VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INFORMAČNÍ SYSTÉM S FOTOGALERIÍ A DIÁŘEM PRO SBOR

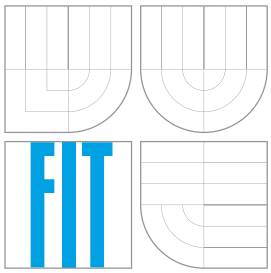
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID HELLEBRAND

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INFORMAČNÍ SYSTÉM S FOTOGALERIÍ A DIÁŘEM PRO SBOR

INFORMATION SYSTEM WITH A PHOTOGALLERY AND A DIARY FOR A CHOIR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID HELLEBRAND

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVLA SEHNALOVÁ

BRNO 2008

Zadání a licenční smlouva jsou uvedeny v archivním výtisku uloženém v knihovně FIT VUT v Brně.

Abstrakt

Úkolem této bakalářské práce je dle konkrétních požadavků zadavatele vytvořit informační systém pro Vox Iuvenalis – pěvecký sbor VUT v Brně. Hlavním cílem je umožnit komunikaci mezi sbormistrem a členy sboru a také mezi členy navzájem. Implementace systému proběhla na základě Zend Frameworku s použitím jazyků PHP, MySQL, XHTML, CSS a JavaScript.

Klíčová slova

informační systém, fotogalerie, XHTML, PHP, Zend Framework, pěvecký sbor

Abstract

The goal of this bachelor's thesis is to develope an information system for the choir Vox Iuvenalis according to the specified requirements of the proposer. The main purpose is to provide the communication between the conductor and choir's members and between choir's members each other. The thesis describes information technologies which were utilized for realization: Zend Framework, PHP, MySQL, XHTML, CSS and JavaScript.

Keywords

information system, photogallery, XHTML, PHP, Zend Framework, choir

Citace

David Hellebrand: Informační systém s fotogalerií a diářem pro sbor, bakalářská práce, Brno, FIT VUT v Brně, 2008

Informační systém s fotogalerií a diářem pro sbor

Prohlášení

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

.....
David Hellebrand
11. května 2008

Poděkování

Děkuji své vedoucí Ing. Pavle Sehnalové za odborné vedení a podněty, které mi při řešení tohoto projektu poskytla.

Obsah

Kapitola 1

Úvod

Webové aplikace přináší v porovnání s kompilovanými aplikacemi řadu výhod. Není zapotřebí instalovat na klientské stanice žádný software. Aplikace běží ve webovém prohlížeči, který je dnes samozřejmou součástí softwarového vybavení počítačů. Použití webového prohlížeče přináší také platformní nezávislost aplikace. Dalším kladem je skutečnost, že uživatelé se k aplikaci dostanou z kteréhokoliv místa na světě. K největším výhodám však patří to, že aplikace běží na serveru. Všichni uživatelé tedy v jeden moment používají stejnou verzi aplikace. To umožňuje snadné přidávání nových funkcí a aktualizaci systému.

Druhá kapitola pojednává o technologiích použitých k vytvoření systému a vysvětluje důvody, proč jsem si je vybral. Najdete zde stručné představení a krátké seznámení s historií těchto technologií.

Ve třetí kapitole se zabývám specifikací a analýzou požadavků na výsledný systém. Je zde zobrazen a detailně popsán diagram případů užití.

Další kapitola pojednává o návrhu aplikace. Uvádím v ní vysvětlení pojmů normalizace a normální formy. Dále obsahuje návrh datového modelu systému reprezentovaný ER diagramem. Poslední částí kapitoly je pojednání o správně vytvořeném uživatelském rozhraní aplikace.

V páté kapitole popisuji implementaci systému. Jsou zde rozebrány jednotlivé části aplikace a popsáno jejich fungování. Další částí je popis cizího kódu použitého při vývoji a systémové požadavky aplikace.

Předposlední kapitola přináší seznámení s průběhem testování systému. Popisuje jednotlivé typy testování a shrnuje poznatky získané testováním.

Závěrečná kapitola obsahuje zhodnocení dosažených výsledků a uvádí budoucí možná rozšíření systému.

Na přiloženém CD najdete kompletní zdrojové kódy systému a elektronickou verzi tohoto textu.

