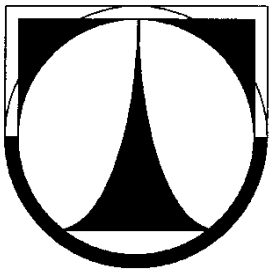
TECHNICKÁ UNIVERZITA V LIBERCI
FAKULTA STROJNÍ
Katedra aplikované kybernetiky



ADMINISTRAČNÍ SYSTÉM PRO ISP
ADMINSTRATION SYSTEM FOR ISP

BAKALÁŘSKÁ PRÁCE

Vojtěch Kočí

Leden 2011

TECHNICKÁ UNIVERZITA V LIBERCI
FAKULTA STROJNÍ

Katedra aplikované kybernetiky

Studijní program
Strojírenství
Obor
Výrobní systémy
Zaměření
Inženýrská informatika

ADMINISTRAČNÍ SYSTÉM PRO ISP
ADMINISTRATION SYSTEM FOR ISP

Bakalářská práce

Vojtěch Kočí

Vedoucí bakalářské práce: Ing. Michal Moučka, Ph.D.
Konzultant bakalářské práce: Ing. Ivo Jelínek

Počet stran:	56
Počet obrázků:	22
Počet příloh:	1
Počet zdrojových kódu:	41

Leden 2011

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Katedra aplikované kybernetiky Studijní rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení: **Vojtěch KOČÍ**

Studijní program: **B2341 Strojní inženýrství**

Obor: **2301R030 Výrobní systémy**

Zaměření: **Inženýrská informatika**

Ve smyslu zákona č. 111/1998 Sb. o vysokých školách se Vám určuje bakalářská práce na téma:

Administrační systém pro ISP

Zásady pro vypracování:
(uveďte hlavní cíle bakalářské práce doporučené metody pro vypracování)

1. Navrhněte strukturu administračního systému pro poskytovatele internetových služeb (Internet Service Provider) na kompletní administraci zákazníků, který bude splňovat požadavky firmy METRONET.

   Systém musí umožňovat evidenci údajů o jednotlivých vysílačích ISP, evidenci informací o klientech a službách jím poskytovaných. Nezbytnou součástí systému musí být modul pro generování a následné odesílání faktur klientům. Funkce správy uživatelských účtů administračního systému je samozřejmostí.
2. Návrh zrealizujte.
3. Ověřte funkčnost a správnost řešení.

Forma zpracování bakalářské práce:

- průvodní zpráva: cca 35 stran + přílohy
- grafické práce: dle potřeby

Seznam literatury (uveďte doporučenou odbornou literaturu):

[1] *Zend Framework Documantation - Component Goals & Benefits.* http://framework.zend.com/manual/components (accessed Nov 24, 2010)

[2] *Zend Framework Documantation - Programmer's Reference Guide.* http://framework.zend.com/manual/en (accessed Nov 24, 2010)

[3] *Zend Framework Documantation - API Guide.* http://framework.zend.com/manual/apidoc/core (accessed Nov 24, 2010)

[4] GILMORE, J., W.: *Velká kniha PHP a MySQL 5 - Kopendium pro začátečníky i profesionály.* Brno: Zoner Press, 2007.

[5] KOFLER,M. - ÖGGL, B.: *PHP 5 a MySQL 5 - Průvodce webového programátora.* Praha: Computer Press, 2007.

[6] GUTMANS, A. - BAKKEN, S., S. - ROTHANS, D.: *Mistrovství v PHP 5.* Praha: Computer Press, 2007.

Vedoucí bakalářské práce: Ing. Michal Moučka, Ph.D.

Konzultant bakalářské práce: Ing. Ivo Jelínek (METRONET, s.r.o.)

L.S.

Ing. Michal Moučka, Ph.D.
vedoucí katedry

doc. Ing. Miroslav Malý, CSc.
děkan

V Liberci dne 24. 11. 2010

Platnost zadání bakalářské práce je 15 měsíců od výše uvedeného data (v uvedené lhůtě je třeba podat přihlášku ke SZZ). Termíny odevzdání bakalářské práce jsou určeny pro každý studijní rok a jsou uvedeny v harmonogramu výuky.

Téma

Administrační systém pro ISP

Anotace

Bakalářská práce se zabývá administračním systémem pro ISP, který má za úkol evidovat zákazníky, generovat faktury včetně kontroly úhrad jednotlivých faktur. Systém se bude skládat z několika modulů, a bude plně komunikovat s účetním systémem POHODA přes XML rozhraní, bankou Česká Spořitelna pomocí rozhraní multi-cash, externím trouble ticket systémem, zákaznickou zónou a s VoIP systémem.

Cílem této práce je naprogramovat funkční webovou aplikaci pomocí skriptovacího jazyku PHP s napojením na databázi MySQL, který bude splňovat výše uvedené požadavky.

Klíčová slova: php, mysql, xml, informační systém.

Theme

Administration system for ISP

Annotation

This aim of this work is to program administration system for ISPs, which aims to record customers, generate invoices, including checking invoices and payments. The system will consist of several modules, and will fully communicate with the accounting system POHODA via an XML interface, Bank of the Česká Spořitelna via a multi-cash interface, an external trouble ticket systems, customer‘s zone and VoIP systems.

The aim of this work is programmed functional Web application using a scripting language PHP with connection to the MySQL database, which will meet the above requirements.

Key words: php, mysql, xml, information system.

Zpracovatel: TU v Liberci, Fakulta strojní, Katedra aplikované kybernetiky

Dokončeno: 2011

Archivní označení zprávy:

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

V …… dne ……..…… ……

Podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce Ing. Michalu Moučkovi, Ph.D. za rady a konzultace při psaní této práce. Konzultantu Ing. Ivo Jelínkovi za rady, připomínky a trpělivost při vývoji systému, Martinu Mrázovi za backendy, které pracovaly s routery, a byly nutné pro funkčnost systému.

Dále bych rád poděkoval rodičům a všem mým blízkým za podporu během celé doby studia na FS TUL.

Obsah

PŘEHLED POUŽITÝCH ZKRATEK A NÁZVŮ

AJAX	asynchronous javascript and xml	technologie pro interaktivní stránky
AP	access point	přístupový bod do sítě
API	application programming interface	rozhraní pro programování aplikace
CERN		výzkumný ústav
CSS	cascading style sheet	kaskádové styly
FTP	file transfer protocol	protokol pro přenos soubroů
HTML	hypertext markup language	hypertextový značkovací jazyk
IETF	internet engineering task force	vývoj a podpora internetových standardů
JS	javascript	objektově orientovaný skriptovací jazyk
ISP	internet service provider	poskytovatel internetu
MVC	model view controller	návrhový vzor
MySQL		databázový systém
OOP	object-oriented programing	objektově orientované programování
PDF	portable document format	formát souboru
PHP	PHP hypertext preprocessor	skriptovací programovací jazyk
PX		pixel (jeden bod na obrazovce)
SMTP	simple mail transfer protocol	protokol určený pro přenos emailů
SCP	secure copy	protocol pro zabezpečený přenos souborů
SFTP		ssh ftp
SQL	structured query language	strukturovaný dotazovací jazyk
SSH	secured shell	bezpečná komunikace mezi počítači
TCPDF		API pro generování .pdf souboru v PHP
URL	uniform resource locator	www adresa
XHTML	extensible html	rozšiřitelné HTML
XML	extensible markup language	rozšiřitelný značkovací jazyk
VoIP	voice over internet protocol	telefonie přes internet
W3C	world wide web consortium	vyvíjí standardy pro www
WYSIWYG	what you see is what you get	editor pro tvorbu webových stránek
WWW	world wide web	aplikace internetového protokolu
ZF	zend framework	framework pro PHP

1 ÚVOD

Podíváme-li se na tvorbu webových stránek, můžeme je rozdělit do dvou základních kategorií a to *statické* webové stránky a *dynamické* webové stránky. Pokud bychom například tvořili jednoduchou webovou prezentaci pro nějakou firmu, vystačíme si se statickými stránkami psanými v (X)HTML a CSS popř. s nějakými efekty v JS. Na druhou stranu pokud chceme vytvořit administrační systém, v němž je potřeba mít věci dynamické, určitě se neobejdeme bez nějakého skriptovacího jazyku, který umí pracovat s databázovými systémy a dynamicky generovat obsah stránky jako je třeba PHP, Perl, Python.

Pro komplexnější aplikace je dobré zvolit nějaký framework, který využívá návrhový vzor MVC, jež oddělí programovou část od zobrazovací části. Při tvorbě nějaké speciální aplikace může být použito tzv. API (je-li pro danou aplikaci vyvinuto). API se dá použít i při tvorbě webové aplikace za použití frameworku (softwarová struktura, která napomáhá při vývoji softwarových projektů. Může obsahovat podpůrné programy, API, návrhové vzory, …).

Důležitou součástí vývoje jakékoli aplikace je vývojové prostředí. Pro vývoj webových aplikací je hned několik vývojových prostředí, přičemž každé je vhodné pro jiný druh aplikace. Pro méně rozsáhlé aplikace je vhodné využít program PSPad, který je volně dostupný na internetu pro platformu Windows (pod Linuxem lze spustit pomocí wine). Pro větší aplikace jsou vhodnější nějaké komplexnější programy, například Eclipse for PHP, taktéž volně ke stažení, popř. komerční Zend Studio, vhodné pro vytváření aplikací v kombinaci se Zend Frameworkem.

1.1 Cíle bakalářské práce

V této bakalářské práci byl vyvíjen administrační systém psaný pomocí skriptovacího jazyku PHP s využitím Zend Frameworku (dále ZF) na linuxovém serveru. Zobrazení stránky bylo řešeno pomocí XHML 1 a CSS. Pro události před novým načtením stránky bylo využito frameworku jQuery, díky kterému mohlo být velmi snadno dosaženo jednoduchých grafických efektů a AJAXových dotazů. Pro komunikaci s ekonomickým systémem pohoda bylo využito XML a práce s MSSQL databází, pro komunikaci s účtem České spořitelny se využívá program multi-cash, ze kterého se zpracovávají vygenerované soubory, které se upravují.

1.2 Používané styly písma

V této práci bylo využito několika stylů písma, aby byl oddělen vlastní text práce, důležité informace a ukázkové zdrojové kódy. Jedná se o tyto styly písma:

Písmo pro klasický text: Zde je ukázka klasického textu.

Písmo pro důležité informace: *Toto je velmi důležitá informace.*

Písmo pro zdrojové kódy: `<span>Ahoj světe</span>`

Písmo pro komentáře ve zdrojových kódech: `Komentář ve zdrojovém kódu`

2 WEBOVÉ STRÁNKY

Webové stránky jsou děleny na dynamické a statické, jak již bylo uvedeno výše. V následující kapitole budou ukázány základy jazyků HTML, CSS, PHP a popsány rozdíly mezi statickými a dynamickými stránkami.

2.1 Statické webové stránky

Statické webové stránky, jsou stránky, jejichž obsah je neměnný. Pro vytváření tohoto typu webových stránek postačí programátorovi základní znalosti jazyka (X)HTML ideálně ve spojení s jazykem CSS pro upravení vzhledu dané stránky. Problém ovšem nastane, jakmile by chtěl programátor provést nějakou změnu. Dané změny by musel provádět přímo ve zdrojovém kódu dané stránky. Jako příklad lze využít datum, jež se zobrazuje na stránkách.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Statická stránka</title>
  </head>
  <body>
    <h1>Toto je statická stránka</h1>
  </body>
</html>
```

Zdrojový kód 1: Kód statické stránky

Obr. 1: Zobrazení statické stránky v prohlížeči

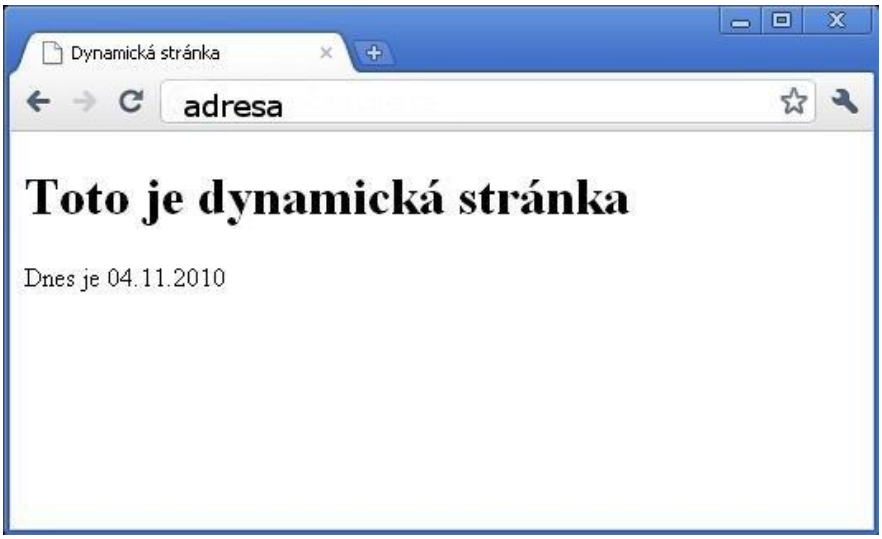
2.2 Dynamické webové stránky

Dynamické webové stránky zjednodušeně řečeno oživí statické stránky. Stránky budou reagovat na uživatelovy požadavky.

Dynamické webové stránky umožňují programátorovi přebírat z formulářových polí data, s nimiž dále pracuje (ukládá je do databáze, odesílá je emailem, vyhledává podle nich v databázích, apod.). Tyto stránky umožňují dynamicky měnit barvu pozadí podle výběru uživatele, chatovat a další věci. Ke statickým (X)HTML stránkám se přidá např. PHP, JS kód a hned získává programátor daleko větší možnosti. Při použití základní funkce v PHP může programátor jednoduše měnit obsah elementů ve stránce (např. elementu `<div>`, viz. kapitola 2.3 (X)HTML), tím zamezí načítání celé stránky, což urychluje načítání. To samé se dá samozřejmě udělat při použití jakéhokoli jiného skriptovacího jazyka.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Dynamická stránka</title>
  </head>
  <body>
    <h1>Toto je dynamická stránka stránka</h1>
    <p>Dnes je <?= date("d.m.Y") ?></p>
  </body>
</html>
```

Zdrojový kód 2: Kód dynamické stránky



Obr. 2: Dynamicky generovaná stránka

Běžný uživatel není schopen rozeznat dynamickou a statickou webovou stránku, ale díky zápisu `<?= date("d.m.Y") ?>` se mu bude datum každý den měnit. Využití zápisu uvedeného výše zajistí, že dojde ke změně data, aniž by musel programátor cokoli měnit ve zdrojovém kódu, neboť si PHP skript převezme datum od serveru.

2.3 (X)HTML

V této kapitole bude popsána historie a vývoj jazyka (X)HTML, včetně popsání základní struktury HTML souboru a několika vzorových ukázek zdrojových kódů psaných v jazyce XHTML 1.

