci standardizovaných komunikačních protokolů.
2. Nezávislost na programovacím jazyce
3. Modulárnost – nově vznikající webové služby mohou být budovány za pomoci stávajících služeb.

3.4 SOAP

Jde o komunikační protokol sloužící k výměně zpráv mezi aplikacemi. Definuje formát zasílaných zpráv pomocí XML. Komunikace probíhá s využitím HTTP protokolu. Je standardizován organizací W3C [10]. Podporuje vzdálené volání procedur (více o RPC viz. [9]) s využitím XML.

3.4.1 Architektura SOAP



Obrázek 3.1 Architektura SOAP

Výměna zpráv probíhá formou požadavek-odpověď. Klient i služba mají implementovanou část zpracovávající SOAP zprávy.

3.4.2 Výhody SOAP

Mezi výhody SOAP např. patří:

- Nezávislost na operačním systému
- Nezávislost na programovacím jazyce
- Je založen na XML
- Je poměrně jednoduchý a rozšiřitelný
- Umožňuje průchod i přes firewally
- Standardizován W3C

3.4.3 Nevýhody SOAP

SOAP má i některé nevýhody, mezi které patří:

- Přenášené zprávy mohou díky struktuře XML mít větší velikosti
- Podporované jazyky hlavně .NET nebo JAVA

3.4.4 SOAP zpráva

SOAP zprávu můžeme rozdělit do tří základních částí a jedné nepovinné, která obsahuje informace o chybách vzniklých při výměně zpráv.

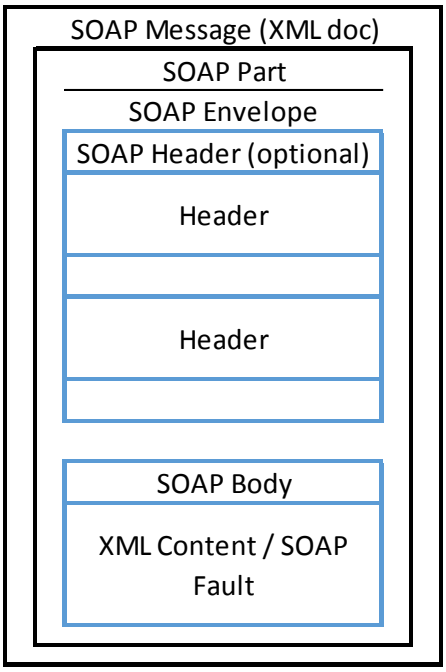
1. Envelope (obálka) – identifikuje XML dokument jako SOAP zprávu
2. Header (hlavička) – obsahuje pomocné informace [1]
3. Body (tělo) – informace o volání a odpovědi
4. Fault (chyba) – obsahuje informace o chybách a je součástí těla zprávy

Zpráva musí dodržovat určitá syntaktická pravidla:

1. Musí být kódována jako XML
2. Musí být specifikován envelope jmenný prostor
3. Musí být specifikován encoding jmenný prostor
4. Nesmí obsahovat DTD referenci
5. Nesmí obsahovat XML procesní instrukci [2]

Syntaxe SOAP zprávy je uvedena na následujícím obrázku:

[1] Může obsahovat informace, které musí být zpracovány.
[2] Jsou to texty určené programům, program je zpracovává podle implementace



Obrázek 3.2 SOAP zpráva

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding"> <!--
xmlns:soap urci typ zpravy -->
<SOAP-ENV:Header>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
</SOAP-ENV:Body>
<SOAP:Envelope>
```

Figure 3.1 Minimální XML zpráva

Předchozí figura má základní značky SOAP zprávy, které jsou běžné u SOAP zpráv.

3.4.5 Odeslání zprávy pomocí HTTP

Následující schéma ilistruje možnost odeslání SOAP zprávy pomocí HTTP:

```
POST /data HTTP/1.1
Host: url
Content-type: application/soap; charset="utf-8"
Content-length: delka

SOAP_ZPRAVA = XML dokument
```

Figure 3.2 Odeslání HTTP

3.4.6 WSDL

Jde o specifikaci, která popisuje metody dostupné prostřednictvím webových služeb. Jde o abstraktní popis, jelikož jsou metody nezávislé na programovacím jazyce. Obsahuje také informace o způsobu

volání, tedy jak navázat spojení. Většinou jsou metody přístupné pomocí HTTP, FTP nebo SMTP. Pro každý způsob přenosu bude mít popis vlastní sekci. WSDL tedy určuje:

- Jak je možné zavolat metodu webové služby
- Jaké parametry metoda očekává
- Jakou odpověď čekat
- Kterými protokoly je možné komunikovat
- Jak mají být data uspořádána
- Jaká je URL pro danou službu

3.4.7 UDDI

Jde o registr, který popisuje, jak může potencionální uživatel webové služby získat informace o nabízených službách. Využívá WSDL soubory k popisu přístupných webových rozhraní. Záznamy v tomto registru mají tři části:

1. Bílé stránky – obsahují základní informace jako kontaktní jména a telefonní čísla a popis nabízené služby
2. Žluté stránky – klasifikují služby podle kritérií
3. Zelené stránky – říkají, jak naimplementovat klienta pro využívání služby

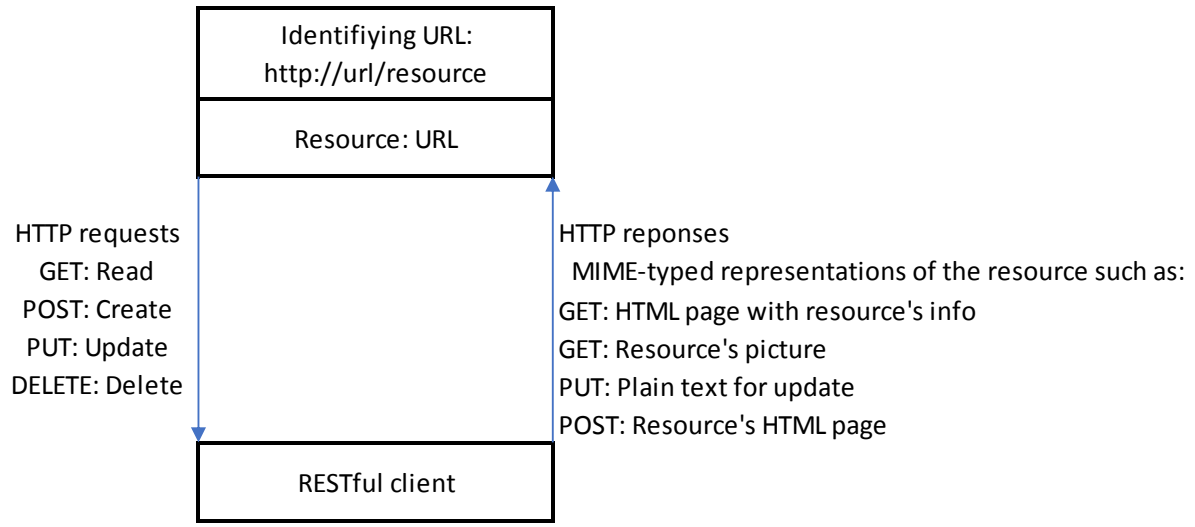
3.5 REST

Autorem popisu REST služeb je Roy Fielding. Popisuje architekturu webových služeb ve své disertační práci. Oproti SOAP, nemá REST standardy a je svým způsobem protiklad složité konstrukce SOAPu. REST využívá identifikátor URI pro přístup ke zdrojům. Zdrojem jsou např. reprezentace dat dostupných na nějaké URL (třeba v XML formátu nebo JSON formátu).

3.5.1 Základní operace se zdroji (resources)

- POST – vytvoří nový zdroj z požadovaných dat
- GET – získá zdroj
- PUT – změní zdroj z požadovaných dat
- DELETE – smaže zdroj

3.5.2 Architektura REST



Obrázek 3.3 Architektura REST

V podstatě REST využívá hlavně URI a HTTP požadavek-odpověď dotazování pro komunikaci mezi klientem a serverem.

3.5.3 PHP a REST

Pro komunikaci s ústřednou je využit programovací jazyk PHP. PHP je speciálně navržený pro vývoj aplikací na webu. Jelikož je REST pouze jakýmsi doporučením, jak vytvářet webové služby, jsou v této práci využity právě tyto tipy. Webová aplikace bude:

1. Přijímat parametry pomocí POST/GET metody
2. Zpracovávat tyto parametry pomocí PHP
3. Vracet JSON/XML formát

Předávat parametry PHP bude AJAX.

3.5.4 AJAX

AJAX je knihovna napsaná javascriptem. Pomocí javascriptu lze měnit obsah webových stránek bez nutnosti znovunačtení obsahu HTML. Zvládá asynchronní a synchronní přenos. Základní syntaxe:

