VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

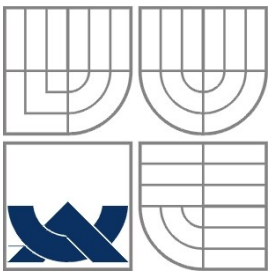
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KOMPLEXNÍ SYSTÉM PRO TVORBU A SPRÁVU WEBŮ

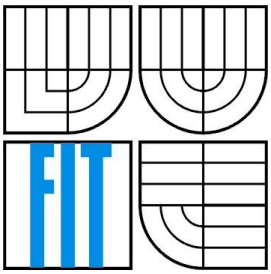
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE VIKTOR JABLONSKÝ
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KOMPLEXNÍ SYSTÉM PRO TVORBU A SPRÁVU WEBŮ

COMPLEX SYSTEM FOR WEB PAGE CREATING AND EDITING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — VIKTOR JABLONSKÝ
AUTHOR

VEDOUCÍ PRÁCE — ING. MICHAEL KUNC
SUPERVISOR

BRNO 2008

Abstrakt

Práce popisuje systém, který umožňuje registraci zákazníků, jejich správu a s tím spojené úkony. Zahrnuje též možnost evidovat platby za poskytnuté služby. Především však registrovaným uživatelům dovoluje jednoduše vytvářet vlastní internetové stránky, měnit jejich obsah a vzhled. Toho je docíleno pomocí šablon a modulů. Dokument analyzuje požadavky na takovou aplikaci, popisuje etapy návrhu řešení a implementace a diskutuje další pokračování tohoto projektu.

Klíčová slova

CSS, Databáze, HTML, JavaScript, Modul, MySQL, PHP, Průvodce, Smarty, Šablona, Web

Abstract

This thesis describes a system which offers customer registration and user administration-related tasks. It also facilitates an apparatus for supervising service fees. Above all, it provides users with tools with which they can create their own web site with ease and also change its content and appearance. This is achieved by using templates and modules. The Document also analyses specifications for this type of application, describes phases of design and implementation and draws out possible continuation of this project.

Keywords

CSS, Database, HTML, JavaScript, Module, MySQL, PHP, Smarty, Template, Web, Wizard

Citace

Viktor Jablonský: Komplexní systém pro tvorbu a správu webů, bakalářská práce, Brno, FIT VUT v Brně, 2008

Komplexní systém pro tvorbu a správu webů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michaela Kunce. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Viktor Jablonský
Datum

Poděkování

Chtěl bych poděkovat svému vedoucímu, panu Ing. Michaelu Kuncovi, za veškerou jeho pomoc, cenné připomínky a rady.

Obsah

1 Úvod

Tvorba webových stránek je dnes pro lidi předmětem potřeby, zábavy a dokonce i obživy. Ať už chceme dát světu vědět o našem dech beroucím experimentu se zdrojem vysokého napětí, o tom, že se naší fence Astě narodila tři štěňátka, či o jedinečné slevové akci na kuchyňské vybavení, jedním z nejlepších způsobů jak to provést, je vytvořit si internetové stránky.

K tomu jsou ale nutné jisté znalosti a dovednosti a tak se nezřídka stává, že si web necháváme vyrobit na zakázku od profesionálů. Pak může docházet k problémům ve vzájemné komunikaci při jeho vzniku. Chceme-li u hotové prezentace provést sebemenší změnu textu, většinou znovu voláme o pomoc, protože sami nevíme, jak na to. Takové „drobnosti“ pak obtěžují obě strany a zbytečně plýtvají časem.

Vznikl proto nápad, že by bylo šikovné nějak umožnit lidem, kteří se nechtějí nebo nedokáží naučit jazyk HTML, aby si vyrobili a spravovali svoje stránky sami. Přístupů k řešení je mnoho a všechny mají své výhody i omezení. Tento dokument pojednává právě o jednom takovém.

V druhé kapitole stručně nastíníme zadání a shrneme požadavky. Prozkoumáme, zda existují i jiná řešení, odhadneme, do jaké míry splňují zadání a jestli by pro nás nebylo výhodnější, abychom je převzali a upravili, namísto vytváření nového projektu. Uvedeme některé alespoň částečně vyhovující alternativy. Nastíníme strukturu vytvářeného systému, seznámíme čtenáře s konceptem modulů a šablon a nakonec popíšeme požadavky vyvíjeného projektu pomocí diagramu případů užití a konceptuálního modelu.

Třetí kapitola čtenáři přiblíží etapu implementace, od výběru vhodných technologií, přes detaily týkající se tvorby administrační i prezentační části aplikace, až po analýzu bezpečnostních rizik a potencionálních hrozeb pro systém. Na konci také zmíníme některé problémy, které při implementaci nastaly.

Ve čtvrté kapitole prozkoumáme současné silné a slabé stránky projektu, rozebereme jeho potencionální komerční i nekomerční využití a nastíníme perspektivu do budoucna.

V závěru zhodnotíme dosažené výsledky, zmíníme přínos a nové přístupy k řešené problematice, navrhneme směr, jakým by se měl systém nadále ubírat a nastíníme, co pozitivního tato práce autorovi přinesla.

2 Analýza problému a návrh řešení

V této kapitole shrneme požadavky na aplikaci a rozebereme teoretická východiska pro návrh řešení. Prozkoumáme stávající možnosti a zvážíme, zda je vůbec nutné podobný projekt realizovat. Rozdělíme problém na logické součásti a rozmyslíme nejlepší způsob řešení. Na konci se podíváme na případy užití a konceptuální model.

2.1 Požadavky

2.1.1 Souhrn

Aplikace má jako primární cíl urychlit tvorbu a správu webů, a to automatizováním procesu vytváření (případně změny) jejich zdrojového kódu. Musí umožnit registraci uživatelů (nejlépe pomocí webového formuláře) a jejich následnou správu. Každý uživatel si může zřídit jedny nebo více webových stránek. O uživatelích je vhodné uchovávat kromě přihlašovacího jména a hesla pro vstup do systémů také jméno (příp. název firmy), adresu či jiné kontaktní údaje a bankovní spojení, dále pak evidovat platby za poskytované služby. Každý registrovaný uživatel bude jednoznačně identifikován svým přihlašovacím jménem.

Systém může být spravován větším počtem správců, nesmí proto chybět možnost jejich administrace. O správcích se budou uchovávat kontaktní údaje, přihlašovací jméno a heslo. Každý správce bude jednoznačně identifikován svým přihlašovacím jménem.

K dispozici by měly být přehledy zákazníků (uživatelů), přehledy vytvořených stránek a také přehledy došlých plateb.

Je třeba důsledně oddělit vzhled stránek od prezentovaných dat. Preferované řešení je vytvoření systému šablon a modulů, kde šablony určují vzhled prezentace a moduly představují prezentovaná data z hlediska obsahu (např. obrázky v galerii, produkty v katalogu) a jejich logického členění (např. strom produktů v katalogu, jednotlivé kolekce fotografií v obrázkové galerii,...).

Moduly mimo jiné umožňují zobrazení dat, jejich zálohování a editaci. V základní sadě modulů by měly být například obrázková galerie, návštěvní kniha, článek (= souvislý text) apod.

Šablony lze vytvářet pomocí jednoduchého a přehledného průvodce, nebo ručně (tj. přímo programovat jejich kód) s možností importovat ji do systému. Šablony, které si uživatel vytvoří (bude jejich vlastníkem), může používat a spravovat pouze on sám. Na výběr by mělo být také několik výchozích šablon, které budou dostupné všem (nemají vlastníka). Výchozí šablony vytvářejí a ruší správci.

Aplikace musí poskytovat nástroje pro manuální i automatickou zálohu stránek (Tím se rozumí záloha dat v jednotlivých modulech a záloha aktuálně používané šablony.)

Systém eviduje měsíční platby zákazníků a v případě zmeškání termínu splatnosti automaticky zablokuje účet uživatele i jeho stránky. Možnost zablokovat účet by měl mít i správce systému a to kdykoliv. Odblokování stránek může být provedeno automaticky, při obdržení platby, či manuálně správcem.

Součástí je i diskuzní fórum registrovaných uživatelů, které bude sloužit především k informování zákazníků o nových službách a jako místo, kde mohou uživatelé klást dotazy nebo se podělit o své zkušenosti/nápady/návrhy.

2.1.2 Přehled v bodech

Následuje stručný seznam požadavků na aplikaci.