2.3.1 Historie a vývoj (X)HTML

První verze jazyka HTML se objevila v roce 1991 a vytvořil ji Berners-Lee jako součást projektu WWW, který měl vědcům zabývajícím se fyzikou vysokých energií umožnit sdílet své výsledky po celém světě [1]. Díky tomu projekt vznikal v CERNu. Tato verze HTML byla označována jako HTML 0.9. Postupně začalo na vývoji tohoto jazyka spolupracovat více vědců, kteří začínali přidávat do tohoto jazyka další možnosti jako vytváření tabulek, stylování textů, psaní matematických vzorců a mnohé další. Kvůli zachování kompatibility Berners-Lee vytvořil pod hlavičkou IETF návrh standardu HTML 2.0, která měla dvě úrovně. První úroveň „Level 1“ pouze málo rozšiřoval předchozí verzi HTML. Ve druhé úrovni „Level 2“ navíc přibyla práce s formuláři. Po HTML 2.0 přišlo HTML 3, ale to bylo natolik náročné, že ho žádný prohlížeč nepodporoval. Na to reagovali členové W3C, shodli se na vlastnostech z HTML 3, o které rozšíří HTML 2 a vytvořili tak HTML 3.2, ovšem z HTML 3 zůstali více méně jen „okleštěné“ tabulky. Nároky klientů se stále zvyšovaly, a tak se přišlo s novým standardem HTML 4.0 (červenec 1997). Nyní se používá HTML 4.1, popř. XHTML 1.0.

2.3.2 Základy (X)HTML

Stránky (X)HTML jsou psány pomocí tzv. značek (tagů). Klasický jazyk HTML má dva druhy značek – párové a nepárové. Párová značka je složena z otevírací a uzavírající značky (např. `<strong>text</strong>`). Oproti tomu nepárové značky se nemusejí „uzavírat“ a značka může být osamocena (např. `<br>`). Toto je jedna z největších odlišností XHTML od HTML. Jelikož XHTML vycházelo z jazyka XML, tak nezná „neuzavřené“ značky, takže značky musí být uzavřené (např. `<br />`).

Každý kód, ať už HTML nebo XHTML, se skládá ze čtyř částí. Jsou to doctype, html, head, body.

Část doctype je úplně nejdůležitější, protože definuje, jakým jazykem je daná stránka psaná. Např. doctype pro HTML 4.01 vypadá následovně: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">` oproti tomu doctype pro XHTML 1.0: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`. Ihned po deklaraci doctypu následuje párová značka `<html></html>`. U XHTML má tato značka další atributy, viz. ukázkové kódy u statických a dynamických stránek. Mezi těmito značky se nacházejí zbývající dvě části.

Taktéž část head je velmi důležitá. Nacházejí se v ní informace o kódování stránky, kontakt na autora, informace pro fulltextové roboty, název stránky, klíčová slova a popis stránky.

```
…
<head>
   Nastavení kódování na utf-8
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <meta name="description" content="Popis stránky" />
   <meta name="author" content="Jméno autora (email)" />
   <meta http-equiv="Content-language" content="cs" />
   <meta name="keywords" content="Klíčová slova" />
   Informace pro fulltextové roboty – budou moci načíst obsah stránky
   <meta name="robots" content="index,follow" />
   <title>Název stránky</title>
   Vložení souboru s CSS stylem (viz. kapitola CSS)
   <link rel="stylesheet" type="text/css" href="styl.css" media="screen" />
</head>
…
```

Zdrojový kód 3: Vzorová hlavička psaná v jazyku XHTML 1

Bezprostředně po značce `</head>` následuje párová značka `<body></body>`, v němž je zapsán celý obsah stránky. Pokud bude potřebné na stránce zobrazit nějaký ze speciálních znaků (např. <, >, €, ¥ apod.), musí být tyto znaky nahrazeny tzv. HTML entitami. Například u znaků „<, >“ by si prohlížeč myslel, že se jedná o nová značka a text zapsaný mezi těmito znaky by vůbec nezobrazil.

2.4 CSS

V předchozí kapitole bylo naznačeno jak psát stránky pomocí jazyka (X)HTML. Tato kapitola ukáže, jak stránky správně formátovat pomocí kaskádových stylů CSS. Pro lepší pochopení bude uvedeno několik příkladů.

2.4.1 Úvod a historie CSS

Webové stránky se dříve stylovaly pomocí atributů jednotlivých značek, to činilo kód chaotickým a nepřehledným. Proto byl vytvořen souhrn metod pro úpravu a vzhled stránek. V roce 1994 byl zveřejněn první návrh normy, v roce 1996 byla vydána specifikace CSS 1, v roce 1998 CSS 2 a v dnešní době se vyvíjí CSS 3, ale stále ještě nemá plnou podporu od všech prohlížečů. CSS styly se využívají k formátování obsahu (X)HTML, XML. Na rozdíl od atributů CSS styly umožňují *přesně nadefinovat* jediným zápisem element pro *celý dokument*, nikoli jako atribut v každé značce příslušného elementu. Rovněž se může nadefinovat individuální vzhled jediného výskytu určitého elementu v celém dokumentu, čímž se zdrojový kód zkrátí a stane se přehlednějším.

2.4.2 Definování stylů

Každý styl je souhrn pravidel. Pravidlo je složeno ze dvou částí – *selektor* a *deklarace*. Selektor je název elementu, na který má dané pravidlo platit. Deklarace určuje, co má pro daný element platit, je uzavřena ve složených závorkách. Pokud chceme, aby každý odstavec začínal odsazením 20px, pak do stylu zadáme `p { text-indent: 20px; }`, kde p je selektor, uvnitř složených závorek je deklarace. Každá vlastnost v deklaraci je ukončena středníkem,

jejichž počet je neomezený. Pokud bude mít element více než jednu vlastnost, je dobré pro lepší přehlednost psát každou vlastnost na nový řádek s odsazením.

```
p {
	text-indet: 20px;
	color: red;
}
```

Zdrojový kód 4: Strukturování CSS kódu

Každému elementu může být pro odlišení od ostatních elementů přiřazen identifikátor. Identifikátory jsou dva – *třída (class)* nebo *id*. Musí být dodrženo, že identifikátor „id“ se na stránce vyskytne pouze jednou. Oproti tomu třída má neomezený počet výskytů. Identifikátory se přidávají jako atributy k jednotlivým značkám např. `<p id=“odstavec“> text</p>`, popřípadě `<p class=“jinyOdstavec“>text</p>`. Ve stylu se k elementu s identifikátorem id přistupuje pomocí symbolu „#“, k elementu s identifikátorem třídy pomocí „.“ (tečka).

(x)HTML:

```
…
<body>
	<div>
		Text v elementu div
		<p id=“odstavec“>Nějaký text</p>
		<p class=“jinyOdstavec“>Nějaký text</p>
	</div>
</body>
…
```

CSS:

```
body { color: #000000; } Barva zadaná v šestnáctkovém tvaru (00 – FF)
div { background-color: #eee; } Jelikož element div nemá identifikátor, tak se dané vlastnosti budou aplikovat na všechny elementy div na stránce
#odstavec { text-indent: 20px; color: #FF0000; }
.jinyOdstavec { text-indent: 15px; }
```

Zdrojový kód 5: Používání identifikátorů

Z příkladu je vidět, jak se přistupuje k elementům s identifikátorem. Pokud má být omezeno id=“odstavec“ pouze na element `<p></p>` (použitelné v případě že by id=“odstavec“ měl ještě jiný element než element `<p></p>`), tak by zápis vypadal takto `p#odstavec { vlastnost; }`, analogicky u identifikátoru třídy.

2.4.3 Dědičnost

Mezi jednotlivými selektory se dané vlastnosti dědí. Je-li nadefinováno, že mezi elementy `<body></body>` bude mít veškerý text černou barvu, tak tuto vlastnost zdědí všechny elementy uvnitř elementu `<body></body>`, jestliže nebude definováno jinak. Bude-li

pracováno s předchozí ukázkou, budou všechny texty černé barvy, kromě textu v odstavci s identifikátorem id=“odstavec“, který by měl červenou barvu textu.

2.4.4 Váha vlastností

Je-li pracováno s velmi důležitou vlastností, přidává se klauzule `!important,` která zajistí, že daná vlastnost bude platit, i když bude později ve stylu předefinována. Bude-li ve stylu nadefinováno:

```
p { text-indent: 20px !important; }
p { text-indent: 50px; }
```

Zdrojový kód 6: Klauzule !important

platí první pravidlo. Bude-li klauzule `!important` využito i ve druhém případě, bude platit pravidlo, jež bylo definováno později: `p { text-indent: 50px !importat; }`.

2.4.5 Vkládání stylů do stránky

Existuje-li představa o tom, jak má stránka vypadat, je možné využít definice vlastností určitých elementů. Definice mohou být do stránky zapracovány třemi různými způsoby. Jedním z nich je vytvoření externího souboru (*.css), přičemž je nejjednodušší vložit do elementu head link na soubor:

```
<link rel=“stylesheet“ type=“text/css“ media=“screen“ href=“cesta/nazev.css“ />
```

Tento způsob má obrovskou výhodu, v pozdější editaci stylu, kdy nemusí být pracováno s několika soubory. Všechny informace o zobrazování elementů jsou v jednom souboru, kde je možné provádět potřebné změny. Atribut „media“ určuje, zda jde o styl pro stránku „screen“, o styl pro tisk „print“, další možnosti jsou například „projection“ pro projektory, „tv“ pro přístroje na bázi televize, „handheld“ pro zařízení „do ruky“ s malým displejem a omezeným spojením, atd. Do atributu href lze dát url adresu.

Druhou možností je vložení stylu přímo do hlavičky stránky, pomocí párové značky `<style type=“text/css“>Zde bude vepsán CSS kód</style>`. Atribut „type“ není povinný, ale u XHTML 1 není kód validní. Uvnitř značky `<style></style>` nemusí být nutně jen CSS kód, ale může se tam naimportovat celý soubor pomocí `@import "cesta/styl.css",` opět může být zadána url `@import "http://server/styl.css"`.

Poslední možností je využití atributu „style“ (u jakékoli značky ve stránce). Nevýhodou této možnosti je, že pokud bude někdy nutné jakkoli upravit styl stránky, tak bude muset dojít ke změně na více místech než jen v souboru se stylem např. `<p style=“text-indent: 20px“>text</p>`. Jednotlivé vlastnosti jsou opět odděleny středníkem. Počet vlastností není omezen.

2.5 PHP

Když jsou vytvořeny statické stránky, existuje možnost jejich oživení, k čemuž slouží PHP. Jedná se o skriptovací jazyk, který řeší dotazy od klienta na straně serveru a klientovi posílá odpověď v podobě vygenerované (X)HTML stránky.

2.5.1 Historie a vývoj PHP

V roce 1994 Rasmus Lerdorf vytvořil aplikaci psanou v Perlu na počítání přístupů k jeho stránkám. Díky značnému vytěžování www-serveru (neustálé spouštění interpretu Perlu) přepsal aplikaci do jazyka C. Systém se stal natolik oblíbeným, že uživatelé přicházeli s požadavky na vylepšení celého systému. Autor proto systém rozšířil, doplnil o dokumentaci a uvolnil jej pod názvem Personal Home Page Tools. Později autor přidal do systému využívání SQL-dotazů a práci s formuláři, tento systém byl pak označován jako PHP/FI 2.0.

Brzy potom vznikla verze PHP 3.0, kterou vytvořil Andi Gutmans a Zeev Suraski, původní zkratka dostává nový význam PHP: hypertext preprocessor.

V roce 2000 byla oficiálně uvolněna verze PHP 4, která byla mnohem výkonnější než verze 3.0 a přidávala možnosti sessions, bufferingu a větší zabezpečení zpracovávání vstupů uživatele.

V červnu 2003 oficiálně vyšla betaverze PHP 5, kde byla největší změna v objektovém modelu. PHP se přibližuje k ostatním jazykům podporujícím objektově orientované programování (dále jen OOP). Současná verze je PHP 5.3.3 (vydána 22. 7. 2010) a ve vývoji je verze PHP 6.

2.5.2 Co je potřeba ke psaní PHP skriptů

Základní nástroj, který bude při psaní PHP skriptů potřeba, je editor. Jelikož se PHP skripty velmi často kombinují s (X)HTML kódem, vystačí jakýkoli editor pro tvorbu webových stránek. Editory usnadňují práci například doplňováním názvů funkcí, barevným odlišováním proměnných od názvů funkcí apod.

Dalším nástrojem pro psaní skriptů je www-server, který podporuje PHP skripty. Server ani nemusí být přístupný z internetu. Pro „testovací“ potřeby stačí programátorovi nakonfigurovat server na jeho počítači. Na platformě Windows se dá využít program Xampp nebo EasyPHP, přičemž oba dva umějí spolupracovat s MySQL databází. Na platformě UNIX (Linux) je nejlepší využít balíky (Debian like distribuce) pro instalaci serveru Apache s podporou PHP skriptů (často označováno zkratkou LAMP – Linux, Apache, Mysql, PHP).

2.5.3 Test konfigurace serveru

PHP skripty jsou obyčejné stránky doplněné o výkonné příkazy. Aby od sebe server odlišil statické (X)HTML stránky a PHP skripty, ukládají se PHP skripty do souborů se zvláštní příponou. Bývají to nejčastěji *.phtml, *.php – dle konfigurace serveru.

Pro kontrolu, zda na serveru vše funguje, je dobré jako první skript na serveru vytvořit soubor index.php s obsahem `<?php phpinfo() ?>`. Soubor musí být umístěn v DocumentRootu webového serveru, aby bylo možné vidět výsledek v prohlížeči (na UNIXových systémech nejčastěji /var/www/). Z tohoto příkladu je patrné, že PHP skripty začínají `<?php`, `<?` (popř. `<?=` které zastupuje příkaz echo), následuje obsah skriptu (`phpinfo()`) a jsou ukončeny pomocí `?>`. Výslednou stránku naleznete na obr.3.

Obr. 3: Výsledek spuštění funkce phpinfo()

Pokud je vypsána stránka podobná této, podpora PHP skriptů na www-serveru funguje. Zde se navíc zjistí, které moduly mají PHP zapnuté, např. zda můžeme využít databázi MySQL nebo nějakou jinou databázi.

2.5.4 Základy syntaxe

PHP skript je složen ze sekvence příkazů. Jejich syntaxe je velmi podobná jazyku C. Jednotlivé příkazy se dají ukončovat dvěma způsoby. Jednou možností je uvedení každého příkazu mezi znaky `<?php` a `?>`:

```
<?php $a = 5 ?>
<?= $a ?>
```

Zdrojový kód 7: Oddělování příkazů v PHP

Druhou možností je uvést více příkazů najednou, musí se oddělit středníkem. Je dobré dodržovat pravidlo: „co příkaz to jeden řádek“, z důvodu přehlednosti kódu:

```
<?php
	$a = 20;
	echo $a;
?>
```

Zdrojový kód 8: Oddělování příkazů v PHP pomocí středníků

V PHP se dají použít jakákoli písmena (malá, velká, či jejich kombinace). Je tedy jedno, zda se zapíše `echo`, `Echo` nebo `ECHO`. Na velikosti písmen záleží pouze u názvu proměnných. `$a`, `$A` jsou dvě různé proměnné.

