WWW prezentace firmy Salvete, spol. s. r. o.

Bakalářská práce

Jindřich Čejka

Vedoucí bakalářské práce: PaedDr. Petr Pexa

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

2008

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne

Anotace

Cílem této práce je vytvořit oficiální webovou prezentaci s možností upravovat obsah stránek samotným správcem stránek pomocí administrátorské sekce. Práce tedy bude rozdělena do dvou částí, kterým bude předcházet návrh databázové struktury.

1) první část:

Tou první je vytvořit webové stránky, které bude používat (prohlížet si) běžný návštěvník stránek. To představuje navrhnout kvalitní design, jenž v dnešní době hraje velkou roli v návštěvnosti stránek, a ten sestavit do HTLM podoby dle standardu XHTML 1.0 Strict samozřejmě s využitím formátovacích stylů CSS 2.1. Celá tato část samozřejmě bude využívat technologii PHP a MySQL, přičemž obsah sekcí stránek bude umístěn v databázi.

2) druhá část:

Cílem druhé části bude vytvořit opět webové stránky, které budou přístupné pouze správci stránek, administrátorovi. Tato sekce umožní administrátorovi upravovat, mazat či přidávat obsah stránek. Opět s využitím PHP a MySQL .

Abstract

The main aim of this diploma work is to create the official web presentation with the site content editing options by the administrator himself with a help of administrator section. So the work will be divided into two parts. Those two parts will be preceding a database design structure.

1) first part

This part concentrates on the creation of web sites which will be used by a common web visitor. That means to form quality design that plays the main role in today's web site visiting. This design will be cut to pieces and gradually put together to HTLM image by XHTML 1.0 Strict standard - with the use of formatting styles CSS 2.1, naturally. All this part will be using PHP and MySQL technology, whereas the sites contain will be downloaded from the database.

2) second part

The aim of the second part is to create the web sites again. Those sites will be, however, accessible only to the administrator. This section will allow to the administrator edit, erase and add sites contain. Once again with PHP and MySQL.

Poděkování

Rád bych na tomto místě poděkoval vedoucímu mé bakalářské práce PaedDr. Petru Pexovi za skvělé vedení a cenné odborné rady.

Obsah

1 Úvod

Tématem mé bakalářské práce je „WWW prezentace firmy pomocí technologií PHP a MySQL“. Jmenované téma jsem zvolil, neboť tímto odborným směrem bych se chtěl ubírat i během své profesní kariéry. Poptávka po podobných produktech neustále roste, protože odpovídající webová prezentace na internetu je dnes již povinností každé firmy.

Cíl této práce je vytvořit komerční webovou prezentaci firmy Salvete, spol. s. r. o. v souladu s jejími požadavky na obsahovou část. Prezentace bude realizována tak, aby pověření pracovníci vlastníka stránek, firmy Salvete, spol. s. r. o., mohli jednoduše obsahovou část prezentace sami editovat.

K dosažení uvedeného cíle je nutné celý projekt rozdělit do tří samostatných fází. První a základní fáze, nutná ke splnění tohoto cíle, je prokonzultování nároků na obsahovou část prezentace s majiteli firmy Salvete, spol. s. r. o. a následné navržení vhodné databázové struktury, která umožní realizovat uchovávání požadovaných dat a bude v souladu s pravidly pro vytváření databází (tzv. „Normální formy“). Dalším krokem bude naprogramovat uživatelské prostředí zobrazující se běžným návštěvníkům prezentace na internetu. Poslední, nejobtížnější a zároveň nejdůležitější fází tohoto projektu bude naprogramovat jednoduché a přehledné administrátorské prostředí, které bude zabezpečené proti vniknutí nežádoucích osob a umožní kompletně spravovat celý obsah prezentace.

Při splňování tohoto cíle je nutné dodržovat normy pro tvorbu moderních webových prezentací stanovených institucí W3C (World Wide Web Consorcium), v tomto případě normy XHTML 1.0 Strict a CSS 2.1. Dále je vhodné, aby moderní web splňoval i zásady přístupnosti podle projektu Blind Friendly Web a pravidla pro tvorbu přístupného webu WCAG 1.0. Zcela samozřejmé bude i použití optimalizačních technik pro vyhledávací služby (SEO).

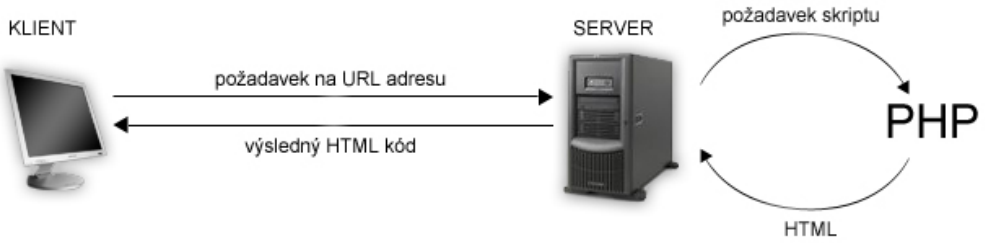
2 Použité technologie

2.1 PHP

PHP, původně zkratka „Personal Home Page“, vytvořil Lerdof Rasmus, aby mohl sledovat návštěvnost svých osobních stránek. Po roce 1994, krátce po svém vzniku, se tento programovací jazyk začal prudce rozvíjet, poskytovat více možností a projevovat se v profesionálních řešeních. Nárůstem profesionality se tato technologie ujala jako „PHP: Hypertext Preprocesor“.

PHP je skriptovací jazyk, který se v dnešní době řadí mezi ty nejpopulárnější. Důstojně konkuruje i mnohem mladším technologiím, jako je např. technologie ASP .NET, kterou vyvinula firma Microsoft. Mezi jeho nesporné výhody oproti jiným jazykům patří dostupnost a nezávislost na platformě. To znamená, že jazyk PHP je možno používat nejen na Unixových operačních systémech, jako je např. Linux, na kterém byl vyvinut, ale také na většině ostatních platformách, jako je např. Microsoft Windows či snad Macintosh.

PHP je určeno pro skriptování ne na straně uživatele, jako je běžný návštěvník webové prezentace, ale pro skriptování na straně serveru. Z toho vyplývá, že napsaný kód v jazyce PHP musí být vždy uložen na webovém serveru, který umí PHP kód zpracovat. Při každé návštěvě webové prezentace napsané v jazyce PHP prostřednictvím webových prohlížečů server načte a zpracuje PHP kód podle toho, jak jej programátor naprogramoval, a odešle výsledek webovému prohlížeči, z něhož přišel požadavek. Odeslaný kód již prvky PHP neobsahuje a webový prohlížeč s ním může pracovat jako se statickou stránkou v podobě HTML.



Obrázek 1: PHP v modelu klient/server při požadavku na webovou stránku.

Kód jazyka PHP se prolíná přímo s kódem HTML. To znamená, že se jeho kód vkládá do HTML kódu, ve kterém se PHP kód od HTML kódu odděluje značkami *<?php ?>* nebo *<? ?>*.

```
<!--TOP IMAGES-->
<div class="top">
    <?php
        include('links.php');
    ?>
</div>
```

Obrázek 2: Ukázka kódu PHP v HTML

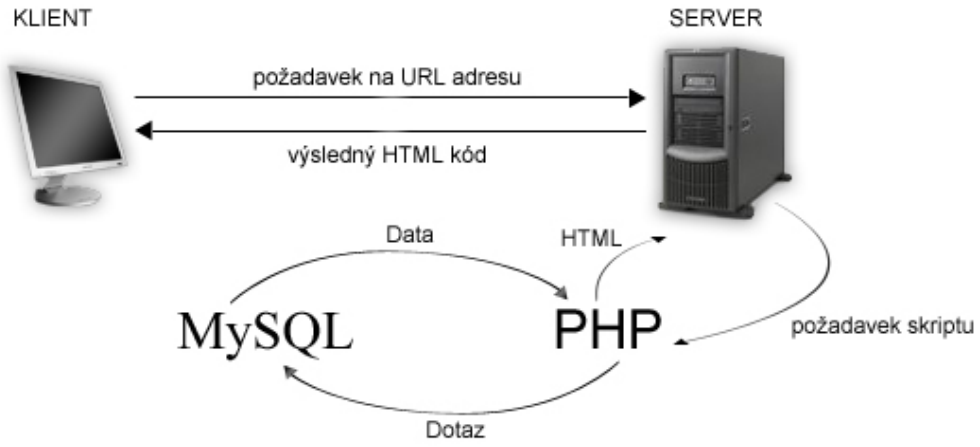
2.2 MySQL

Databáze MySQL se stejně jako skriptovací jazyk PHP řadí mezi nejoblíbenější a mezi nejlepší databázové systémy s veřejným kódem. Opět velmi dobře konkuruje drahým databázovým systémům, jako je Microsoft SQL Server či Oracle.

Svým uživatelům může nabídnout vysokou spolehlivost a kvalitní výkon při minimálních nákladech na provoz. To vše činí MySQL databázi jednou z nejoblíbenějších a nejpoužívanějších po celém světě.

MySQL je databázový systém sloužící k ukládání či uchovávání dat v textové podobě. Systém je založen na tabulkové bázi, kde se jednotlivé tabulky skládají z řádků. Řádky reprezentují jednotlivé záznamy, které se rozkládají do sloupců obsahující jednotlivé hodnoty záznamu.

Díky implementaci databází ve webových aplikacích lze jejich data zpracovávat pomocí PHP skriptů. To znamená, že se pomocí PHP a databázových dotazů načtou data z databáze, stroj PHP je zpracuje spolu s PHP kódem a následně odešle webovému prohlížeči, ze kterého přišel požadavek na zobrazení webové prezentace.



Obrázek 3: MySQL v modelu klient/server při požadavku na webovou stránku.

2.3 CSS

CSS, zkratka „Cascading Style Sheets“, v překladu „kaskádové styly“, vznikly jako sada vlastností, které nám umožňují grafickou úpravu webových prezentací. Jejich kořeny sahají až do roku 1994, kdy se začaly objevovat jejich návrhy. Samotná specifikace CSS1 se objevila v roce 1996, na kterou v roce 1998 navázala specifikace CSS2. Tyto specifikace a jejich normy jsou vyvíjeny organizací W3C (World Wide Web Consorcium), která definuje normy i pro HTML.

Jak již bylo řečeno, kaskádové styly se využívají k úpravě grafické podoby dokumentů HTML, XHTML ale také XML. Rozšiřují nám formátovací možnosti HTML elementů. Umožňují tedy přesně určit, jak a s jakými vlastnostmi se daný element zobrazí. Umožňují nám také definovat, jak bude

vypadat určitý element pro celý dokument. Například si můžeme nadefinovat barvu odkazů či podtrhávání odkazů jedním příkazem pro celý dokument. Samotné příkazy se mohou vrstvit i duplikovat, ale vždy platí jen ten poslední zpracovávaný prohlížečem.

2.4 XHTML 1.0 Strict

Ještě než se dozvíme něco o XHTML, je třeba se zmínit o jeho předchůdci, ze kterého vychází. Jde o HTML.

HTML, zkratka „HyperText Markup Language“. Jak z názvu vyplývá, jedná se o značkovací jazyk. To znamená, že veškerý obsah stránek se vkládá do značek, odborně HTML tagů či elementů, které říkají prohlížeči, jak se má objekt či text ve značkách zobrazit.

Jeho nejmodernějším nástupcem je jazyk XHTML (eXtensible HyperText Markup Language). Je kombinací jazyka XML (eXtensible Markup Language) a zmiňovaného jazyka HTML. Během svého vývoje proděl jazyk spoustu radikálních změn, mezi které patří definice nových tagů a upravení některých stávajících. Také došlo k jakémusi vyčištění jeho kódu.

Každý XHTML dokument se řídí definicí DTD standardu, kterých je více. Můžeme tedy mluvit o několika verzích XHTML.

XHTML verze:

- XHTML 1.0 Transitional
- XHTML 1.0 Strict
- XHTML 1.1

V tomto projektu jsem se řídil standardem XHTML 1.0 Strict. Proto bylo nutné na každou samostatnou stránku webové prezentace vložit DTD hlavičku, neboli Doctype.

Použitý Doctype:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

2.5 SEO

SEO (Search Engine Optimization) neboli optimalizace pro webové vyhledávače se zabývá technikou, jak co nejvíce zviditelnit internetovou prezentaci v celosvětovém bludišti webových stránek.

Vyhledávání stránek na internetu bychom mohli rozdělit do dvou základních částí. Na placené a neplacené. Placené jsou ty, kde si můžeme zaplatit pozici, na které chceme mít svou stránku/prezentaci zobrazenou. Samozřejmě, že čím více zaplatíme, tím více se zviditelníme. To se však vztahuje vždy pouze k portálu, který tuto možnost zviditelnění nabízí. Druhá část je nekomerční, tedy neplacená, a právě tou se SEO zabývá.

Jejím cílem je webovou prezentaci optimalizovat tak, aby se umisťovala mezi prvními místy při vyhledávání. K tomu slouží několik metod a postupů. Ty se mohou rozdělit na dvě velké části, kterým se říká čisté a nečisté. Nečisté se snaží zmást vyhledávací internetové roboty a přesměrovat je na vaše stránky. Čisté se snaží optimalizovat zdrojový kód stránek tak, aby byl co nejefektivnější pro vyhledání.

Existuje zhruba 10 tipů, které nám pomohou naše stránky zviditelnit. Není to však zaručený postup jak dostat naše stránky na první místo ve vyhledávačích.