- Registrace a správa uživatelů
- Administrace správců
- Přehledy plateb, stránek, uživatelů
- Moduly pro uchování, správu a zobrazení prezentovaných dat a jejich administrace
- Šablony pro definování vzhledu internetových stránek a průvodce pro jejich vytváření
- Zálohování dat
- Blokování stránek
- Diskuze registrovaných uživatelů

2.2 Existující řešení problému

2.2.1 Zasazení do kontextu

V současné době je k dispozici mnoho tzv. opensource CMS, neboli otevřených redakčních systémů. Většinou bývají úzce zaměřené na jednu oblast webové prezentace, například e-shop, blog, diskusní fórum, wiki systémy atd. Jen na naší fakultě letos například vznikl informační systém pro tvorbu katalogů [1] či redakční systém pro stránky Nadace Partnerství [2].

Záměrem projektu vytvářeného v rámci této práce je však poskytnutí obecných možností, tzn. vytvoření systému, který v určitých případech může všechny výše uvedené zastoupit. Na rozdíl od nich má být také co nejjednodušší – tak, aby za pomoci průvodců (které rozdělují požadovanou činnost do několika logických kroků) dokázal stránky vytvořit i nezasvěcený uživatel, který se systémem nikdy předtím nepřišel do styku.

2.2.2 Možné alternativy

Z již hotových řešení by zadání této práce asi nejvíce vyhovoval Website Baker [3], který je volně šiřitelný pod Obecnou veřejnou licencí GNU (GNU GPL). Shoduje se v systému modulů a v zaměření na jednoduchou tvorbu webů (ačkoliv metodu průvodců nerozvíjí do takové hloubky).

Zásadním rozdílem je však způsob použití, neboť Komplexní systém pro tvorbu a správu webů je zamýšlený jako systém, do kterého se registrují uživatelé, jimž je nabízena správa a technická podpora. Záměrem celého projektu je komerční provoz jako doplňková služba pro zákazníky firmy, která působí v oblasti internetu, reklamy a telemarketingu. Tudíž aplikace zahrnuje správu registrovaných uživatelů, systém plateb za poskytované služby a také prostředky pro komunikaci se zákazníky.

Ve stávajících systémech také často chybí možnost „ruční práce“ na grafické podobě, tedy uspokojování potřeb zákazníků, kterým z nějakého důvodu nestačí základní, předem vytvořené šablony. Návrháři webů by proto měli dostat možnost výsledný vzhled ručně vylepšit či dokonce od základu vytvořit. Proto se jeví jako přijatelnější řešení vývoj zcela nového projektu, než úprava některého stávajícího.

2.3 Dekompozice, logika aplikace

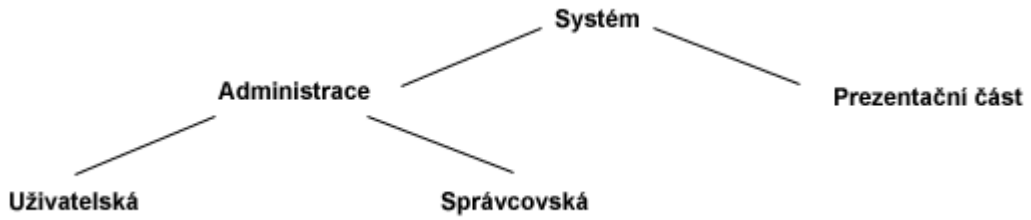
Nyní rozdělíme celý systém na logické součásti a popíšeme je. Navrhneme mechanismy, které usnadní splnění požadavků a do budoucna ponechají prostor pro různá vylepšení a rozšíření.

2.3.1 Struktura systému

Začneme-li co nejobecněji, můžeme systém rozdělit na část administrační a prezentační.

2.3.1.1 Administrační část

Máme zadán požadavek na správu uživatelů a možnost mít v systému více správců, administrace tedy musí mít dvě úrovně přístupů: na té vyšší je to správcovská administrace, ve které mohou zaměstnanci firmy dohlížet na provoz systému, aktivity uživatelů a další záležitosti s tím spojené. V administraci určené pro uživatele jsou pak základními úkoly práce s vlastním profilem a vytváření/změna internetových stránek. Situaci znázorňuje obrázek č. 1.



Obrázek 1. - Základní rozdělení systému

2.3.1.2 Prezentační část

Prezentační část je víceméně automatizovaný prvek, protože výsledný vzhled internetových stránek určují dvě specifické komponenty – grafická šablona a modul (resp. moduly). Každé z nich je věnována samostatná část této podkapitoly.

V tomto textu se vyhneme bližšímu popisu principu fungování modulů. Neuvedeme ani diagramy případů užití, ani entitně-vztahové diagramy, neboť moduly jsou proměnlivou součástí systému a s jeho návrhem úzce nesouvisí (nic nám nebrání je ze systému odebrat nebo naopak přidat).

2.3.2 Šablony

Nyní se podíváme na záležitosti spojené s vizuálním stylem webových stránek.

2.3.2.1 Problém ideální šablony

Vzhledem ke snaze docílit co největší uživatelské přívětivosti aplikace vznikl požadavek, aby šablona stránek definovala vzhled prezentace v co největším rozsahu, ideálně aby ji určovala zcela. Protože se však předpokládá, že během užívání sytému (jak se objevují nové nároky a přání zákazníků) budou postupně přibývat další různorodé moduly, musí být schopnost šablony při určování vzhledu stránek co nejvíce nezávislá na komponentách, ze kterých se stránky skládají. Ideální by bylo, kdyby šablona dokázala přesně naformátovat jakýkoliv obecný obsah.

Tohoto stavu však není možno rozumným způsobem docílit, neboť modul může nést prakticky jakákoliv data, od prostého, neformátovaného textu, až po složité formuláře a tabulky. Přidáme-li do systému nový modul, bylo by přinejmenším nešikovné, abychom museli veškeré existující šablony aktualizovat o nějaká pravidla určující jejich chování při jeho zobrazení. Ale jak bychom bez takových metadat mohli docílit například toho, že šablona rozpozná modul obsahující strukturovaný životopis a zobrazí jednotlivá hesla přehledně pod sebou, a že zároveň v jiném případě korektně identifikuje zobrazované položky jako seznam vstřelených branek v hokejovém zápase a bude vědět, že je vhodné je vypsat za sebou, oddělené čárkami? Představme si, o jak komplikovaný princip by šlo, pokud bychom ještě navíc chtěli, aby šablona automaticky rozeznala nějaký zdrojový

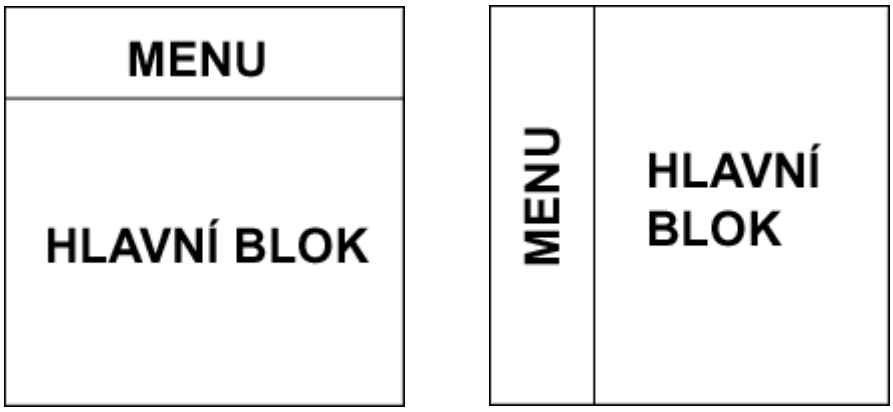
kód (např. v jazyce C) a zvýraznila syntaxi. A podobné nároky bychom mohli klást takřka do nekonečna.

Univerzální šablona takové komplexity je jistě zajímavá problematika, kterou lze rozebírat do velkých podrobností, což ovšem přesahuje rámec této práce a vyžadovalo by to zároveň hlubší studium. Museli bychom uvážit nejen psychologické a typografické aspekty, ale také různé normy a zvyklosti, brát ohled na druh zobrazovaných dat atd. Samozřejmě existuje možnost tuto funkcionalitu jistým způsobem rozumně omezit – například tím, že předdefinujeme určité normy pro moduly, vytvoříme soubor nekompletních pravidel, která pokryjí nejběžnější případy, využijeme principů fuzzy logiky atp. Takové řešení je ovšem kompromis, který nemusí fungovat vždy a jeho realizace také není zcela bez komplikací.

2.3.2.2 Zjednodušené pojetí šablon

My se proto v rámci tohoto projektu spokojíme s tím, že šablona bude určovat pouze základní rozložení webové stránky, barvy, typy a velikosti písem, barvy a vzory (tzn. obrázky) pozadí a případně také vzhled některých konkrétních HTML elementů (rámečky tabulek, zvýraznění řádku tabulky pod kurzorem, formulářových políček, tlačítek,...). O ostatní (tedy zejména o správnou interpretaci dat a jejich relativní pozici v dokumentu) se postará modul sám, resp. jeho prezentační vrstva (podrobněji viz. část věnovanou modulům).