U dlouhých skriptů je dobré psát komentáře. Komentář nikterak neovlivňuje kód, je to jen poznámka programátora pro ty, kteří po něm budou číst kód popř. nějak jej upravovat. Jsou dva druhy komentářů – řádkový komentář a více řádkový komentář. Jednořádkový komentář se zapisuje pomocí znaku `//` (dvě lomítka). Vše od této dvojice znaků až dokonce řádku je ve skriptu ignorováno. Druhý druh komentáře je vhodný pro delší poznámky.

Poznámka se nachází mezi dvojicí znaků `/*` a `*/`. Tento druh komentáře může zabírat více řádků než jen jeden.

Proměnné v PHP vždy začínají znakem `$`, po kterém následuje jméno proměnné. V PHP není potřeba deklarovat proměnné předem jako je tomu například v jazyce C, proměnná je totiž deklarována okamžikem, kdy je poprvé použita. Pokud se do proměnné neuloží žádná hodnota přiřazovacím příkazem (`=`), má proměnná hodnotu prázdného řetězce. V okamžiku přiřazení se automaticky určí její typ. PHP má pět typů *integer, double, string, array, object.*

Nadefinované proměnné můžeme porovnávat a dále s nimi pracovat pomocí tzv. operátorů. Operátorů jsou v PHP 4 druhy a to *matematické operátory, operátory pro manipulaci s bity a čísly, logické operátory* a *operátory pro spojování řetězců*. Matematické operátory se používají pro matematické operace jako je sčítání (`+`), odčítání (`-`), násobení (`*`) a dělení (`/`). Dále je zde operátor k určení zbytku po celočíselném dělení (`%`). Při použití jakéhokoli matematického operátoru platí matematická pravidla (násobení má přednost před sčítáním apod.). Operátory pro manipulaci s bity a čísly jsou logický součin (`&`), logický součet (`|`), nonekvivalence (`^`), negace (`~`) a operatory pro bitový posun (`<<`, `>>` - podle toho zda budeme chtít posunout bity doleva či doprava). Logické operátory se používají převážně jako podmínka v cyklech nebo podmíněných výrazech a nabývají hodnot *true* nebo *false*. Mezi tyto operátory patří operátor porovnání (`==`, popř. `===`), nerovnosti (`!=`, `>`, `>=`, `<`, `<=`), logický součin (`&&`) a logický součet (`||`). Rozdíl u porovnávání mezi `==` a `===` je velice důležitý. Při použití operátoru `==` se porovnává pouze hodnota proměnných, ale při použití operátoru `===` se porovnává hodnota i datový typ proměnných.

```
$a = 5;
$b = "5";
$a == $b vrátí true
$a === $b vrátí false protože se liší datové typy proměnných (int a string)
```

Zdrojový kód 9: Porovnávání v PHP

Operátor pro spojování řetězců (`.`) funguje tak, že své operandy převede nejprve na znakové řetězce a pak je spojí v jeden řetězec.

Každý složitější program potřebuje reagovat na okolní podmínky. Podle nich se pak různě větví aby ošetřil všechny možnosti, které mohou nastat. PHP nabízí 2 příkazy, které zajišťují další větvení programu. Tyto příkazy jsou `if` a `switch`. Příkaz `if` slouží k podmíněnému provedení provedení příkazu, pokud je splněna určitá podmínka. Podmínka se musí zadat jako výraz, který vrací logickou hodnotu.

```
If( $jmenovatel == 0 ){ echo „jmenovatel musí být různý od nuly“;}
```

Zdrojový kód 10: Podmínka IF na ošetření nenulového jmenovatele

Při složitějších podmínkách je samotný příkaz `if` nedostačující a je nutné ho skombinovat s příkazy `else`, popřípadě `elseif`. Výsledná podmínka pak může vypadat následovně:

```
if( $a < $b ){
	echo „a je menší než b“;
} elseif( $a == $b ) {
	echo „a je rovno b“;
} else {
	echo „a je větší než b“;
}
```
Zdrojový kód 11: Větvení podmínek

V případě, že na základě hodnoty jednoho výrazu potřebujeme provést jednu z několika větví skriptu, je dobré využít příkaz `switch`. Syntaxe příkazu switch je velmi následující:

```
switch( $pismeno ){
	case ‚a‘:
		echo „Písmeno a“;
	break;
	case ‚b‘:
		echo „Písmeno b“;
	break;
	default:
		echo „Takové písmeno neznám“;
	break;
}
```
Zdrojový kód 12: Příkaz switch

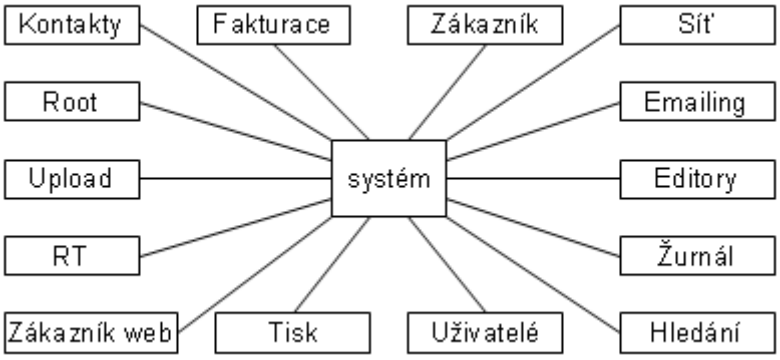
Příkaz `switch` postupně prochází hodnoty zapsané za slovem `case`, dokud nenalezne shodu s výrazem (v tomto příkladu `$pismeno`), pokud nenalezne žádnou shodu, tak se provedou příkazy za slovem `default`.

Často je potřeba spustit některou část skriptu opakovaně. K tomuto PHP nabízí 4 cykly a to cyklus `while`, `do-while`, `foreach` a cyklus `for`. Každý je vhodný pro něco jiného. Cyklus `for` se hodí v případě, že přesně víme počet opakování, u ostatních cyklů se zadává logická podmínka, která cyklus zastavuje. Ukázka syntaxe cyklů:

```
while( podmínka ){ příkazy; }
do{ příkazy }while( podmínka )
for( $i=0; $i<$konec; $i++ ){ příkazy; }
foreach( $pole AS $k=>$h ){ příkazy; }
```
Zdrojový kód 13: Syntaxe cyklů

3 NÁVRH SYSTÉMU

Jelikož systém vycházel ze starší verze, která byla rozdělena do několika modulů, bylo rozhodnuto, že i nový systém bude z důvodu přehlednosti a zaběhlosti rozdělen na moduly (obr. 4). Každý z modulů je určen pro správu různých věcí. Dále systém obsahuje rozhraní pro komunikaci s bankou Česká Spořitelna, ekonomickým systémem Pohoda, externím trouble ticket systémem a VoIP systémem. Veškerá data uložená v systému musí být přístupná zákazníkům v zákaznické zóně. Při navrhování systému se zvažovalo, zda systém psát funkcionálně, nebo zda využít ZF.



Obr. 4: Moduly v systému

3.1 Moduly v systému

- **Modul Kontakty:** Modul pro správu údajů o zákaznících, včetně mazání a přidávání zákazníků. Tento modul bude sloužit k povolení/zakázání zákazníka pokud nebude veden jako ISP.
- **Modul Síť:** Modul pro správu informací o síti, evidence informací o vysílačích, shaperech (rozdělení šířky pásma dle daných kritérií) a skupinách uživatelů.
- **Modul Zákazník:** Jedná se o nejdůležitější modul. Zde budou uvedeny jednotlivé služby poskytované daným zákazníkům. Dále je zde umožněn provádět veškeré operace s položkami v modulu zákazník včetně vkládání fotek, dokumentů a zaznamenaných telefonních hovorů.
- **Modul Fakturace:** V tomto modulu se generují faktury vždy první den v měsíci dle podkladů získaných v modulu zákazník. Mimo jiné je zde evidence faktur dle měsíců. S jednotlivými fakturami lze plně pracovat (editovat, mazat, hradit).
- **Modul Emailing:** Modul sloužící k odesílání hromadných e-mailů s možností přiložit PDF soubor. Taktéž je zde automatické odesílání emailu o provedené fakturaci včetně možnosti přílohy PDF faktury.
- **Modul Editory:** Zde se dají editovat vzorové dokumenty pro zákazníky, jako jsou smlouva, dodatek smlouvy, technická specifikace, předávací protokol, protokol deinstalace, text k emailové zprávě o provedené fakturaci a statická část konfiguračního souboru pro smokeping (aplikace pro monitorování dob odezvy strojů v síti).
- **Modul Žurnál:** V tomto modulu se nacházejí veškeré informace o provedených operacích v systému a zákaznické zóně s možností vyhledávání.

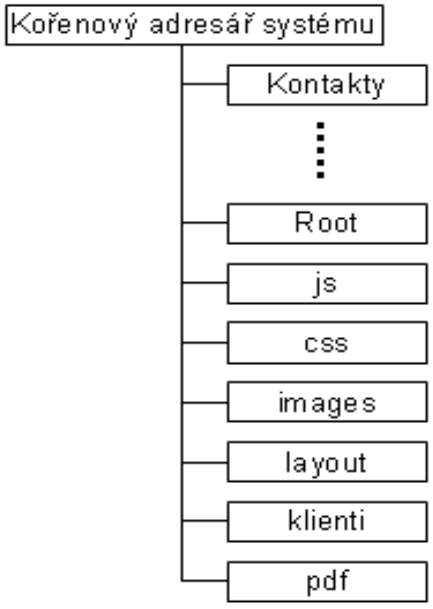
- **Modul Tiskové výstupy:** Tato část systému je určena zejména pro manažery. Obsahuje grafy tržeb za poslední rok, uzavřených/vypovězených smluv, počet připojených zákazníků dle technologie. Další výstupy jsou výpis zákazníků s platebními prázdninami, výpis potencionálních zákazníků dle data podání žádosti o připojení, výpis zákazníků dle data připojení atd.
- **Modul Uživatelé:** V tomto modulu se nachází správa uživatelů, kteří mají přístup do systému včetně nastavování individuálních oprávnění k jednotlivým modulům.
- **Modul Hledání:** Modul slouží k rychlému vyhledávání kontaktů dle IP adresy, MAC adresy, číslo smlouvy, telefonu popř. variabilního symbolu faktury.
- **Modul Rt:** Modul Rt je určen k propojení s externím trouble ticket systémem. Eviduje zákazníky, kteří mají ticket v externím systému.
- **Modul Zákazník web:** Tento modul je určený pro správu zákaznické sekce např. nahrávání souborů „ke stažení“, úvodní text v systému po přihlášení.
- **Modul Upload:** Modul slouží k nahrávání veškerých dokumentů zákazníků, fotek z instalací, zaznamenaných hovorů, fotek a dokumentů vysílačů.
- **Modul Root:** Tento modul je určen pro administrátory sítě. Nachází se zde evidence routerů, IP subnetů, vlanů (virtuální lan – vice logických lan sítí nad jedním fyzickým přenosovým mediem) a aktualizace konfiguračního souboru pro aplikaci smokeping.

3.2 Systém programovaný funkcionálně

Při tomto způsobu programování je vždy na začátku souboru PHP kód, který například přebírá data z formulářů, kontroluje, zda je uživatel přihlášen (sezení), nebo se dotazuje na databázi. Po tomto PHP skriptu následuje část (X)HTML, která určuje, co všechno se má na dané stránce zobrazit. Jelikož jsou obě části v jednom souboru, tak se může stát, že celý skript bude nepřehledný.

3.2.1 Adresářová struktura

Z důvodu přehlednosti je dobré od sebe oddělit jednotlivé moduly, CSS styly, JS skripty, layout stránky atd. (obr.5).



Obr. 5: Adresářová struktura

V kořenovém adresáři má každý modul svou složku, ve které jsou skripty pro jednotlivé operace v daném modulu. Ve složce „js“, jsou veškeré javascriptové soubory (jquery.js, jquery.plot.js, …), obrázky jsou ve složce „images“, hlavička a patička stránky, které se vkládají do stránky jsou ve složce „layout“. Složka „klienti“ je určená pro fotky, dokumenty, záznamy hovorů jednotlivých zákazníků. Ve složce „pdf“ jsou uloženy skripty, fonty pro generování *.pdf souborů. V kořenovém adresáři je ještě soubor index.php ve kterém je úvodní stránka pro přihlášení do systému. Dále zde může být soubor na spojení s databázovým serverem, popř. se pro tyto konfigurační soubory může udělat speciální složka.

3.2.2 Nastavení webového serveru (apache2 – platforma linux)

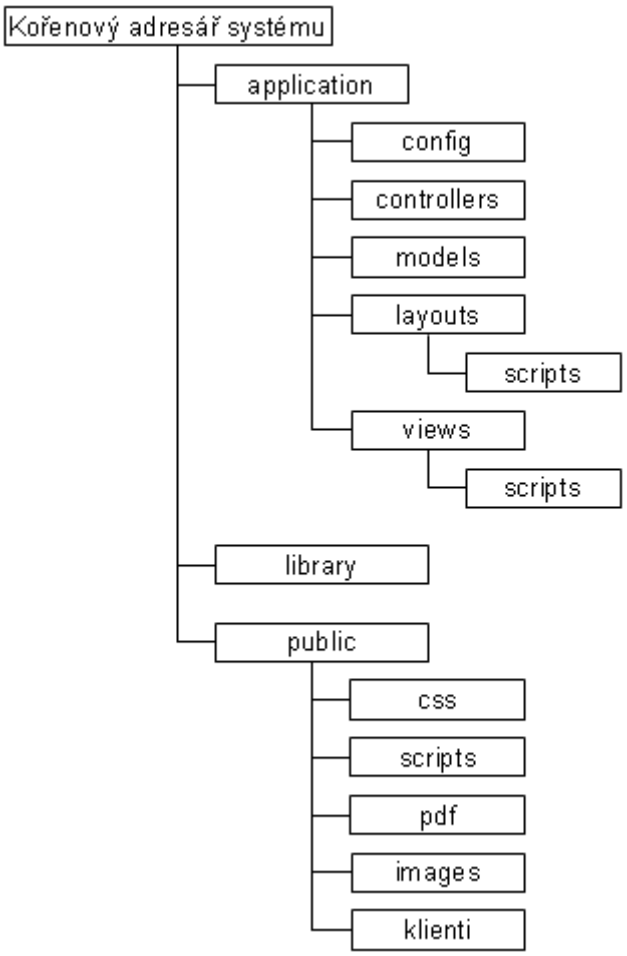
Po nainstalování web serveru s podporou PHP5 a MySQL stačilo pouze upravit výchozí konfigurační soubor. V souboru /etc/apache2/sites-enabled/sites.conf se upravily direktivy ServerName a DocumentRoot. Direktiva ServerName nastavuje „hostname“ pod kterým se server identifikuje, DocumentRoot nastavuje cestu k souborům daného webu (v tomto případě cesta ke kořenovému adresáři systému).

3.3 Systém programovaný objektově s využitím ZF

ZF je framework, který využívá OOP pro tvorbu webových aplikací psaných v jazyce PHP. Díky OOP jsou případné úpravy jednodušší, než je tomu v případě procedurálního programování, protože se úpravy dělají jen na jednom místě (v třídě) a nemusí se dělat ve více souborech.