Kapitola 2

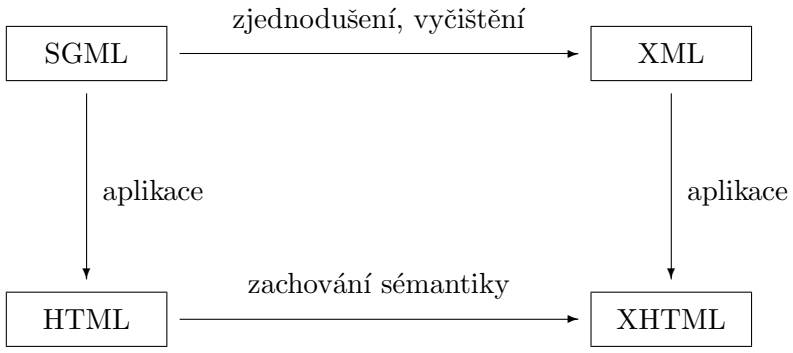
Použité technologie

Aplikace, kterou jsem v rámci bakalářské práce vytvořil, musí být dynamická. Z tohoto důvodu bylo na její realizaci nutné použít několik různých nástrojů. Jejich představení a stručný popis jsou obsahem této kapitoly.

2.1 Jazyk XHTML

Extensible Hypertext Markup Language (dále jen XHTML) je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW. XHTML je nástupcem jazyka HTML verze 4.01. Jazyk XHTML, konkrétně ve verzi 1.0 Strict, byl použit pro tvorbu aplikace v mé bakalářské práci.

XHTML má stejnou mocnost jako HTML, ale vyhovuje také syntaxi XML (*Extensible Markup Language*). Zatímco HTML je aplikací rozsáhlého univerzálního značkovacího jazyka SGML (*Standard Generalized Markup Language*), XHTML je aplikací jazyka XML (více omezující podmnožiny SGML). XHTML dokumenty je díky tomu možné zpracovávat výkonnými standardními XML nástroji – na rozdíl od HTML dokumentů. Ty potřebují vlastní komplexní parser. XHTML může být v mnoha směrech chápáno jako průsečík mezi XML a HTML. [5]



Obrázek 2.1: Značkovací jazyky rodiny SGML [9]

Za jedním z hlavních důvodu přepracování HTML a vzniku jazyka XHTML stála potřeba zobrazovat WWW obsah na mobilních zařízeních na úkor zobrazení na výkonných stolních počítačích. Na mobilních zařízeních není možné vyhradit vysoké procento výkonu

na zpracování HTML složitým parserem. Pro tuto situaci se mnohem více hodí jednoduchý parser XML. [5]

Rozdíly mezi XHTML a HTML

- Všechny tagy a atributy musí být napsány malými písmeny.
- Jednotlivé tagy se nesmí křížit.
- Všechny tagy musí být párové.
- XHTML dokument musí mít na začátku XML prolog (pokud není v kódování UTF-8)
- Dokument XHTML požaduje uvedení správného doctype.

2.2 Jazyk CSS

Jazyk CSS (*Cascading Style Sheets*) je nástrojem pro popis vzhledu dokumentů napsaných některým značkovacím jazykem (např. XHTML nebo XML). V současné době jsou kaskádové styly nejrozšířenější technologií pro specifikaci stylu dokumentů. Jazyk CSS je použit v mé bakalářské práci pro definici vzhledu všech prvků webové aplikace.

Oddělení obsahu dokumentů od definice jejich vzhledu přináší mnohé výhody. Obsah dokumentu se stane přehlednějším. Vzhled všech dokumentů bude definován pouze na jednom místě. Bude možné měnit vzhled dokumentů nezávisle na jejich obsahu či způsobu generování.

Kaskádových stylů může být s úspěchem využito pro různou prezentaci jednoho dokumentu v závislosti na použitém zobrazovacím zařízení (monitor, displej mobilního zařízení, tiskárna apod.).

První verze CSS, označovaná jako CSS1, byla vydána konsorciem W3C v roce 1996. V té době se potýkala s malou podporou prohlížečů. Situace se však postupně zlepšovala. Další verze (CSS2) byla vydána v roce 1998 a přinesla oproti první verzi mnohá rozšíření. Podpora prohlížečů je dnes už na velmi dobré úrovni. Stále se však v některých případech objevuje nejednotná interpretace v různých webových prohlížečích. [1] [7]

2.3 Jazyk PHP

Moderní internetové prezentace si nevystačí pouze se statickými stránkami. Vyžadují interakci s uživateli. K vytvoření takovýchto aplikací už nestačí pouze jazyky XHTML a CSS představené v předchozích kapitolách. V mé bakalářské práci jsem pro tvorbu dynamického obsahu použil skriptovací jazyk PHP.