Co se týká rozložení stránky, šablona rozhoduje pouze o tom, které moduly se najednou zobrazí, jaké to budou a kam se umístí. Vždy se musí jednat minimálně o dva: modul obsahující menu a modul, který se zobrazí vybráním nějaké volby z menu. Zjednodušeně řečeno, hlavní menu je v podstatě seznam všech modulů, které stránky obsahují. Po kliknutí na nějaký odkaz menu se jeho cíl (tj. odkazovaný modul) objeví v tzv. hlavním bloku (toto není obecný termín a v rámci této práce je použit jako pojem označující určitý prostor na webové stránce). Dva příklady, jak může takové rozložení vypadat, jsou na následujících náčrtcích (viz. obrázek č. 2).



Obrázek 2. - Dva případy prostorového rozložení modulů na stránce, jak může být definováno šablonou

2.3.2.3 Proč hlavička s logem do šablony nepatří

Při úvahách o těchto záležitostech počáteční předpoklad zněl, že každý uživatel chce mít na svých stránkách nějakou hlavičku, do které například umístí své logo, název své firmy atp. Nesmí také postrádat hlavní menu (pro navigaci na stránkách) a zápatí, případně i panel s novinkami nebo upoutávkami na různé články. Všechny jmenované součásti by měly nějak zapadat do vizuálního motivu webu a proto je logické, že budou součástí šablony. Tím se ovšem omezuje její univerzálnost, neboť ji svazujeme ke konkrétnímu uživateli a dokonce ke konkrétním stránkám. Naproti tomu skutečnost že tyto části „povýšíme“ na moduly nám poskytne potřebnou flexibilitu. Můžeme si tak vyrobit deset různých podob hlaviček a to, kterou použijeme v naší oblíbené šabloně, je úplně jedno.

Uveďme jiný příklad: uživatel má dva weby, jeden pro svojí firmu a druhý pro její dceřinou společnost. Na prvních představuje koberce všemožných vzorů, velikostí a barev, informuje o cenách, způsobu dopravy, adrese kamenného obchodu atp. Na druhých stránkách nabízí služby spojené s čištěním koberců, pokládáním koberců atd. Aby mohl ještě lépe těžit z výhod společné značky a loga, rád by měl obě prezentace provedené ve stejném grafickém stylu (např. písma a pozadí ve firemních barvách).

Kdybychom v navrhovaném systému zavedli „zosobněnou“ šablonu (tzn. variantu, kde hlavička, zápatí atd. jsou součástí šablony), narazíme na nutnost upravit např. kontaktní údaje v zápatí, reklamní slogan v hlavičce. Naproti tomu námi zvolený systém má tu výhodu, že šablonu snadno aplikujeme na druhé stránky, aniž bychom ji museli pracně upravovat – hlavička a zápatí jsou totiž pojaty jako samostatné moduly a změna šablony se jejich obsahu nedotýká. Až si za rok majitel vzpomene, že chce nové firemní barvy, vytvoří prostě novou šablonu a opět ji na obě stránky aplikuje – na obsah hlavičky, zápatí apod. nebude muset ani sáhnout.

2.3.3 Moduly

Velkou pozornost si zaslouží i základní stavební kameny internetových stránek v systému. Výslednou podobu prezentace totiž z celkového hlediska (tzn. nejen z vizuálního, ale také z funkčního a obsahového) definují nejvýrazněji. Fungují jako univerzální stavebnice, ze které lze postavit téměř cokoliv.

2.3.3.1 Vymezení pojmu modul

Než se začneme podrobněji zabývat moduly, pokusme se stručně shrnout, jaký je vlastně jejich účel. Internetové stránky většinou obsahují nějaká užitečná data, často velice různorodá. V běžné praxi (při ručním vytváření stránek) je nějak logicky členíme a seskupujeme (Např. stránky fotoateliéru mají sekci ve které se chlubí svými nejpodařenějšími fotografiemi a jinou sekci, kde uvádí svojí adresu, telefon atd.) Podobně chceme k problému přistupovat i v našem případě, kdy jsou stránky vytvářeny uživatelem za pomoci jednoduchých a přehledných průvodců a HTML kód se generuje víceméně automaticky.

Právě k tomu využijeme moduly. Pod tím pojmem si můžeme představit jakýsi objekt, který uchovává, spravuje a zobrazuje uživatelova data, ať už se jedná o galerii obrázků, novinky ze světa módy či rozvrh hodin. Klíčové funkce, které zajišťuje, jsou:

- vkládání a editace dat
- zálohování a obnovení dat ze zálohy
- zobrazení vložených dat

2.3.3.2 Co může modul představovat

Modulem může být prakticky cokoliv, od těch nejtriviálnějších věci, jako je prostý odstavec neformátovaného textu, až po ty nejkomplikovanější, např. celý informační systém městské knihovny. Protože vytvářený systém má umožnit tvorbu a správu víceméně jakéhokoliv obsahu, není charakter modulu nijak omezen. To do budoucna ponechává otevřené dveře pro široké spektrum úprav a vylepšení – bude možné rozšířit funkcionalitu prakticky v každém ohledu.

Na druhou stranu je nutné uvědomovat si rizika z toho plynoucí! Uveďme na vysvětlení raději příklad: teoreticky by bylo možné celý systém, který popisuje tato zpráva, začlenit jako jeden z modulů. Vznikne z toho však nějaký užitek? Zorientuje se vůbec uživatel v záplavě možností, jež mu vložíme do rukou? Neodradí ho naopak komplexnost takového modulu? Dokážeme vůbec tak složitý mechanismus poskládat?

Z výše prezentovaných úvah lze učinit jasný závěr: modul musí být funkční, ale především co nejjednodušší a snadný na správu. Jeho účel by měl být intuitivně rozpoznatelný a neměl by zavádět k chybnému využití. Nebudeme stavět spletitý modul, raději se pokusíme rozdělit jej do více

nezávislých částí. Při tvorbě nových modulů je též na místě úvaha, zda má vůbec takové počínání smysl. Myšlenka komplexního, automatizovaného systému vytváření internetových stránek není samo spásná – někdy narazíme na problém, u kterého bude vhodnější ruční vytvoření stránek tak říkajíc „na míru“.