3.3.1 Adresářová struktura

Adresářová struktura je komplikovanější než v případě procedurálního programování (obr.6.). ZF využívá návrhový vzor MVC, který rozděluje programovou část (PHP skripty, které přebírají data z formulářů, kontrolují, zda je uživatel přihlášen, …) a zobrazovací část (např. výpis dat vytažených z databáze). To činí zdrojové kódy velice přehledné a jednoduché na úpravu. Velká výhoda této adresářové struktury je, že složky application a library nejsou přístupné z webu. Výhoda spočívá ve větší bezpečnosti, protože ve složce application jsou citlivá data, jako například přihlašovací údaje do databáze.



Obr. 6: Adresářová struktura pro ZF

Z obr. 6. je patrné, že se adresářová struktura dělí na 3 základní složky application, library a public. Ve složce application jsou konfigurační soubory (.ini soubory například pro připojení k databázi) ve složce config, ve složce controllers jsou třídy pro každý modul, které mají jako metody jednotlivé operace v modulu (programová část). Složka models je určena pro vlastní vytvořené třídy, ve složce layout/scripts jsou všechny soubory týkající se rozložení stránky. Soubory, které zobrazují obsah operací v jednotlivých modulech, jsou uloženy ve složce views/scripts ve které jsou složky pro každý modul. Složka library je určena pro veškeré knihovny, které se používají (Zend, ZendX, …), ve složce public jsou dále v jednotlivých složkách soubory .css pro nastylování stránky, JS soubory pro AJAX dotazy a akce před načtením stránky, fonty pro generování .pdf souborů, obrázky, a složka pro soubory jednotlivých zákazníků.

3.3.2 Nastavení webového serveru (apache2 – platforma linux)

Konfigurace web serveru je skoro stejná jako v předchozím případě. Stačí změnit direktivy ServerName a DocumentRoot (u aplikace využívající ZF je DocumentRoot nastaven na složku public), k tomu ještě ale musí být zapnut rewrite mod. Rewrite mod se povol9 na platform2 Linux příkazem `a2enmod rewrite`. Aby rewrite mod fungoval, musela se

v konfiguračním souboru nastavit direktiva AllowOverride na hodnotu all, což umožní využít soubor .htaccess pro nastavení rewrite modu.

```
RewriteEngine on
RewriteRule !\.(js|ico|gif|jpg|png|jpeg|css|htm|rar|pdf|swf|mp3|wav)$ index.php
```

Zdrojový kód 14: Soubor .htaccess – nastavení rewrite módu

První řádek zapíná rewrite mod a druhý definuje pravidlo, které bude, vše co nemá jednu z výše uvedených koncovek přepisovat na soubor index.php, který zpracuje požadavek a vypíše odpověď.

Druhá možnost je zapnout rewrite mód přímo v konfiguračním souboru.

```
<Location />
   RewriteEngine on
   RewriteRule !\.(js|ico|gif|jpg|png|jpeg|css|htm|rar|pdf|swf|mp3|wav)$ index.php
</Location>
```

Zdrojový kód 15: Nastavení rewrite módu v konfiguračním souboru pro VirtualHost

3.4 Zhodnocení návrhů

Každý z návrhů se hodí pro různé aplikace, má své výhody a nevýhody, proto bylo důležité správně zhodnotit obě možnosti a zvolit tu vhodnější pro tento systém. Je potřeba vzít v potaz, že systém je modulový, takže by se moduly měli do systému moduly jednoduše přidat a odebrat.

3.4.1 Zhodnocení procedurálně programovaného systému

+ Stačí pouze základní znalost PHP
+ Jednoduchá adresářová struktura
+ Jednoduché přidávání a odebírání modulů
+ Jednoduchá konfigurace web serveru

− Kontrola přihlášení v každém souboru
− Složitá validace údajů
− Složité zabezpečení komunikace s databázovým serverem
− Nepřehledné kódy

3.4.2 Zhodnocení systému programovaného objektově s využitím ZF

+ Jednoduché přidávání a odebírání modulů
+ Větší zabezpečení citlivých údajů
+ Kontrola přihlášení v jednom souboru
+ Předepsané validační třídy
+ Jednoduché úpravy díky OOP (v jednom souboru)
+ Přehledné kódy

- Složitější adresářová struktura
- Složitější konfigurace web serveru

3.4.3 Volba systému

Po zhodnocení obou návrhů je patrné, že procedurálně programovaný systém je lepší spíše pro méně rozsáhlé aplikace, kvůli úpravám na více místech. Naopak systém programovaný objektově je vhodný pro rozsáhlé aplikace, protože případné úpravy se provedou v definici třídy a ty se projeví při každé instanci dané třídy a nemusí se hledat každý soubor, kde se daná třída využívá. Jelikož je požadovaný systém rozsáhlý, bylo rozhodnuto, že systém bude programovaný objektově s využitím ZF zejména pro velký počet předepsaných tříd, zejména třídám k validaci vstupních dat.

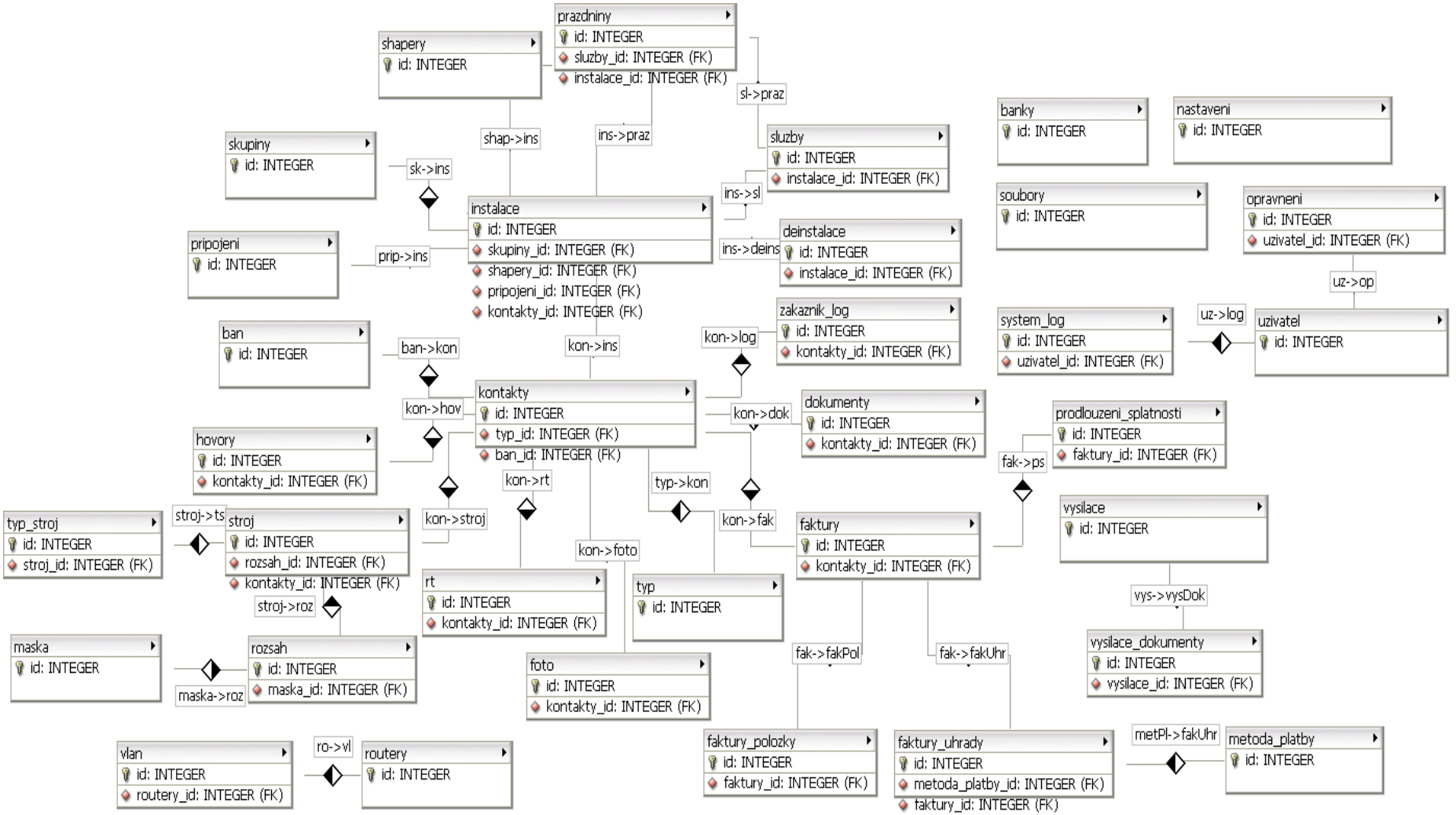
4 NÁVRH DATABÁZE

Jelikož celý systém běží na linuxovém serveru, tak se hned nabízejí 2 volné databázové servery a to MySQL a PostrgeSQL. Protože systém je založen na starší verzi systému, který využíval databázi MySQL, bylo zvoleno využití databáze MySQL pro snadnější migraci dat ze starého do nového systému. Původní návrh databáze se skládal ze třiceti šesti tabulek obr. 7 (zadavatel, nechtěl zobrazit kompletní strukturu databáze z důvodu bezpečnosti). V databázi se dále nachází 5 pohledů, které měli usnadnit a urychlit zejména výpis faktur, žurnálů a zjednodušit SQL dotazy při práci s jednotlivými stroji.

4.1 Pohledy

Pohled je statický dotaz, se kterým se pracuje jako s tabulkou, která má dynamický obsah vypočítaný v okamžiku použití. Používají se zejména při zjednodušování složitých dotazů. Například dotaz: `SELECT tb1.sloupec1, tb2.sloupec3 FROM tb1 LEFT JOIN tb2 ON tb2.id = tb1.tb2_id` se dá zapsat jen `SELECT * FROM pohled`. Pohled může spojovat 0 a více tabulek. Vytvářejí se příkazem `CREATE VIEW nazev_pohledu AS SQL dotaz`.

- **Pohled kontakty – log:** Tento pohled slučuje údaje z tabulky zakaznik_log a kontakty. Používá se zejména v tiskových výstupech u výpisu poruch. Zjednodušení dotazu je nejvíce znatelné na formátu data a sloučení jména. V dotazu se už nemusí používat funkce `CONCAT()` pro spojení 2 a více polí, a funkce `DATE_FORMAT()` pro úpravu data do formátu na který jsme zvyklí.
- **Pohled rozsah – gw:** Pohled rozsah – gw je využíván pouze při synchronizaci používaných rozsahů v systému s údaji na routerech. Problém nastal při porovnání údajů z routeru, které nebyly vždy IP rozsahu, ale občas to byla IP adresa brány. Z IP adresy brány by se musela zjistit IP adresa rozsahu, který by se mohl porovnávat vůči databázi, a tím už vzniká možnost chyby, kterou tento pohled vymezil.
- **Pohled stroj – kontakty:** Tento pohled slučuje data z tabulek stroj, typ_stroj, maska, rozsah, kontakty. IP adresa u každého stroje se skládá z rozsahu a konečného čísla, které tento pohled složí dohromady, přidá k nim údaje o masce, typu stroje a informace o uživateli, který má daný stroj přiřazený. Největší využití má u detailu zákazníka v modulu kontakty.
- **Pohled uživatel – log:** Tento pohled má obdobnou funkci jako pohled kontakty – log, akorát slučuje tabulky system_log a uzivatel. Nejvíce se využívá v modulu žurnál.
- **Pohled uživatel – faktury:** Nejkomplikovanější pohled, slučuje data za tabulek kontakty, faktury, faktury_polozky, faktury_uhrady. Vypočítával, zda je daná faktura před nebo po splatnosti, likviditu faktury, bohužel byl pohled natolik náročný, že při větším množství faktur trval jednoduchý dotaz přes 10 sekund. Řešení bylo vytvořit 2 pomocné tabulky sum_faktury_polozky a sum_faktury_uhrady, do kterých se ukládaly sumy cen a úhrad. Doba vykonávání dotazu se zkrátila na 6 ÷ 7 sekund. 6 sekund na jeden dotaz je pořád hodně, proto se tento pohled využívá pouze při automatickém zakazování, kde nevadí dlouhá doba vykonávání dotazu.



Obr. 7: Struktura databáze

5 TVORBA SYSTÉMU

Po zvolení způsobu programování systému a navržení databáze bylo možné přistoupit k samotnému programování systému.

5.1 Příprava ZF

Aby ZF správně fungoval, je nutné si připravit adresářovou strukturu (obr.6), nastavit web server dle sekce 4.2.2 poté je nutné vytvořit soubor index.php v adresáři public, aby rewrite mod měl kam přeposílat požadavky. V tomto souboru se definují konstanty, include_path, spouští se automatické nahrávání souborů s třídami, které se budou dále využívat a nahrává se soubor (bootstrap.php ve složce application) s nastavením návrhového vzoru MVC.

```
Definování konstant a nastavení include path
define('APPLICATION_ENV', 'development');
define('APPLICATION_PATH', realpath(dirname(__FILE__) . '/../application/'));
set_include_path(get_include_path() . PATH_SEPARATOR
 . '../application/models' . PATH_SEPARATOR
 . '../application/controllers' . PATH_SEPARATOR
 . '../library' . PATH_SEPARATOR
);

Spuštění automatického nahrávání souborů s třídami
require_once 'Zend/Loader/Autoloader.php';
$autoloader = Zend_Loader_Autoloader::getInstance();
$autoloader->setFallbackAutoloader(true);

Pokus o nahrání souboru bootstrap.php
try {
    require '../application/bootstrap.php';
} catch (Exception $exception) {

V případě, že se nahrání nezdaří vypíše se chyba
    echo '<html><body><center>'
       . 'Při spouštění aplikace nastala chyba.';
    if (defined('APPLICATION_ENV') && APPLICATION_ENV != 'production' ) {
        echo '<br /><br />' . $exception->getMessage() . '<br />'
           . '<div align="left">Stack Trace:'
           . '<pre>' . $exception->getTraceAsString() . '</pre></div>';
    }
    echo '</center></body></html>';
    exit(1);
}
```

Zdrojový kód 16: Soubor index.php

```
nastavení časové zóny
date_default_timezone_set('Europe/Prague');

kontrola nadefinovaných konstant
defined('APPLICATION_PATH')
    or define('APPLICATION_PATH', dirname(__FILE__));

defined('APPLICATION_ENV')
    or define('APPLICATION_ENV', 'development');
```

```
spuštění controlleru (reaguje na uživatelské akce)
$frontController = Zend_Controller_Front::getInstance();
$frontController->setControllerDirectory(APPLICATION_PATH . '/controllers');
$frontController->throwExceptions(true); Pro vypnutí zadat false
$frontController->registerPlugin(new Zend_Controller_Plugin_ErrorHandler());

nastavení zobrazovací části
$pohled = new Zend_View();
$pohled->setScriptPath(APPLICATION_PATH . '/views/scripts/');

Konfigurace spojení s databází (viz. Konfigurační soubor app.ini)
$config = new Zend_Config_Ini('../application/config/app.ini', 'config');
$db = Zend_Db::factory('Pdo_Mysql', $config->db->toArray());
Zend_Registry::set('db',$db);

Konfigurace sessions
$session = new Zend_Config_Ini('../application/config/app.ini', 'session');
Zend_Session::setOptions($session->toArray());

Nastavení Locales
$locale = new Zend_Locale('cs_CZ');
Zend_Registry::set('Zend_Locale', $locale);

Registry
$registry = Zend_Registry::getInstance();
$registry->configuration = $config;
$registry->dbAdapter = $db;

Spuštění layoutu
Zend_Layout::startMvc(APPLICATION_PATH . '/layouts/scripts');

Nastavení layoutu
$view = Zend_Layout::getMvcInstance()->getView();
$view->doctype('XHTML1_STRICT'); Nastaví doctype

Vyčištění proměnných
unset($frontController,$view,$config,$registry, $pohled, $router, $locale);
```

Zdrojový kód 17: Soubor bootstrap.php

```
[config]
db.type     = pdo_mysql
db.host     = server
db.username = uživatel
db.password = heslo
db.dbname   = název databáze

[session]
name = maa
save_path = /var/www/cookies
use_only_cookies = on
remember_me_seconds = 864000
```

Zdrojový kód 18: Konfigurační soubor app.ini

5.2 Spuštění ZF

Po vytvoření adresářové struktury, souborů index.php, bootstrap.php a app.ini, se může ZF spustit.