```
$.ajax({
    method: "POST/GET",
    // data k odeslani
    data: {
        param1: param1
    },
    // url adresa k volani
    url: "kamZaslatData.php",
    success: function(data) {
        // co provest s navracenymi daty
        // pokud vse probehlo v poradku
    }
})
```

Figure 3.3 Základní syntaxe AJAX

3.6 REST vs SOAP

SOAP je metoda pro předávání zpráv. Tyto zprávy jsou ve formátu XML. REST umožňuje odesílat a přijímat data ve více formátech. REST také není standardizován, jde spíše o návod jak se dostat k informacím a jak je zpracovat, přitom využívá URI pro identifikaci zdrojů dat. SOAP má definovaný svůj standard a je nutné implementovat více aplikačních částí. REST je mnohem jednodušší na implementaci, proto byl v této práci zvolen právě REST pro otestování odeslání DTMF kódu prostřednictvím internetových stránek.

4 Asterisk-java

Asterisk-java je framework napsaný v javě určený pro usnadnění budování aplikací pro ústřednu Asterisk. Momentálně je vyvíjena nová verze asterisk-java 1.4 pro Asterisk 12 a současně používanou verzí je 1.0.0.CI-SNAPSHOT. Tato současná verze umožňuje pracovat s FastAGI a Manager API.

4.1 FastAGI

Je implementací AGI, která komunikuje pomocí TCP protokolu. Hlavním využitím je odlehčení výpočetní zátěže ústředny. Obecně AGI jsou skripty, které lze volat pomocí dialplanu. Tyto skripty, které mohou zpracovávat hovory stejně jako by byly definovány v dialplanu, je možné umístit na server s FastAGI službou a následně je volat z dialplanu, když na ně přijde řada při vykonávání pomocí příkazu AGI(agi://host). Defaultně nastavený port pro tuto službu bývá 4573.

Ukázku volání AGI demonstruje následující schéma:

```
exten=>volej, 1, AGI(agi://192.168.0.1)
exten=>volej, 1, AGI(agi://192.168.0.1/VolajiciPozadujePodporuServiceDesku)
exten=>volej, 1,
AGI(agi://192.168.0.1:9876/VolajiciPozadujePodporuServiceDesku)
```

Figure 4.1 Ukázka volání AGI[3]

Předávání parametrů se provádí pomocí AGI proměnných. Mezi tyto proměnné patří např.:

- EXTEN – právě volaná extensiona
- UNIQUEID – unikátní identifikátor právě probíhajícího hovoru
- CALLERID – informace o volajícím

Předání parametrů by vypadalo:

```
exten=>volej, 1,
AGI(agi://192.168.0.1/VolajiciPozadujePodporuServiceDesku,${CALLERID(name)}
)
```

Figure 4.2 Parametry v AGI

4.2 Manager API (AMI)

AMI je rozhraní sloužící k řízení ústředny. Lze pomocí něj měnit nastavení ústředny, vykonávat hovory a zobrazovat konfiguraci jednotlivých částí.

Toto rozhraní je podrobněji popsáno v kapitole 4.2. Popis je už specifičtější, jelikož toto rozhraní používám s jazykem PHP pro řízení ústředny.

4.3 Ukázka implementace AMI v javě

Na následujícím schématu Figure 4.3 Originate pomocí javy je zobrazena implementace volání pomocí třídy asterisk-java. Kód provede zalogování uživatele „tomek“ do managera a zahájí hovor s 30 sekundovým čekáním na zvednutí volaného „Tom“, který je registrován jako uživatel komunikačního kanálu SIP.

[3] Název za /VolajiciPozaduje… znamená místo, kam skočit následně v dialplanu pro pokračování vykonávání.