Historie PHP

Počátky jazyka PHP spadají do roku 1995. Tehdy programátor Rasmus Lerdorf vytvořil Perl/CGI skript, který mu umožnil zjišťovat, kolik čtenářů navštěvuje jeho web. Lerdorf dal veřejnosti k dispozici svou sadu nástrojů a pojmenoval ji Personal Home Page (PHP).

Později své dílo stále rozšiřoval. Snažil se převádět data z HTML formuláře do takové podoby, která by umožňovala jejich export na jiné systémy. Ve vývoji pokračoval v jazyce C a Perl opustil. V roce 1997 vydal PHP 2.0 neboli Personal Home Page Form Interpreter (PHP-FI).

PHP se těšilo celosvětové oblibě a Lerdorf na jeho vývoji spolupracoval s celým týmem vývojářů. Drželi se původní myšlenky začleňování kódu přímo do HTML a přepsali celý engine PHP. Tak v roce 1998 vznikla verze PHP 3.0. Tou dobou už akronym PHP znamenal „Hypertext Preprocesor“.

Následoval další mohutný vývoj, během něhož byly přidány stovky nových funkcí. V roce 1999 přesahoval počet uživatelů hranici jednoho milionu. Pracovalo se na přepsání celého jádra PHP, což mělo zvýšit výkon ve složitých aplikacích. V roce 2000 bylo vydáno PHP 4.0 se zcela novým Zend Enginem. Tato verze obsahovala i další nové klíčové prvky, jako např. podporu mnoha WWW serverů, HTTP sessions nebo bezpečnější zpracování vstupů od uživatelů.

Verze PHP 5.0 vydaná v roce 2004 znamenala další předěl v evoluci jazyka PHP. Byl vytvořen nový Zend Engine 2 a přibyla zdokonalená podpora objektově orientovaného programování. [8]

Popis jazyka

PHP je skriptovací programovací jazyk obzvláště vhodný pro tvorbu webových aplikací. Jedná se o jazyk interpretovaný, to znamená, že je vyhodnocován za běhu a nepřevádí se do podoby spustitelné binární aplikace. PHP zpravidla běží na webovém serveru, jako vstup bere PHP kód a jako výstup vrací HTML stránku. Může být ovšem také využito pro skriptování v příkazové řádce nebo pro tvorbu klientských GUI (*graphical user interface*) aplikací. PHP může být nasazeno na mnoha webových serverech a na mnoha operačních systémech a platformách. K dalším kladům bezesporu patří i to, že je dostupné zdarma. [4]

PHP je vágně typovaný jazyk. Není tedy zapotřebí explicitně vytvářet nebo přetypovávat proměnné. PHP tuto práci dělá automaticky. Vytváří proměnné za pochodu podle pořadí jejich volání ve skriptu. Datový typ proměnné se určí v okamžiku přiřazení hodnoty. Po skončnení běhu skriptu zlikviduje proměnné a vrátí prostředky operačnímu systému. [8]

PHP skripty jsou spouštěny na straně serveru. Webový server je zpracovává PHP procesorem. Ten provádí všechny příkazy, které skript obsahuje. Výsledek vygenerovaný skriptem je pak zaslán zpět klientovi stejným způsobem, jako běžná statická stránka.

Díky tomuto přístupu odpadají problémy s kompatibilitou na straně klienta. Pro korektní funkci aplikace (tzn. zobrazení statické stránky) stačí klientovi prohlížeč zobrazující HTML.

Velkou výhodou PHP je dlouhý seznam různých funkcí, které jsou v tomto jazyce obsaženy. Jedná se např. o funkce pro práci se soubory, s grafikou, s více druhy databází (MySQL, PostgreSQL aj.), s různými internetovými protokoly (HTTP, SMTP, POP3 aj.). PHP tyto funkce obsahuje přímo ve svém jádru, nebo jsou dostupné pomoci dynamických knihoven (extension).

2.4 Databáze MySQL

Obsahem této kapitoly bude stručné seznámení s databází MySQL. O tomto databázovém systému píšu ze dvou důvodů – jedná se v současnosti o jeden z nejrozšířenějších systémů a také jsem ho použil při vývoji mé bakalářské práce.