10 tipů:

- Unikátní obsah stránek
- Pořízení vhodné domény

- Titulek pro každou stránku by měl být jedinečný a výstižný
- Vkládání META tagů na stránky
- Sémantický obsah webu
- Minimum parametrů v URL adrese
- Vyvarovat se duplicitnímu obsahu
- Odkazy z jiných webových portálů
- Trpělivost

2.6 JavaScript

JavaScript je dalším jazykem z dlouhé řady skriptovacích jazyků určených pro webové aplikace. Hlavním a podstatným rozdílem od již zmiňovaného jazyka PHP je fakt, že JavaScript není serverovým skriptovacím jazykem, ale klientským. Tudíž veškeré operace probíhají na straně klienta. To umožňuje menší zatěžování serveru, kdy se data nemusí nutně odesílat a zpracovávat na serveru, ale všechny výpočty a úkony se provádějí přímo na klientském počítači.

Tento fakt nám umožňuje skvělou práci s formuláři, kde se zadávaná data nemusí odesílat vždy na webový server. Tam by se vždy musela složitými postupy ověřovat jejich správnost. Pomocí JavaScriptu mohu přistupovat k jednotlivým prvkům formulářů a hlídat jejich obsah přímo na klientském počítači. Popřípadě zakázat jejich odeslání na server, pokud uživatel například nevyplnil textové políčko, které je třeba vyplnit.

3 Návrh databázové struktury

Návrh databázové struktury je prvním a nejdůležitějším krokem k vytvoření kvalitní dynamické webové prezentace. Je třeba si uvědomit, že celá prezentace bude využívat právě tuto databázovou strukturu, neboli námi navrženou databázi. Proto je nutné věnovat jí vysokou pozornost a projednat strukturu prezentace se zadavatelem zakázky.

Od zvolené struktury webové prezentace se bude odrážet celý vývoj databáze a bude jí přizpůsobována. Je tudíž nutné dokonale si promyslet rozčlenění tabulek a hlavně jejich návaznost mezi sebou. Nemalou pozornost je také potřeba věnovat datovým typům a jejich parametrům, které se v jednotlivých tabulkách budou vyskytovat.

3.1 Obecné rozdělení

Po dlouhém přemýšlení a zdlouhavých konzultacích s vedením firmy Salvete, spol. s. r. o. jsme došli k závěru, že celá prezentace bude rozdělena do celkem 5 sekcí.

Sekce:

- Home, úvodní stránka
- Služby, přehled služeb firmy
- Reference, přehled referencí a jejich databází fotografií
- Certifikáty či normy, které firma splňuje
- Kontakty, přehled kontaktů

Přičemž sekce *home* bude obsahovat pouze jen textové odstavce a rychlý, neboli zkrácený seznam poskytovaných služeb. Sekce *služby* bude obsahovat celkový přehled poskytovaných služeb firmou spolu se stručným popiskem

služby. Nejdůležitější částí webové prezentace bude sekce *reference*. Tato sekce bude návštěvníkům prezentace představovat firmu pomocí realizovaných projektů, které budou řazeny do jednotlivých kategorií. Každá reference tak ještě umožní zobrazení nejrůznějších informací o projektu a popřípadě i náhled na ni ve formě fotogalerie. Sekce *certifikáty* představí certifikáty/normy splňované firmou a poslední sekce *kontakty* umožní nalezení vhodného kontaktu.

3.2 Způsoby vytváření tabulek

Je několik způsobů jak vytvářet tabulky. V této práci vás jen okrajově seznámím, s jakými typy se můžete setkat a jaký postup jsem volil já.

Nejzákladnější a zároveň nejstarší způsob tvoření tabulek je pomocí databázových příkazů. Konkrétně pomocí příkazu *CREATE TABLE*, za kterým následuje název tabulky a v kulatých závorkách potom posloupnost názvů sloupců a jejich datových typů, popřípadě vlastností.

Použití příkazu CREATE:

```
CREATE TABLE about_text(
        id INT(9) UNSIGNED NOT NULL AUTO_INCREMET,
        text TEXT NOT NULL ,
        PRIMARY KEY (id)
    );
```

Dalším způsobem, dnes asi nejpopulárnějším a zároveň tím který jsem volil já, je vytváření tabulek přímo na webu pomocí produktu známého jako *phpMyAdmin*. Ten je poskytován většinou poskytovatelů webhostingu, u kterých si můžete objednat hosting svých stránek s podporou PHP.

Jedná se o grafické prostředí, kde si nemusíte pomatovat tu škálu databázových příkazů určených pro MySQL. Tento systém je nastaven a naprogramován tak, aby jeho uživateli co nejvíce zpříjemnil a zjednodušil

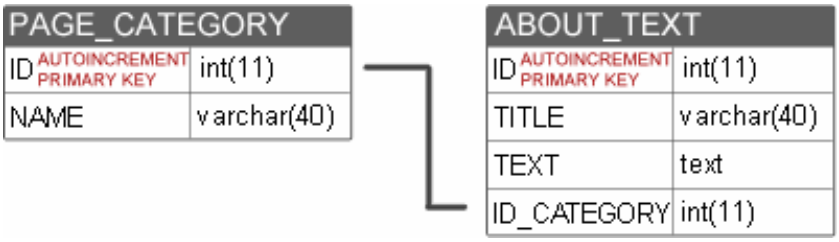
práci při navrhování databází. Vytváření a úprava tabulek zde probíhá na bázi vyplňování formulářových políček a vybírání datových typů pro jednotlivé sloupce z přednastavených hodnot. Díky tomu se s tímto systémem během krátké doby naučí pracovat i jeho naprostý neznalec.

3.3 Realizace vytváření tabulek

3.3.1 Články a jejích kategorie

Pro některé čtenáře se možná bude zdát divné, že jsem nazval tuto kapitolu jako *články* namísto *home*, o které jsem hovořil v předchozí kapitole. Avšak skutečnost je taková, že sekce *home* nepotřebuje žádnou svojí speciální tabulku či snad složitější databázovou strukturu. V této sekci nám vystačí pouze obyčejná tabulka, která bude uchovávat data jednotlivých článků.

To má své nesporné výhody. Především velikou užitečnost, neboť prezentace bude naprogramována tak, aby každá sekce zobrazovala článek, který jí přísluší. K tomu nám poslouží dvě tabulky. Tabulka *PAGE_CATEGORY* a *ABOUT_TEXT*.



Obrázek 4: Tabulky pro textový obsah všech sekcí.

První tabulka je přizpůsobena k uchovávání možných kategorií/sekcí, kde se mohou články zobrazit. Tabulku jsem pojmenoval *PAGE_CATEGORY* a obsahuje dva sloupce. První sloupec se jmenuje *ID*, obsahující celé jedinečné číslo reprezentující danou kategorii/sekci, a druhý sloupec se jmenuje *NAME*. Sloupci *ID* jsem nastavil datový typ *int(11), AUTOINCREMENT* a nastavil ho

na *PRIMARY KEY*. To vše mi zajistí automatické přičítání hodnot, a tudíž i jedinečnost hodnot nacházejících se v tomto sloupci. Sloupec *NAME* jsem nastavil na datový typ *varchar(40)*.

Druhá tabulka má o dva sloupce více. Vyskytuje se v ní opět sloupec *ID*, se stejnými vlastnostmi jako v předchozím případě a ještě sloupce *TITLE* s datovým typem *varchar(100)*, *TEXT* s datovým typem *text* a nakonec sloupec *ID_CATEGORY* typu *int(11)*.

3.3.2 Služby

Další sekcí, pro kterou jsem vytvářel tabulku, je sekce informující o službách firmy. Pro ukládání služeb jsem potřeboval pouze jednu tabulku, neboť vše, co je o službách třeba ukládat, je pouze název služby a její popis.

Výsledná tabulka s názvem *SERVICES* má však tři sloupce. Sloupec *NAME* s datovým typem *varchar(70)*, *TEXT* s datovým typem *text* a ještě sloupec *ID* se stejnými parametry, jako tomu bylo v předchozích případech a bude tomu tak i v těch následujících.

SERVICES	
ID AUTOINCREMENT PRIMARY KEY	int(11)
NAME	varchar(70)
TEXT	text

Obrázek 5: Databázová tabulka pro služby.

Ke každé službě přísluší i nějaký obrázek vyjadřující činnost oné služby, a proto by se mohlo zdát, že je nutné vytvořit ještě jednu tabulku, ve které by se ukládaly názvy obrázků a *ID* služby, ke které obrázek patří. V mnohých projektech tomu tak bývá, ale já jsem tento problém řešil jinak. Každý obrázek, který přísluší některé ze služeb, jsem přejmenovával podle *ID* služby. Díky tomu jsem si ušetřil práci se zbytečně složitým načítáním obrázků z databáze.

3.3.3 Reference a jejich kategorie

Sekce reference je nejdůležitější část této práce. Proto jí věnuji nebývalou pozornost. Je to nejsložitější část, ať už z hlediska propracovanosti návrhu databázových tabulek tak i samotného naprogramování pomocí PHP, ale o tom až později. Teď k samotnému návrhu tabulek.

Než jsem začal tabulky navrhovat, musel jsem si dobře promyslet jak jednotlivé reference nejvhodněji implementovat do databázové struktury. K dispozici jsem měl pouze nároky kladené firmou Salvete, spol. s. r. o. na jednotlivé reference.

Kladené nároky:

- Každá reference musí obsahovat textovou část zahrnující název, jméno zadavatele, období realizace a jakkoliv dlouhý popis realizace.
- Každou referenci musí být možné zařadit do nějaké kategorie.
- Kategorie si lze kompletně editovat a kromě názvu kategorie si lze zadat i její stručný popis.
- Ke každé referenci je možné přidat libovolný počet obrázků.

Po dlouhém přemýšlení a zvážení všech faktorů ovlivňující tuto část jsem došel k závěru, že pro co nejjednodušší a nejefektivnější realizaci mně postačí pouze tři tabulky.

První tabulka je nejjednodušší. Je to tabulka pro ukládání kategorií referencí. Tuto tabulku jsem pojmenoval *REF_CATEGORY* a obsahuje tři sloupce. První sloupec je *ID*, které zde slouží opět k jednoznačné identifikaci kategorie. Dalším sloupcem je sloupec uchovávající stručný název kategorie. Je známý jako sloupec *NAME* nastavený na datový typ *varchar(80)*. Posledním

sloupcem v této tabulce je sloupec *TEXT* reprezentující popis kategorie. Je nastaven jak jinak než na typ *text*.

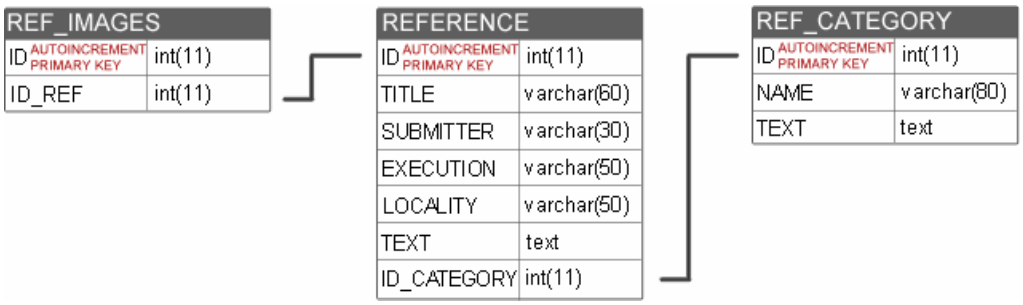
Po vytvoření tabulky pro uchovávání kategorií referencí přichází další krok. Vymyslet, jak ukládat reference do databáze a zároveň aby každé referenci příslušel různý počet obrázků, které ji reprezentují. Způsobů je určitě spousta. Jedním z nich je například ukládání fotografií do jednotlivých složek. Tedy tak, že každá reference bude mít vytvořenou složku. To by však bylo velice programově složité. Nejen, že by se musely složky vytvářet a mazat, ale ještě by se musely pokaždé kompletně procházet kvůli získání jejich struktury. Tedy k získání všech fotografií reference.

Bylo tedy třeba vymyslet něco, co by tyto složité postupy obešlo. Obejít tato složitá řešení mi umožnilo rozdělení referencí do dvou tabulek. Jedna tabulka bude uchovávat potřebné informace o referencích, spolu s *ID* kategorie do které patří, a druhá bude obsahovat seznam všech fotografií spolu s *ID* reference, ke které patří. Tohle vše nám usnadní a ošetří ještě jednu věc. A to duplicitu jmen fotografií. Ty se budou vždy jmenovat podle *ID* z tabulky, kde se uchovává seznam fotografií.

Tabulka pro uchovávání jmen fotografií se jmenuje *REF_IMAGES* a má tedy dva sloupce. Již zmiňovaný sloupec *ID* a sloupec *ID_REF*. Ještě se vrátím ke jménům fotografií. Jak jsem řekl, ty se jmenují vždy podle *ID*. To znamená, že pokud je řádek reprezentující fotografii například s *ID* rovno jedné, fotografie se jmenuje 1.jpg. Systém je nastaven tak aby mohl načítat obrázky jen typu JPG či JPEG, ale o tom později.

Poslední tabulka bude uchovávat textové data o referencích a jmenuje se *REFERENCE*. Její nedílnou součástí je opět sloupec *ID*. Dalšími jsou sloupce *TITLE* nastavený na *varchar(60)*, SUBMITTER neboli zadavatel s typem *varchar(30)*, *LOCALITY* typu *varchar(50)*, *EXECUTION* typu *varchar(50)*,

sloupec *TEXT* s datovým typem *text* a jako poslední již zmiňovaný sloupec *ID_CATEGORY*. Ten říká, do jaké sekce reference patří.