```
package asterisk;
import java.io.IOException;
import org.asteriskjava.manager.AuthenticationFailedException;
import org.asteriskjava.manager.ManagerConnection;
import org.asteriskjava.manager.ManagerConnectionFactory;
import org.asteriskjava.manager.TimeoutException;
import org.asteriskjava.manager.action.OriginateAction;
import org.asteriskjava.manager.response.ManagerResponse;
/**
 *
 * @author u10700
 */
public class Asterisk {

    private ManagerConnection managerConnection;

    public Asterisk() throws IOException
    {
        ManagerConnectionFactory factory = new ManagerConnectionFactory(
                "172.25.4.211", "tomek", "tomek");

        this.managerConnection = factory.createManagerConnection();
    }

    public void run() throws IOException, AuthenticationFailedException,
            TimeoutException
    {
        OriginateAction originateAction;
        ManagerResponse originateResponse;
        originateAction = new OriginateAction();
        originateAction.setChannel("SIP/Tom");
        originateAction.setContext("uzivatel");
        originateAction.setExten("volej");
        originateAction.setPriority(new Integer(1));
        originateAction.setTimeout(new Integer(30000));
        // pripojeni k asterisku
        managerConnection.login();
        // posle Originate prikaz, ktery ceka po dobu 30 sekund do zvednuti
        originateResponse = managerConnection.sendAction(originateAction,
30000);
        // zjisteni odezvy
        System.out.println(originateResponse.getResponse());
        // odhlaseni
        managerConnection.logoff();
    }

    public static void main(String[] args) throws Exception
    {
        Asterisk helloManager;
        helloManager = new Asterisk();
        helloManager.run();
    }
}
```

Figure 4.3 Originate pomocí javy

5 VoIP Asterisk

Ústředna Asterisk (momentální verze je 12.0.0) je flexibilní a dostupnou variantou pro telefonování po síti. Existuje několik profesionálních řešení jako Cisco Call Manager, který stojí řádově sta tisíce, nicméně vzhledem k dostupným informacím o ústředně a jejího nastavení si může ústřednu nakonfigurovat téměř kdokoliv, včetně připojení k telefonní síti a ve zmíněné verzi Asterisku je i ARI, což umožňuje vyvíjet aplikace pro usnadnění správy ústředny. Sponzorem ústředny je společnost Digium, která jej vydává jako open source framework, takže je vhodný pro malé i velké firmy.

5.1 Architektura ústředny

Asterisk je postaven na modulech. Každý modul nabízí svou funkčnost a záleží jen na programátorovi, zda modul využije, některé moduly jsou ovšem vyžadovány k provozu telekomunikace.

5.2 Softphone

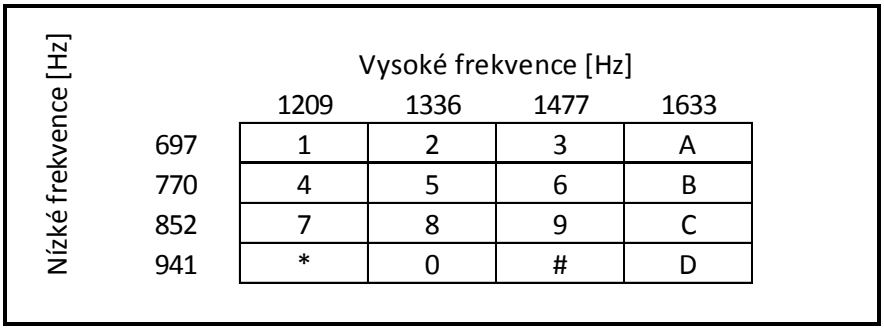
Pro otestování spojení je nezbytně nutné mít na počítači nainstalován minimálně softphone. Jde o simulaci běžného ip hardwarového telefonu a je implementován různými technologiemi. Nejvíce se při testování v této práci osvědčil softphone s názvem „X-Lite“ pro jednoduchou konfiguraci připojení k ústředně. Nicméně postrádá možnost spravovat více účtů najednou, ale tento problém v daném případě nevadí. Pro zaregistrování telefonu ústřednou je potřeba nastavit název účtu, protokol, uživatelské id, doménu, na které běží ústředna a heslo uživatele.

Doména ústředny je IP adresa virtualizovaného Ubuntu serveru na kterém běží Asterisk[4].
Uživatelské id je název uživatele, v našem případě to jsou emailové adresy, ale v příkladech se může objevit i třeba jméno, které jsem požíval během testování (sip.conf).
Heslo je nastaveno uživatelem (sip.conf).
Název účtu je informativní pro uživatele používající X-Lite.

5.3 DTMF kódy

DTMF kódy jsou vlastně tóny jednotlivých kláves na telefonu. Každá klávesa od 0-9, A-D (nejsou tak často), * a # má svůj unikátní tón. Tón se skládá z dvou frekvencí, kdy jedna je nízká a druhá vysoká. Tento způsob kódování vznikl z důvodu identifikace telefonních čísel v telekomunikačních sítích. Na obrázku Obrázek 5.1 DTMF rozložení je vidět, které frekvence se pro daný znak skládají. Jednotlivé tóny si ústředna dokáže dekódovat na číslo.

[4] Oracle VirtualBox nabízí možnost přemostění síťového spojení, pokud získáváme adresu pomocí DHCP, je serveru nastavena automaticky. Pokud není je nutné jí nastavit. Pro výpis nastavení sítových rozhraní slouží na unixu příkaz ifconfig.



Obrázek 5.1 DTMF rozložení

5.3.1 Sip.conf

Je to konfigurační soubor pro kanál SIP, jak pro příchozí tak pro odchozí hovory. Každý záznam začíná hranatými závorkami a identifikuje uživatele. Parametrů může být uvedeno více než na následujícím schématu, nicméně pro tuto práci postačí pouze uvedené.

```
[email@uzivatele.cz]
type=friend
host=dynamic
secret=heslo
context=uzivatel
```

Figure 5.1 sip.conf

Context určuje, kterým plánem v extensions.conf (detaily dále) bude příchozí nebo odchozí hovor zpracován.
Typ může nabývat tří hodnot:

1. Peer – uživatel s touto hodnotou musí být zaregistrován, tedy být přihlášen pomocí svého telefonu k ústředně aby mohl volat a přijímat.
2. User – ústředna zpracovává pouze příchozí hovory (uživatel může volat ústředně, ústředna uživateli ne).
3. Friend – kombinace dvou předchozích, obousměrná komunikace s ústřednou a uživatelem, uživatel musí být zaregistrován jako v prvním bodě (Peer).

Host určuje IP adresu identifikující uživatele. Může být statická nebo dynamická. Pro webové stránky byla v testu zvolena dynamická.
Secret heslo uživatele.

5.3.2 Extensions.conf

Tento konfigurační soubor je plánem, pro realizaci hovorů. Je rozdělen podle kontextů, pro snadné rozlišení jednotlivých částí dialplanu, např. část zpracovávající příchozí hovory na servisní podporu by v dialplanu mohlo být pojmenováno jako `[ServiceDesk]`. Při volání je uveden volající a kontext, který se má vykonat. Pro testování odesílání DTMF kódů byl pro účely této práce vytvořen plán, který se chová jako automat.

1. Přihlášený uživatel na webové stránce zavolá uživateli a působí jako ústředna (pro otestování funkčnosti postačí navázat spojení ústředna-uživatel).
2. Uživatel příjme hovor.
3. Ústředna čeká na zadání DTMF kódu (1 znak).
4. Uživateli, který zadal DTMF, je ústřednou zpátky vrácen zvuk zmáčknutého kódu.

5. Po zadání DTMF kódu se opakuje krok 3, dokud není hovor ukončen volaným

Pro zobrazení dostupných příkazů k použití v dialplanu slouží příkaz (v Asterisk CLI) `core show application` následovaný tabulátorem. Pro detail o příkazu pak `core show application nazev_prikazu`. V dalších podkapitolách jsou stručně popsány příkazy k tvorbě automatu v dialplanu.

Základní syntaxe:

```
exten=>number,priority,application(param1, ...)
same=>n,application ; same je stejne jako exten, ale nezapisuje se znova
number
                    ; jde tedy o slovo urcujici pokracovani
                    ; n urcuje ze jde o nasledujici krok
;ekvivalentni zapis:
exten=>id_volajiciho,1,application(param1, ...)
exten=>id_volajiciho,2,application(param1, ...)
```

Figure 5.2 Základní syntaxe exten

5.3.2.1 Příkaz GOTO

Při zpracování hovoru dialplanem je s tímto příkazem možné skočit do jiné části dialplanu a pokračovat ve vykonávání následujícími příkazy.

```
same=>n,Goto(kontext,extension,priority)
same=>n,Goto(vstup,volej,1)
same=>n,Goto(vstup) ; staci pouzit navesti pro navrat ve stejne casti
```

Figure 5.3 Funkce GOTO

5.3.2.2 Příkaz READ

Tento příkaz umožňuje přehrát volajícímu audio nahrávku a následně od něj přijmout DTMF kód a uložit jej do proměnné. Délka může být maximálně 255 znaků. Volající může ukončit zadávání stisknutím „#“ nebo ukončením hovoru.

```
same=>n,Read(promenna,soubor,maxZnaku,volby,pokusu,casovyLimit)
```

Figure 5.4 Funkce READ

Pro automat postačí pouze proměnná a počet znaků. Volby mohou nabývat hodnot:

- n – čte, i když nepřichází data
- i – přehraje audio soubor z indikačního souboru
- s – konec pokud není spojení

5.3.2.3 Příkazy NOOP, HANGUP, WAIT, SAYDIGITS

- `NOOP` je užitečný k ladění dialplanu. Vypisuje programátorem zadaný text jako parametr při vykonávání jednotlivých částí.
- `WAIT` je klasická funkce k vložení pauzy, udává se v sekundách.
- `SAY DIGITS` slouží k přehrání audio souboru pro dané číslo.

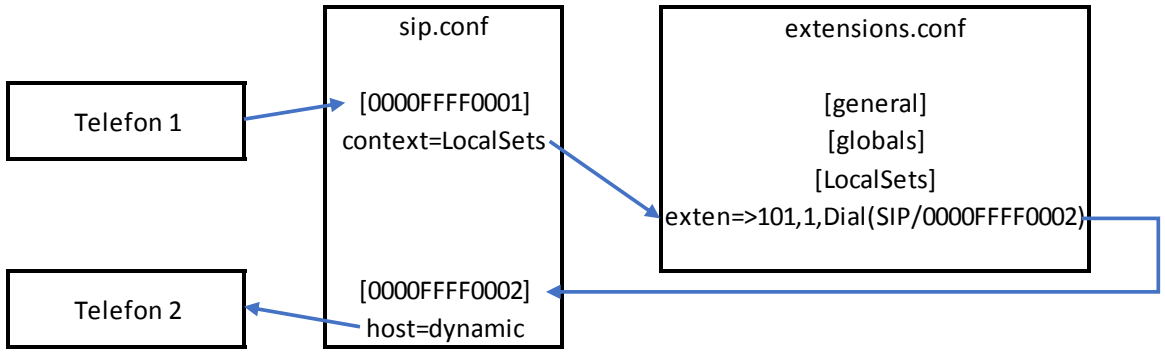
Smyčka v extensions tedy bude vypadat následovně:

```
[uzivatel]
exten=>volej,1,noop(zpracovavam prichozi hovor)
same=>n(vstup),Read(NUMBER,,1,n)
same=>n,Wait(1)
same=>n,SayDigits(${NUMBER})
same=>n,Goto(vstup)
same=>n,Hangup
```

Figure 5.5 Extensions smyčka

Pokud zalogovaný uživatel webové stránky volá jinému uživateli, provede ústředna spojení s tímto uživatelem.

5.3.3 Komunikace mezi sip.conf a extensions.conf



Obrázek 5.2 sip a extensions komunikace

Tyto dva konfigurační soubory jsou velmi úzce provázány. Následuje popis komunikace předchozího obrázku:

1. Na ústřednu přijde požadavek o vytvoření hovoru
2. Ústředna podle identity volajícího najde kontext, v případě na obrázku je volající [0000FFFF0001] a kontext je LocalSets
3. Tomuto kontextu ústředna předává kontrolu pro vyřízení hovoru
4. V extensions.conf se vykonávají příkazy daného kontextu, na příkladu je vidět, že se provede extensiona 101, příkaz Dial(SIP/0000FFFF0002), to znamená, že se požaduje spojení s telefonem [0000FFFF0002]

Shrnutím předešlých kroků zjistíme, že volající vytočil číslo 101. Číslo 101 je natvrdo nastaveno, kdokoliv jej zavolá, žádá o spojení s telefonem [0000FFFF0002] na komunikačním kanále SIP. Asi je možné představit si, že nastavovat každému telefonu, co provést, při zavolání jakéhokoliv čísla, je více než nereálné. Proto byl pro účely této práce zvolen název extensiony „volej“. Vyvolána je pomocí příkazu originate, který je popsán v kapitole 5.4.4.4.

5.3.4 Moduly

Hlavní konfigurační soubor, který rozhoduje o použitých modulech, se nachází v etc/asterisk/modules.conf. Moduly můžeme rozdělit podle typu:

5.3.4.1 Aplikační modul

Tyto funkce se používají pro zpracování hovorů. Funkce tohoto modulu se vyskytují v souboru extensions.conf a často je označován jako srdce Asterisku. Patří zde funkce s názvem dial, originate bez kterých se tato práce neobejde.

5.3.4.2 Přemosťující modul

Zajišťuje přemosťování komunikačních kanálů. Přemostěním komunikačních kanálů lze vytvářet konferenční hovory.

5.3.4.3 CDR

CDR umožňuje ukládat informace o volaných číslech. Data jsou ukládány do souboru defaultně. Dalšími možnostmi jsou uložení do databáze, RADIUS nebo systémového logu.

5.3.4.4 CEL

Opět sběr informací, tentokrát na komunikačních kanálech. Nabízí detailnější popis uskutečněných hovorů.

5.3.4.5 Ovladače komunikačních kanálů

Bez tohoto modulu by Asterisk nemohl uskutečnit žádné hovory. Každý ovladač je specifický svým protokolem nebo technologií, kterou podporuje (SIP, ISDN…).

5.3.4.6 Překladač kodeků

Zajišťuje překlad mezi různými formáty přenášeného audia během hovoru.

5.3.4.7 Překladač formátů

Podobný překladači kodeků, ale pracuje se soubory. Vyhodnocuje, který formát souboru bude použit na daném kanálu.

5.3.4.8 Dialplan funkce

Doplňuje aplikační modul. Umožňuje zpracování stringů, zpracování času, dat a nabízí možnost ODBC připojení.

5.3.4.9 PBX moduly

Jsou periferními moduly, které poskytují lepší kontrolu a konfiguraci důležitých konfiguračních souborů.

5.3.4.10 Zdrojové moduly

Spojují Asterisk s externími zdroji. Tyto moduly jsou v etc/asterisk jako textové soubory.

5.3.4.11 Přídavné moduly

Jsou vytvářeny komunitou s různým využitím. Defaultně nejsou instalovány ani kompilovány. K aktivaci těchto modulů se využije příkaz menuselect[12] během instalace Asterisku.

5.3.4.12 Testovací moduly

Jsou využívány vývojáři Asterisku.

5.4 Asterisk Manager Interface

Je systém umožňující monitorování a řízení ústředny. Pomocí HTTP serveru se lze do manageru přihlásit a vykonávat příkazy, tzv. akce. K zobrazení dostupných akcí slouží příkaz `manager show commands`, který se zadá do asterisk CLI. Jednotlivé informace o příkazu se zobrazí příkazem `manager show command <název příkazu>`.

5.4.1 Základní princip

1. Před zahájením vykonávání příkazu je nutné navázat spojení s AMI a vytvořit tzv. manager session (AMI po úspěšné autorizaci vytvoří cookie)
2. Data mohou být odesílány v obou směrech po navázání spojení
3. Data můžeme rozdělit podle směru, odkud jsou zasílány:
    a. Klient → ústředna, první řádek je akce
    b. Ústředna → klient, první řádek je buď typu „event“ nebo „response“
4. Oddělení jednotlivých řádků se provádí znaky CR/LF

5.4.2 Tvorba požadavku

K vykonání příkazu na rozhraní manageru je potřeba mu jej nějakým způsobem zaslat. Tyto příkazy budeme předávat ústředně pomocí URI adresy. Ta se skládá z adresy serveru, portu a samotného příkazu s parametry, což je uvedeno zde:

```
"http://172.25.4.211:8088/mxml?Action=Login&username=tomek&secret=tomek"
```

Figure 5.6 Požadavek s akcí

Na portu 8088 naslouchá HTTP server zabudovaný v ústředně. Jeho povolení je probráno v následující kapitole 5.4.3. Za portem následuje lomítko a typ odezvy, který bude ústředna vracet. Na výběr je z několika možností:

1. mxml – odezva je formátována jako xml dokument.
2. rawman – obyčejný text.
3. json – formát pro využití javascriptem.
4. manager – odezva je formátována pomocí HTML.

Za typem odezvy následuje akce oddělená znakem „?“ a označená klíčovým slovem `Action`. Akce se nastaví pomocí rovnítka, pokud akce předává parametry, pak se přidávají pomocí znaku „&“ s názvem parametru a jeho hodnotou nastavenou opět rovnítkem.

5.4.3 Povolení AMI

V konfiguračním souboru manager.conf je nutné upravit následující:

1. Enabled = yes
2. Webenabled = yes

3. Vytvořit uživatele pro přístup k AMI

Konečný soubor tak bude vypadat následovně:

```
[general]
enabled=yes
webenabled=yes
port=5038
allowmultiplelogin=no

;uzivatele
[uzivatel@email.cz]
permit=0.0.0.0/0.0.0.0 ; prihlasit se muze odkudkoliv
secret=heslo
read=all
write=all
[uzivatel2@email.cz]
...
```

Figure 5.7 manager.conf

5.4.4 Akce

AMI nabízí velký výčet akcí, které lze vyvolat vzdáleně. V případě komunikace pro zasílání DTMF kódů si vystačíme s následujícími akcemi.

5.4.4.1 Login

Přihlásí uživatele k AMI. Uživatel může vykonávat příkazy, podle svých nastavených práv.

```
Action: Login CR/LF
ActionID: 0-xxxx nepovinné CR/LF
Username: uzivatel@email.cz CR/LF
Secret: heslo CR/LF CR/LF
```

Figure 5.8 AMI Login

5.4.4.2 Logoff

Odhlásí uživatele.

5.4.4.3 UpdateConfig

Pomocí této akce, můžeme přistoupit k jakémukoliv konfiguračnímu souboru ústředny. Tedy i sip.conf a manager.conf. Těmito dvěma soubory budeme registrovat uživatele. To zobrazuje následující schéma:

```
Action: UpdateConfig
SrcFilename=manager.conf
DstFilename=manager.conf
Reload=yes
Action-000000=NewCat
Cat-000000=Email uzivatele

Action-000001=Append
Cat-000001=Email uzivatele
Var-000001=promenna
Value-000001=hodnota

Action-000002=Append
Cat-000002=Email uzivatele
Var-000002=promenna
Value-000002=hodnota
…
```

Figure 5.9AMI UpdateConfig

5.4.4.4 Originate

Tento příkaz umožňuje realizovat hovory.