5.2.1 Url v ZF

Protože ZF využívá rewrite módu jsou url trochu odlišná od klasických adres. Máme-li adresu http(s)://server.domena.cz/slozka/soubor rewrite mód způsobí, že část /slozka/soubor se přepíše na soubor index.php, pokud koncovka souboru nebude jedna ze seznamu v sekci 4.2.2. Díky tomu může url vypadat http(s)://server.domena.cz/modul/akce-v-modulu/. Kde modul je controller (programová část) určený pro daný modul a akce-v-modulu je metoda v daném controlleru. Pokud není určen controller, ve výchozím nastavení ZF použije IndexController, to samé platí v případě, že není určena akce (indexAction).

5.2.2 Pomocné třídy

Pomocné třídy jsou třídy, ze kterých vychází každý controller, zajišťují kontrolu přihlášení, připojení k databázovému serveru a podobně. V systému se používají 2 pomocné třídy. Jedna pro controllery před přihlášením (Index) a druhá pro controllery po přihlášení (Controller). Pro třídu po přihlášení se navíc načítají data o nastavení, jinak jsou třídy prakticky stejné. Aby správně fungoval návrhový vzor MVC, obě třídy rozšiřují třídu Zend_Controller_Action.

```
class Index extends Zend_Controller_Action {
	…
}
```

Zdrojový kód 19: Deklarace pomocných tříd

Vše co je potřeba nastavit, zkontrolovat se zadá do veřejné funkce init().

```
public function init(){
	kontrola zda je uživatel přihlášen
	$auth = Zend_Auth::getInstance();
	if ( $auth->hasIdentity() ) {
	uživatel je přihlášen - načtou se data o něm
		$data = $auth->getStorage()->read();
		$this->view->data = $data;
		$this->_userData = $data;
	}

	spojení s db
	$this->_db = Zend_Registry::get("db");
	$this->_db->query("SET names utf8"); nastavení kódování databáze
	přesměrovávání
	$this->_redirector = $this->_helper->getHelper('Redirector');

	složení základní url
	$request = new Zend_Controller_Request_Http();
	switch( $_SERVER['SERVER_PORT'] ){
	case 80:
		$http = 'http://';
	      break;
```

```
            case 443:
                   $http = 'https://';
            break;
            default:
                 $http = 'https://';
            break;
      }
      $baseUrl = $http . $_SERVER['SERVER_NAME'] . $request->getBaseUrl() . '/';
      $this->view->baseUrl = $baseUrl;
}
```

Zdrojový kód 20: Veřejná metoda init()

5.2.3 Úvodní stránka

K zobrazení úvodní stránky je potřeba vytvořit v programové části systému soubor IndexController.php a v zobrazovací části složku index, ve které budou soubory pro zobrazení obsahu jednotlivých akcí v controlleru. Funkce úvodní stránky spočívá pouze ve zjištění, zda je uživatel přihlášen nebo ne, a podle toho zobrazit menu systému popřípadě přihlašovací formulář.

```
class IndexController extends Index {

      konekce na db, kontrola přihlášení
      public function init(){
            parent::init();
      }

      úvodní stránka
      public function indexAction(){

      }
}
```

Zdrojový kód 21: Soubor IndexController.php

Ze zdrojového kódu 21 je patrné, že třída IndexController rozšiřuje pomocnou třídu Index. Díky tomu se dalo využít dědičnosti v OOP, kontrolu přihlášení, nastavení spojení s databázovým serverem atd. se tím zkrátilo na jeden řádek (`parent::init()`). Pro úvodní stránku je dále nutné vytvořit soubor index.phtml v zobrazovací části, ve složce index.

```
<div class="nadpis">Pro správnou funkčnost je nutno zapnout cookies</div>
```

Zdrojový kód 22: Obsah souboru index.phtml

5.3 Layout

Pod pojmem layout se skrývá rozložení stránky. Protože se starším systémem pracovalo hodně lidí, rozhodlo se využít stávající layout, aby se zaměstnanci nemuseli učit pracovat s celým systémem znovu.

5.3.1 Funkce layoutu

Funkce layoutu spočívá v rozdělení statického rozložení stránky, a dynamického prvku stránky, který generuje například PHP skript. To činí skripty ještě více přehledné, když je v zobrazovací části skriptu složitější PHP kód. Hlavní rozložení stránky se v ZF skládá z jednoho popřípadě více souborů *.phtml.

5.3.2 Vytvoření layoutu v ZF

Ve zdrojovém kódu 17 se nachází řádek `Zend_Layout::startMvc(APPLICATION_PATH . '/layouts/scripts');` který definuje cestu ke složce se soubory s rozložením stránky. V případě, že se nenadefinuje jinak, ZF hledá v této složce soubor layout.phtml ve kterém se nachází statické rozložení stránky. Pokud se v layoutu vyskytují PHP skripty (např. kontrola práv, kontrola přihlášení, …), je vhodné vložit tyto části do externích souborů a do layoutu je vložit pomocí funkce `require()` popř. `include()`. Výsledný layout je vidět na obr. 8.

Obr. 8: Rozložení stránky v systému

Vše co má černé pozadí se nachází v externích souborech, pro zpřehlednění kódu celkového layoutu. Do elementu (při rozložení stránky nejčastěji element `<div></div>`), ve kterém se dynamicky mění obsah (obr.8 - žluté pozadí), se zobrazení příslušného obsahu zajistí PHP kódem: `<?= $this->layout()->content ?>`.

5.4 Přihlašování a odhlašování ze systému

Celý mechanismus přihlášení není úplně jednoduchý, proto bylo rozhodnuto vytvořit třídu „Prihlaseni“ ve které budou metody pro přihlášení a odhlášení uživatele. Tento krok rozdělil přihlašování na dvě části – validace dat a samotné přihlašování.

Validací dat je myšlena kontrola údajů vyplněných v přihlašovacím formuláři. Kontrolovat se v tomto případě bude délka řetězců vyplněných v polích login a heslo.

```
$validator = new Zend_Validate_StringLength(2,20);
if( !$validator->isValid( $login ) ){
      $chyby[] = "Login musí mít 2 ÷ 20 znaků";
}
if( !$validator->isValid( $heslo ) ){
      $chyby[] = "Heslo musí mít 2 ÷ 20 znaků";
}
```

Zdrojový kód 23: Validace délky předaných řetězců

V případě, že jsou oba řetězce validní, provede se dvojí hash hesla, který se společně se loginem předá metodě pro přihlášení uživatele. Dvojí hash hesla se provádí kvůli zlepšení

bezpečnosti. Pokud login a heslo odpovídají nějakému záznamu v databázi, načtou se údaje o oprávnění z databáze a uloží se do cookies.

Metoda pro odhlášení akorát odstraní veškeré údaje uložené o uživateli, pokud vše řádně proběhne, je uživatel přesměrován na úvodní stránku.

5.5 Modul Kontakty

Jelikož se jedná o modul pro správu zákazníků, budou zde potřeba akce na výpis, přidání, editaci, detail, smazání, zakázání a povolení jednotlivých zákazníků. Dále do tohoto modulu byl umístěn backend pro našeptávače.

5.5.1 Příprava controlleru

Ke kontrole přihlášení, spojení s databází a načtení informací o oprávnění uživatele stačí pouze rozšířit pomocnou třídu Controller. Protože má modul své menu, je dále potřeba určit jaké menu se má vložit do layoutu. Controllery pro všechny moduly se definují stejně, jen se mění název controlleru a proměnná `$layout->menu`.

```
class KontaktyController extends Controller {
	spojení s db + kontrola autentizace
	public function init(){
		parent::init();
		$layout = new Zend_Layout;
		$layout->menu = 'kontakty.phtml';
	}
}
```

Zdrojový kód 24: Definice třídy pro modul kontakty

Každá funkce začíná kontrolou práv. Práva jsou řešena pomocí písmen r,w,d (čtení, zápis/editace, smazání).

```
if( strpos($this->_userData->root, 'w') !== false ){
} else {
	$this->_helper->viewRenderer->render(‚prava‘, true, true);
}
```

Zdrojový kód 25: Kontrola práv uživatele

5.5.2 Přidání a editace zákazníka

Přidání a editace zákazníka spočívá v převzetí dat vyplněných ve formuláři, které se následně validují. Jediný rozdíl mezi těmito skripty je, že při editaci uživatele se převzatá data do tabulky kontakty nevkládají, nýbrž jen editují podle unikátního identifikátoru kontaktu (id) a na začátku skriptu se data o zákazníkovi vloží do formuláře.

```
$data = array(
		'jmeno' => $jmeno,
		'prijmeni' => $prijmeni,
			.
			.
		'faktury' => $faktura
);
$this->_db->insert('kontakty', $data);
```

Zdrojový kód 26: Vložení dat do databáze

Veškeré další přidávání a editace čehokoli funguje na stejném principu – převzetí dat, validace, vložení respektive editace údajů v databázi.

5.5.3 Zakázání a povolení zákazníka

Zakázání a povolení zákazníků spočívá pouze v úpravě jednoho sloupečku v tabulce kontakty, který určuje, zda je daný zákazník povolen popř. zakázán. Dále systém vygeneruje jednou denně seznam zakázaných ip adres a uloží je do souboru.

```
$zakaz = array(
	'ban_id' => $banId
);
$this->_db->update('kontakty', $zakaz, 'id = ' . $id);
```

Zdrojový kód 27: Editace záznamu v tabulce kontakty

5.5.4 Smazání zákazníka

Při této akci jde pouze o vymazání záznamu v databázi. Je však nežádoucí, aby hned po kliknutí na odkaz došlo ke smazání daného zákazníka, proto se po kliknutí na odkaz spustí skript psaný v JS, který vyžaduje potvrzení, o smazání daného zákazníka. Toto potvrzení se dá jednoduše přiřadit jakémukoli odkazu pomocí identifikátoru `class="potvrzeni"`.

```
$(".potvrzeni").click(function(elem){
	akce = $(this).attr('title');
	a = confirm("Opravdu chcete " + akce);
	if( !a ){
		elem.preventDefault();
	} else {
		window.location($(this).attr('href'));
	}
});
```

Zdrojový kód 28: Potvrzení události v JS (využívá framework jQuery)

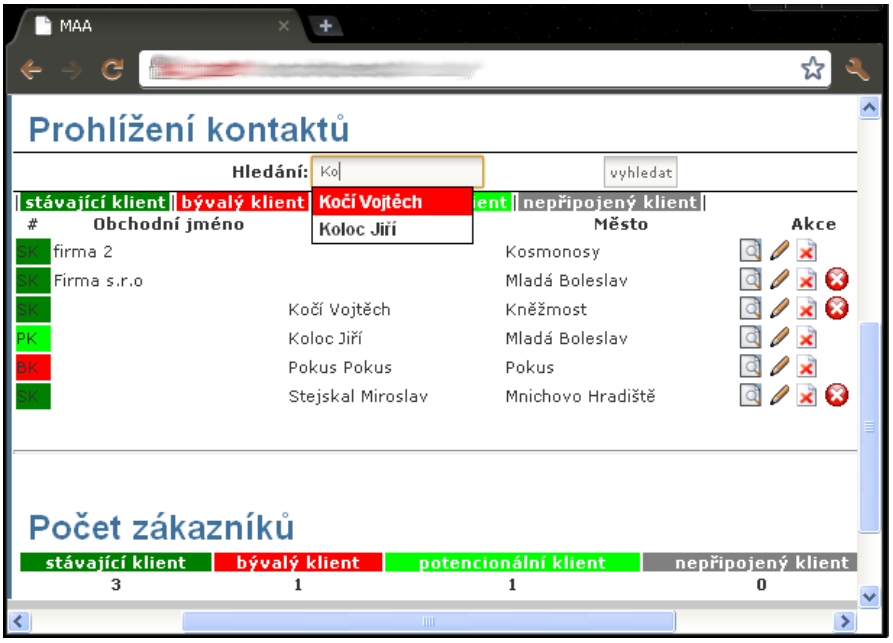
Jako u přidávání a editace zákazníka, se jedná o univerzální skript pro mazání čehokoli z databáze včetně potvrzení události v JS.

5.5.5 Výpis, detail zákazníků a backendy pro našeptávače

Ve výpisu zákazníků a u backendů, jde o totéž. Dotaz na databázi, ze které se vytáhnou data, co se mají zobrazit a vyhovují podmínce, kterou si zadal uživatel systému. Pro detail zákazníka se z databáze vytahují data o konkrétním zákazníku, které se zobrazí ve zvláštním okně. Zobrazují se osobní informace, stroje a platební morálka zákazníka. U backendů pro našeptávače se navíc musí vypnout layout systému, který je v tomto případě nežádoucí (uživatel tyto výpisy nevidí, využívá je pouze AJAXový dotaz, jež data pak zobrazí v požadované podobě).

```
$this->_helper->layout->disableLayout();
```

Zdrojový kód 29: Vypnutí layoutu



Obr. 9: Ukázka našeptávače pro vyhledávání v kontaktech

5.6 Modul Síť

Tento modul je rozdělen na tři části – shapery, skupiny a vysílače.

V části shapery se přidávají, editují, mažou a prohlížejí shapery, které se dále přidávají zákazníkům do smlouvy (tarif).

Část skupiny je určena pro správu (přidávání a mazání) skupin zákazníků, tyto skupiny se využívají při rozesílání emailů s informacemi o údržbě sítě.

Pro správu vysílačů (přidávání, mazání, editace, prohlížení) slouží část vysílače. Dále je zde odkaz na modul upload, kde se nacházejí soubory vysílače.

Veškeré přidávání, editace, mazání a prohlížení se provádějí stejným způsobem, jak je popsáno v modulu kontakty.

Obr. 10: Validace dat při přidání shaperu

5.6 Modul Zákazník

V tomto modulu se pracuje se zákazníky, kteří se přidali v modulu kontakty. Spravují se zde instalace jednotlivých klientů, deinstalují se klienti, spravují se platební prázdniny (v případě, že si zákazník předplatí služby dopředu), dále jsou zde backendy pro přidávání strojů (kontrola zadaných ip adres, zda se již nepoužívají, data do formulářových polí) a nakonec je zde ping na AP zákazníka (pokud je připojen pomocí bezdrátového připojení).