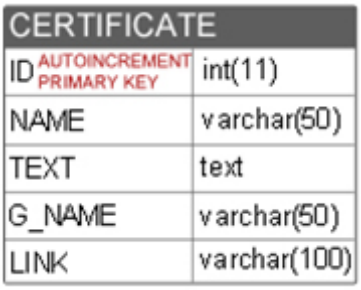


Obrázek 6: Výsledné schéma tabulek pro reference.

3.3.4 Certifikáty

Další a předposlední sekcí, pro kterou jsem navrhoval tabulky, respektive tabulku, je sekce certifikáty. Princip zobrazení a princip práce s certifikáty je stejný jako u sekce služby. Tabulka je tedy také jednoduchá.

Tabulku jsem pojmenoval *CERTIFICATE* a je tvořena sloupci *ID, NAME*, *TEXT*, *G_NAME*, *LINK*. Všechny sloupce jsou známé již z předchozích tabulek. Přibyly zde však dva nové. Sloupec *G_NAME*, který je datového typu *varchar(50)* a slouží k uchování jména společnosti, kterou byl firmě Salvete, spol. s. r. o. certifikát vydán. Další sloupec *LINK* slouží k uchování odkazu na internetové stránky certifikující společnosti. Tomuto sloupci přísluší datový typ *varchar(100)*.



Obrázek 7: Výsledné schéma tabulek pro reference.

3.3.5 Kontakty

Poslední sekcí, kterou jsem se zabýval v navrhování databázových tabulek, je sekce kontakty. U této sekce bylo opět nutné podobně jako u referencí prokonzultovat, jaké typy kontaktů se na stránkách budou zobrazovat. Jelikož tato firma se pomalu rozrůstá a do budoucna plánuje vybudovat pobočky i mimo území, kde sídlí nyní, s čímž souvisí i příval nových zaměstnanců, je nutné zvolit dva druhy kontaktů. Kontakty osob a kontakty poboček.

Z těchto faktů vyplývá, že bylo nutné navrhnout pro každou skupinu svojí vlastní tabulku, neboť kontaktní údaje osob se dost podstatně liší od kontaktních údajů poboček firmy.

Tabulka kontaktů osob se jmenuje *C_EMPLOYEE* a má 8 sloupců. Jsou to sloupce *ID*, *F_NAME*, *S_NAME*, *TITLE_1*, *TITLE_2*, *TITULAR*, *POSITION* a sloupec *TEL*. Sloupec *F_NAME* typu *varchar(15)* slouží k uchovávání křestního jména a sloupec *F_NAME* typu *varchar(20)* k ukládání příjmení. Dalšími sloupci jsou *TITLE*_1 a *TITLE_2*. Oba dva sloupce slouží k přidávání různých poznámek ke kontaktu a jsou nastaveny na datový typ *text*. Sloupec *TITULAR* je nastaven na *varchar(12)* a poslední sloupec *TEL* na *varchar(13).*

C_EMPLOYEE	
ID AUTOINCREMENT PRIMARY KEY	int(11)
F_NAME	varchar(15)
S_NAME	varchar(20)
POSITION	varchar(40)
TITLE_1	text
TITLE_2	text
TITULAR	varchar(12)
TEL	varchar(13)

Obrázek 8: Tabulka pro kontakty na osoby.

Úplně poslední tabulkou, kterou jsem se zabýval, je tabulka kontaktů pro sídla firmy. Tu jsem pojmenoval *C_PLACES*, neboli kontaktní místa. Tato tabulka je složená ze sloupců *ID*, *INSTITUDE* neboli typ kontaktního místa a je nastaven na datový typ *varchar(30)*, *RECIPIENT* neboli název, pod kterým vystupuje. Ten se používá například na dopisových obálkách. Dále se tam vyskytuje sloupec *STREET* obsahující název ulice. Oba tyto sloupce jsou nastaveny na datový typ *varchar(30)*. Dalším sloupcem je sloupec *NUMBER* uchovávající číslo popisné. Mohlo by se zdát, že tento sloupec bych mohl nastavit na datový typ *decimal*, který umožňuje uchovávat pouze číselné hodnoty. Ale není tomu tak. Je třeba si uvědomit, že některá popisná čísla obsahují i lomítko, a to není číselný znak. Proto jsem sloupec nastavil na *varchar(10)*. Sloupec *CITY* uchovává název města, kde se sídlo firmy nachází, a je typu *varchar(30)*. Dalším sloupcem je sloupec *POST_CODE*. Ten uchovává poštovní směrovací čísla a teprve u něj jsem mohl použít datový typ, který umožňuje uchovávat pouze číselné znaky. Tento sloupec jsem ještě omezil na délku 5. Výsledný datový typ jsem tedy zvolil *decimal(5,0)*. Dalšími sloupci jsou sloupce *TEL* a *FAX*, které jsou nastaveny na *varchar(13)*. Poslední sloupec se jmenuje *MAIL* a je nastaven na typ *varchar(30)*.

C_PLACES	
ID AUTOINCREMENT PRIMARY KEY	int(11)
INSTITUDE	varchar(30)
RECIPIENT	varchar(30)
STREET	varchar(30)
NUMBER	varchar(10)
CITY	varchar(30)
POST_CODE	decimal(5,0)
TEL	varchar(13)
FAX	varchar(13)
MAIL	varchar(13)

Obrázek 9: Tabulka pro kontakty poboček firmy.

4 Uživatelská sekce

Další částí tohoto projektu, kterou jsem se zabýval, bylo navržení a naprogramování uživatelské sekce internetové prezentace. Tato část na internetu prezentuje firmu Salvete, spol. s. r. o. širokému okolí, a proto jsem jí věnoval velkou pozornost.

Prioritně jsem se zabýval především grafickým zpracováním stránek. To znamená výběrem barev, návrhem designu stránek a vhodným naformátováním textů. Při grafickém zpracování jsem však musel vycházet z požadavků zákazníka. Celá prezentace měla co nejvíce obsahovat firemní barvy, kterými jsou tmavě šedá a červená s příměsí vínové.

Neméně velikou pozornost si zasloužilo i rozčlenění samotné prezentace. To mělo být jednoduché a velmi přehledné, aby zabránilo zbytečnému bloudění po stránkách. Tak je tomu u spousty dnešních internetových prezentací, kde mnohdy není poznat, čemu je prezentace věnována, nebo čím se firma, kterou stránky prezentují, zabývá.

4.1 Obecná koncepce stránek

Samotné rozčlenění stránek bylo závislé na typu menu, které je součástí každého webu. Stejně jako u jiných stránek, které jsem zpracoval, se nabízely dvě možnosti. Vytvořit stránky s horizontálním nebo vertikálním menu. Jako nejvhodnější řešení pro tyto stránky jsem zvolil horizontální menu, protože vertikální, alespoň dle mých zkušeností a úsudku, se hodí spíše pro velké internetové portály s bohatým členěním a obsahem a do internetových obchodů, kde se velice často používá stromová struktura menu.

Po zvolení horizontálního menu byla koncepce, neboli rozvržení stránky více méně jasné. Stránka bude rozdělena na tři základní horizontální oddíly, které na sebe budou navazovat. Stejně jako je tomu u většiny internetových stránek s horizontálním menu.

Části:

- Horní část top area
- Obsahová část kontent area
- Autorská část map area

4.1.1 Horní část „top area“

Celé horní části je přiřazen jeden základní oddíl, který zahrnuje všechny další. Tomuto oddílu je přiřazeno obrázkové pozadí s horizontálním opakováním. Pozadí má charakter přechodu z tmavě šedé až černé do světlejší šedi. Tyto přechody jsou dnes hojně využívány, neboť dělají internetové stránky vzdušnějšími a mnohem prostornějšími.

Do horní části je zasazen další oddíl s pevnou šířkou nastavenou na 800px, pevnou výškou nastavenou na 223px, obrázkovým pozadím, relativní pozicí a vnějšími okraji nastavenými na hodnotu *auto*. Poslední dvě zmiňované vlastnosti nám zajistí, aby tento oddíl byl vždy zarovnán na střed okna prohlížeče.

Jako poslední je do oddílu s pevnou šířkou vložena tabulka, která obsahuje jednotlivé odkazy. Této tabulce je nastavena vlastnost absolutní pozice a umístění 5px od spodní části bloku a 30px od levé části bloku. Tato absolutní pozice nám zajistí umístění odkazů přesně na to místo, kde je v prostoru 800px x 223px chceme mít umístěny.

```
<!--TOP-->
<div class="top_out">
    <div class="top">
        <!--import links table-->
        <? include('links.php'); ?>
    </div>
</div>
```

Obrázek 10: XHTML kód horní části.

4.1.2 Obsahová část „content area“

Celé textové části, stejně jako tomu bylo u linkové a bude tomu tak i u autorské, je přiřazen oddíl rozpoložený po celé šířce prohlížeče. V tomto případě je bloku nastaveno pozadí v podobě jednoduché barvy, která přímo navazuje na barvu z horní části.

Opět je zde vložen blok s vlastnostmi, které zajistí zarovnání na střed při šířce bloku 800px. Tomuto bloku je také nastaveno obrázkové pozadí, které má za účel vystínovat celý tento blok. Jako by vrhal stín na pozadí.

Celý tento blok je rozdělen ještě na tři části. První z nich je oddíl s pevnou šířkou nastavenou na 135px a zarovnání vpravo. Vždy bude obsahovat informaci, v jaké části se zrovna vyskytujete.

Další, velmi podobná část je zarovnána doprava a rovněž s pevnou šířkou nastavenou na 220px. Tato část je poměrně významná. Obsahuje vždy rychlé odkazy na reference, které firma získala během své dlouholeté funkčnosti na trhu.

Třetí částí je samotný obsah stránek. Ten je však uzavřen v bloku s pevnou šířkou nastavenou na 440px a s levým a pravým vnějším okrajem nastavenými na 136px a 222px.

Všechny tři vyjmenované části obsahují ještě jeden vnořený blok, který má vnější okraje nastaveny na 10px. Tím zajistíme, že vkládaný text bude odsazen od okrajů a odstavce z různých sekcí se nebudou dotýkat ani se příliš přibližovat. Velké zhuštění textu by negativně působilo na čitelnost a přehlednost stránek.

Jelikož jsem u předchozích dvou částí použil zarovnání bloku do strany, tzv. *float*, je třeba za ně vložit ještě jeden oddíl, který prohlížeči potvrdí konec těchto bloků. Pokud bychom tak neučinili, prohlížeč by stránku a oddíly se zarovnáním špatně zobrazoval. Oddílu je proto nutné nastavit vlastnost *clear*

s hodnotou *both*. Jelikož nechci, aby se mi tento oddíl zobrazoval na stránkách, nastavil jsem mu ještě jednu vlastnost. A to *visibility* nastavenou na hodnotu *hidden*.

```
<!--TEXT AREA-->
<div class="text_area_out">
    <div class="text_area">

        <div class="cat_area">
            <div class="in">
                <!-- info where you are -->
            </div>
        </div>

        <div class="content">
            <div class="in">
                <!-- content part -->
            </div>
        </div>

        <div class="fast_ref">
            <div class="in">
                <!-- fast link to reference -->
            </div>
        </div>

        <div class="cleaner"><!--invisible float cleaner--></div>

    </div>
</div>
```

Obrázek 11: XHTML kód obsahové části.

4.1.3 Autorská část „map area“

Tato část se využívá k informaci, kdo stránky vytvářel. Řešení této části je téměř stejné s horní částí. V bloku, který pokrývá celou šíři obrazovky, se opět vyskytuje obrázek na pozadí. Jen s opačným přechodem, který navazuje barevností na obsahovou část.

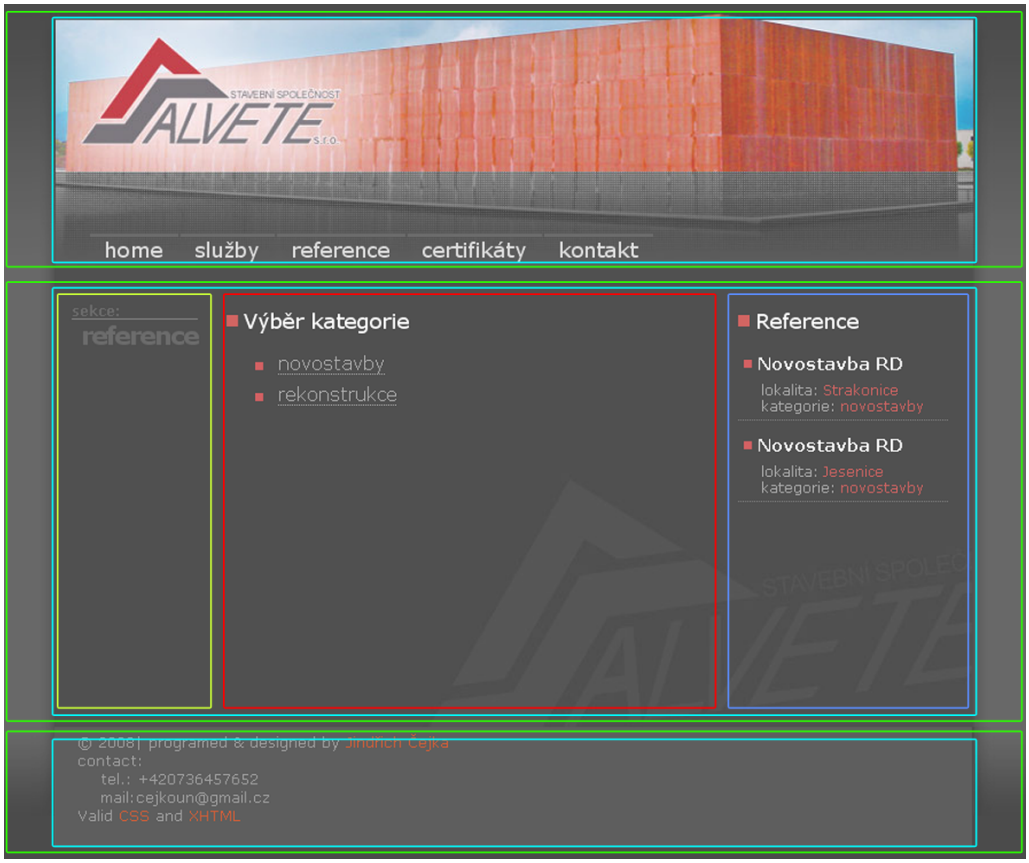
Uvnitř bloku se opět nachází obrázek na pozadí, šířka bloku je nastavena na 800px s relativní pozicí a automatickými vnějšími okraji. Výška bloku je dynamická vůči obsahu a jediné, co zde přibylo, jsou vnitřní okraje nastavené na 20px.

```
<!--map area-->
<div class="map_out">
    <div class="map">
        <!--signature-->
    </div>
</div>
```

Obrázek 12: XHTML kód autorské části.