```
Action: Originate
Channel=SIP/email uzivatele
Exten=volej
Context=uživatel
CallerID=email_volajiciho
Priority=1
```

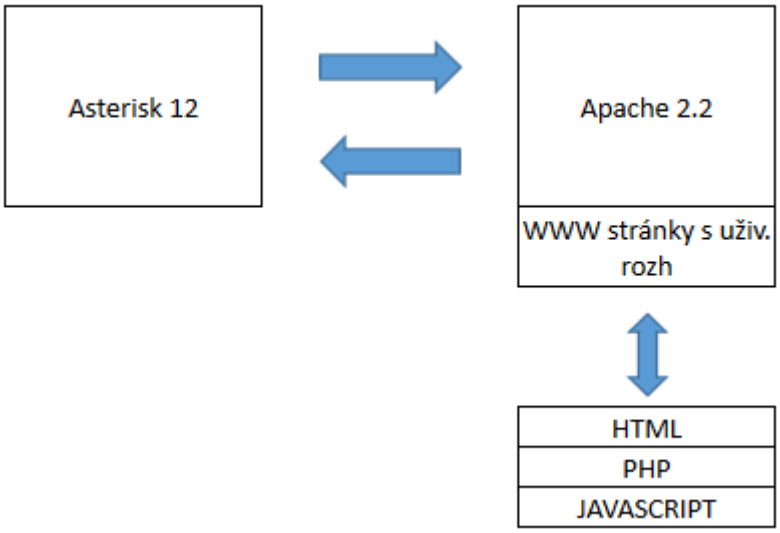
Figure 5.10 AMI Originate

Zavoláním toho příkazu je zajištěno předání řízení na ústřednu, ta vybere kontext a začíná extensionou „volej“, kde je definována smyčka k příjmu DTMF kódu od volajícího.

6 Návrh systému

6.1 Základní koncept

Obrázek 6.1 Koncept zobrazuje základní koncept pro realizaci zadání, tedy zasílání DTMF kódu.

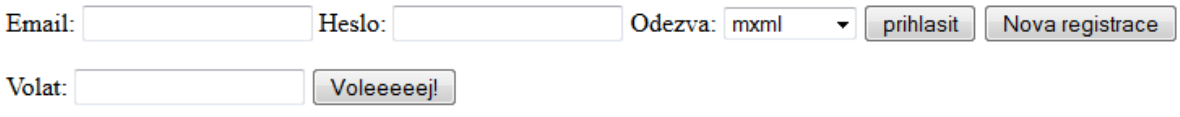


Obrázek 6.1 Koncept

Pro otestování možností komunikace s ústřednou bylo využito virtualizace. Všechny komponenty fungují na jednom fyzickém stroji. Asterisk 12 je virtualizován pomocí softwaru Oracle VM VirtualBox. Je nainstalován na unixové distribuci Ubuntu Server, bez GUI. Apache 2.2 běží jako lokální server přímo na operačním systému, kterým je Windows 7 Professional SP1.

Komunikace začíná na straně uživatele. Uživatel si načte stránky poskytované Apache serverem, následně se např. přihlásí, tzn., že uživatel odesílá požadavek opět na Apache server. Apache tento požadavek zpracuje pomocí PHP. Během zpracování PHP je odeslán další požadavek HTTP serveru ústředny. Ústředna odpoví zpět Apachi, PHP část čeká a jakmile jí příjme, promítne se požadavek uživateli samotnému, třeba přihlášením do AMI a možnosti zavolat.

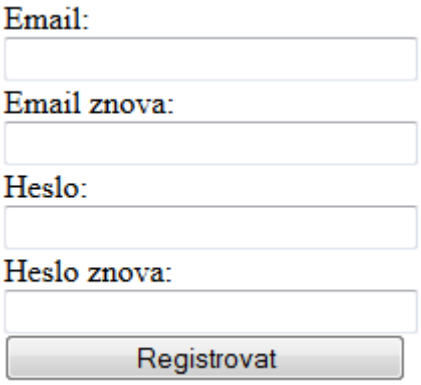
6.2 Návrh uživatelského rozhraní www



Obrázek 6.2 Návrh uživ. rozhraní [5]

Tlačítko nová registrace zobrazí registrační formulář:

[5] **Odezva: mxml** tato část nebude uživateli zobrazena, slouží jen pro testování během implementace



Obrázek 6.3 Registrační form.

6.3 Registrace

Registrace proběhne následovně (dle obrázku výše):

1. Uživatel si zobrazí registrační formulář
2. Uživatel vyplní email a heslo
3. Uživatel potvrdí zadané údaje
4. Údaje se zašlou pomocí ajaxu stisknutím tlačítka Registrovat na PHP stránku s názvem registraceUzivatele.php

Registrace.php zpracuje informace v pořadí:

1. Vytvoří spojení s ústřednou, konkrétně vytvoří manager session definovaným uživatelem v kapitole 5.4.3.
2. Během tohoto spojení proběhne zaslání příkazu UpdateConfig s parametry zadané uživatelem, tedy jméno a heslo. Příkaz obsahuje ještě dodatečné parametry jako kontext, který má název „uživatel“.

6.4 Přihlášení

Po vyplnění údajů a stisknutí tlačítka přihlásit se opět zpracuje požadavek pomocí ajaxu a php. Ústředna zareaguje dle následujícího schématu:

```
== HTTP Manager 'tomek' logged on from 172.25.4.99
```

Figure 6.1 CLI Odezva

Během přihlášení je vytvořená cookie, bez ní by uživatel nemohl provádět příkazy, protože by ho ústředna nerozpoznala. Ta je uložena v adresáři na straně Apache serveru. Po odhlášení uživatele je odstraněna. Při vypršení časového limitu připojení k ústředně sice cookie zůstane, ale při příštím přihlášení je obnovena.

6.5 Volání

Volání je realizováno příkazem originate viz kapitola 5.4.4.4. Po zadání jména volaného a potvrzení tlačítkem „Voleeeej“ (registrovaný uživatel je již přihlášený), je opět přenos dat zajištěn pomocí ajaxu a php. Po zvednutí uživatelem ústředna čeká na zadání DTMF kódu, uživatel zadá jeden znak a

ústředna mu ho zpátky přehraje v podobě mluveného slova. V ústředně je tento hovor zaznamenám následovně na schématu Figure 6.2 CLI volání.

```
== Using SIP RTP CoS mark 5
-- Called Tom
-- SIP/Tom-00000005 is ringing
-- SIP/Tom-00000005 answered
Executing [volej@uzivatel:1] NoOp("SIP/Tom-00000005", "zpracovavam prichozi
hovor") in new stack
Executing [volej@uzivatel:2] Read("SIP/Tom-00000005", "NUMBER,,1,n") in new
stack
-- Accepting a maximum of 1 digits.
DTMF end '1' received on SIP/Tom-00000005, duration 250 ms
DTMF end passthrough '1' on SIP/Tom-00000005
-- User entered '1'
-- Executing [volej@uzivatel:3] Wait("SIP/Tom-00000005", "1") in new stack
-- Executing [volej@uzivatel:4] SayDigits("SIP/Tom-00000005", "1") in new
stack
-- <SIP/Tom-00000005> Playing 'digits/1.gsm' (language 'en')
-- Executing [volej@uzivatel:5] Goto("SIP/Tom-00000005", "vstup") in new
stack
-- Goto (uzivatel,volej,2)
-- Executing [volej@uzivatel:2] Read("SIP/Tom-00000005", "NUMBER,,1,n") in
new stack
-- Accepting a maximum of 1 digits.
DTMF end '6' received on SIP/Tom-00000005, duration 250 ms
DTMF end passthrough '6' on SIP/Tom-00000005
-- User entered '6'
-- Executing [volej@uzivatel:3] Wait("SIP/Tom-00000005", "1") in new stack
-- Executing [volej@uzivatel:4] SayDigits("SIP/Tom-00000005", "6") in new
stack
-- <SIP/Tom-00000005> Playing 'digits/6.gsm' (language 'en')
-- Executing [volej@uzivatel:5] Goto("SIP/Tom-00000005", "vstup") in new
stack
-- Goto (uzivatel,volej,2)
-- Executing [volej@uzivatel:2] Read("SIP/Tom-00000005", "NUMBER,,1,n") in
new stack
-- Accepting a maximum of 1 digits.
-- User disconnected
```

Figure 6.2 CLI volání

Na tomto výpisu lze dobře vidět, jak funguje smyčka vytvořená příkazem GOTO.

7 Implementace

V této kapitole bude ukázána implementace části webové aplikace s využitím AJAXu, PHP a HTML. Stylování zde není rozebráno, ale je dostupné na přiloženém CD.

7.1 Přihlášení

Nejprve konstrukce HTML formuláře pro zadání uživatelovy emailové adresy a hesla. Vystačí dva popisné prvky (`label`) s dvěma vstupními (`input`) a jedním potvrzovacím, tedy tlačítkem (`button`).