5.6.1 Přidání instalace, platebních prázdnin a deinstalace

Při přidání instalace se mimo vkládání dat do databáze musí vytvořit složka zákazníka, do které se budou vkládat nahrané fotky, hovory, dokumenty a vzorové dokumenty (smlouva, technická specifikace, předávací protokol, …). Dále se zde generují přihlašovací údaje, do zákaznické zóny a následně ukládají do databáze. Přidávání strojů a služeb je řešeno pomocí JS, který dynamicky přidává/odebírá formulářové pole, díky tomu je možné přidat více strojů a služeb najednou, aniž by se stránka musela několikrát načítat.

```
if( !file_exists($slozka.$idInstalace) ){
	mkdir($slozka . $idInstalace);
	$fajl = array('dodatek.txt', 'dp.txt', 'pp.txt', 'prolongace.txt',
'smlouvam.txt', 'ts.txt');
	for( $i=0; $i<count($fajl); $i++ ){
		copy("./vzory/".$fajl[$i] ,$slozka.idInstalace."/".$fajl[$i]);
	}
}
```

Zdrojový kód 30: Vytvoření složky zákazníka a překopírování vzorových dokumentů

Přidávání platebních prázdnin a deinstalace, probíhá stejně, jak je popsáno u modulu kontakty.

5.6.2 Editace instalace

Pro editaci instalace je nutné do formuláře vložit nejen informace o instalaci, ale také vypsat veškeré stroje a služby, které zákazník využívá, aby se mohli editovat popřípadě smazat. Po odeslání formuláře se opět veškeré data validují, a v případě, že validace dat je bezchybná, změny se zapíší do databáze.

5.6.3 Smazání deinstalace, platebních prázdnin a instalace

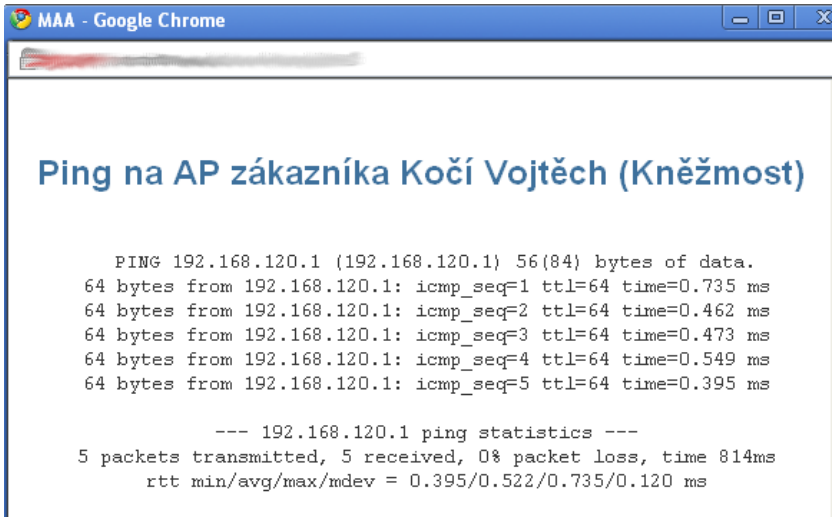
Smazání provádí skript, který po potvrzení vymaže konkrétní data z databáze. U smazání instalace skript navíc změnu typ zákazníka ze stávajícího na bývalého.

5.6.4 Ping na AP

Tato akce se využívá pro rychlou kontrolu, zda zákazníkovo AP je přístupné ze sítě, tím se dá rychle zjistit, jestli síť funguje nebo ne. Pokud má zákazník AP, jeho ip adresa se vytáhne z databáze a po dokončení příkazu ping se do okna vypíší statistiky.

```
shell_exec("ping -c 5 -i 0.2 " . $ip);
```

Zdrojový kód 31: Funkce pro spuštění příkazu ping



Obr. 11: Ping na zákazníkovo AP

5.6.5 Backendy

Modul zákazník nevyužívá na rozdíl od modulu kontakty backendy pro našeptávače, při vyhledávání v databázi, nýbrž pro doplnění formulářových polí a okamžitou kontrolu zadaných dat.

```
poradi = maa.zjistiPoradi(id, c);
$.get("/zakaznik/zjisti-ip/?rid="+id+"&poradi="+poradi, function(data){
	$("input#ip"+c).val(data);
});
```

Zdrojový kód 32: Dotaz pro zjištění první volné ip adresy a následné vložení do formuláře

5.7 Modul Faktury

Modul vyhrazený pro práci s fakturami. Zde dochází ke generování fakturace za minulý měsíc, vytvoření individuální faktury, výpisy vystavených faktur dle měsíců a v neposlední řadě práce s jednotlivými fakturami (uhrazení, smazání úhrady, prodloužení splatnosti, editace faktury včetně přidávání a odebírání položek).

5.7.1 Generování fakturace

Faktury se generují všem zákazníkům, kteří využívali nějaké služby minulý měsíc. Pro zákazníky, kteří nebyli připojení před prvním dnem v měsíci, se dopočítává pouze poměrná částka do konce měsíce. Zákazníci s platebními prázdninami mají slevu, dle začátku a konce prázdnin. Sleva se přidává jako druhá položka faktury. Protože byla zavedena pomocná tabulka sum_faktury_polozky, musí se celková cena faktury jako při generování fakturace vložit do této tabulky. Provedená fakturace, se v zákaznické zóně zobrazí až po povolení zobrazení od uživatele.

5.7.2 Vytvoření individuální faktury

V případě, že data ve formuláři budou řádně vyplněné, uloží se do databáze. Celková cena faktury musí být vložena do pomocné tabulky sum_faktury_polozky jako u generování fakturace.

5.7.3 Výpisy faktur

Výpisy faktur jsou rozděleny do tří skupin – provedené fakturace, individuální fakturace a faktury po splatnosti.

V provedené fakturaci se nacházejí pouze faktury vygenerované automaticky první den v měsíci. Faktury jsou rozdělené podle data uskutečnění plnění.

U výpisu individuálních faktur jsou pouze faktury, které nebyly vygenerované automaticky. Dělení faktur je podle měsíce, ve kterém byly vytvořeny.

V posledním typu výpisu se vypisují faktury, které nebyly uhrazeny včas.

5.7.4 Editace faktury, přidávání/odebírání položek

Pro přidávání položek je připraven formulář, kam se zadá popis položky a cena bez DPH. Jsou-li data vyplněna správně, dopočítá se cena s DPH (velikost DPH se dá změnit v nastavení systému), položka se uloží do databáze, následně dojde k přepočítání celkové ceny faktury a ta se následně upraví v pomocné tabulce sum_faktury_polozky.

Obdobně je tomu i u odebírání položek z faktury. Po potvrzení smazání položky, se položka vymaže z databáze, dojde k přepočítání celkové ceny faktury a cena v pomocné tabulce se upraví na aktuální hodnotu.

V případě editace faktury je průběh stejný jako u jakékoli jiné editace – převzetí dat, validace dat, v případě korektních dat jsou data editována v databázi.

5.7.5 Úhrady faktur

Úhrady faktur se v systému dělí do dvou skupin – úhrada na pokladně a úhrada bankovním převodem

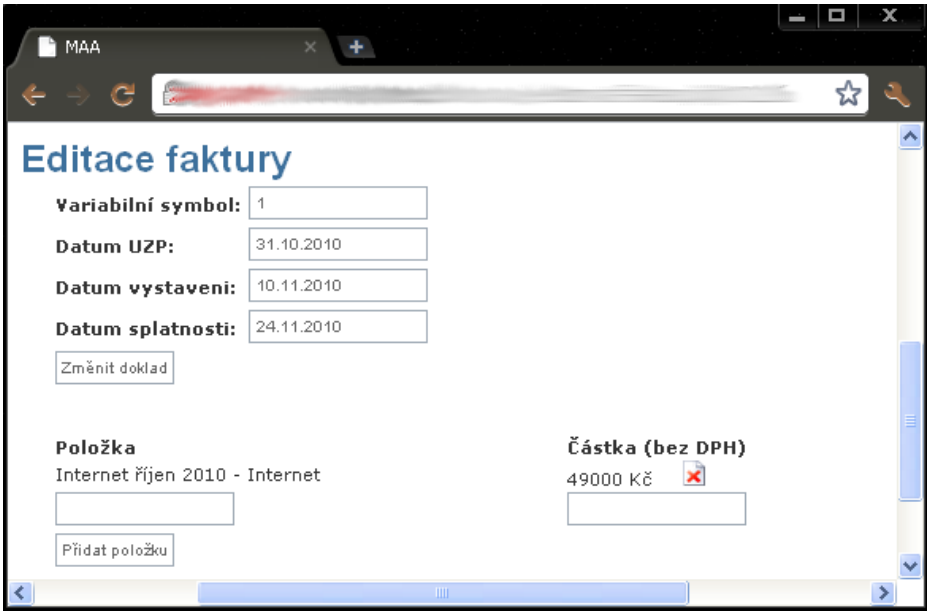
Při úhradě na pokladně, se okamžitě hradí celá částka faktury, která ještě nebyla uhrazena (pro případ částečně uhrazené faktury), úhrada se vloží do databáze, pokud se jedná o částečně uhrazenou fakturu, upraví se záznam v pomocné tabulce sum_faktury_uhrady, v opačném případě se záznam v pomocné tabulce vytvoří.

Oproti tomu úhrada bankou vede na formulář, kam se zadává datum platby, částka popř. variabilní symbol, pokud jsou data vyplněna korektně, platba se uloží do databáze a upraví, popřípadě vytvoří záznam v pomocné tabulce.

5.8.5 Smazání faktury, položky a úhrady

Smazání položky a úhrady je v podstatě stejné. Dojde ke smazání dat ať už položky nebo úhrady v databázi a upraví se pomocná tabulka.

Při smazání faktury se z databáze vymažou veškerá data, která s tabulkou souvisí, vymažou se veškeré položky, úhrady, a informace o faktuře. Data se vymažou i z pomocných tabulek.



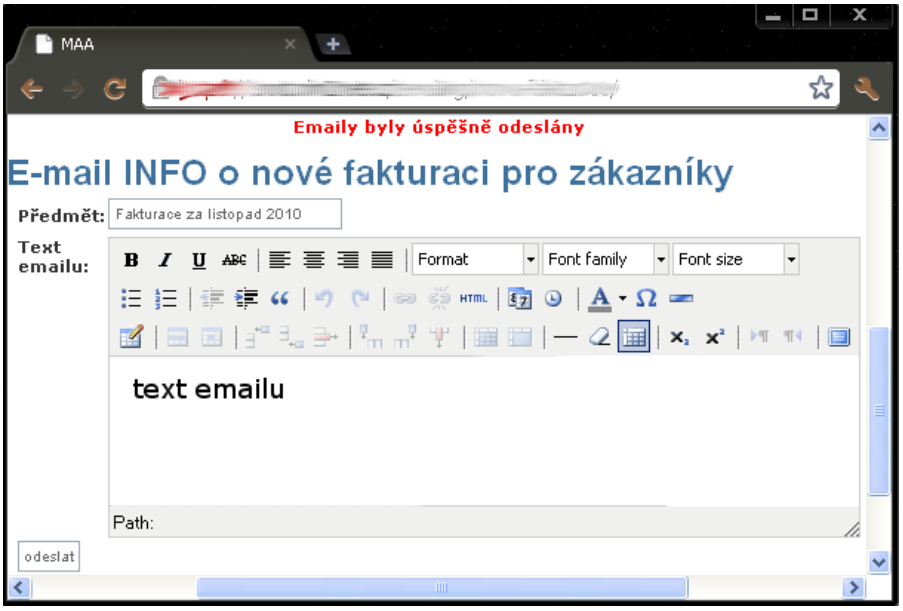
Obr. 12: Editace faktury + přidávání a mazání položek

5.9 Modul Emailing

Tento modul je určen pro informování zákazníků prostřednictvím emailů. Obsahuje dvě akce – Info o proběhlé fakturaci a Info o údržbě na síti. Rozdíl mezi těmito dvěma akcemi je pouze ve výběru příjemců. Při rozesílání informace o proběhlé fakturaci jsou příjemci všichni, kterým se za poslední měsíc vygenerovala faktura, kdežto u rozesílání informace o údržbě na síti jsou příjemci pouze členové skupin, které se vybrali ve formuláři před odesláním emailů.

```
$mail = new Zend_Mail('UTF-8');
$mail->setHeaderEncoding(Zend_Mime::ENCODING_BASE64);
$bodyText = html_entity_decode($htmlText, ENT_NOQUOTES, 'UTF-8');
$bodyText = stripslashes(strip_tags($bodyText));
if( $priloha != false ){
      $mail->addAttachment($priloha);
}
$mail->setBodyText($bodyText);
$mail->setBodyHtml($htmlText);
$mail->setFrom('obchod@metronet.cz', 'Maa');
$mail->setReplyTo('obchod@metronet.cz', 'Obchod');
$mail->setSubject($predmet);
$mail->addTo($a['email'], $a['jmeno']);
$mail->send($transport);
```

Zdrojový kód 33: Vytvoření a odeslání emailu s přílohou (pokud je vložena)



Obr. 13: Odesílání emailů ze systému

5.10 Modul Editory

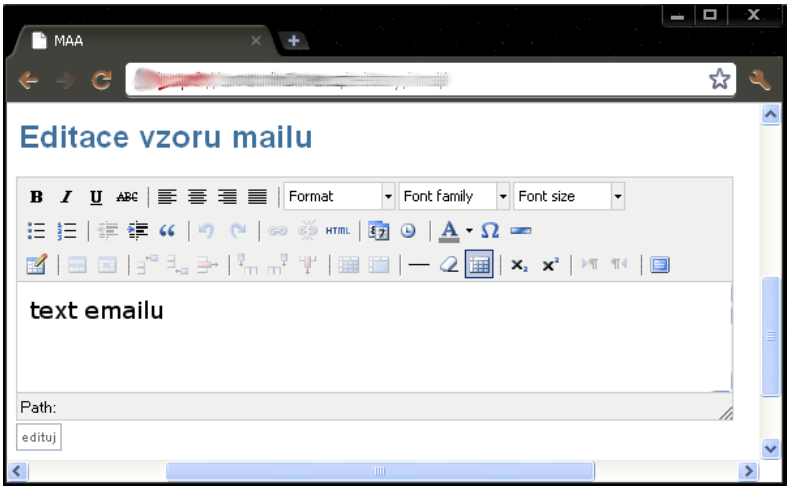
Tento modul slouží k editování vzorových dokumentů, dokumentů zákazníka, textu emailu o proběhlé fakturaci a statické části konfiguračního souboru pro smokeping.

5.10.1 Editace dokumentů zákazníka

Při přidání instalace se zákazníkovi překopírovali vzorové soubory do jeho složky. Tím se zajistí, že případné změny se provedou pouze v dokumentu u požadovaného zákazníka a ne u všech zákazníků. Obsah stávajícího souboru se vloží do formulářového pole, na kterém je spuštěn WYSIWYG editor, kde se provedou požadované změny, kterými se upravovaný soubor přepíše.