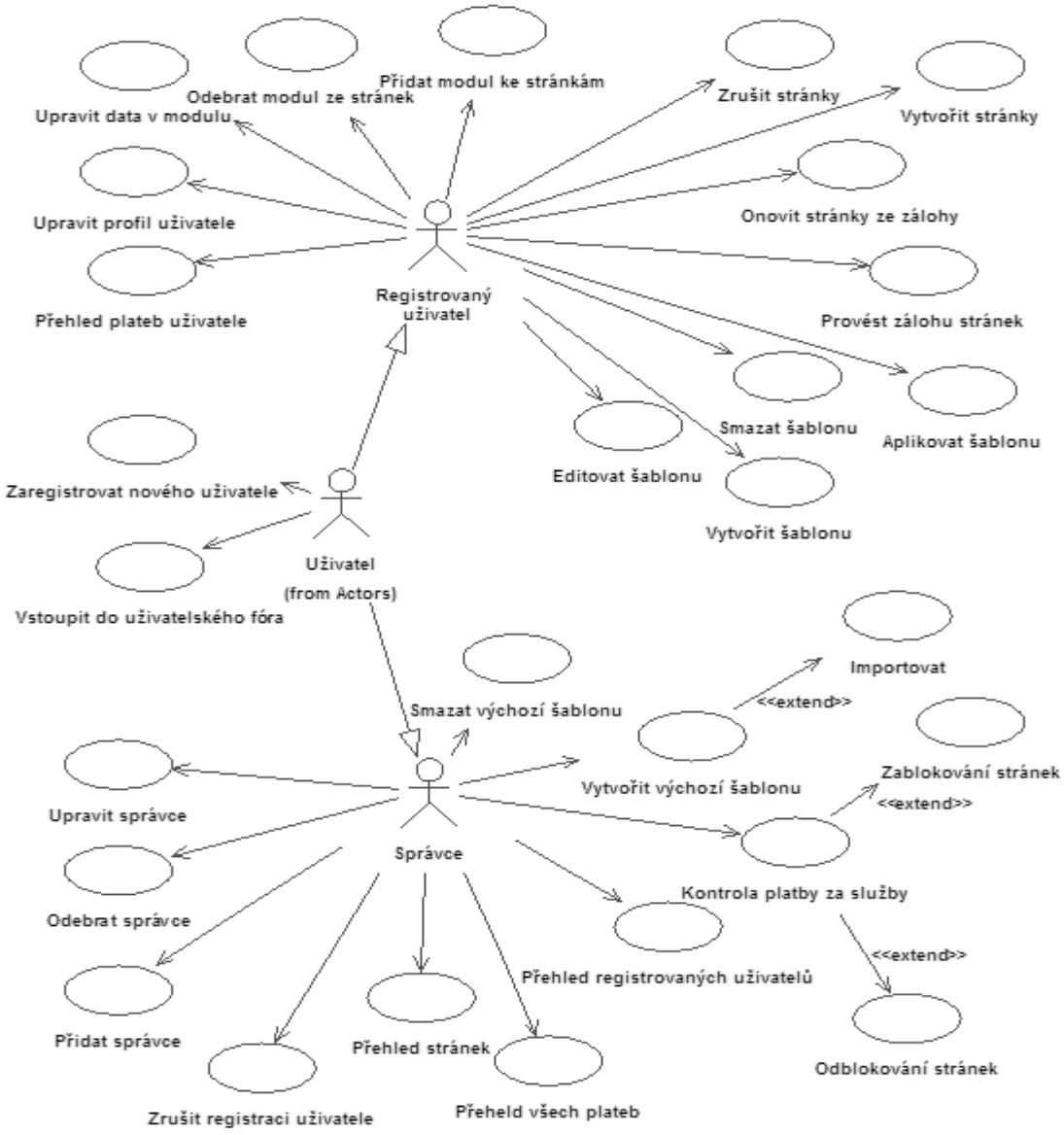
2.3.3.3 Části modulu

Abychom dodrželi požadavek na důsledné oddělení vzhledu stránek od prezentovaných dat, musíme zcela separovat úkony týkající se práce s daty a úkony související s jejich zobrazením. Jak už jsme rozebrali v části věnované problematice šablon, je nutné převést jistou zodpovědnost za vzhled webu na moduly samotné. Šablona se stane jakýmsi vodítkem k pospojování těchto „stavebních kamenů“, budeme z ní vždy vycházet. Konečné úpravy prezentace a zobrazení obsahu jako takové je ovšem záležitostí modulu, přesněji jeho prezentační vrstvy. Dokonce mu umožníme i „přebít“ nastavení šablony co do barvy a velikosti textu, pozadí atp. Nesmí se to ovšem stát běžnou praxí a k podobným krokům bychom se při definování zobrazovací funkce modulu měli obracet jen v krajních případech (nebude-li jiného zbytí).

Chceme-li do systému snadno přidávat nové moduly i v budoucnosti (resp. během jeho provozu), je vhodné spolu s nimi poskytovat i nástroje pro jejich přidání/odebrání do/ze stránek a nástroje pro správu dat v nich (tzn. vložení, úpravu, mazání a zálohování/obnovu dat modulu). Je to výhodnější, než se spolehnout na nějaký univerzální nástroj společný pro všechny. Tuto funkcionalitu lze považovat za datovou, resp. aplikační vrstvu.

2.4 Diagram případů užití

Jaké akce lze v systému vykonávat a kdo je bude provádět, nejlépe demonstrujeme na diagramu případů užití (tzv. Use Case diagramu), který si můžete prohlédnout na obrázku č. 3. Kvůli přehlednosti v něm pomineme roli času (jako spouštěče naplánovaných úloh) a roli návštěvníka stránek. V následujícím textu vyobrazené případy užití popíšeme a stručně vysvětlíme.



Obrázek 3. - Use Case diagram pro administrační část sytému

2.4.1.1 Obecná role uživatele

Obecný uživatel systému má možnost pouze vstoupit do uživatelského fóra (jež slouží jako jeden z prostředků interakce mezi zákazníky a poskytovatelem služby) a nebo se zaregistrovat. Registrací se z něj stává speciální případ uživatele – registrovaný uživatel.

Správce se sám zaregistrovat nemůže (takový úkon smí provést pouze některý již existující správce) a proto zde tento případ není uveden.

2.4.1.2 Role správce

Předpokládá se, že firma provozující systém pověří jeho vedením několik správců. Proto tvoří jednu z náplní jejich činnosti administrace správců. Správce může založit účet nového správce, či zrušit účet některého stávajícího. Měnit údaje může každý administrátor pouze u svého profilu. Nepředpokládá se, že by svých pravomocí správci nějak zneužívali, neboť se jedná o zaměstnance firmy, kteří zodpovídají za bezproblémový chod systému. Nicméně v případě potřeby lze v budoucnosti snadno doplnit přísnější bezpečnostní mechanismy.

Další úkoly správců se týkají uživatelů systému (zákazníků) – to je především kontrola plateb, blokování stránek a zrušení registrace uživatele. Administrátor může kdykoliv zkontrolovat, zda zákazník platí za poskytované služby. Případným protiopatřením je možnost zablokovat uživateli stránky, které pak nepůjdou zobrazit. Samozřejmostí je pak také možnost stránky opět zprovoznit.

Nesmíme zapomenout ani na možnost zobrazit nejdůležitější přehledy (plateb, stránek, uživatelů).

Poslední, ale neméně důležitou oblastí jsou také nástroje, s jejichž pomocí mohou správci pomáhat uživatelům s definováním vzhledu jejich stránek – tzn. vytvářet a přidávat šablony. Tyto se pak (na rozdíl od těch vytvořených uživatelem) stávají výchozími a může je na své stránky aplikovat kterýkoliv uživatel. Měla by existovat i možnost vložit do systému šablonu ručně vytvořenou.

2.4.1.3 Role registrovaného uživatele

Základními úkony, které bude chtít jistě zákazník čas od času provádět, je změna některých osobních údajů. Dále ho také bude zajímat, zda jsou v systému správně evidovány jeho platby, musí proto dostat možnost je prohlížet.

Co však bude zřejmě středem pozornosti uživatele, je vytváření webových stránek, tato funkce by měla být realizována formou přehledného průvodce. S tím souvisí i případ, kdy chceme stránky zrušit.

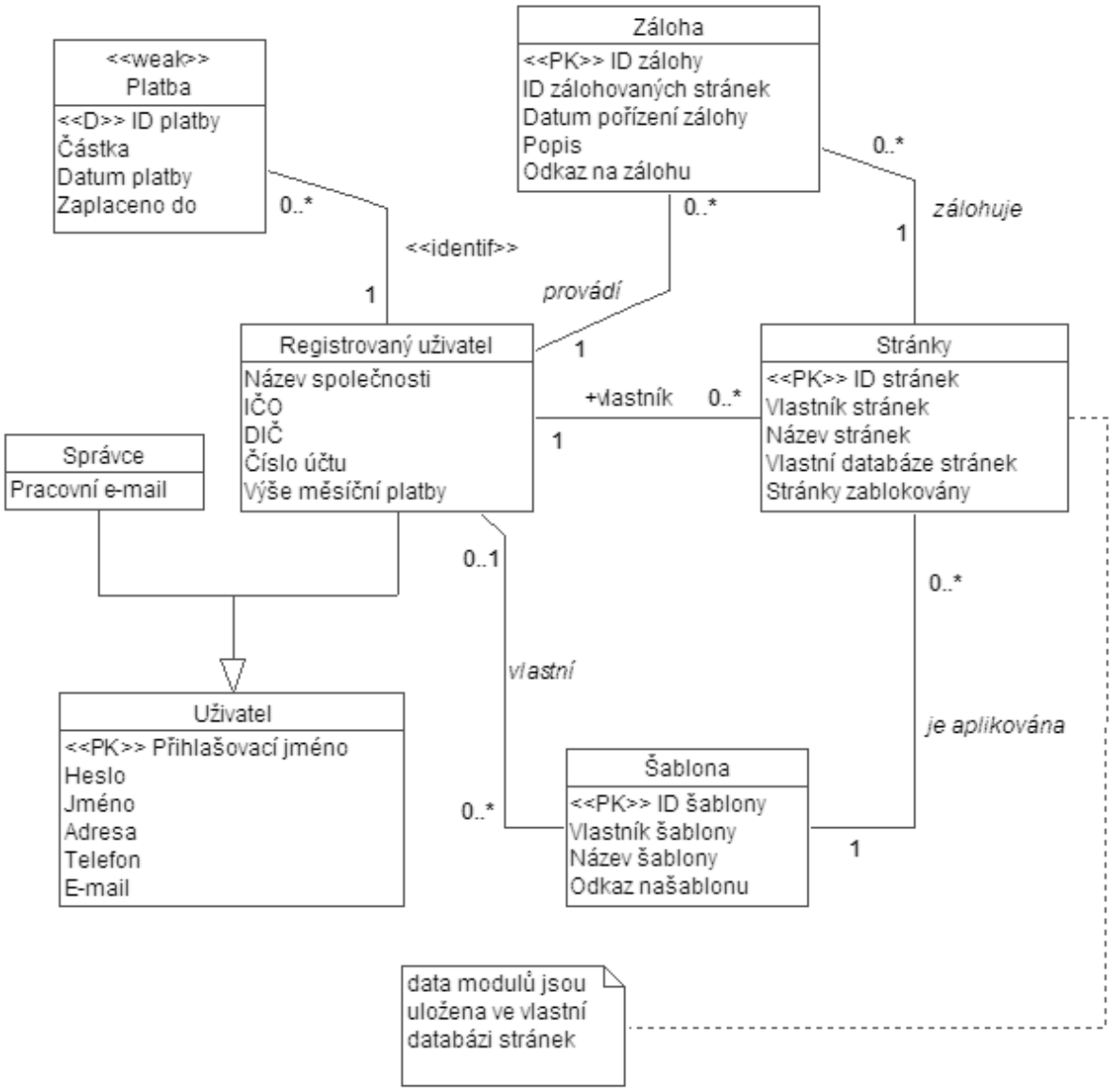
K vytvořeným stránkám bude třeba přidávat nové moduly, odebírat je či upravovat data v nich. To samé platí o vytváření, mazání a úpravě šablon, nelze opomenout také aplikování šablony (tj. případ, kdy určíme šablonu používanou při zobrazení webu).