4.1.4 Výsledná koncepce

Pokud dáme postupně dohromady všechny tři sekce, dostaneme výsledek, který můžeme vidět na obrázku pod odstavcem. Jsou na něm barevně znázorněny jednotlivé bloky, které byly popsány a ze kterých je složena každá stránka/sekce uživatelského rozhraní.



Obrázek 13: Výsledné složení třech částí stránek.

4.2 Metodika řešení

Než jsem se pustil do samotné realizace naplňování stránek pomocí PHP skriptů a načítání dat z MySQL databáze, bylo třeba si důkladně promyslet, jakým způsobem budu přistupovat k jednotlivým sekcím stránek, tedy jakou metodiku řešení zvolit.

U každého projektu programovaného PHP skriptem se naskýtají vždy dvě možnosti postupu. Každá má své klady, ale také zápory.

První možnost je přistupovat ke každé sekci jako k samostatné stránce. Tato metoda se používá u běžných statických WWW prezentací. Pro každou sekci se vytvoří jedna kompletní stránka, která může být zobrazena sama o sobě a přistupuje se k ní pomocí odkazování na ni.

Druhým přístup je určen pouze pro skriptovací jazyky jako je třeba PHP. Tento přístup je více profesionální, ale ne vždy využívaný. Princip spočívá ve vytvoření jedné šablony neboli kostry pro všechny sekce, do které se obsah doplňuje na základě parametrů. Tato metoda má mnoho výhod, ale i mnoho nevýhod. Její nespornou výhodou je ušetření práce vytvořením právě jedné kostry, která se nemusí zbytečně kopírovat do každé stránky sekce. Tak jak je tomu u statických stránek. Její nevýhodou oproti statickým stránkám je složité větvení podmínek *if* nebo rozsáhlé odkazovací seznamy *switch*. Toto tvrzení platí zejména u gigantických portálů s nespočetným množstvím sekcí. V tomto případě se vytvoří jen holá kostra dokumentu, tak jak jsem ji popisoval v kapitole *Koncepce*, a zbylý kód se ukládá do externích souborů, které se volají PHP příkazem *include()*. Pro každou sekci se vytváří právě jeden tento externí soubor.

Pro tuto část projektu jsem si vybral první způsob. Vedlo mě k němu několik důvodů. Tím nejhlavnějším byla pevně stanovená struktura celé prezentace spolu se složitým větvením struktury při projíždění přehledu

jednotlivých referencí. Avšak k tomuto postupu budu přistupovat tak, abych na prezentaci mohl vždy aplikovat druhý zmiňovaný způsob.

4.3 Odkazové menu „links“

Jelikož každá prezentace má několik sekcí, je třeba se na ně nějakým způsobem odkazovat. K tomu nám slouží HTML odkazy, které sem vkládal do jedné tabulky, respektive do jednotlivých buněk tabulky. To mi umožnilo dobré naformátování a umístění odkazů pomocí CSS.

```
<table class="links"  cellpadding="0" cellspacing="0" style="margin:0px" >
    <tr>
        <td><a href="index.php">home</a></td>
        <td><a href="sluzby.php">služby</a></td>
        <td><a href="reference.php">reference</a></td>
        <td><a href="certifikaty.php">certifikáty</a></td>
        <td><a href="kontakt.php">kontakt</a></td>
        <td> </td>
    </tr>
</table>
```

Obrázek 14: XHTML kód tabulky s odkazy

Tabulka s odkazy je stejná pro všechny sekce prezentace. Bylo by tedy zbytečné, aby se tabulka vkládala na každou stránku zvlášť. Proto jsem tuto tabulku uložil do externího souboru s názvem *links.php*, který se vkládá do horní části zvané top každé stránky PHP příkazem pro vnoření externích souborů *include()*.

```
<!--TOP IMAGES-->
<div class="top">
    <?php include('links.php') ?>
</div>
```

Obrázek 15: Vnoření externího souboru links.php.

4.4 Spojení s databází

Jako každá jiná, tak i tato dynamická prezentace načítá svůj textový obsah kompletně z databáze. Prvním krokem předcházejícím úspěšnému načtení dat z databáze, je navázání spojení s databází, ke které se následně připojíme. K připojení k databázi jsem používal PHP příkaz *mysql_connect()* s třemi parametry. Parametry *hostname*, neboli adresa hostitelského počítače, *db_name*, jméno databáze a *password*. Tedy heslo pro připojení k databázi.

Výsledek připojení se ukládá do proměnné. A to z několika důvodů. Jedním může být například ošetření navázání spojení, kdy se proměnná s výsledkem vloží do podmínky *if*, která nám ošetření zajistí. Pokud se spojení nepodařilo navázat, proměnná nabývá hodnoty 0, se kterou PHP podmínka pracuje jako s hodnotou *FALSE*. Tedy neúspěch, který zabrání dalšímu pokračování.

```
$CONN = mysql_connect("sql.web4u.cz", "salvetest" , "trtvl550");
```

Obrázek 16: PHP skript pro navázání s pojení s databází.

Jak již bylo zmiňováno, tak z databáze se přenáší jen data v podobě textu. Proto je třeba myslet na jejich korektnost. Tím je myšleno zajistit správné načítání jednotlivých znaků. K tomu mně pomohlo několik PHP příkazů, u kterých je navoleno, jaké znakové sadě všechny načítané či odesílané znaky přísluší. V každém případě jsem k tomuto nastavení užil příkaz *mysql_query()*, jen s jiným parametrem.

```
mysql_query('SET character_set_results=UTF8');
mysql_query('SET character_set_connection=UTF8');
mysql_query('SET character_set_client=UTF8');
mysql_query('SET NAMES utf8');
mysql_query('SET CHARACTER SET utf8');
```

Obrázek 17: Nastavení znakové sady transportu dat mezi databází a serverem.

Pro transport dat jsem zvolil stejnou znakovou sadu jako pro celou prezentaci. Tedy *UTF 8*. První příkaz nastavuje znakovou sadu pro výsledky, které se vrací po dotázání na databázi. Druhý nastavuje znakovou sadu spojení a třetí znakovou sadu klientskému počítači, tedy běžnému návštěvníkovi prezentace. Čtvrtý a pátý nastavují kódování globálně pro všechny úkony spojené s transportem dat mezi databází, serverem a klientem.

Skripty uvedené v této kapitole jsem vložil do externího souboru *set_connection.php*, neboť navázání spojení a nastavení kódování je stejné pro celý tento projekt a je zbytečné jej vkládat do každé stránky zvlášť.

Tento soubor se opět vnořuje příkazem *include()*. Je nutné vnořit tento soubor do stránky vždy dříve, než se začnou provádět operace s databázemi. Z tohoto důvodu jsem jej vkládal zpravidla na začátek XHTML dokumentu.

4.5 Jednotlivé sekce

4.5.1 Společné rysy sekcí

Všechny sekce této prezentace mají několik společných rysů. Jedním z nich jsou články sekcí. Prezentace je naprogramována tak, aby každá sekce prezentace mohla obsahovat libovolný počet článků, které budou vypovídat o tom, co se v dané sekci nachází.

K výpisu článků jsem používal dvě tabulky. Tou první je tabulka *ABOUT_TEXT* obsahující název, obsah a id kategorie článku. Druhou tabulkou je *PAGE_CATEGORY* obsahující seznam kategorií, kterým mohou být články přidruženy.

Samotný výpis je rozdělen do dvou fází. První fáze spočívá v získání záznamů z databáze a druhou fází je zpracování těchto výsledků.

Načítání záznamů bylo provedeno PHP příkazem *mysql_query()* s parametrem v podobě databázového dotazu typu *SELECT*, ve kterém je užito

spojování dvou tabulek pomocí klíčového slova *INNER JOIN* a klíčového slova *ON*, které nastavuje na které sloupce se má návaznost uplatnit. V tomto případě jsou to sloupce *ID* a *ID_CATEGORY*. Nedílnou součástí dotazu je také omezující podmínka *WHERE*, kterou zajistíme získání výsledků pouze zařazených do potřebné sekce. Výsledky dotazu se ukládají do proměnné, která po načtení obsahuje tabulku s výsledky.

```
$RESULT = mysql_query("SELECT TITLE , TEXT
                       FROM salvetest.ABOUT_TEXT
                       INNER JOIN salvetest.PAGE_CATEGORY
                       ON ABOUT_TEXT.ID_CATEGORY = PAGE_CATEGORY.ID
                       WHERE NAME = 'home'"));
```

Obrázek 18: Ukázka MySQL dotazu pro získání výsledků sekce home.

Poslední fází byl samotný výpis získaných dat. Výpis se provádí cyklickým procházením tabulky uložené v proměnné *RESULT*. Jako procházecí cyklus jsem použil cyklus *while*, kde jsem v každém kole získal pomocí PHP příkazu *mysql_fetch_array()* jednu řádku z tabulky. Řádka je v každém kole ukládána do proměnné *ROW*, která se chová jako asociativní pole. K jednotlivým prvkům pole se tedy přistupuje pomocí jména, které je shodné se jménem sloupce hodnoty.

```
while($ROWS = mysql_fetch_assoc($RESULT)){
    echo '<div class="partion" >';
    echo '<h1><img src="dot.jpg" alt="dot" />'.$ROWS["TITLE"].'</h1>';
    echo '<p class="text_move">'.$ROWS["TEXT"].'</p>';
    echo '</div>';
};
```

Obrázek 19: Ukázka cyklického výpisu článků.

4.5.2 Home

Sekce *home* je výchozí stránka celé prezentace firmy Salvete, spol. r. o. Jejím obsahem jsou vždy články popisující firmu. Za nimi následuje zkrácený výpis služeb spolu s obrázkem, který je ke službě přiložen.

Výpis článků tedy probíhá přesně tak, jak jsem jej popsal v předcházející kapitole, kde jsem použil jako ukázku kód pro výpis článků pro sekci *home*.

Výpis služeb popisuji v další kapitole, která je určena celé sekci služby a odkud je použit jeho zdrojový kód a MySQL dotaz pro načtení dat z databáze. Avšak jeden rozdíl je patrný. V sekci *home* se nevypisují podrobnosti o jednotlivých službách. Vypisuji zde jen název služby a její obrázek.

4.5.3 Služby

Tato sekce je určena pro výpis seznamu služeb, které firma poskytuje. Výpisu služeb však předcházejí články patřící do této kategorie. Tento výpis probíhá stejně jako pro sekci *home*, jen s jiným parametrem v podmínce *WHERE*.

Výpis služeb a jeho databázový dotaz je poněkud jednodušší než výpis článků. K výpisu bylo zapotřebí použít pouze jednu databázovou tabulku, konkrétně *SERVICES*, což zjednodušilo celý MySQL dotaz, ve kterém jsem nemusel používat návaznost tabulek jako u výpisu článků. V tomto případě jsem nemusel použít ani omezující podmínku *WHERE*, neboť se provádí výpis všech služeb.

```
$RESULT = mysql_query("SELECT * FROM salvetest.SERVICES");
```

Obrázek 20: MySQL dotaz pro získání služeb.

Zatímco se databázový dotaz zjednodušil, zpracování výsledků se ztížilo. Opět výsledek prochází cyklem, ale je obtížněji zpracováván. Každou službu vkládám do samostatného bloku, který obsahuje další dva bloky. První má nastavené zarovnání vlevo a obsahuje obrázek přiložený k sekci. Tento obrázek se jmenuje stejně jako *ID* služby, jen s příponou *jpg*. Při načítání obrázku

využívám právě zmiňované *ID*, které skládám dohromady s příponou. Druhý oddíl má zarovnání vpravo a obsahuje název služby a její detailní popis.

```
while($ROWS = mysql_fetch_array($RESULT)){
    echo '<div class="text_move">';
        echo '<table><tr><td>';
                    echo '<img src="ser_pic/'.$ROWS["ID"].'.jpg" alt="services pic." />';
                echo '</td><td>';
                    echo '<h2>'.$ROWS["NAME"].'</h2>';
                echo '</td></tr></table>';
    echo '</div>';
    };
```

Obrázek 21: Kód pro výpis služeb.

4.5.4 Reference

Tato sekce je nejdůležitější a nejnáročnější etapou této části projektu. Poprvé se zde vyskytuje členění na základě parametru *category*. Pomocí tohoto parametru z globálního pole *_GET* určuji, jestli se má provést výpis kategorií do kterých jsou reference zařazeny nebo výpis referencí zvolené kategorie.