Historie MySQL

Databáze MySQL vychází ze starší databáze mSQL. Začátek vývoje se datuje do roku 1994, kdy švédská firma TcX začala vyvíjet webové aplikace. Pro potřeby firmy vytvořil programátor Michael Widenius databázový systém UNIREG. Tento systém se však na poli dynamicky generovaných webových stránek neujal. Proto se TcX poohlédla po jazyce SQL (*Structured Query Language*) a databázi mSQL. Tou dobou byl výkon mSQL ve srovnání s UNIREGem nízký.

Michael Widenius kontaktoval autora mSQL – Davida Hughese a navrhnul mu spojení mSQL s UNIREGem. TcX se rozhodla na základě tohoto spojení vytvořit databázový systém, který by plně odpovídal jejim požadavkům. Tím byl položen základ pro vznik MySQL.

Programátoři TcX vytvořili na základě UNIREGu aplikační rozhraní, které bylo podobné jako u mSQL, ale lépe vyhovovalo potřebám společnosti. Výsledkem bylo, že uživatelům mSQL, kteří chtěli přestoupit na nový vyspělejší systém TcX, stačilo udělat pouze drobné změny do existujících kódů.

V roce 1995 měla TcX konečně v rukou databázi, která vyhovovala jejim potřebám – MySQL 1.0. Po nátlaku obchodních partnerů zveřejnila TcX svou databázi na internetu. Výsledkem tohoto kroku bylo mnohem větší rozšíření MySQL než jejího předchůdce mSQL.

Od prvního vydání MySQL na internetu byla tato datábaze doplněna o možnost provozovat ji na mnoha operačních systémech. Dnes je aktuální verze 5.1 a MySQL se řadí k nejpoužívanějším databázovým systémům na internetu. Díky tomu, že je základní verze poskytována zdarma, je populární jak pro osobní použití, tak pro rozsáhlejší komerční projekty.[3]

Popis MySQL

MySQL je multiplatformní databázový systém. Tzn. že je schopen běžet na více systémech (GNU/Linux, Microsoft Windows, FreeBSD, Sun Solaris a mnoho dalších). Komunikace klientů s databázovým serverem probíhá pomocí jazyka SQL. SQL je standardizovaný jazyk používaný v mnoha databázových systémech. V roce 1999 byl přijat zatím poslední standard tohoto jazyka, a to pod názvem SQL99. Ve většině databázových systémů je ke standardu SQL99 přidáno ještě několik specifických rozšíření. Díky tomu má každá databáze svá specifika.

MySQL je v současné době velmi populární databáze. Mezi důvody početného nasazení na webhostinzích patří to, že se jedná o volně šiřitelný multiplatformní software. Dále pak snadná správa, rychlost databáze a také spolupráce s jazykem PHP. Mezi negativa by se dala zařadit nedostupnost některých funkcí a možností v porovnání s konkurečními databázovými systémy. Společně s operačním systémem Linux a se serverem Apache tvoří MySQL základ mnoha webových aplikací (často se pro spojení těchto technologií používá zkratka LAMP – Linux, Apache, MySQL, PHP). [14]

2.5 Jazyk JavaScript

Jazyk JavaScript je skriptovací jazyk určený pro tvorbu aplikací běžících na klientské straně (obvykle ve webovém prohlížeči). Jádro jazyka je syntakticky velmi podobné C, C++ nebo Javě.

Nejčastěji se používá jako součást webových stránek a poskytuje interakci s DOM modelem (*Document Object Model*) stránky. Umožňuje měnit vzhled nebo obsah dokumentu, aniž by byla nutná komunikace se serverem. Díky tomu, že JavaScript běží ve webovém prohlížeči na straně klienta, může aplikace reagovat velmi rychle.[7] [10]

Historie jazyka

Ačkoliv má podobný název jako jazyk Java, není Javascript zjednodušenou verzí Javy. Jde o samostatný jazyk, který byl původně vyvíjen firmou Netscape pod názvem Mocha, později LiveScript a nakonec byl přejmenován na JavaScript. Změna názvu byla provedena z marketingových důvodů a souvisela s přidáním podpory technologie Java do prohlížeče Netscape Navigator. JavaScript byl poprvé představen v roce 1995. [2]

Základní vlastnosti jazyka

- Jedná se o interpretovaný jazyk. Program je vykonáván přímo prohlížečem. Není nutné jej překládat do podoby spustitelné binární aplikace.
- Syntakticky vychází z jazyků C, C++ nebo Java.
- Je objektově orientovaný.
- Má potlačenou typovou kontrolu.