Kromě automatického systému zálohování je zde i možnost provádět tyto úkony manuálně.

2.5 Konceptuální model

Popišme nyní vztahy mezi jednotlivými entitami formou ER diagramu, který je na obrázku číslo 4. Předpokládáme, že čtenář je s problematikou ER diagramů alespoň minimálně seznámen a proto snad

není třeba grafické znázornění příliš podrobně komentovat. V následujícím textu pouze stručně osvětlíme, k čemu jsou jednotlivé entitní množiny dobré a co představují.



Obrázek 4. - Znázornění vztahů mezi entitami pomocí ER diagramu

2.5.1.1 Stručná charakteristika entitních množin

K uchování informací o hlavních aktérech systému slouží množiny Registrovaný uživatel a Správce. V požadavcích na aplikaci bylo uvedeno, abychom o nich evidovali kontaktní a jiné údaje. Oba druhy uživatelů jsou v systému jednoznačně identifikováni svým uživatelským jménem, které musí být unikátní i z toho důvodu, že slouží k přihlašování do administrace.

Na množině Registrovaný uživatel je závislá slabá entitní množina Platba. Ta uchovává informaci o platbě za služby v daném období.

Pro přehledné uchovávání informací o záloze stránek vytvoříme množinu záloha, jež ponese kromě jednoznačného identifikátoru také textový popis zálohy (pro možnost přidat k ní vysvětlující poznámku) a také datum, kdy došlo k jejímu vytvoření. Odkazem na zálohu se rozumí místo, kde jsou data fyzicky uložena.

V hlavní databázi systému je také nutné udržovat přehled o existujících stránkách jednotlivých uživatelů. K tomu poslouží množina Stránky. Jednotlivé záznamy identifikuje tzv. ID stránek, které musí být samozřejmě unikátní. Jelikož pro každé stránky vzniká vlastní databáze, je velice důležité, abychom zde ukládali také informace o tom, jak se k ní připojit. Významnou položkou je i příznak o tom, že jsou stránky zablokovány (např. pokud uživatel nezaplatí za služby).

Ještě jsme se také nezmínili o množině Šablona. Ta nese údaje o tom, kdo je jejím vlastníkem, jak se šablona jmenuje a kde se nachází. Každou položku jednoznačně určuje tzv. ID šablony.

2.5.1.2 Poznámky ke konceptuálnímu modelu

Za pozornost stojí entitní množiny Registrovaný uživatel a Správce, které vznikly specializací z množiny Uživatel. Přestože Use Case a ER diagramy spolu nijak nesouvisí a každý slouží k modelování zcela jiného problému, je zajímavé uvědomit si jistou korespondenci se situací, kdy nám v diagramu případů užití z „uživatele“ jeho registrací vznikl „registrovaný uživatel“ resp. „správce“. Vidíme, že „příbuznost“ obou aktérů vychází i ze společných atributů (díváme-li se na ně z hlediska dat).

Povšimněme si také skutečnosti, že entitní množiny Registrovaný uživatel a Stránky figurují ve většině vztahů. To jenom potvrzuje jejich důležitost a napovídá, na jakou část systému je kladen důraz především a co je hlavním cílem projektu.

2.5.1.3 Data modulů

Konečně se dostáváme k problematice uložení dat jednotlivých modulů (tedy obsahu, který se prezentuje na uživatelově webu). Ta jsou uložena ve vlastních databázích jednotlivých stránek. Pomůže to udržet přehled v hlavní (chcete-li – systémové) databázi a výrazně se tím sníží počet tabulek v ní.

3 Implementace

V této kapitole zvolíme a popíšeme technologie pro implementaci systému. Rozebereme některé významnější detaily, zastavíme se u otázky bezpečnosti a nakonec budou zmíněny některé problémy, jež se během vývoje objevily a zkomplikovaly práci.

3.1 Volba technologií

Při výběru technologií záleží především na výkonu, poskytovaných funkcích, jejich dostupnosti a také na kvalitě dokumentace. V první řadě musíme být schopni rozumně realizovat požadavky – čím širší možnosti máme k dispozici, tím je naše práce snazší.

Abychom zajistili, že aplikace hladce poběží na většině hostingů, bez nutnosti dožadovat se na poskytovateli instalace dodatečných služeb, měli bychom volit implementační technologie z některých běžně používaných. S tím souvisí také fakt, že bude snazší k nim sehnat odbornou literaturu, praktické návody, volně dostupné programové knihovny a také cenné rady od rozsáhlé vývojářské komunity.

V následujícím textu tedy stručně popíšeme technologie, které jsme zvolili a tuto volbu zdůvodníme.

3.1.1 XHTML

Extensible Hypertext Markup Language (Rozšiřitelný hypertextový značkovací jazyk) je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW. Vyvinulo ho společenství W3C (The World Wide Web Consorcium – viz.[6]). XHTML verze 1.0 je přepracování HTML 4.01 do XML a kombinuje sílu HTML 4 s výhodami XML. Tím se ulehčuje tvorba a údržba kódu, zvyšuje se dostupnost na různých platformách a pro různá zařízení (od mobilních telefonů přes televizory až po automobily) [6].

XHTML 1.0 existuje ve třech verzích – strict, transitional a frameset. Jeho předností je důslednější oddělení formátování textu od zbytku kódu pomocí kaskádových stylů (CSS). To zpřehledňuje výsledný dokument, usnadňuje případné změny a vylepšení a v kombinaci se šablonovacím systémem Smarty (viz. dále) poskytuje vynikající možnosti pro vývoj složitějších aplikací.

3.1.2 PHP

PHP, nebo přesněji „PHP: Hypertext Preprocessor," je široce používaný, opensource skriptovací jazyk, který je zvláště vhodný pro tvorbu dynamického webu. Může být začleněn přímo

do (X)HTML. Jeho syntaxe je podobná dobře známým jazykům jako je C, Java či Perl, a dá se velmi snadno naučit. Hlavním cílem PHP je umožnit webovým vývojářům psát dynamicky generované internetové stránky, a to velice rychle a efektivně. [4]

K jeho dalším přednostem patří podpora široké řady souvisejících technologií, formátů a standardů; skvělá kompatibilita s webovým serverem Apache; snadná komunikace s databázemi jako je MySQL, PostgreSQL aj.; je multiplatformní a lze jej provozovat s většinou webových serverů a na většině dnešních operačních systémů; je podporován mnoha poskytovateli webhostingových služeb.

PHP ve verzi 4 už také podporuje objektový přístup k programování, přestože má ještě určitá úskalí (velké množství z nich se podařilo odstranit ve verzi 5). Vzhledem k tomu, že firma, pro kterou je systém vyvíjen, má k dispozici server s PHP ve verzi 4, zvolíme ji pro implementaci projektu.

Přestože některé další technologie závislé na PHP (zejména Smarty – viz. dále) používají objektově orientovaný přístup, my se tomu vyhneme. Jedním z důvodů je fakt, že ve verzi PHP 4 není jeho podpora stoprocentní (například absence privátních metod, špatné hlídání přístupu k nedeklarovaným atributům v rámci třídy,...).

3.1.3 Kaskádové styly (CSS)

Zkratka znamená „tabulky kaskádových stylů“ (Cascading Style Sheets). Je to jazyk pro popis způsobu zobrazení webových stránek napsaných v jazyce HTML či XHTML, avšak lze jej aplikovat prakticky na každý druh XML dokumentu. Byl navržen organizací W3C, stejně jako XHTML [7].