Princip výběru je jednoduchý. Pokud je nastaven parametr *category,* provede se výpis referencí pro danou kategorii. V opačném případě se provede výpis kategorií. Členění řeším pomocí podmínek *if* a příkazu *isset()*, který vrací hodnotu *TRUE,* pokud je testovaná proměnná nastavena. Pokud není, vrací hodnotu *FALSE*.

```
if(!isset($_GET['category'])){
    /* VÝPIS KATEGORIÍ */
}else{
    /* VÝPIS REFERENCÍ ZVOLENÉ KATEGORIE */
    };
```

Obrázek 22: Kód ukázka členění.

4.5.4.1 Výpis kategorií

Veškeré kategorie, do kterých se všechny reference člení, jsou uloženy v tabulce *REF_CATEGORY* obsahující název a identifikační číslo kategorie. K načítání kategorií se používá pouze tato tabulka. To znamená, že data

získávám z databáze nejjednodušší verzí dotazu *SELECT*. Stejně jako při dotazování se na jednotlivé služby.

```
$RESULT = mysql_query("SELECT * FROM salvetest.REF_CATEGORY");
```

Obrázek 23: Dotaz pro získání kategorií referencí.

Jednotlivé záznamy uložené jako vždy v proměnné RESULT zpracovávám cyklem. V každém kole potom vytvářím jeden prvek XHTML seznamu. Abych však dosáhl onoho členění, musím jednotlivé kategorie vkládat do XHTML odkazu, který se odkazuje sám na sebe. Tedy na stránku, kde se nachází. Do *URL* adresy však ještě přidávám parametr *category*, kterému je přiřazována hodnota jména kategorie.

```
echo '<hl><img src="dot.jpg" alt="dot" />Výběr kategorie</hl>';
echo '<ul>';
while($ROWS = mysql_fetch_array($RESULT)){
    echo'<li>';
        echo '<a href="reference.php?category='.$ROWS["NAME"].'"
                name="reference">'.$ROWS["NAME"].'</a>';
    echo '</li>';
};
echo '</ul>';
```

Obrázek 24: Kód výpisu seznamu odkazů na jednotlivé kategorie.

4.5.4.2 Výpis referencí

Výpis referencí je plně závislý na zvolené kategorii a je podstatně složitější než výpis samotných kategorií. K tomuto výpisu se využívá všech tabulek určených referencím. Jsou to tabulky *REFERENCE*, *REF_IMAGES* a *REF_CATEGORY*. První tabulka uchovává veškeré informace o referenci a druhá v sobě nese informaci, který obrázek patří k jaké referenci.

Jelikož bylo potřeba získávat pouze reference z dané kategorie, musel jsem v databázovém dotaze použít propojení dvou tabulek, konkrétně *REFERENCE* a *REF_CATEGORY*, a jeden parametr, kterým volím zvolenou kategorii.

```
$RESULT = mysql_query("SELECT REFERENCE.ID, REFERENCE.TITLE, LOCALITY,
                       EXECUTION, REFERENCE.TEXT, REF_CATEGORY.NAME
                       FROM salvetest.REFERENCE
                       INNER JOIN salvetest.REF_CATEGORY
                       ON REFERENCE.ID_CATEGORY = REF_CATEGORY.ID
                       WHERE REF_CATEGORY.NAME =  '".$_GET['category']."'");
```

Obrázek 25: Dotaz pro získání referencí ze zvolené kategorie.

Tím jsem si zajistil získání informací o všech referencích a mohl pokračovat postupným zpracováním jednotlivých záznamů pomocí cyklu *while*. Každé referenci přísluší libovolný počet obrázků a bylo nutné, aby obecný výpis referencí vždy doprovázela jedna fotografie k referenci. Bylo tudíž nezbytné v každém kole vytvořit ještě jednu proměnnou, do které se ukládá výsledek databázového dotazu pro získání názvů obrázků patřících k dané referenci. Ta je reprezentována svým identifikačním číslem, které bylo použito jako podmínka v dotazu.

```
$RESULT2 = mysql_query("SELECT * FROM salvetest.REF_IMAGES
                        WHERE ID_REF = '".$ROWS['ID']."'");
```

Obrázek 26: Dotaz pro získání názvů obrázků reference.

Po získání všech záznamů bylo zapotřebí vybrat pouze jeden. Vždy jsem volil ten první. Vybíral jsem tedy první řádek z proměnné *RESULT2*, který má vždy pořadové číslo 0. K tomu mi posloužil PHP příkaz *mysql_result()*, jehož výsledek jsem ukládal do proměnné *ID_FIRST*.

```
$ID_FIRST = mysql_result($RESULT2,0,"ID");
```

Obrázek 27: Získání jména prvního obrázku reference.

Samotný výpis probíhá stejně jako výpis služeb. Každá reference je vložena do jednoho bloku, který obsahuje opět dva další. První se zarovnáním vlevo obsahuje obrázek a druhý se zarovnáním vpravo obsahuje základní informaci o referenci. Pod tímto blokem je umístěn ještě odkaz s parametrem nesoucím identifikační číslo reference. Tento odkaz se odkazuje na stránku *ref.php*

zobrazující podrobné informace o jedné referenci spolu se všemi obrázky patřící k ní.

4.5.4.3 Zobrazení konkrétní reference

Jak již bylo řečeno, podrobnosti o referenci spolu s náhledem fotografií patřících k referenci se zobrazují na stránce uložené jako *ref.php*. Celá tato stránka je podmíněna vstupním parametrem *ref_id* z globálního pole *_GET*. Tento parametr udává identifikační číslo reference, kterou chceme zobrazit. Slouží tedy jako parametr do databázového dotazu *SELECT*.

```
$RESULT = mysql_query("SELECT * FROM salvetest.REFERENCE
                       WHERE ID = '".$_GET['ref_id']."'");
```

Obrázek 28: Získání dat pro konkrétní referenci.

Po získání vždy následoval výpis získaných informací. I v tomto případě tomu tak bylo. Jediný rozdíl byl v tom, že získané informace nebylo nutné procházet cyklem, neboť jsme získali pouze jeden řádek tabulky, ke kterému jsem mohl přistupovat přímo.

```
$ROWS = mysql_fetch_assoc($RESULT);
echo '<h1>'.$ROWS["TITLE"];
    echo ', <span class="red">'.$ROWS["LOCALITY"].'</span>';
echo '</h1>';
echo '<p>'.$ROWS["TEXT"].'</p>';
echo '<span class="red"> realizace: </span>';
echo '<span class="white">'.$ROWS["EXECUTION"].'</span>';
```

Obrázek 29: Výpis informací o referenci.

Hned po vypsání informací následuje výpis obrázků. Ten se skládá ze dvou kroků. Nejprve je třeba načíst první v plné velikosti a potom všechny zmenšeniny. Obrázek načtený v plné velikosti slouží zároveň jako zobrazovací prostor pro všechny ostatní obrázky. Musel jsem mu tedy nastavit atribut *name="big"*, na který se odkazuji. To jsem zajistil JavaScriptem, kde jsem u zmenšenin použil akci *onclick*, odkazující se právě na tento atribut. Akce

onclick s nastaveným novým zdrojem obrázku zajistí zobrazení nového velkého obrázku namísto původního.

```
$RESULT = mysql_query("SELECT * FROM salvetest.REF_IMAGES
                       WHERE ID_REF = '".$ID."'");
$ID_FIRST = mysql_result($RESULT,0,"ID");

echo '<div class="big_image_box">';
    echo '<img src="ref_images/'.$ID_FIRST.'.jpg" name="big" alt="big" />';
echo "</div>";
```

Obrázek 30: Kód pro vložení zvětšeniny obrázku.

Po načtení velkého obrázku bylo třeba načíst všechny zmenšeniny obrázků patřících k referenci. I když jsem již používal dotaz pro získání jejich jmen, je nutné tento dotaz provést znovu a následně cyklem *while* provést výpis všech. V následující ukázce je vidět již zmiňovaný JavaScript.

```
$RESULT = mysql_query("SELECT * FROM salvetest.REF_IMAGES
                       WHERE ID_REF = '".$ID."'");
while($ROWS = mysql_fetch_array($RESULT)){
echo "<img src=\"ref_images/".$ROWS["ID"]."s.jpg\" alt=\"small\"
    onclick=\"big.src='ref_images/".$ROWS["ID"].".jpg';return true;\" />";
};
```

Obrázek 31: Kód pro vložení zmenšenin obrázků.

4.5.5 Certifikáty

Načtení a výpis certifikátů je kombinací sekce reference a služby. Byl zde použit opět nejjednodušší typ dotazu *SELECT* stejně jako u sekce služby a podobné cyklické zpracování. Ze sekce reference jsem převzal členění výpisu celé stránky. Celý tento výpis je závislý na parametru *cert_id*. Pokud je tento parametr nastaven, provede se podrobný výpis certifikátu, vedený pod identifikačním číslem shodným s parametrem. Parametr jsem v tomto případě použil i v podmínce dotazu *WHERE* a výsledek dotazu se nezpracovával cyklicky, neboť byl vrácen pouze jeden záznam. Stejně jako u konkrétní reference. Pokud však není nastaven, provede se výpis všech certifikátů.

Při výpisu seznamu certifikátů je postup stejně jako u výpisu služeb. Stejné rozložení a stejné načítání obrázků. Jen v pravé sekci, obsahující informace, přibyl odkaz s parametrem. Ten zajišťuje nastavení parametru *cert_id* a tím pádem i výpis jednoho konkrétního certifikátu.

4.5.6 Kontakty

Sekce kontakty obsahuje dvě kategorie kontaktů. Jsou to kontakty na osoby, tedy na vybrané zaměstnance firmy, a na místa, tedy pobočky firmy. Každá tato kategorie má svojí specifickou tabulku, a proto je výpis kontaktů rozdělen do dvou částí. Výpis kontaktů zaměstnanců a výpis kontaktů sídel firmy.

4.5.6.1 Výpis kontaktů zaměstnanců

Kontaktní údaje jednotlivých zaměstnanců jsou uloženy v tabulce *C_EMPLOYEE*. Jak je již patrné, databázový dotaz typu *SELECT* byl jednoduchý a bezpodmínkový. Pouze volá všechna data z tabulky.

```
$RESULT = mysql_query("SELECT * FROM salvetest.C_EMPLOYEE");
```

Obrázek 32: Dotaz na kontaktní osoby.

Data přijatá ze serveru, uložená v proměnné *RESULT*, se zpracovávají cyklem pomocí PHP příkazu *mysql_fetch_array()*.

4.5.6.2 Výpis kontaktů na sídla firmy

Výpis kontaktů na pobočky firmy byl proveden stejným způsobem. Jediný rozdíl byl v použití jiné tabulky, konkrétně *C_PLACES* obsahující jiné sloupce než *C_EMPLOYEE*, a jiným zpracováním jednotlivých sloupců uvnitř zpracovávajícího cyklu.

```
$RESULT = mysql_query("SELECT * FROM salvetest.C_PLACES");
```

Obrázek 33: Dotaz na kontaktní místa.

5 Administrátorská sekce

Administrátorská sekce slouží pověřeným zaměstnancům firmy Salvete, spol. s. r. o. k editaci obsahu celé internetové prezentace. Tedy přidávat, upravovat či mazat jednotlivé záznamy jako jsou například kontakty, články a mnoho dalších.

Administrační část prezentace je uložena ve složce *admin* na serveru, kde je uložena celá prezentace. Spuštění administrátorské části vyvoláme přidáním řetězce */admin* za URL adresu prezentace.

5.1 Metodika řešení

I v této části, obdobně jako při realizaci uživatelské sekce, se naskýtají dvě metodiky řešení. Stejné jako v předchozím případě. Při realizaci administrátorské sekce jsem se však rozhodl užít druhou metodiku. Ta je více v souladu s vizí dynamičnosti WWW stránek při využívání skriptovacího jazyka PHP a databázového systému MySQL.

Celý princip spočívá ve vytvoření jedné společné kostry pro všechny sekce, kde se obsah sekcí vybírá na základě parametru. Název parametru jsem zvolil *page,* kterému je přiřazován název sekce, ve které se administrátor zrovna vyskytuje nebo do které vstupuje.

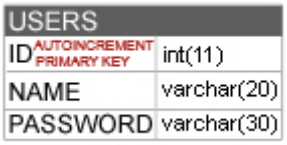
5.2 Přihlašování uživatelů

Prvním požadavkem firmy byl autorizovaný přístup do administrátorské sekce prezentace, neboť je nepředstavitelné, aby si každý návštěvník mohl přidávat jakýkoliv typ záznamu, což toto prostředí umožňuje.

5.2.1 Databáze uživatelů

Abych mohl zabránit vniknutí do systému neoprávněným osobám, bylo třeba vytvořit ještě jednu tabulku v databázi. Tabulku jsem pojmenoval *USERS*

a uchovává jména a jim přidružená hesla. Tabulka obsahuje sloupce *ID* pro pozdější editaci uživatelů, sloupec *NAME* a sloupec *PASSWORD*.



Obrázek 34: Tabulka uživatelů.

5.2.2 Princip a realizace přihlašování

Základem přihlašování bylo vytvořit stránku s formulářem, který se zobrazí vždy po zadání již zmiňované URL adresy. Tento formulář obsahuje dvě políčka *input* a jedno potvrzovací tlačítko se jménem *log_button*. Jedno s atributy *type="text" name="login"* a druhý s atributy *type="password" name="password"*. Formulář je uložen v externím souboru, pojmenován jako *login_form.php*, a vkládá se do startovací stránky administrátorské sekce.