5.10.2 Editace vzorových souborů

Editace jakéhokoli ze vzorových souborů probíhá stejně jako editace dokumentů zákazníka, s rozdílem, že se soubory nehledají u zákazníka ve složce, nýbrž ve složce vzory, kde jsou tyto soubory uloženy.



Obr. 14: Editování textu e-mailu při posílání informace o provedené fakturaci

5.11 Modul Žurnál

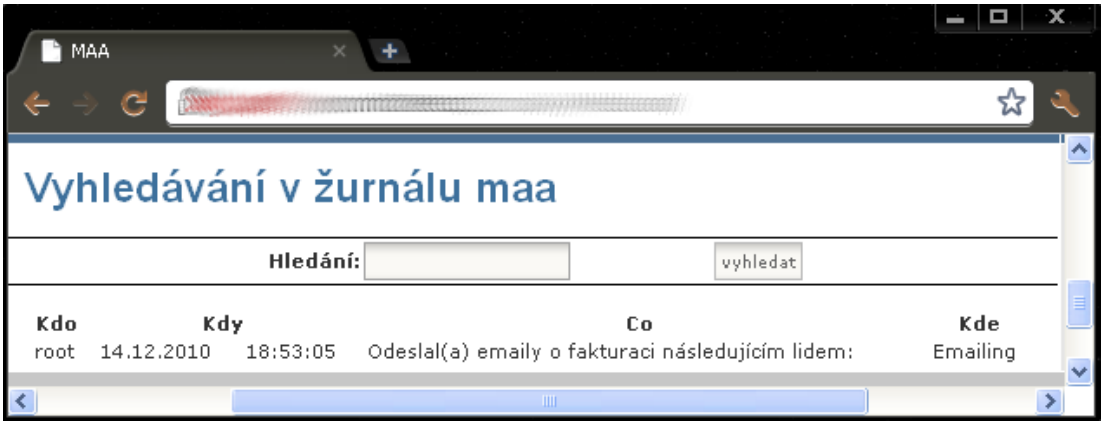
Modul žurnál je rozdělen na dvě části – žurnál zákazníka a žurnál systému.

5.11.1 Žurnál systému

V žurnálu systému se uchovávají veškeré akce provedené v systému, s možností vyhledávání. Výpis akcí je realizován přes pohled v databázi, což urychluje načítání stránky. Vyhledávání v žurnálu je realizováno pomocí fulltextu.

```
$sql = "SELECT * FROM pohled_uzivatel_log
        WHERE MATCH(login, akce, modul) AGAINST( '". $hledej ."' IN BOOLEAN MODE )
        ORDER BY lid DESC
        LIMIT $poz,$max";
$this->view->log = $this->_db->fetchAll($sql);
```

Zdrojový kód 34: Fulltextové vyhledávání v databázi



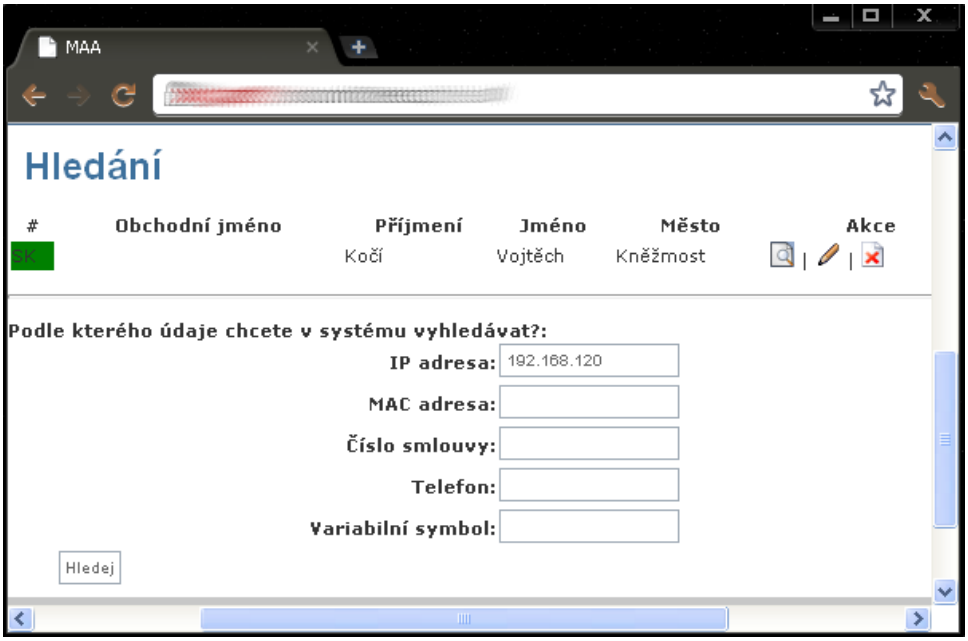
Obr. 15: Vyhledávání slova „emailing“ v žurnálu

5.11.2 Zákaznický žurnál

Zákaznický žurnál slouží k dvěma věcem – výpis akcí zákazníků z uživatelské sekce, ale především k evidenci nahlášených poruch. Uživatelé systému mohou vytvářet záznamy v žurnálu, při vytvoření záznamu se automaticky vytvoří ticket v externím trouble ticket systému. Při potvrzení vyřešení ticketu se automaticky záznam v žurnálu označí jako vyřešený. Ke každému záznamu je možno přiložit *.pdf soubor.

5.12 Modul Hledání

V tomto modulu se dají hledat zákazníci, popřípadě rozsahy dle různých parametrů. Fulltextové hledání se zde nevyužívá (pouze co se SQL dotazu týče, jinak modul hledání je rozdělen na dvě části – fulltextové hledání a klasické vyhledávání), protože fulltext v MySQL nedokáže doplňovat slova, v případě, že se uživatel nezná celé jméno, popřípadě ip adresu je velmi nežádoucí. Vyhledávat lze podle: ip adresy, mac adresy, čísla smlouvy, telefonu, variabilního symbolu, ve fulltextovém vyhledávání dle jména, příjmení, emailu, obchodního jména, ičo, dič, město, ulice.



Obr. 16: Vyhledávání dle ip adresy

5.13 Modul Uživatelé

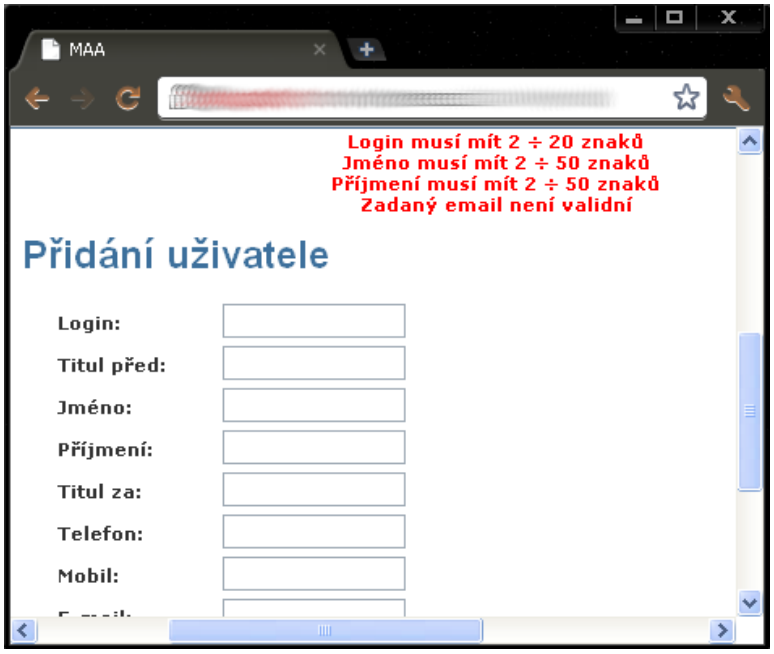
Modul pro správu (přidání, smazání, editace, editace práv, prohlížení, výpis) uživatelů, kteří mají přístup do systému.

5.13.1 Přidání uživatele

V případě, že data vyplněná ve formuláři projdou validací bez chyby, vygeneruje se uživateli heslo (které si může kdykoli změnit), odešle se mu uvítací email, ve kterém najde přihlašovací údaje do systému a přidá záznam o přidaném uživateli do tabulky oprávnění.

```
$valLogin = new Zend_Validate_StringLength(2,20);
$valJm = new Zend_Validate_StringLength(2,50);
$valEmail = new Zend_Validate_EmailAddress();
$validator = new Zend_Validate_Db_RecordExists(
            array(
                  'table' => 'uzivatel',
                  'field' => 'login',
                  'adapter' => $this->_db
            )
);
if( $validator->isValid($login) ){ $chyby[] = "Login se již používá"; }
if( !$valLogin->isValid($login) ){ $chyby[] = "Login musí mít 2 ÷ 20 znaků"; }
if( !$valJm->isValid($jmeno) ){ $chyby[] = "Jméno musí mít 2 ÷ 50 znaků"; }
if( !$valJm->isValid($prijmeni) ){ $chyby[] = "Příjmení musí mít 2 ÷ 50 znaků"; }
if( !$valEmail->isValid($email) ){ $chyby[] = "Zadaný email není validní"; }
if( sizeof($chyby) == 0 ){
      vložení uživatele do db
} else {
      vrácení dat do formulářových polí a výpis chyb
}
```

Zdrojový kód 35: Validace vyplněných údajů při přidání uživatele



Obr. 17: Validace dat při přidání uživatele do systému

5.13.2 Editace a editace práv uživatele

Při obou editacích nejdříve dochází k vytažení dat z databáze pro konkrétního uživatele, které se vypíšou do formuláře, aby se moli upravit. Po úpravě opět proběhne validace dat, v případě, že je validace bezchybná, data se upraví v databázi.

5.13.3 Smazání uživatele

Pro smazání uživatele je opět vyžadování potvrzení vymazání, kdyby se na odkaz pro smazání kliklo omylem. Po vymazání uživatele, je dále vymazán s právy daného uživatele.

5.13.4 Výpis uživatelů a Prohlížení uživatele

Výpis uživatelů a prohlížení uživatele je řešeno stejně, jako je tomu při výpisu a detailu zákazníka v modulu kontakty, jen se nevypisují data z tabulky kontakty, nýbrž z tabulky uzivatel.

5.14 Modul Tiskové výstupy

Zde nacházejí důležité informace manažeři sítě. Jsou zde jak textové, tak grafické textové výstupy, ze kterých se dají zjistit důležité informace.

5.14.1 Textové informace

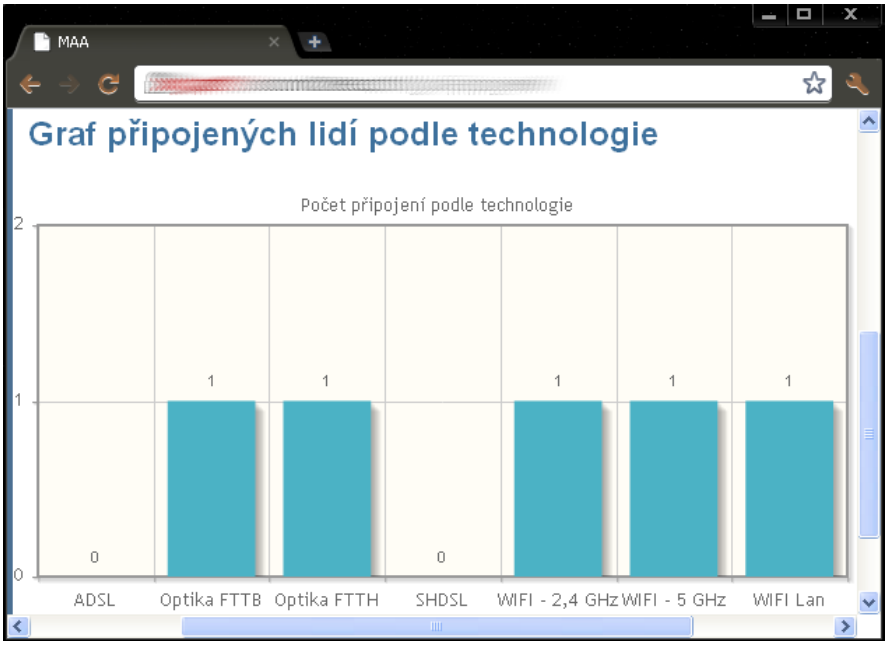
Mezi textové informace patří výpis zákazníků, kteří mají prázdniny, zákazníci dle data připojení, lidé kteří žádají o připojení, nemají vyplněné veškeré údaje, zákazníci dle určitého počtu dní před koncem smlouvy, zákazníci kteří nezaplatili faktury do data splatnosti, zákazníci dle typu připojení, nahlášené poruchy zákazníků, výpovědi zákazníků, tržby z určitého typu připojení a nakonec neplatící zákazníci. Veškeré tyto výpisy jsou řešeny pomocí dotazu na databázi, každý s jinými podmínkami, následně jsou data vypsána pomocí tabulek.

5.14.2 Grafické informace

Grafické informace jsou také data vytažená z databáze, rozdíl je v tom, že data jsou vykreslena v grafu. V tomto modulu jsou grafy smluv (uzavřené/vypovězené), graf tržeb za poslední rok, graf počtu připojených zákazníků podle druhu připojení. Grafy jsou vykreslovány v JS, protože grafy vykreslované pomocí PHP nejsou tak přehledné, díky JS se zobrazují přesné hodnoty po najetí kurzorem do jakéhokoli bodu grafu.

```
var data = $.ajax({type: "GET",url:"/tisk/pripojeni-data/",async:
false}).responseText;
var a = data.split("\n");
hodnoty = new Array();
max = 0;
for( var i=0; i<a.length; i++){
        d = a[i].split('|');
        hodnoty.push([d[0],d[1]]);
        if( eval(d[1]) > max ){
                max = eval(d[1]);
        }
}
```

Zdrojový kód 36: Zpracování hodnot, aby se mohli vykreslit v grafu



Obr. 18: Sloupcový graf počtu připojení dle technologie

5.15 Modul Zákazník web

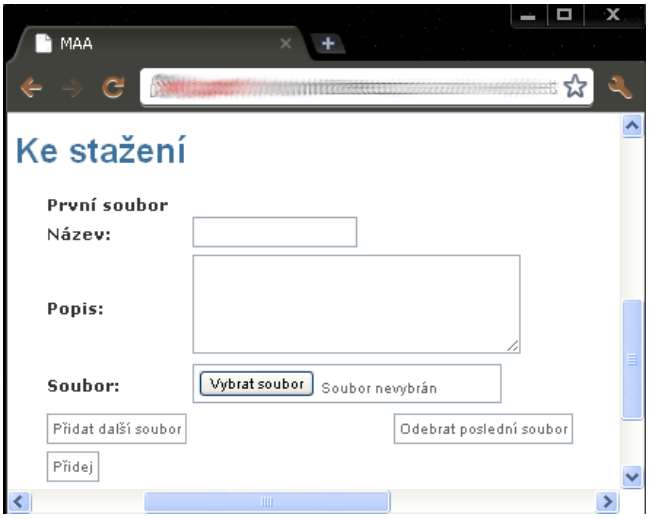
Modul určený pro práci s webem zákaznické zóny. Nahrávání souborů do sekce „ke stažení“, a možnost měnit úvodní text, který se zobrazuje po přihlášení do zákaznické sekce.