Je primárně určen jako doplněk XHTML pro tvorbu internetových stránek, aby definoval barvy, písma a rozložení dokumentu. CSS specifikuje schéma priorit pro aplikování pravidel, pro případ, že na daný element se vztahuje více než jedno. V této tzv. kaskádě jsou vypočítány priority (jinak řečeno váhy) a přiřazeny jednotlivým pravidlům.

V tomto projektu upotřebíme kaskádové styly hlavně jako doplněk systému šablon Smarty (viz. dále) a také pro určování vzhledu administrační části aplikace.

3.1.4 Smarty

Smarty je šablonovací nástroj pro PHP. Usnadňuje oddělení aplikační logiky a obsahu od výsledné prezentace. To je výhodné například v situaci, kdy programátor aplikace a návrhář šablony jsou dvě různé osoby. [5]

Jedním z unikátních aspektů Smarty je kompilování šablon. To znamená, že Smarty načte šablonu a vytvoří z ní PHP kód. Napříště jsou při dotazu na příslušnou stránku zavolány tyto přeložené skripty, což má obrovský vliv na rychlost zpracování dotazu.

K dalším výhodám patří: použití PHP parseru pro kompilaci; rekompilují se pouze šablony, které byly změněny; snadné vytvoření vlastních funkcí a modifikátorů,...

Při vývoji našeho systému použijeme Smarty k vytvoření systému šablon a modulů, tedy zejména v prezentační části aplikace. Za pomoci PHP skriptů a Smarty šablon oddělíme aplikační logiku a prezentační logiku, výhodou je také možnost jednoduše vytvořit šablonu ručně, což se bude při provozu systému hodit zejména k uspokojování náročnějších zákazníků.

V aplikaci je Smarty použito verzi 2.6.14. Ta vyžaduje ke správnému provozu server s podporou PHP 4.0.6 nebo vyšší, což je v dnešní době splněno skoro na každém webhostingu.

3.1.5 Java Script

Java Script je multiplatformní, objektově orientovaný skriptovací jazyk. V současné době je to jedno z nejrozšířenějších řešení pro tvorbu dynamického webu a obvykle se používá pro ovládání různých interaktivních prvků, jako jsou textová políčka či tlačítka. Syntaxí je velmi podobný jazykům C, C++, Java; slovo Java je však součástí jeho názvu pouze z marketingových důvodů. [7]

Program v Java Scriptu se obvykle spouští až po načtení WWW stránky z internetu do uživatelova prohlížeče, čili na straně klienta. Je tak vhodným doplňkem jazyka PHP, který naopak běží na straně serveru.

V naší aplikaci jej využijeme především pro kontrolu dat v různých formulářích a také k ovládání některých interaktivních prvků na stránkách. Vzhledem k tomu, že pro kódování webu používáme XHTML, bude elegantnější vkládat jej z externích souborů (tudíž odpadne nutnost vkládat jej do tagu CDATA abychom dodrželi předepsaný standard). Tato praxe je výhodná také z toho důvodu, že některé formuláře v systému jsou si velice podobné a pro kontrolu správnosti vyplněných údajů můžeme snadno využít jednu a tu samou funkci.

S vkládáním externích skriptů do XHTML stránek se při vytváření systému objevil nepříjemný nedostatek prohlížeče Internet Explorer (verze 7). Podrobněji se o tom lze dočíst v podkapitole „Problémy při implementaci.“

3.2 Implementace administrační části

V následujícím textu se budeme zabývat problematikou implementace administrační části (tj. těch prvků systému, které poskytují zázemí pro plnění hlavní funkce systému – tvorbu a správu webů). Přestože správa jednotlivých modulů stránek (resp. dat v nich) se provádí z administrační části, nebudeme ji zde rozebírat. Pro přehlednost je tato problematika ponechána u popisu modulů, který však byl vzhledem k jejich účelu zařazen do podkapitoly věnující se implementaci prezentační části.

3.2.1 Vytvoření databáze

K vytvoření databáze pro administrační část byl vytvořen soubor obsahující MySQL skript. Je zformulován na základě převodu ER diagramu (popsaného v předchozí kapitole) na tabulky databáze. Upotřebíme jej hlavně při instalaci systému na webhosting, ale po jednoduché úpravě může být použit i k jeho přeinstalování, neboť dokáže všechny tabulky zrušit a vytvořit znovu.

Nicméně toto není standardní chování a změny je nutné provést odkomentováním příslušných řádků ve zdrojovém textu (jsou viditelně označeny a doplněny vysvětlením). Ve výchozím nastavení bude skript již vytvořené tabulky ignorovat a jejich obsah nijak nezmění.

3.2.2 Vstupní bod do administrace

Místem, kde mohou registrovaný uživatel i správce vstoupit do administrace, je hlavní webová stránka systému. Ta by mimo jiné měla poskytovat také odkaz na formulář pro registraci nového uživatele. V okamžiku, kdy bude systém zprovozněn, může zároveň posloužit jako základní zdroj informací pro potencionální zákazníky.

V implementační verzi, která je součástí této práce, je pro systém použit pracovní název Modulární webový manažer (MWM). Kromě uvedených odkazů také stránka neobsahuje žádné další informace. Důvod bude uveden později v této práci (při zhodnocení výsledků a perspektivě projektu).

3.2.3 Přihlašování do systému

K přihlášení uživatele (tj. registrovaného uživatele či správce) a uchování informace o tom poslouží proměnná „session“ v PHP. Obsluha této činnosti je realizována prostřednictvím jednoho skriptu, který umí rozlišit, jaký typ uživatele se přihlašuje.

Po zadání uživatelského jména a hesla do příslušných políček formuláře se data odešlou na server, kde se zkontrolují proti záznamům v databázi. Pokud vyhoví, uživatel je přihlášen, informace o tom je uložena do „session“ a spolu s ní také čas přihlášení. Časový údaj se pak aktualizuje s každým požadavkem, který uživatel odešle na server. Je-li prodleva v činnosti příliš dlouhá, sezení se prohlásí za neplatné a uživatel bude přesměrován na přihlašovací stránku, kde musí znovu zadat své jméno a heslo, aby mohl pokračovat v práci.

Délka časového intervalu je definována v konstantě SESSION_MAX_IDLETIME, která se nachází v souboru *funkce.php* a je zadávána ve vteřinách.

3.2.4 Správcovská administrace

Správce je po přihlášení přesměrován na hlavní nabídku správcovské administrace. Zde má k dispozici přehled úkonů, které může vykonávat.

Všechny možnosti a volby zde uvedené vyplývají z diagramu případů užití (a popsali jsme je při jeho rozboru), tudíž je zde nebudeme znovu opakovat. Podotkněme jen, že po vykonání nějaké úlohy (nebo při zrušení jejího provádění) je administrátor vrácen zpět na tuto nabídku, odkud může dále pokračovat v práci.

3.2.5 Uživatelská administrace

Podobně jako v předchozím případě se registrovaný uživatel po přihlášení do systému ocitne v hlavní nabídce uživatelské administrace, kde si může zvolit, jakou akci chce provést. Po každém úkonu se také znovu vrací sem.

I v tomto případě upustíme od podrobného popisu a čtenáře hledající přehled poskytovaných úkonů odkazujeme na diagram případů užití, který jsme uvedli dříve.

3.3 Implementace prezentační části

Nyní zmíníme podstatné skutečnosti týkající se vytváření prezentační části systému. Podkapitola se zaměřuje především na problematiku šablon a modulů z hlediska jejich implementace. Součástí je i obecný popis editačních nástrojů povinných pro moduly, které jsou sice přístupné z administrační části, ale s moduly jsou svojí podstatou tak těsně spojeny, že danou problematiku pro přehlednost neoddělujeme od popisu jejich prezentační vrstvy.

3.3.1 Realizace šablony

Jak jsme již nastínili při volbě technologií, základními pilíři pro konstrukci šablony jsou XHTML, CSS a Smarty. Šablonu jako takovou nemůžeme uložit do databáze (resp. možné to je, ale není to vhodné), neboť je jakousi základní kostrou webové stránky.
Například obsahuje případné http hlavičky, prvky nutné pro vytvoření validního XHTML dokumentu (tj. zejména prvky <*html*>, <*head*>, <*body*> aj.) a v neposlední řadě pomocí oddílu <*div*> poskytuje základní rozvržení webu, do kterého poté prostřednictvím Smarty vkládáme obsah jednotlivých modulů.

Jak již bylo několikrát zmíněno, šablona nemá dostatek informací k tomu, aby mohla sama o sobě zobrazit obsah stránek, k naformátování některých dat zkrátka potřebuje pomoc modulu.