Princip celého ověřování autorizace uživatele je poměrně jednoduchý. Používám pro realizaci globální pole *_SESSION*, konstantu pojmenovanou *GROUP* definovanou vždy na začátku každé sekce administrátorské části a ještě načítání dat z databáze, konkrétně jmen a hesel. Abych mohl pracovat s polem *_SESSION* a konstantou *GROUP*, je nutné si je na začátku každé stránky nastavit. Proto jsem jejich nastavení uložil do externího souboru *for_auth.php*, který vkládám vždy na začátek každé sekce.

```
session_start();
header("Content-Type: text/html; charset=utf-8");
define('GROUP','salvete');
```

Obrázek 35: Obsah souboru for_auth.php.

A jak to celé funguje? Po zadání uživatelského jména a hesla se stránka zavolá znovu. Na začátku stránky je umístěn PHP skript ověřující platnost zadaných hodnot. Tento PHP skript je celý v podmínce, kde kontroluji, jestli bylo tlačítko stisknuté nebo ne. Jestliže tlačítko bylo zmáčknuté, skript ověří existenci hodnot v databázi. Akce se provádí pomocí dotazu *SELECT*, kde se v podmínce *WHERE* použije právě ono zadané jméno a heslo. Ty jsou uloženy v globálním poli *_POST*, kam se ukládají vždy prvky zadávané do formulářů.

Výsledek dotazu ukládám do proměnné *RESULT*, ze které si pomocí PHP funkce *mysql_fetch_array()* přiřadím všechny vrácené záznamy do proměnné *ROW*. Pokud proměnná *RESULT* obsahuje alespoň jeden záznam, tedy alespoň jednu řádku, zadané jméno a heslo existuje a konstantu *GROUP* mohu vložit do globálního pole *_SESSION*. K tomuto ověření jsem použil podmínku *if* a PHP funkci *mysql_num_rows()*, která vrací počet záznamů v proměnné zadané jako parametr.

```
if($_POST['log_button']){
    $RESULT = mysql_query("SELECT * FROM salvetest.USERS
                           WHERE LOGIN='".$_POST['login']."'
                           AND PASSWORD='".$_POST['password']."'");

    $ROWS = mysql_fetch_array($RESULT);

    if(mysql_num_rows($RESULT)>0){
        $_SESSION[GROUP]['LOGIN'] = $ROWS['LOGIN'];
    }else{
        $CHYBA = "Špatne zadané jméno nebo heslo!";
    };
};
```

Obrázek 36: Ověřování pravdivosti přihlašovacích údajů.

K tomu, abychom zabránili zobrazení obsahu stránek nežádoucím osobám, bylo třeba provést ještě jeden triviální krok. Tento krok spočívá ve vložení jedné podmínky, ve které se ptám, jestli je nastavena konstanta *GROUP* v globálním poli *_SESSION*. Pokud je nastavena, může se provést výpis zabezpečeného obsahu. V tomto případě jednoduché menu s výběrem

upravovaných sekcí nebo obsah konkrétní sekce pokud je nastaven parametr *page*. Pokud konstanta není nastavena, zobrazí se znovu formulář pro zadávání přihlašovacích údajů spolu s chybovou hláškou uloženou v proměnné *CHYBA*.

```
if($_SESSION[GROUP]['LOGIN']){
    /* ZABEZPEČENÝ OBSAH */
    echo '<...
}else{
    /* FORMULÁŘ */
    include...
}
```

Obrázek 37: Kontrola nastavení konstanty.

5.3 Odhlašování uživatelů

Odhlašování jsem provedl vložením odkazu s parametrem *action=logout*. Tento odkaz je vždy nastaven na výchozí stránku administrátorské sekce. Při načítání této stránky se vždy na začátku kontroluje, jestli není nastaven parametr *action* na hodnotu *logout*. Pokud ano, vymaže se konstanta *GROUP* z globálního pole *_SESSION*. Tím jsem zabránil další manipulaci se zabezpečeným obsahem.

```
if($_GET['action']=='logout'){
    unset($_SESSION[GROUP]['LOGIN']);
};
```

Obrázek 38: Odhlašování uživatelů.

5.4 Editační sekce

5.4.1 Společné rysy sekcí

Každá sekce má vytvořenou svojí složku s pevnou strukturou svého obsahu. Všechny složky obsahují nejméně tři soubory a jmenují se podle názvu sekce.

Prvním souborem je vždy *index.php,* obsahující zdrojový kód obsahu celé sekce. Tento soubor se vnořuje pomocí příkazu *include()* přímo do kostry z požadovaného adresáře na základě parametru *page.*

```
if(isset($_GET['page'])){
    include($_GET['page'].'/index.php');
}else{
    /*MENU PRO ZVOLENI SEKCE*/
    echo '<...
}
```

Obrázek 39: Výběr a vnoření obsahu stránek.

Druhým souborem je vždy soubor *action.php.* Tento soubor se vkládá do kostry pomocí příkazu *include()* vždy zcela nahoru před veškerý zdrojový kód. Vnořování je však ještě podmíněno parametrem *page.* Pokud je nastaven, vloží se soubor *action.php* z požadovaného adresáře. Pokud není, vloží se soubor *action.php* ze složky *admin*, kde je umístěna celá administrátorská sekce. Tomuto souboru předchází pouze vnoření souborů pro navázání spojení s databází, nastavení kódování a soubor obsahující nastavení pole *_SESSION* a konstanty *GROUP*. Obsahem souboru *action.php* jsou vždy PHP skripty pro úpravy, mazání a přidávání dat a fotografií.

```
include('admin_setting/for_auth.php');
/*connection*/
include('../setting/open_connection.php');
include('../setting/set_coding.php');
if(isset($_GET['page'])){
    include($_GET['page'].'/action.php');
}else{
    include('action.php');
}
```

Obrázek 40: Vnoření nastavovacích souborů a souboru action.php.

Celá administrátorská sekce je postavena na obsluze a zpracování dat z formulářů. Proto dalším a posledním souborem v adresáři je soubor *validity_control.js*, který obsahuje skripty pro kontrolu zadaných dat do

formulářů pro danou sekci. Tyto skripty jsou napsány jazykem JavaScript a celý soubor je vložen do kostry pomocí tagu *scrip* v těle tagu *head*.

5.4.2 Editace uživatelů „users“

Tato sekce slouží k editaci uživatelů, kteří se mohou do systému přihlásit. Celá tato sekce je podmíněna parametry *id* a *action*. Parametru *action* se přiřazuje název akce, která se bude provádět, a parametru *id* pro jakého uživatele se má akce provést.

Při vstupu do sekce není nastaven žádný parametr a provede se cyklický výpis všech uživatelů. Každému uživateli je přiřazen odkaz sloužící k editaci uživatele s parametrem *action=edit* a *id*, jemuž je přiřazeno identifikační číslo, pod kterým vystupuje. Pro přidání nového uživatele je v menu nastaven odkaz s parametrem *action=add*.

Při výběru jednoho z tlačítek se nastaví parametry a na základě podmínky se namísto výpisu uživatelů zobrazí formulář pro zadávání údajů.

Pokud uživatel použije tlačítko pro přidání uživatele, zobrazí se formulář s nevyplněnými políčky. V opačném případě se načtou data z databáze, pro uživatele s identifikačním číslem rovnému *id*, které se zobrazí ve formuláři v příslušných políčkách.

Na konci formuláře je ještě vložen blok určený pro potvrzovací tlačítka. Jejich výpis je také podmíněn parametrem *action*. Pokud tento parametr nabývá hodnoty *add*, zobrazí se tlačítka *přidat* a tlačítko pro zrušení akce *storno*. Tlačítku *přidat* je přiřazena JavaScriptová návratová metoda *controlNew()* reagující na akci *onclick* a ošetřující validitu zadaných hodnot. Pokud formulář neprojde validitou, metoda vrátí hodnotu *false* a systém nepokračuje. V opačném případě systém pokračuje v akci.

Jestliže nabývá parametr *action* hodnoty *edit*, provede se zobrazení tlačítek *storno*, *uložit* a *smazat*. Tlačítku *smazat* je přiřazena kontrolní metoda *controlDel()*, tlačítku *uložit* metoda *controlEdit()* a tlačítko *smazat* zůstane bez kontroly.

```
if($_GET['action']=='add'){
    echo "<input type='submit' name='add' value='přidat' onclick='return controlNew();'>";
    }
if($_GET['action']=='edit'){
    echo "<input type='submit' name='accept' value='uložit' onclick='return controlEdit();'>";
    }
echo "<input type='submit' name='storno'  value='storno' >";
if($_GET['action']=='edit'){
    echo "<input type='submit' name='del' value='smazat' onclick='return controlDel();'>";
    }
```

Obrázek 41: Výběr a zobrazení tlačítek.

Při potvrzení a úspěšné validaci zadaných dat se stránka zavolá znovu a vyvolá akci ze souboru *action.php*. Tato akce je závislá na zvoleném tlačítku se specifickým jménem uloženým v atributu *name*. Toto jméno se ukládá do pole *_POST* a akce se vybírá testováním právě tohoto pole pomocí podmínek *if*.

Pro tuto sekci jsou vytvořeny tři akce. První je pro přidání uživatele a používá databázový dotaz *INSERT*. Mazání funguje pomocí dotazu *DEL* a parametru *id* a změna pomocí dotazu *UPDATE* a zase parametru *id*.

```
if(isset($_POST['add'])){
    mysql_query("INSERT INTO salvetest.USERS(login,password)
                VALUES ('{$_POST['login']}',
                        '{$_POST['pass_1']}')");
}
```

Obrázek 42: Přidání uživatele.

5.4.3 Editace článků „articles“

Tato editační sekce je určena pro editaci všech článků, které mohou uživatelské sekce obsahovat. Je velice podobná jako předchozí popisovaná. Po vstupu do sekce se zobrazí jedno odkazové tlačítko s parametrem *action=add*

a standardní výpis všech článků. Každý článek je vložen do jednoho bloku, pod kterým jsou umístěna dvě odkazová tlačítka patřící ke článku. Tlačítko smazat nastaví parametr *action=del* a tlačítko *upravit* nastavuje parametr *action=edit* a parametr *id*.

Po stisknutí jednoho z tlačítek se vyhodnotí parametr a provede se jemu příslušná akce. Při akci *add* či *edit* se zobrazí formulář. Buďto vyplněný nebo prázdný. Potvrzující tlačítka jsou ošetřena JavaScriptem a opět zde probíhá výběr vkládaných tlačítek dle parametru *action*. Vždy se zobrazí tlačítko *storno* spolu s tlačítkem *přidat* nebo *změnit*. Po potvrzení formuláře se provede akce příslušná tlačítku ze souboru *action.php*.

```
/*ZMĚNA ČLÁNKU*/
if(isset($_POST['accept_change'])){
    mysql_query("UPDATE salvetest.ABOUT_TEXT
                    SET TITLE = '{$_POST['title']}' ,
                        TEXT = '{$_POST['text']}',
                        ID_CATEGORY = '{$_POST['sel_cat']}'
                    WHERE ID = '{$_POST['id']}'");
```

Obrázek 43: Databázový dotaz na změnu článku.

5.4.4 Editace sekcí služby a certifikáty „services & certificate“

Editace služeb a certifikátů pracuje rovněž na stejném principu jako obě předchozí sekce. Obsahují jedno tlačítko pro přidání služby či certifikátu a při vstupu se provede výpis všech záznamů, tak jak jsem jej prováděl v uživatelské části. Ale jen s tlačítky upravit a smazat.

V těchto sekcích je zapotřebí přidávat obrázky jednotlivým záznamům. K tomu mi posloužilo formulářové políčko *input* s atributem *type=“file“*. Obrázek se po načtení a potvrzení formuláře nejprve nahraje na server do globálního pole *_FILE,* odkud je zpracováván. Prvním krokem bylo přesunout obrázek z pole *_FILE* na určené místo, jehož cestu mám uloženou v proměnné

target. Přesunutí jsem provedl pomocí metody *move_uploaded_file()*. Následně obrázku změním velikost a přejmenuji jej podle *id* vkládaného záznamu.

```
if($_FILES['upload_file']['type']=="image/jpeg"
    or $_FILES['upload_file']['type']=="image/pjpeg"){

    $target = "./empty/" .$_FILES['upload_file']['name'];
    $name = $_FILES['upload_file']['tmp_name'];
    $copy = move_uploaded_file($name, $target);
    chmod ($target, 0644);
    $new_name = next_id();
    resize();
    rename($target, "../../ser_pic/".$new_name.".jpg");

    mysql_query("INSERT INTO salvetest.SERVICES(name,text)
            VALUES ('{$_POST['name']}','{$_POST['text']}')");

}else{
    $_GET['action']='error';
}
```

Obrázek 44: Proces přidávání nové služby s obrázkem.

Při přejmenování jsem musel brát v úvahu, jestli záznam pouze upravuji nebo přidávám nový. Pokud je záznam upravován, znám *id* a mohu rovnou obrázek přejmenovat. V opačném případě využívám návratovou metodu vracející *id*, které bude přiděleno následujícímu vkládanému záznamu. Tedy službě nebo certifikátu.

```
/*zjisti id vkladane sluzby*/
function next_id(){
    mysql_select_db("salvetest");
    $RESULT = mysql_query("SHOW TABLE STATUS LIKE 'SERVICES'");
    $ROW = mysql_fetch_assoc($RESULT);
    $NEXT_ID = $ROW['Auto_increment'];
    return $NEXT_ID;
}
```

Obrázek 45: Metoda vracející ID později vkládaného záznamu do tabulky SERVICES.