```
<label>Email:</label>
<input type="text" id="inpEmail" />
<label>Heslo:</label>
<input type="password" id="inpHeslo" />
<label>Odezva:</label>
<select id="selOdezva">
    <option value="mxml">mxml</option>
    <option value="amxml">amxml</option>
    <option value="static">static</option>
    <option value="arawman">arawman</option>
    <option value="rawman">rawman</option>
</select>
<input type='button' id='btnPrihlasit' value='prihlasit' />
```

Figure 7.1 Přihlášení HTML

Po vyplnění emailu a hesla je nutné tyto informace předat PHP k zpracování. Značka `selOdezva` slouží k přepínání formátu odpovědí, je to pouze pro vývoj aplikace, finální uživatelské rozhraní toto nemusí mít. Předání není nijak složité, s pomocí javascriptu, konkrétně AJAXem bude takový problém řešen následovně:

```
function prihlasUzivatele(email, heslo) {
    $.ajax({
        method: "POST",
        url: "prihlasUzivatele.php",
        data: {
            email: email,
            heslo: heslo,
            odezvaV: $('#selOdezva').val()
        },
        success: function(data, textStatus, jqXHR) {
            if (data === alreadyAuthenticated) {
                poPrihlaseni(true);
            } else if (data === 'Authentication accepted') {
                poPrihlaseni(true);
            } else {
                poPrihlaseni(false);
            }
        }
    });
}
```

Figure 7.2 AJAX předání parametru

Na předchozím příkladu je použita metoda POST k odeslání informací o emailu a heslu k zpracování PHP. `odezvaV` je pouze informativní při vývoji, tato možnost přepíná formát dat, které ústředna vrací. Poté stačí přiřadit událost po kliknutí tlačítku a tuto funkci použít. Pro vylepšení a zabezpečení by se

měly hesla kódovat, nicméně v této práci bezpečnost aplikace není řešena, jde hlavně o vyzkoušení přístupu k ústředně z prostředí webu. Nyní následuje samotné zpracování pomocí PHP.

```php
<?php
include './nastaveni.php';
$typOdezvy = "mxml"; # po ? nasleduje prikaz

if (isset($_REQUEST['email']) && isset($_REQUEST['heslo'])) {
    $typOdezvy = $_REQUEST['odezvaV'];
    $email = $_REQUEST['email'];
    $heslo = $_REQUEST['heslo'];
    $cLogin = "$typOdezvy" . "?Action=Login&username=$email&secret=$heslo";
    $uri = "http://" . $server . ":" . $port . "/" . $cLogin;

    $ch = curl_init();
    $options = array(
        CURLOPT_URL => $uri,
        CURLOPT_HEADER => true, # vraci hlavicku
        CURLOPT_COOKIEFILE => "cookies/asterisk_cookies_" . "$email" .
".txt", # soubor s cookies pro asterisk
        # pokud uz cookie je, je tam ulozena
        CURLOPT_COOKIEJAR => "cookies/asterisk_cookies_" . "$email" .
".txt", #pokud cookie neni, vytvori ji
        CURLOPT_RETURNTRANSFER => true
    );
    curl_setopt_array($ch, $options);
    $data = curl_exec($ch);
    curl_close($ch);

    $pole = explode("\r\n\r\n", $data);

    $hlavicka = $pole[0];
    $xml = $pole[1];

    $exploded = (explode("\r\n", $hlavicka));

    $h = array();
    for ($i = 0; $i < count($exploded); $i++) {
        if ($i != 0) {
            list($key, $value) = explode(":", $exploded[$i]);
            $h["$key"] = trim($value);
        } else {
            $h["$exploded[$i]"] = '';
        }
    }
    $xmlParsed = simplexml_load_string($xml);
    $xpath = $xmlParsed->xpath("//@message");
    echo $message = $xpath[0];
```

Figure 7.3 PHP zpracování přihlášení

Pro nastavení manager session byla zvolena knihovna cURL pro její jednoduché a robustní využití. Po připravení URL k odeslání je dalším krokem příprava požadavku cURLem.

1. Inicializace cURL session pomocí `curl_init()`
2. Nastavení parametrů pro přenos pomocí `curl_opt()`
3. Provedení příkazů `curl_exec()`
4. Ukončení session `curl_close()`

7.2.1 cURL možnosti (CURL_OPT)

U schématu Figure 7.3 PHP zpracování je použito asociativní pole k nastavení parametrů pro přenos. Nejdříve se do pole vloží jednotlivé volby. Celé pole se pak promítne v session pomocí příkazu `curl_setopt_array($ch, $options)`. Možností co lze nastavit je spousta. Níže jsou uvedeny alespoň v této práci použité volby:

7.2.1.1 CURLOPT_URL

Základní volba, bez které se neobejde nikdo. Nastavuje URL, se kterou probíhá komunikace.

7.2.1.2 CURLOPT_HEADER

Vrací hlavičku odpovědi.

7.2.1.3 CURLOPT_COOKIESESSION

Načítá soubor s cookies z předešlých session.

7.2.1.4 CURLOPT_COOKIEJAR

Určuje název souboru s cookies, do kterého jsou zapsány informace o session.

7.2.2 cURL odezva

Po vykonání kódu z Figure 7.3 PHP zpracování je odezva ústředny:

```
HTTP/1.1 200 OK
Server: Asterisk/12.0.0
Date: Fri, 02 May 2014 14:38:32 GMT
Connection: close
Cache-Control: no-cache, no-store
Content-Length: 144
Content-type: text/xml
Cache-Control: no-cache;
Set-Cookie: mansession_id="c8230b56"; Version=1; Max-Age=300
Pragma: SuppressEvents