Tento fakt vedl k nutnosti zavést jakousi pomocnou prezentační vrstvu v rámci modulu, která šabloně dodá potřebné informace o tom, jak zarovnat a zformátovat jeho data. I tuto vrstvu realizujeme pomocí Smarty. Díky funkci *{include}*, kterou Smarty poskytuje, ji snadno začleníme do hlavní šablony.

Výsledná šablona tedy vlastně připomíná kostru prázdné (avšak validní) webové stránky, s tím, že na příslušných místech se díky prezentační vrstvě modulu zobrazí správná data. K tomu ještě přidružíme kaskádové styly (zpravidla definované v samostatném souboru), abychom nastavili velikost, typ a barvu písma, barvu (případně obrázek) pozadí a vzhled dalších prvků (tabulky, formuláře atd.).

3.3.2 Tvorba modulů

Již několikrát jsme v textu zmínili prezentační a aplikační vrstvu modulů. Nyní je prozkoumáme z hlediska implementačního a pozastavíme se nad tím, jaká na úskalí přitom můžeme narazit a jak je vyřešíme.

3.3.2.1 Zobrazení modulu

Typický modul stránek se skládá z šablony Smarty, která je (při potřebě zobrazit jej) vložena do šablony stránek. Tento mechanismus tvoří prezentační vrstvu modulu a realizuje záměr, který jsme si dali při návrhu aplikace, když jsme řešili problém ideální šablony (tzn. nutnost přenést část práce při zobrazení stránek na samotné moduly).

Druhou nezbytností je PHP skript, který se stará o načtení dat a jejich předání prezentační vrstvě.

3.3.2.2 Správa modulu

Neopomenutelným prvkem jsou skripty, jež umožní vložení modulu do stránek (tzn. jeho vznik, s čímž je spojeno vytvoření případných tabulek v databázi stránek), odebrání modulu ze stránek (čili jeho zánik a odstranění eventuálních záznamů v databázi), editaci obsahu a provedení zálohy modulu.

Tuto funkcionalitu můžeme souhrnně označit jako aplikační vrstvu, do níž samozřejmě patří i výše zmíněný skript pro předání dat prezentační vrstvě. Musíme mít na paměti, že možnost vytvoření, zrušení a editace modulu by měla být přístupná pouze z administrační části aplikace.

3.4 Bezpečnost

V této podkapitole se zaměříme na různá bezpečnostní hlediska a potencionální hrozby pro server samotný, pro uživatele sytému i pro jejich data.

3.4.1 Relace (session)

Protokol HTTP je bezstavový, z čehož plynou jistá omezení. Pro přenos kontextu mezi jednotlivými stránkami v aplikaci proto používáme tzv. session (česky relace nebo sezení). Jedná se o mechanismus PHP, který nám umožní uchovat data i po ukončení běhu skriptu a dovolí je použít v některém dalším.

Nicméně tento modul PHP nemůže sám o sobě zajistit, že data uložená během relace konkrétního uživatele nebudou zneužita třetí osobou. Návštěvník webové stránky dostává přidělen unikátní identifikátor (tzv. session id), který však může být potencionálním útočníkem zcizen.

Abychom minimalizovali škody při takové situaci, neukládáme do relace žádná citlivá data. Například ověření uživatelského hesla oproti tomu uloženému v databázi se provádí při přihlášení, samotné heslo však do relace neukládáme (je to navíc i zbytečné). Při všech úkonech spojených s nějakou nevratnou operací nebo zásadní změnou jej také budeme od uživatele znovu požadovat.

Protože relace slouží především k udržování informací o přihlášení uživatele, existuje v aplikaci mechanismus, který umožní nastavit a kontrolovat maximální dobu platnosti přihlášení při nečinnosti. Délka tohoto intervalu je nastavitelná formou programové konstanty a může být velmi jednoduše změněna. Pokud ji uživatel překročí, musí se znovu přihlásit, než bude moci pokračovat v práci.

Pokud bychom chtěli systém do budoucna ještě více zabezpečit, můžeme zavést politiku povinného používání tzv. cookies (koláčky) pro ukládání relace na straně uživatele. K zabezpečení komunikace mezi klientem a serverem proti odposlechu lze také implementovat technologii SSL (Secure Sockets Layer).

3.4.2 Uživatelská hesla

Tím se dostáváme k dalšímu problému spojenému s přihlašováním uživatelů a to je jejich autentizace. Uživatelé mají svá přístupová hesla, která musíme ukládat v databázi. Protože žádný systém nikdy nebude stoprocentně bezpečný, může se stát, že dojde ke zcizení databáze a s tím i všech uživatelských hesel.

Bohužel je poměrně obvyklé, že běžný uživatel má pouze jedno heslo, které používá hned v několika aplikacích (pro svůj osobní e-mail, pro uživatelský účet na PC v práci, pro přihlášení ke službě ICQ atp.). Dojde-li tedy k jeho zcizení, má útočník teoreticky přístup i k dalším citlivým datům poškozeného.

Vhodným protiopatřením je ukládání zašifrovaných hesel do databáze (například pomocí algoritmu MD5). Nicméně objevuje se zde ještě jeden zádrhel – průměrní uživatelé často volí jako své heslo nějaké slovo či slovní spojení, které lze uhádnout za pomoci slovníku. Nastane-li tato

situace, může útočník vytvořit tzv. Rainbow table (duhovou tabulku), kdy pomocí stejného algoritmu zašifruje slovníková hesla a pak je porovnává se záznamy v databázi. V takovém případě je „prolomení“ otázkou několika málo hodin.

Abychom podobný druh útoku co nejvíce ztížili, použijeme techniku, které se říká „password salting,“ což v doslovném překladu znamená „osolení hesel.“ V podstatě se jedná o proces, kdy k heslu před jeho zašifrováním připojíme ještě nějaký řetězec, čímž zkomplikujeme porovnání výsledné šifry s předem vygenerovanou tabulkou.

V PHP naštěstí existuje funkce crypt(), která plní podobnou úlohu. Pro její podrobný popis viz. [4].

3.4.3 Import šablon

V původním návrhu systému měl registrovaný uživatel možnost nahrát na server vlastní šablonu (přesněji řečeno importovat ji ze souboru) a následně ji použít pro své internetové stránky. Později však bylo jako hlavní nástroj pro vytváření šablon určeno Smarty. To přineslo mnoho výhod, ale také některá úskalí.

Smarty totiž umožňuje vložit PHP kód přímo do šablony (pomocí vestavěné funkce {php} – podrobnosti viz. [5]). Existuje zde proto riziko, že kdosi vloží (úmyslně či nevědomky) do systému šablonu obsahující škodlivý kód a pak získá moc provést na serveru téměř cokoliv (od krádeže či změny zdrojových kódů až po smazání důležitých dat).

Z těchto důvodů byl návrh systému pozměněn a možnost importovat šablonu mají pouze správci systému.

3.4.4 Nahrávání souborů na server

Podobné nebezpečí jako při importování šablon číhá i při nahrávání souborů na hosting prostřednictvím webových formulářů. I zde se musí zajistit, aby někdo nemohl nahrát na server škodlivý skript.

V rámci administrační části jde především o případ, kdy uživatel vkládá obrázek s pozadím, který se má začlenit do šablony vzhledu. Větší riziko existuje při vkládání dat do jednotlivých modulů. Tyto případy jsou hodně specifické, záleží totiž na charakteru modulu a na typech souborů, které budeme chtít umožnit do něj nahrávat.

Ošetření tohoto problému je plně na zodpovědnosti tvůrce modulu. Obecně se dá aplikovat zásada, že se na server mohou nahrávat soubory s obrázky (tzn. formáty jako je GIF, JPEG, PNG apod.) a případně také PDF dokumenty, ostatní typy souborů pak jedině ve formě komprimovaného archívu (například zip, tar.gz aj.).

3.5 Problémy při implementaci

V této podkapitole jsou uvedena různá úskalí, která zkomplikovala vývoj systému. Některé z problémů bylo velice těžké odhalit, protože souvisely s chybami či odlišnostmi webových prohlížečů.

3.5.1 Externí Java Script v XHTML a IE 7

Tato v podstatě jednoduchá, avšak obtížně detekovatelná chyba v MS Internet Exploreru 7 nezpůsobila příliš velké komplikace, spíše promarnila spoustu času, než byla objevena.

Jak už jsme uvedli dříve, díky svým výhodám je v aplikaci vkládán Java Script do XHTML stránek externě, pomocí tagu <*script*> a jeho hodnoty *src*. V tomto případě je vhodný zkrácený zápis ve tvaru <*script* />, který norma vytvořená organizací W3C povoluje jako stoprocentně validní. Prohlížeč Mozilla Firefox jej bez problémů rozpozná a stránku normálně zobrazí.