5.15.1 Nahrávání souborů

U nahrávání souborů pro web zákaznické zóny, nastal problém, protože web zákaznické zóny je umístněn na jiném serveru, než na kterém je umístěn tento systém. Byly zvažovány dvě možnosti přesunu souborů – pomocí protokolu FTP a pomocí protokolu SCP (samotné SCP na přenos souboru stačilo, SFTP bylo zbytečné). Pro využití protokolů SCP a SFTP bylo nutné zkompilovat rozšíření SSH2, která přidá podporu SCP a SFTP do PHP.

```
phpize && ./configure --with-ssh2 && make
echo „extension=ssh2.so“ >> /etc/php5/apache2/php.ini
cp ssh2.so /usr/lib/php5/20060613+lfs/
```

Zdrojový kód 37: Zkompilování rozšíření ssh2 pro PHP (debian)



Obr. 19: Nahrání souborů do zákaznické zóny

Při nahrávání souboru se do databáze ukládá název a popis souboru, který se pak zobrazuje zákazníkům v zákaznické sekci. Připojování ke vzdálenému serveru je navíc řešeno pomocí klíčů – nikde ve zdrojových kódech se neobjevují hesla.

```
$spojeni = @ssh2_connect(ZAKAZNIK, 22, array('hostkey'=>'ssh-dss'));
$auth = @ssh2_auth_pubkey_file($spojeni, USER,
                          'verejny_klic',
                          'privatni_klic');
$soubor = @ssh2_scp_send($spojeni, $file, SLOZKA . strtolower($file), 0655);
```

Zdrojový kód 38: Ukázka připojení ke vzdálenému serveru a nakopírování souboru

5.15.2 Mazání souborů

Pro mazání souborů je protokol SCP ve spojení s PHP nedostačující, bylo nutné využít protokol SFTP. Připojení k serveru je úplně stejné, po připojení se využije funkce, která daný soubor vymaže.

```
$smazani = ssh2_sftp_unlink($sftp, SLOZKA . $soubor);
```

Zdrojový kód 39: Smazání souboru ze vzdáleného serveru

5.15.3 Procházení souborů

Díky tomu, že se soubory (název, popis) ukládají do databáze, ke zjištění nahraných souborů není nutné se připojovat ke vzdálenému serveru a procházet složku s nahranými soubory. Názvy souborů s jejich popisem se pouze vytáhnou z databáze a vypíší se uživateli.

5.15.4 Úvodní text zákaznické zóny

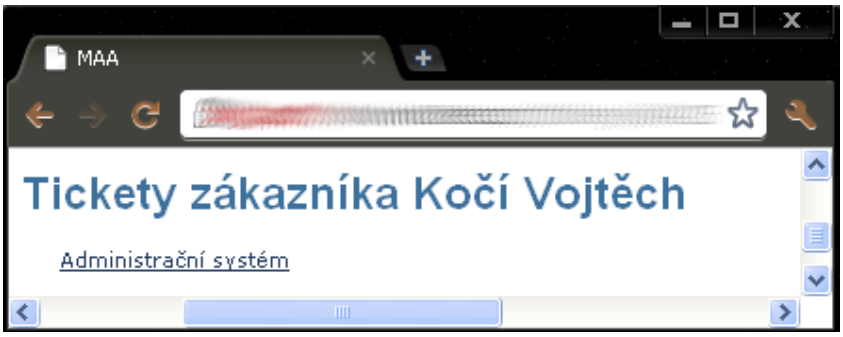
Protože web pro zákazníky je napojen do stejné databáze, může se úvodní text uložit do tabulky pro nastavení, a zákazníkům po přihlášení jednoduše zobrazit. Při úpravě úvodního textu se opět využívá WYSIWYG editor, aby nebylo nutné psát celý text v jazyce (X)HTML.

5.16 Modul RT

V tomto modulu se uchovávají informace o ticketech, které mají zákazníci uložené v ticketovém systému. V případě, že se uzavře ticket, jež byl vytvořen v modulu „žurnál“ při vytvoření poruchy u zákazníka, automaticky se zde vytvoří záznam o daném ticketu. Tickety uložené v systému se dají spravovat (editovat, mazat, přidávat).

5.16.1 Správa ticketů

Poněvadž se jedná o napojení na externí systém, v databázi se ukládá pouze unikátní identifikátor ticketu a jeho předmět. U výpisu ticketů je vypsán odkaz na externí systém, kde se zobrazí podrobnosti o daném problému. Údaje o ticketu se dají upravovat, popř. mazat, kde je opět vyžadováno potvrzení.



Obr. 20: Výpis ticketů u zákazníka

5.17 Modul Upload

Tento modul slouží ke správě fotek, hovorů, dokumentů jednotlivých zákazníků. Dále se zde spravují fotky a dokumenty k vysílačům (Veškeré dokumenty jsou nahrávány ve formátu *.pdf). Pro nahrávání fotek byla vytvořena speciální třída, která automaticky při nahrávání fotky vytvoří malý náhled fotky, který se umístí na stránku (pro rychlejší nahrávání celé stránky – menší fotka, méně dat ke stažení).

5.17.1 Nahrávání souborů

Při nahrávání jakéhokoli dokumentu, hovoru nebo fotek je nutné vybrat zákazníka popřípadě vysílač, ke kterému dané soubory patří. Dalším předpokladem je, že uživatel, pod kterým běží web server (na Linuxu ve výchozím nastavení uživatel www-data) má právo zapisovat do složky klienti respektive vysílače.

Při nahrávání fotek je možné formuláři dynamicky přidávat elementy pro vložení souboru v případě, že je potřeba nahrát více jak jednu fotku. U přidávání hovorů tato možnost není, protože nahrané hovory mohou mít několik megabajtů a mohl by být překročen maximální limit pro nahrání souboru. Po nahrání souboru se soubor ukládá do databáze, pro rychlejší výpisy – dotaz na databázi je rychlejší než procházení několika stovek složek.

```
$velikost = getimagesize($adresa_org); zjištění rozměrů obrázku
$sirka = $velikost[0];
$vyska = $velikost[1];
$nahledVyska = 80; výška obrázku po zmenšení(px)
$nahledSirka = $sirka*(80/$vyska);
V $funkce je uložen název fce pro načtení obrázku dle koncovky obrázku
```

```
$org = $funkce($adresa_org);
$tn = imagecreatetruecolor($nahledSirka, $nahledVyska);
ImageCopyResampled($tn,$org,1,1,0,0,$nahledSirka,$nahledVyska,$sirka,$vyska);
V $obrazek je uložen název fce pro vytvoření obrázku dle koncovky obrázku
$obrazek($tn,$adresa);
$obrazek($tn);
ImageDestroy($tn);
ImageDestroy($org);
```

Zdrojový kód 40: Generování náhledu fotky

Obr. 21: Vygenerované náhledy fotek

5.17.2 Mazání souborů

Při mazání souborů ať už od zákazníka, nebo od vysílače se nejdříve smaže soubor na disku, v případě, že vše proběhne v pořádku, vymaže se záznam o souboru z databáze. Je to z důvodu, kdyby se nepovedlo smazat soubor z disku (například z důvodu nečekané změně práv k souboru), tak by soubor zůstal fyzicky na disku, ale nešel by vymazat přes webové rozhraní.

5.18 Modul Root

Modul je určen zejména pro administrátory sítě, spravují se zde vlany, rozsahy a routery. Generuje se zde dynamická část konfiguračního souboru pro smokeping, která se následně pomocí SCP nahrává na monitorovací stroj sítě.

5.18.1 Správa routerů

Pod správu routerů patří přidávání, editace, výpis a mazání (s potvrzením) routerů. Navíc jsou zde dvě akce – synchronizace a přetažení vlanů.

Pod pojmem synchronizace se rozumí spuštění bashového skriptu od administrátora sítě, který se přihlásí na daný router, stáhne všechny rozsahy, které jsou na něm aktivní. Tyto rozsahy se porovnají s údaji v databázi a vypíší se případné rozdíly.

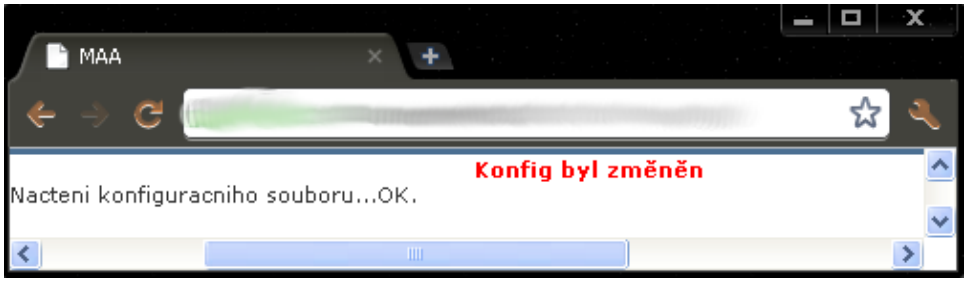
Akce pro přetažení vlanů je zde z důvodu, kdyby se daný router rušil, vlany které na něm končí, by najednou nebyly nikde ukončeny. Proto se můžou naráz všechny „přetáhnout“ na jiný router.

5.18.2 Správa rozsahů a vlanů

Jedná se o přidávání, editaci a mazání routerů respektive vlanů. Veškerá data musejí být validní – nesmějí již být v databázi, ip adresy musí být validní (ZF validuje IPv4 i IPv6). Pro smazání routeru popřípadě vlanu je opět vyžadování potvrzení.

5.18.3 Generování konfiguračního souboru pro smokeping

Smokeping je aplikace, která monitoruje latence jednotlivých strojů v síti. Aby bylo menu generované systémem přehledné, bylo rozhodnuto, že jednotliví zákazníci budou rozděleny, dle skupin, do které byly přidány během přidání instalace. Konfigurační soubor se skládá ze statické části, kterou lze editovat v modulu „editory“ a dynamické části, která se skládá ze strojů jednotlivých zákazníků, kterým bylo zapnuto monitorování pomocí aplikace smokeping. Sloučený konfigurační soubor se odešle na monitorovací stroj, aby se nový konfigurační soubor načetl, musí skript smokeping vypnout a znovu zapnout.



Obr. 22: Nahrání konfiguračního souboru pro smokeping

6 IMPLEMENTACE SYSTÉMU

Pro implementaci systému, bylo nutné napsat exportní skript ze starého systému. Problém byl v kódování staré databáze, která byla kódovaná v kódování latin1_swedish_ci, ale data v databázi byly kódované ve windows-1250 (cp1250). Protože nový systém je celý kódovaný v utf-8, bylo nutné data ze staré databáze překódovat na utf-8.

```
Nastavení kódování pro starou databázi
mysql_query('SET NAMES utf8', $maaOld);
Funkce pro nahrazení špatně převedených znaků
function diakritika($str){
      $prevod = array('?' => 'š', '' => 'ť', 'è' => 'č', 'Ø' => 'Ř', 'ø' => 'ř',
'' => 'ť', 'ì' => 'ě', 'È' => 'Č', 'ò' => 'ň', 'ù' => 'ů', 'ï' => 'ď', '
<U+008D>' => 'ť', '<U+009D>' => 'Ť') ;
        $str = strtr($str, $prevod);
        return $str;
}
```

Zdrojový kód 41: Funkce pro úpravu kódování na utf-8

Po úspěšném exportu dat ze staré do nové databáze následovalo provedení zkušební fakturace, zdali systém počítá stejně jako systém starý. Po provedení fakturace byly odhaleny chyby v algoritmu, který generuje fakturaci.

Úplné spuštění systému bylo ještě ovlivněno nutnými úpravami webu zákaznické zóny. Web zákaznické zóny byl postaven na jinou strukturu databáze než nový systém, proto bylo nutné připravit si potřebné úpravy, aby při spuštění nového systému zákazníci mohli stále využívat zákaznickou zónu.

Později se objevili problémy při posílání emailů o proběhlé fakturaci. Problémy nastávali při generování PDF faktury. PHP funkce překročila výchozí maximální povolenou dobu běhu skriptu při zpracovávání unicódových dat. Tento problém byl vyřešen zvětšením doby maximálního běhu skriptu. Doba nebyla zvětšena pro všechny PHP skripty, nýbrž pouze pro skript, který generuje *.pdf souboru.

7 ZÁVĚR

Administrační systém pro ISP psaný v PHP s využitím ZF je dostačující s jednoduchou možností rozšíření. Ovšem teprve se zapojením JS je systém opravdu interaktivní a mnohdy rychlejší. Především díky AJAXovým dotazům je možné data kontrolovat prakticky ihned po zapsání do formulářového pole, dynamicky měnit počet formulářových polí bez opakovaného načítání stránky, což práci se systémem značně urychluje.

V době psaní této práce jsou všechny moduly plně funkční a doplňují se o další funkce, zejména funkce do modulu root. Kam byla přidána mapa rozsahů (využití rozsahu s maskou /24 – u rozsahů s maskou větší než /24 se rozsahy sloučí, aby vytvořili jeden rozsah o velikosti /24 ve kterém bude barevně zvýrazněno označení rozsahu a jeho broadcast), výše zmíněná synchronizace u routerů s přetažením vlanů, upravoval výpis detailu kontaktu, aby obsahoval i informace o deinstalaci zákazníka.

Propojení s ekonomickým systémem Pohoda je nyní napůl hotové. Systém generuje .xml soubor pro import faktur z administračního systému do eokonomického systému Pohoda, dále se testují automatické úhrady, kde skript vytahuje úhrady přímo z SQL databáze systému Pohoda a následně v administračním systému uhrazuje patřičné faktury. Automatické úhrady jsou dále závislé na komunikaci se systémem multi-cash, na které se v současné době pracuje. V poslední řadě se ještě pracuje na propojení s externím VoIP systémem, na které je již připravená databáze.

SEZNAM POUŽITÉ LITERATURY

[1] Kosek, J.; HTML tvorba dokonalých webových stránek, Grada, Praha 1998

[2] Naramore, E.; Gerner, J.; Scouarnec, Y. L.; Stolz, J.; Glass, M. K.; PHP5, MySQL, Apache Vytváříme webové aplikace, Computer press, Brno 2006

[3] Kosek, J.; PHP tvorba interaktivních internetových aplikací, Grada, Praha 1999

[4] Smith, C. W.; CSS využijte kaskádové styly naplno!, Computer press, Brno 2006

[5] Manuál pro Zend Framework, dostupné na: http://framework.zend.com/manual/en/

[6] CSS kaskádové styly, dostupné na: http://www.webtvorba.cz/css/

[7] SSH2 a jeho použití v PHP – transfery dat (SCP a SFTP), dostupné na: http://interval.cz/clanky/ssh2-a-jeho-pouziti-v-php-transfery-dat-scp-a-sftp/

[8] Dokumentace pro framework jQuery dostupné na: http://docs.jquery.com/Main_Page

SEZNAM PŘÍLOH

CD Text bakalářské práce ve formátu .pdf

Software bude odkoupen firmou, proto v této práci nebudou uveřejněny zdrojové kódy administračního systému.