5.4.5 Editace referencí „reference“

Celou tuto sekci jsem rozdělil na dvě části. Na editaci kategorií referencí a na samotné reference. V menu jsou nyní dvě tlačítka. Jedno pro přidávání kategorií a jedno pro přidávání referencí. Změna poznamenala i výpis základního obsahu. Do toho jsou vloženy dva odkazy. Odkaz *editace kategorií* a *editace referencí*.

Editace dvou druhů záznamů v jedné sekci mělo za následek změnu ve větvení zobrazovaného obsahu. Přibil zde parametr *ref_cat*, kterému je přiřazeno jméno editované kategorie. Tento parametr je zde dominantní a může nabývat hodnot *ref* nebo *cat*, které jsou přiřazeny všem odkazovacím tlačítkům v této kategorii. Parametry *action* a *id* jsou zde zachovány, ale při zpracování přicházejí na řadu až po parametru *ref_cat*.

```
if($_GET['ref_cat']=='cat'){
    if($_GET['action']=='edit' || $_GET['action']=='add'){
        $RESULT...
    }

    if(!isset($_GET['action'])||$_GET['action']=='del'){
        $RESULT...
    }
}
if($_GET['ref_cat']=='ref'){
    if(!isset($_GET['action'])||$_GET['action']=='del'){
        $RESULT...
    }
    if($_GET['action']=='edit' || $_GET['action']=='add'){
        $RESULT...
    }
    if($_GET['action']=='edit_gal' || $_GET['action']=='del_pic'){
        echo '<...
    }

}
```

Obrázek 46: Metoda vracející ID později vkládaného záznamu do tabulky SERVICES.

5.4.5.1 Editace kategorií

Pokud si zvolím editaci kategorií, nejprve se mi zobrazí standardní výpis všech záznamů s příslušnými editačními tlačítky. Těmi jsou tlačítko *odebrat* a *upravit*. Po zvolení tlačítka pro přidání nebo upravení záznamu se opět zobrazí příslušný formulář. Princip a způsob provedení je totožný s editací článků.

5.4.5.2 Editace referencí

V této kategorii jsou podstatné změny oproti všem ostatním. Každá reference se skládá ze dvou částí. Z informativních textů a z obrázkové galerie. Při výpisu všech referencí jsem vkládal ke každému záznamu tři tlačítka. Prvním je tlačítko *odebrat*, které nechybí v žádné sekci. Druhým je tlačítko *upravit galerii* nastavující parametr *action* na hodnotu *edit_gal* a parametr *id* nastavuje na hodnotu identifikačního čísla reference, které je galerie přidružena. Posledním je tlačítko *upravit text.* Nastavuje parametr *action=edit* a *id* stejným způsobem jako u *edit_gal*.

Při zvolení úpravy textu se zobrazí formulář s údaji patřící k referenci získané dotazem *SELECT* s podmínkou *WHERE*, kde je použito *id* jako parametr.

Pokud však zvolím úpravu fotografií, zobrazí se galerie, která přísluší referenci. Fotky jsou načítány z databázové tabulky *REF_IMAGES*, kde jsou uložena jejich identifikační čísla shodující se s názvem fotografií a čísla reference fotce přidružené. Každá fotografie je dána do bloku se stylem *float:left* a je jí přiřazen odkaz s parametry *action=del_pic*, *ref_cat=ref*, *id* s identifikačním číslem upravované reference a *id_img*, kterému odpovídá *id* obrázku. V části, kde se editují obrázky, je ještě umístěn formulářový prvek pro nahrávání obrázků. Stejně jako tomu bylo u služeb a certifikátů.

```
$RESULT = mysql_query("SELECT * FROM salvetest.REF_IMAGES
                                WHERE ID_REF = '{$_GET['id']}'");
while($ROW = mysql_fetch_array($RESULT)){
echo '<div class="inbox_pic">';
    echo '<img src="../../ref_images/'.$ROW['ID'].'s.jpg" alt="small" />';
    echo '<a href="?action=del_pic&ref_cat=ref
                    &id='.$_GET['id'].'&id_img='.$ROW['ID'].'" name="del">';
    echo '<img src="../cross.jpg" alt="cross" /> smazat</a>';
echo '</div>';
```

Obrázek 47: Výpis obrázků dané reference.

Při volbě přidání nové reference se nejprve provede zobrazení formuláře, jehož potvrzovací tlačítka jsou ošetřena JavaScriptem. Po jeho úspěšném potvrzení se data uloží do databáze a zobrazí se prázdná galerie, kde je možno přidávat fotografie k referenci. K tomu, aby bylo možné přidávat fotografie, bylo nejprve třeba znát *ID* nově vloženého záznamu. K tomu jsem použil návratovou funkci *mysql_inserted_id()*. Ta vrátí identifikační číslo naposledy vloženého záznamu. Při přidávání fotografií jsem postupoval stejným způsobem jako tomu bylo u služeb a certifikátů. Změna nastala při ukládání, kdy se musel obrázek ukládat dvakrát. Jednou obrázek v určité velikosti a podruhé jeho zmenšenina. Větší obrázek se vždy jmenuje podle *ID* reference a zmenšenina se jmenuje dle *ID* s připojeným písmenem *s*. *S* jako *small*. Pro zmenšování obrázků jsem si vytvořil funkci se vstupním parametrem udávající cílovou výšku, ze které si vypočítám budoucí šířku. Princip skriptu spočívá ve vytvoření bílého plátna o velikosti vypočítaných hodnot, na které nanáším zmenšeninu obrázku. Následně si zjistím jméno obrázku, přejmenuji jej a uložím do složky *REF_IMAGES*.

```
//RESIZE
function resize($width){

    $old_size = getimagesize("./empty/".$_FILES['upload_file']['name']);
    $rate = $width/$old_size[0];
    $new_width = $rate * $old_size[0];
    $new_height = $rate * $old_size[1];

    $old=imagecreatefromjpeg("./empty/".$_FILES['upload_file']['name']);
    $new=imagecreatetruecolor($new_width,$new_height);
    imagecopyresized($new,$old,0,0,0,0,
                    $new_width, $new_height,
                    $old_size[0],$old_size[1]);
    imagejpeg($new,"./empty/".$_FILES['upload_file']['name'],99);
    imagedestroy($new);
    imagedestroy($old);
}
```

Obrázek 48: Skript pro změnu velikosti obrázku.

5.4.6 Editace kontaktů

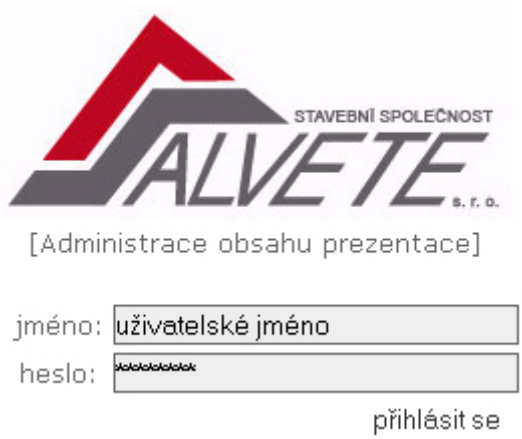
V této kategorii se vyskytují dva druhy kontaktů, a proto zde přibyl jeden dominantní parametr určující, o jaký druh kontaktu jde. Jedná se o parametr *category*, který uchovává hodnotu *emplyee* nebo *places*.

Při vstupu do této sekce se provádí výpis všech kontaktů společně s jim přidruženými tlačítky. Nejprve se provede výpis kontaktů na zaměstnance z tabulky *EMPLOYEE* a následně se provede výpis kontaktů na pobočky firmy z tabulky *PLACES*. Větvení celé této části je shodné s editační sekcí referencí. Menu pro přidávání obsahuje rovněž dvě tlačítka. Pro každý druh kontaktu jedno. Při zvolení některé z akcí, přidání či editaci kontaktu, se jako vždy zobrazí editační formulář. Záleží na zvolené akci, zda je vyplněný nebo prázdný.

6 Uživatelská příručka

6.1 Přihlášení a ohlášení z administračního systému

Administrační systém prezentace spustíme zadáním URL adresy *www.salvete-strakonice.cz/admin* do adresového řádku prohlížeče. Po jeho potvrzení se zobrazí stránka s logem firmy a jednoduchým formulářem pro zadání uživatelského jména a hesla. Pro přihlášení je nutné tyto políčka vyplnit a potvrdit tlačítkem *přihlásit se.*



Obrázek 49:Přihlašovací formulář.

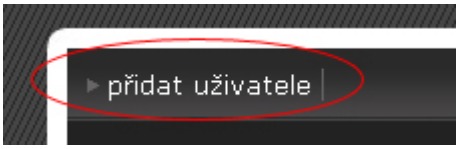
Při neúspěšném přihlášení, nejspíše z důvodu zadání špatného jména či hesla, naběhne formulář znovu. Po úspěšném přihlášení do systému se zobrazí jednoduché obrázkové menu, kde si zvolíte požadovanou sekci pro úpravu záznamů.

Obrázek 50:Menu pro volbu sekce.

6.2 Editace uživatelů

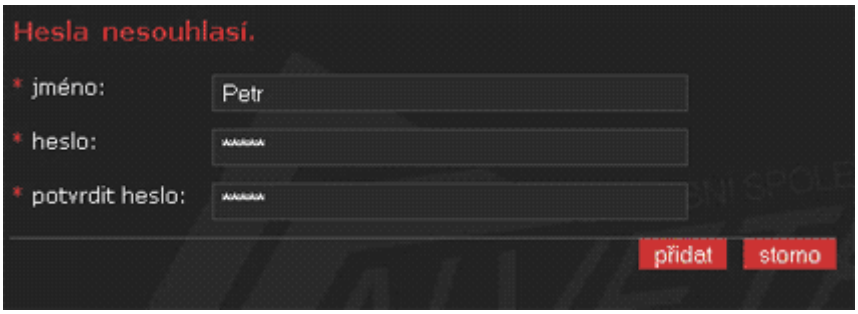
6.2.1 Přidání uživatele

Pro přidání nového uživatele je nejprve nutné stisknout tlačítko *přidat uživatele*, umístěné v menu editační části sekce.



Obrázek 51:Tlačítko pro přidání uživatele.

Po jeho stisknutí se zobrazí formulář vyzývající k zadání nového uživatelského jména a hesla. Tyto údaje jsou povinné a jsou označené červenou hvězdičkou. Heslo je z bezpečnostních důvodů nutné zadat dvakrát. Jednou do kolonky *heslo* a podruhé do kolonky *potvrdit heslo*. Hesla se musí shodovat, jinak nebudou vyplněné údaje systémem přijaty. Objeví se varovná hláška a zadání se musí opakovat. Pro potvrzení zadaných údajů je pod formulářem umístěno tlačítko *přidat*, vedle kterého je umístěno tlačítko *storno* sloužící ke stornování přidání uživatele a návratu do základní části editace uživatelů.



Obrázek 52:Formulář pro registraci nového uživatele.

6.2.2 Úprava údajů a mazání uživatelů

Pro úpravu a mazání uživatelů je třeba zvolit tlačítko *upravit* umístěné vždy u konkrétního uživatele základní části této sekce. Po jeho zmáčknutí se zobrazí editační formulář s vyplněným jménem.

Pokud chcete uživatele odstranit, je nutné zadat platné heslo do kolonky *původní heslo* a zmáčknout tlačítko *smazat*.

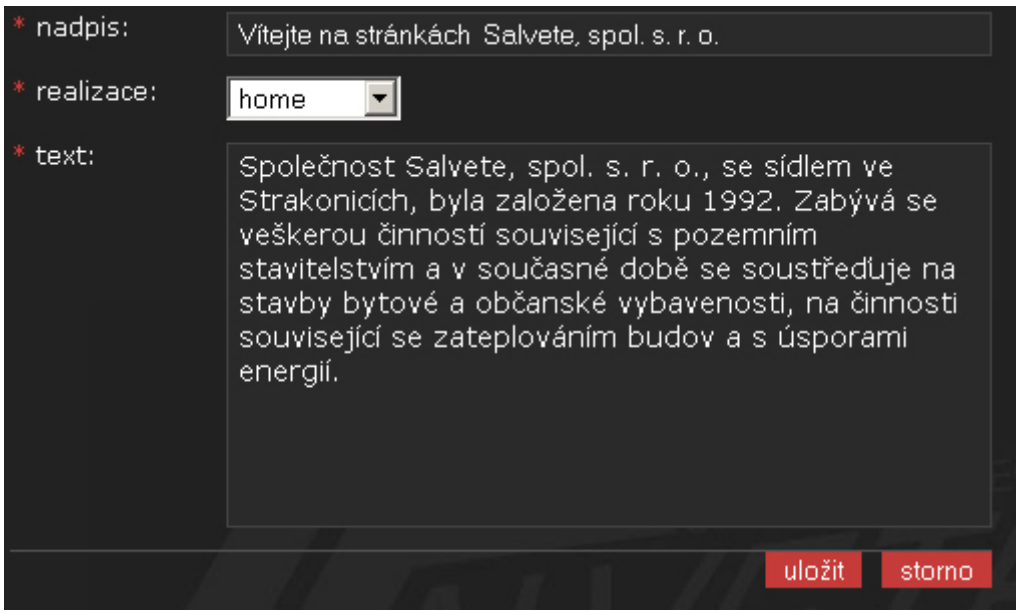
Při změně hesla je nutné zadat původní heslo do kolonky *heslo* a nové heslo zadat do kolonek *nové heslo* a *potvrdit heslo*. Opět se musí obě hesla shodovat, jinak nebude nové heslo systémem schváleno a uloženo.