Zato v IE 7 je situace trochu jiná: po načtení se ukáže jenom bílá obrazovka bez jakéhokoliv obsahu. Celý jev je o to zajímavější, že podíváme-li se na zdrojový kód stránky, bude tento naprosto v pořádku – nechybí jediný znak.

Po dlouhé řadě experimentů se seznam podezřelých částí kódu zúžil právě na zmíněný tag <*script*> a ukázalo se, že toto podivné chování je možné obejít vložením Java Scriptu přímo do XHTML kódu.

Nakonec pomohl blog (dostupné na URL http://www.phpied.com/ie-script-tag-problem/, jaro 2008), kde se autor příspěvku podělil o své řešení, a sice přepsání tagu ze zkráceného zápisu ve tvaru <*script src="... /*> na <*script src="... ></script*>. Tím se problém definitivně vyřešil.

3.5.2 Formátování webu pomocí CSS stylů

Peripetie okolo odlišné interpretace CSS stylů různými prohlížeči jsou dobře známé a bohužel stále velmi časté. Ani při tvorbě tohoto projektu tomu nebylo jinak.

Potíž, která snad nemůže při tvorbě webu pomocí CSS chybět nikdy, je různá šířka elementů v různých prohlížečích. Z větší části je původcem Internet Explorer, neboť implementuje CSS jinak, než alternativní programy a často nedodržuje standardy. V současné době je situace o to zoufalejší, že se přechází na IE verze 7, zatímco starší verze 6 dál existuje vedle ní. S každým dalším prohlížečem, pro který je nutné ladit kód naší aplikace, pak přibývá spousta zbytečné práce a mnoho zmařeného času.

O dalších těžkostech spojených s tímto fenoménem se zde nebudeme zmiňovat. Pokud by to čtenáře zajímalo, na internetu lze nalézt mnoho informací a článků věnujících se zmíněné problematice (někdy se mluví o tzv. „internet explorer hacks“, tedy o kličkách pro internet explorer.)

4 Zhodnocení a perspektiva projektu

Během realizace se projekt z čistě praktického, komerčního řešení přerodil spíše v jakousi studii, neboť se radikálně změnily požadavky na aplikaci a v současné době není ani zcela jasné, zda se nakonec celý záměr zrealizuje. Zjistilo se totiž, že mnoho původních nároků je v praxi spíš na škodu, než k užitku, a že navíc není jisté, zda vůbec bude o podobnou (placenou) službu zájem.

Například systém evidence zákazníků v současné podobě je zbytečně složitý, neboť firma již má údaje o svých zákaznících uloženy v jiném systému. Jeho „duplikátní“ verze (jaká existuje pro potřeby projektu) znamená udržovat dvě různé databáze, které se dříve či později stanou nekonzistentními. Řešením by bylo evidenci uživatelů buďto omezit na nejnutnější minimum, a nebo provázat s existujícím systémem, což však s sebou přináší řadu nových komplikací.

Dalším novým požadavkem je možnost vytvořit kompletní stránky a ty pak nahrát na úplně jiný webhosting, kde budou nadále fungovat bez administrační části a to tak, že sice nepůjde přidávat či odebírat moduly, ale bude existovat možnost spravovat jejich obsah. S tím pak souvisejí i nezbytné změny v systému plateb. Této funkce chce firma využít ke snadnému generování typizovaných internetových stránek (například jednoduché stránky s katalogem a kontaktními informacemi). Zde už vidíme jasný konflikt s tím, co jsme uváděli v implementační části ohledně editace dat v modulech. V současné verzi je administrační část příliš těsně provázána s prezentační, než aby bylo možno tento záměr jednoduše splnit.

Dalším sporným bodem je otázka, zda budou zákazníci ochotni platit za služby, které jsou jinde mnohdy dostupné zdarma. Nicméně dle autorova soukromého názoru lze tento stav do velké míry ovlivnit kvalitou a druhem poskytnutých modulů.

Po vyhodnocení výše uvedených i dalších jiných faktorů zřejmě dojde k vývoji nové aplikace, v mnoha směrech odlišné od stávajícího projektu, nicméně zkušenosti a některé izolované části kódu dobře poslouží jako základ, na kterém je možno stavět.

Všechny tyto skutečnosti navíc nemění nic na tom, že systém tak, jak je navržen, by klidně mohl fungovat v praxi.

Silnou stránku tvoří zajisté koncepce šablon a modulů, která umožňuje snadno rozšířit či doplnit funkcionalitu celé aplikace. Do budoucna tak zůstává prostor pro bezproblémový rozvoj a různá vylepšení a zároveň aplikace zůstává dostatečně flexibilní a přehledná. Taktéž zpracování ovládání formou průvodců je velkou devizou. Může to kladně ovlivnit postoj uživatelů k systému, neboť je to vlastnost u podobných projektů neobvyklá (a snad neprávem opomíjená, neboť její přínos je dobře patrný.)

5 Závěr

Realizovaný Komplexní systém pro tvorbu a správu webů je rozsáhlý projekt, který lze vylepšovat a zvětšovat prakticky donekonečna. Po implementační stránce zbývá stále ještě spousta oblastí, které je možné doplnit či přepracovat. Zejména co se týká modulů, byla uskutečněna jen hrstka nabízejících se možností.

Nový a unikátní je experiment s ovládáním administrace formou průvodců. Ten se v praxi víceméně nevyskytuje. Také k systému modulů a šablon se v projektu přistupuje odlišným způsobem, než bývá zvykem, a autor doufá, že se tato cesta osvědčí alespoň jako použitelná alternativa. Netypická je koneckonců i myšlenka (na jejímž zrodu se autor aktivně podílel) vytvořit nástroj tak mocný, že nám dovolí rozebrat prakticky libovolné internetové stránky na malé, nezávislé kousíčky, které pak zpřístupníme formou modulů, aby je kdokoliv mohl opět poskládat a užít. To vše dostatečně jednoduše, aby to zvládl i nezasvěcený, průměrný uživatel internetu.

Přesto však systém není a ani nemůže být všemocný. Vždy se najdou situace, kdy bude pro zákazníka výhodnější použít nějaký specializovaný redakční (i jiný) systém. Zejména tvorba sofistikovanějších informačních systému za pomoci tohoto nástroje není nejvhodnější a mohla by být zbytečně pracná.

Jak jsme uvedli v předchozí kapitole, projekt se zřejmě v současné podobě nedočká komerční realizace, nicméně stále může uspět a má jistě co nabídnout. Přidat k němu další funkce a služby totiž není příliš náročný úkol.

Během kódování aplikace autor neustále narážel na nové problémy a výzvy. S každým úspěšně vyřešeným bodem získával další poznatky a zkušenosti. Jako dobrá volba se ukázalo použití XHTML a volba technologií celkově, utilizace externích Java Scriptů, oddělení aplikační a prezentační vrstvy,...

Naproti tomu kdyby měl autor stejné zkušenosti jako dnes, využil by například PEAR Framework, což je balík komponent pro PHP, který řeší mnoho typických a znovu se objevujících situací při vývoji webu. Bohužel se o něm blíže dozvěděl až v době, kdy se přepracování stávajících zdrojových kódů již nevyplatilo. Z toho plyne cenná zkušenost do budoucna – důkladněji zjišťovat možnosti a potenciál používaných nástrojů a vždy důsledně hledat, zda již pro daný problém není k dispozici vyzkoušené řešení.

Literatura

[1] Šrahol, Robert: Informační systém pro tvorbu katalogu. Brno, 2008, bakalářská práce, FIT VUT v Brně.

[2] Žert, Zdeněk: Pokročilá aplikace redakčního systému. Bakalářská práce, Brno, FIT VUT v Brně, 2008.

[3] Domovská stránka projektu Website Baker. Dostupná na URL http://start.websitebaker.org (květen 2008).

[4] PHP Documentation Group: PHP Manual. The PHP Documentation Group, 2007. Dostupné na URL http://www.php.net/docs.php (květen 2008)

[5] Ohrt, Monte a Zmievski Andrei: Smarty Manual. New Digital Group, 2007. Dostupné na URL http://www.smarty.net/docs.php (květen 2008)

[6] Domovská stránka W3C. Dostupná na URL http://www.w3.org/ (květen 2008)

[7] Wikipedie, otevřená encyklopedie. Dostupná na URL http://cs.wikipedia.org (květen 2008)

[8] Stejskal, J.: Vytváříme WWW stránky pomocí HTML, CSS a JavaScriptu. Computer Press, 2004.

[9] Gilmore, W. J.: Velká kniha PHP5 & MySQL. Zoner Press, 2005.

[10] Kosek, J.: HTML: tvorba dokonalých WWW stránek. Grada Publishing, 1998.

Seznam příloh

Příloha 1. CD s kompletními zdrojovými kódy a README souborem.