<ajax-response>
<response type='object' id='unknown'><generic response='Success'
message='Already authenticated' /></response>
</ajax-response>
```

Figure 7.4 odezva Asterisku

`Set-Cookie: mansession_id="c8230b56"; Version=1; Max-Age=300` tento řádek, nastaví cookie pro daného uživatele s dobou platnosti 300 sekund.
Odezva je ve spodní části a začíná značkou `<ajax-response>`. Zpráva obsahuje text, který říká, že uživatel je přihlášen a má tedy platnou cookie.

Soubor s cookie je jednoduchý:

```
# Netscape HTTP Cookie File
# http://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

172.25.4.211      FALSE /     FALSE 1399285070  mansession_id     "c8230b56"
```

Figure 7.5 asterisk_cookies_tomek

První je adresa, které bylo nastaveno cookie. `FALSE` vyjadřuje, jestli mohou ostatní webové servery přistupovat k této cookie. Lomítko určuje omezení, v tomhle případě je omezení takové, že další URL s přístupem k cookie musí být z množiny první URL, tedy za lomítkem může být třeba akce. Další v řadě po lomítku říká, zda je nutné přejít na zabezpečený protokol HTTPS k další komunikaci a předání cookie.

7.3 Volání

Zpracování volání ilustruje následující schéma:

Volání je velmi podobné přihlášení, jen se předávají parametry volaného, opět je nutné mít platný soubor s cookie, jinak se uživatel musí znova přihlásit.

Odpovědí je buď „Originate failed“ nebo „Originate successfully queued“. K selhání dojde, pokud uživatel neexistuje nebo hovor odmítne.

```php
<?php

include './nastaveni.php';
$volany = $_REQUEST['volany'];
$email = $_REQUEST['email'];
$cOriginate = $typOdezvy .
        "?Action=Originate&"
        . "Channel=SIP/$volany&"
        . "Exten=volej&"
        . "Context=uzivatel&"
        . "CallerID=$email&"
        . "Priority=1";

$uri = "http://" . $server . ":" . $port . "/" . $cOriginate;

$ch = curl_init();
$options = array(
    CURLOPT_URL => $uri,
    CURLOPT_HEADER => true, # vraci hlavicku
    CURLOPT_COOKIEFILE => "cookies/asterisk_cookies_" . "$email" . ".txt",
# soubor s cookies pro asterisk
    CURLOPT_COOKIEJAR => "cookies/asterisk_cookies_" . "$email" . ".txt",
#pokud cookie neni, vytvori ji
    CURLOPT_RETURNTRANSFER => true
);
curl_setopt_array($ch, $options);
$data = curl_exec($ch);
curl_close($ch);

//print_r($data);
$pole = explode("\r\n\r\n", $data);

$hlavicka = $pole[0];
$xml = $pole[1];

$exploded = (explode("\r\n", $hlavicka));

$h = array();
for ($i = 0; $i < count($exploded); $i++) {
    if ($i != 0) {
        list($key, $value) = explode(":", $exploded[$i]);
        $h["$key"] = trim($value);
    } else {
        $h["$exploded[$i]"] = '';
    }
}

//    print_r($h);
$xmlParsed = simplexml_load_string($xml);
//    echo "<pre>";
$xpath = $xmlParsed->xpath("//@message");
    print_r( $xmlParsed );
echo $message = $xpath[0];
```

Figure 7.6 PHP zpracování volání

8 Závěr

V této bakalářské práci jsem se snažil vysvětlit základní konfiguraci ústředny Asterisk 12.0.0 a jejího ovládání s využitím webových služeb, stavěných na architektuře REST. Tím, že je to verze začínající 12.0 můžeme očekávat změny s dalšími vydáními.

Můj osobní dojem z ústředny a jejího ovládání je velmi pozitivní. Líbí se mi uspořádání konfiguračních souborů a pokud je nutné provádět nějaké změny, nebo zavést část ústředny k provozu, postačí si projít ukázkové konfigurační soubory, které jsou dostupné již po instalaci ústředny a jsou velmi hezky vysvětleny i s příklady daných nastavení.

Jako další kladnou zkušenost, kterou bych dále rád rozvíjel je prostředí Ubuntu serveru. Před touto prací jsem byl hlavně uživatelem produktů firmy Microsoft a myslím si, že je to velká škoda. Unixové systémy sice asi budou trochu složitější pro někoho, kdo v nich nepracuje běžně a ještě ve verzi bez grafického rozhraní, ale když se člověk seznámí s prostředím, zjistí, že spousta věcí je velmi intuitivních.

Zasílání DTMF kódů je nyní z mého pohledu poměrně jednoduchou překážkou s využitím webových služeb stavěných pomocí PHP, HTML a javascriptem. Implementace pomocí druhé varianty služeb, tedy technologií SOAP bude značně složitější už jenom z důvodu větší zátěže pro implementování jak klientské, tak uživatelské aplikační části podle definovaných standardů.

Tato práce by mohla být posunuta ještě dále, např. v rámci diplomové práce. Bylo by zajímavé zjistit, jak připojit ústřednu k reálné telekomunikační síti, nakonfigurovat volání pro menší firmu a vytvořit k tomu patřičné administrátorské rozhraní prostřednictvím webových stránek. Nakonfigurovat hardwarové IP telefony namísto softphonů. Další co ve mne vzbudilo zájem je vyzkoušet pomocí DTMF kódů ovládání elektronického relátka, např. pro zámky u dveří.

Literatura a zdroje:

[1] SAMS Teach Yourself Web Services in 24 Hours ISBN-10: **0672325152**
[2] Java Web Services Up and Running ISBN-10: **0-596-52112-X**
[3] Asterisk The Definitive Guide 4th Edition Ebook ISBN-10: **1-4493-3241-2**
[4] Internet http://en.wikipedia.org/wiki/Internet
[5] HTTP http://en.wikipedia.org/wiki/Hypertext
[6] RFC 2616 dokumentace http://tools.ietf.org/html/rfc2616
[7] URL http://en.wikipedia.org/wiki/Uniform_resource_locator
[8] DNS http://en.wikipedia.org/wiki/Domain_Name_System
[9] RPC úvod http://en.wikipedia.org/wiki/Remote_procedure_call
[10] W3C SOAP http://www.w3.org/TR/soap/
[11] AMI Akce https://wiki.asterisk.org/wiki/display/AST/Asterisk+12+AMI+Actions
[12] Menuselect
https://wiki.asterisk.org/wiki/display/AST/Using+Menuselect+to+Select+Asterisk+Options
[13] PHP http://www.php.net/manual/en