Obrázek 53: Editační formulář.

6.3 Editace článků

6.3.1 Přidávání článků

Pro přidávání článků je v sekci články vloženo tlačítko *přidat článek*. Po jeho stisknutí se zobrazí formulář obsahující kolonky *název* a *text* článku. Součástí tohoto formuláře je také rozbalovací roletka pro zvolení sekce, které chcete článek přiřadit. Všechna tato políčka jsou povinná a jsou označena červenou hvězdičkou.



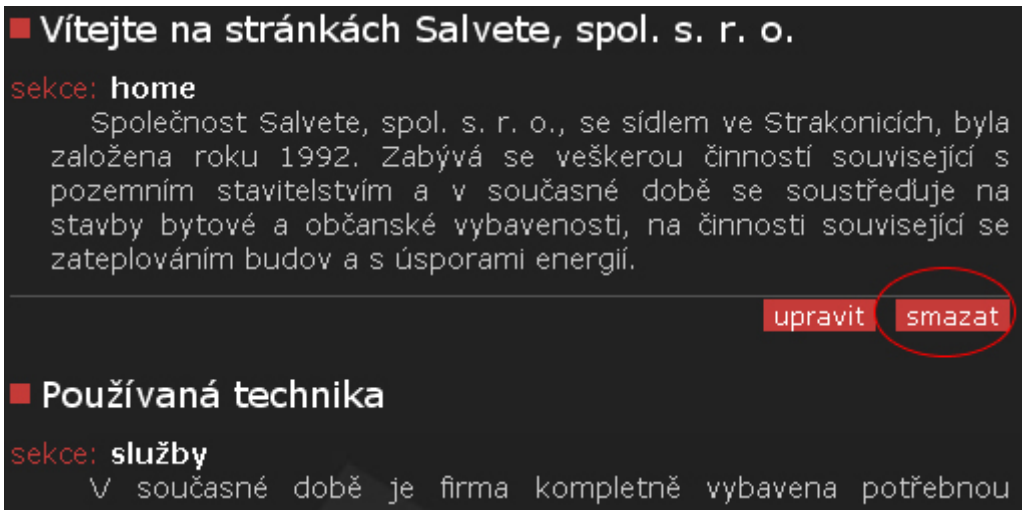
Obrázek 54: Editační formulář článků.

6.3.2 Změna údajů

Pro změnu údajů některého z článků slouží stejný formulář jako pro přidávání. Editaci článku aktivujeme tlačítkem *upravit*, které je umístěno vždy u konkrétního článku v základní části sekce, kde je zobrazen celkový výpis všech článků. Po jeho aktivaci naskočí již zmiňovaný formulář s vyplněnými údaji. Provedení změn se potvrdí tlačítkem uložit.

6.3.3 Smazání článku

Každý článek je možno smazat stisknutím tlačítka *smazat* umístěného v základní části vždy u konkrétního článku.



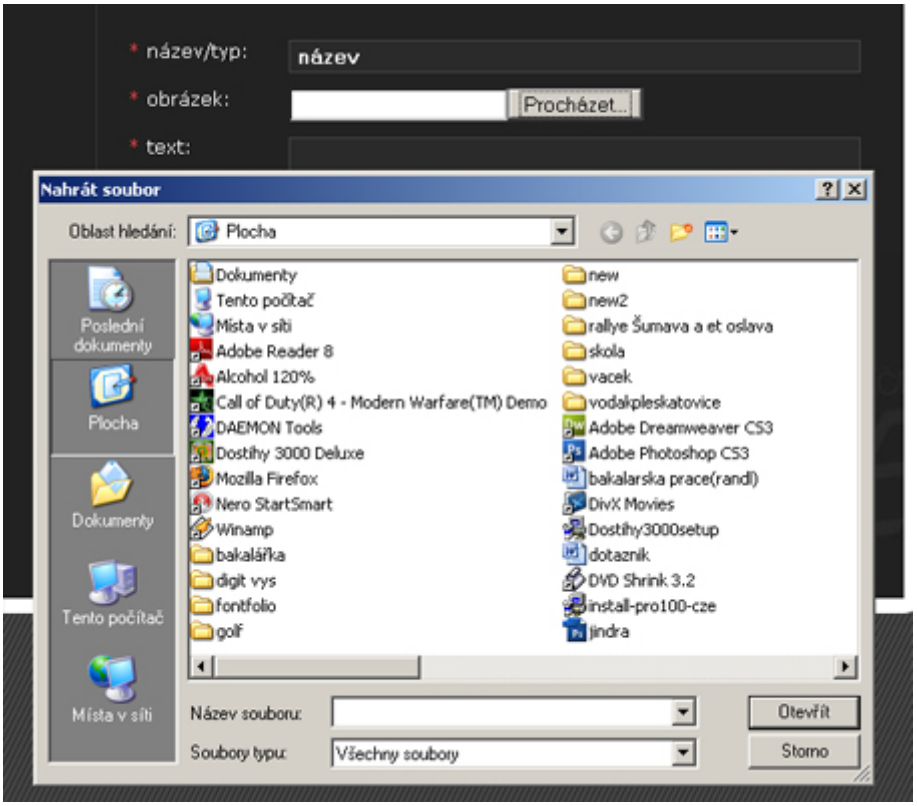
Obrázek 55: Tlačítko pro odstranění článku.

6.4 Editace služeb a certifikátů

Návod na obsluhu certifikátů a služeb jsem zahrnul do jedné kapitoly, neboť práce se službami a certifikáty se zcela shoduje.

6.4.1 Přidání záznamu

Pro přidání záznamu je nutné zvolit tlačítko *přidat službu/certifikát*. Po jeho stisknutí se zobrazí formulář, do kterého se vyplní jednotlivé záznamy. Formulář obsahuje také políčko s tlačítkem *procházet* pro přidávání obrázku k záznamu. Po jeho stisknutí se zobrazí okno, kde si najdete a zvolíte obrázek, který chcete k záznamu přidat. Označíte jej a stisknete tlačítko *otevřít*. Formulář obsahuje povinné prvky označené červenou hvězdičkou. Ty je nutné vyplnit. Po vyplnění formuláře se data a obrázek uloží tlačítkem *přidat*.



Obrázek 56: Okno pro načtení obrázku.

6.4.2 Změna údajů a změna obrázku

Pro editaci existujících záznamů je určeno tlačítko *upravit* v základní části sekce umístěné u každého záznamu zvlášť. Po jeho stisknutí se zobrazí formulář totožný s tím, jenž se zobrazil při přidávání záznamu jen s vyplněnými údaji, které lze měnit. Nevyplněné zůstane pouze políčko pro načítání obrázku. Pokud je nutné obrázek změnit, použijte opět tlačítko *procházet*. Při ukládání se nahradí původní obrázek novým. Pokud necháte políčko nevyplněné, zůstane u záznamu původní obrázek.

6.4.3 Smazání záznamu

Pro smazání záznamu existuje tlačítko *smazat*, které se nachází v základní části hned pod konkrétním záznamem vedle tlačítka *upravit*.

6.5 Editace referencí a jejich kategorií

6.5.1 Přidání kategorie

Pro přidání kategorie je určeno tlačítko *přidat kategorii* po jehož stisknutí se zobrazí formulář, ve kterém se vyplní název a popis kategorie. Pro jeho uložení je třeba stisknout tlačítko *přidat.*

6.5.2 Změna údajů kategorie

Pro úpravu záznamu kategorie je třeba stisknout na tlačítko *úprava kategorií* v hlavní části sekce. Po stisknutí se zobrazí kompletní výpis všech záznamů, u nichž je vždy tlačítko *smazat* a *upravit*. Po zvolení tlačítka *upravit* se zobrazí formulář, kde lze data přenastavit a uložit tlačítkem *uložit*.

Obrázek 57: Volba sekce.

6.5.3 Smazání kategorie

Pro smazání kategorie je nutné kliknout na tlačítko *smazat*, které je umístěno u každého záznamu v celkovém výpisu všech kategorií.

6.5.4 Přidání reference

Pro přidání reference je umístěno tlačítko v hlavní části této sekce. Po jeho zvolení se zobrazí formulář, kde je třeba vyplnit potřebné údaje. Pro jejich uložení je pod formulářem umístěno tlačítko přidat. Po jeho stisknutí se vytvoří nová galerie fotografií a spustí se režim editace galerie.

6.5.5 Změna údajů reference

Pro změnu údajů je nutné nejprve stisknout tlačítko *upravit text*. To je umístěno vždy v kategorii *reference* u konkrétního záznamu. Po jeho stisknutí se zobrazí formulář stejný jako při zadávání nové reference, jen s vyplněnými políčky. Pro uložení změn je třeba stisknout tlačítko *uložit.*

6.5.6 Smazání reference

Referenci lze smazat stisknutím tlačítka *smazat* umístěného v kategorii reference základní části této sekce.

6.5.7 Mazání a přidávání fotografií

Režim pro úpravu fotografií lze spustit dvěma způsoby. Prvním je automatické spuštění při zadávání referencí a druhý je stisknutí tlačítka *upravit galerii* nacházející se u výpisu referencí. V tomto režimu je umístěn jeden formulářový prvek pro načítání fotografií a tlačítko *přidat* pro potvrzení a nahrání fotografie na server.

Pod formulářem jsou umístěny zmenšeniny všech fotografií patřících ke zvolené referenci. U každé z nich je umístěn odkaz *smazat*, po jehož stisknutí se daná fotografie vymaže z databáze.

6.6 Editace kontaktů

6.6.1 Přidání kontaktu

Pro přidání kontaktu je třeba stisknout jedno ze dvou tlačítek v menu. Buďto tlačítko *přidat osobu* nebo *přidat místo*. Záleží na typu vkládaného kontaktu. Po jeho stisknutí se zobrazí odpovídající formulář ke zvolenému typu kontaktu. Po vyplnění je nutné uložit kontakt stisknutím tlačítka *přidat.*

6.6.2 Úprava kontaktu

Úpravu kontaktu provedete stisknutím tlačítka *upravit* umístěného u konkrétního kontaktu v základní části této sekce, kde je zobrazen kompletní výpis všech kontaktů firmy. Po stisknutí tlačítka se zobrazí odpovídající formulář s vyplněnými políčky. Všechny provedené změny je třeba uložit stisknutím tlačítka *uložit*.

6.6.3 Smazání kontaktu

Smazání kontaktu z databáze kontaktů provedete stisknutím tlačítka *smazat* umístěného u konkrétního kontaktu v základní části této sekce.

7 Závěr

Jako cíl jsem si stanovil vytvořit atraktivní a dynamické stránky firmy Salvete, spol. s. r. o. s administrátorským rozhraním, kde majitel nebo jím pověření zaměstnanci mohou upravovat kompletní obsah stránek.

Mohu říci, že se mi cíl podařilo splnit. Naprogramoval jsem dynamickou webovou prezentaci s administrátorským rozhraním, kde jsem skloubil několik nejmodernějších technologií a ověřil tak jejich funkčnost a kompatibilitu. I přes použití několika technologií najednou jsem dosáhl validního kódu podle standardu XHTML 1.0 Strict stanoveným konsorciem W3C.

Při vytváření jsem užil dvě různé metodiky pro tvorbu dynamických prezentací, které se následně pokusím zhodnotit. První metodika spočívala v individuálním přístupu ke každé stránce. Její programování je velice zdlouhavé a hodí se spíše pro statické stránky. Zcela nevyužívá výhod programování pomocí PHP skriptu a není tak flexibilní. Druhá metodika, vytvoření jedné kostry a načítání dat na základě parametru, je mnohem výhodnější a variabilnější. Více využívá praktických dovedností PHP a i načítání stránek je o něco rychlejší, neboť kostra se načte do paměti počítače a prohlížeč čerpá z ní. Většina času je potom věnována jen načítání dat z databáze a ne celé stránce, jako je tomu u statických prezentací. Další nesporná a velká výhoda spočívá v úpravě stránek, kdy není třeba měnit obsah shodný pro všechny sekce zvlášť, ale stačí jej změnit pouze ve vytvořené kostře dokumentu.

Jsem rád, že jsem mohl díky této práci zviditelnit firmu Salvete, spol. s. r. o. ve světě internetu a vstoupit tak do tajů programování dynamických webových prezentací pomocí jazyka PHP a databáze MySQL. To mi bylo a ještě dlouho bude velkým přínosem a doufám, že nabyté zkušenosti uplatním i ve své profesní kariéře.

Musím také konstatovat, že PHP a MySQL jsou díky své dostupnosti v porovnání s konkurenčními komerčními technologiemi ASP.NET a SQL od firmy Microsoft úctyhodným soupeřem a nedivím se, že většina programátorů zůstává věrná právě PHP a MySQL.

Použitá literatura

[1] GUTMANS, Andi, BAKKEN, Stig Seather, RETHANS, Derick. Mistrovství v PHP 5. CP Books, a.s., 2005. 655 s.

[2] KOFLER, Michael, BERND, Öggl. PHP5 a MySQL5 – Průvodce webového programátora. Computer Press, a.s., 2007. 607 s.

[3] KOSEK, Jiří. PHP – tvorba interaktivních internetových aplikací. GRADA Publishing, 1999. 492 s.

[4] ULLMAN, Larry. PHP a MySQL – Názorný průvodce tvorbou dynamických WWW stránek. Computer Press, a.s., 2004. 534 s.

Seznam příloh

[1] CD obsahující zdrojové kódy WWW prezentace a tento dokument.