[7]

2.6 Zend Framework

Framework znamená soubor knihoven, které usnadňují práci při programování. Umožňuje např. snadný přístup k databázi, práci s emaily atd. [12] PHP frameworků existuje celá řada. K nejznámějším patří např. PRADO, PHPCake, Symfony, Zend.

Pomocí Zend Frameworku (dále jen ZF) je možné vytvářet kompletní aplikace. Zvládá například generování do PDF, autentizaci, autorizaci, práci s formuláři a spoustu dalšího. ZF je složen z mnoha menších částí, tzv. balíčků, které lze používat samostatně (tento framework je neinvazivní). Nic nám tedy nebrání využít v našem projektu pouze samotný balíček (např. `Zend_Mail` pro práci s emaily) a zbytek frameworku nemusíme vůbec instalovat.

ZF má mnoho výhod. V první řadě není složitý na konfiguraci. Stačí nastavit několik řádku v *bootstrapu* (viz. dále) a vše bude fungovat. Další podstatnou výhodou je, že kolem ZF existuje velká komunita lidí, kteří pomohou s případným problémem nebo nahlásí vývojářům nalezenou chybu. K ZF je také vypracovaná výborná dokumentace napsaná jasnou a jednoduchou angličtinou. K dispozici jsou i překlady do několika dalších jazyků. Čeština mezi nimi zatím chybí. Celá dokumentace je dostupná online nebo je možné stažení ve formátu pdf. Dalším silným argumentem pro použití ZF je firma Zend, která je autorem PHP a zaštítila vývoj celého frameworku.

Mezi nevýhody ZF by se dala zařadit jeho velikost. Samotný framework zabírá přes 10 MB místa. V dnešní době, kdy hostingy poskytují stovky MB místa na prezentace, už tento problém není naštěstí moc palčivý. Dalším nedostatkem by se mohla zdát pomalost frameworku (stovky knihoven) ve srovnání s klasickým způsobem vytvořenou webovou aplikací (pouze jeden skript starající se o výběr a zobrazení dat). Je potřeba zvážit, jestli

mírné zpomalení aplikace nebude na obtíž. Ve většině případů bude ale toto zpomalení jistě vykoupeno značným usnadněním práce. Řešením problému je nainstalování aplikací Zend Optimizer a Zend Accelerator na webový server. S těmito aplikacemi dosáhne server radikálního zrychlení.

ZF se neustále vyvíji. Stále přibývají nové balíčky. Ve verzi 1.5 se objevily dlouho očekávané `Zend_Form` a `Zend_Layout`. V současné době je poslední stabilní verzí 1.5.1.

Zend Framework patří pod novou *BSD licenci*, což znamená, že za používání Zend Frameworku nemusíte platit žádný poplatek, a to ani při použití pro komerční účely. Framework můžete dále rozšiřovat, vaše třídy však nesmí začínat předponou „Zend“. [11]

2.6.1 MVC architektura

MVC (*Model-View-Controller*) je návrhový vzor (obecně doporučovaný postup) pro tvorbu architektury aplikace. Popisuje, jak co nejlépe vytvořit aplikaci, aby její vývoj byl snadný, neobsahovala chyby a budoucí rozšiřování nebylo noční můrou. [12] Základní myšlenkou MVC je oddělení aplikační logiky od logiky zobrazovací.

Tradiční způsob tvorby webových aplikací je zhruba takový, že v jednom souboru máme na začátku připojení k databázi, poté vytištění hlavičky stránky, dotazy na databázi a zobrazení výsledků databázových dotazů uživateli. Tento způsob se ale stává neefektivním při vytváření nějakých změn. Navíc je nutné provádět změny v různých souborech.

Jednou z možností řešení uvedeného problému a zároveň hlavní charakteristikou MVC je rozdělení kódu do tří samostatných částí (a většinou i do samostatných souborů). Viz. 2.1. [6] Oddělení rozdílných částí aplikace má za důsledek snazší vývoj a údržbu.

Model	Model je část aplikace starající se o výběr dat, která budou zobrazena.
View	View se stará o prezentaci dat uživateli (nejčastěji pomoci HTML).
Controller	Má na starost vytvoření modelu. Volá správný View.

Tabulka 2.1: Popis Model-View-Controller architektury

MVC architektura je využita i v Zend Frameworku.

2.6.2 Bootstrap

Bootstrap je soubor zpracovávající všechny požadavky aplikace. Obvykle se jedná o soubor `index.php`. Pro základní funkčnost stačí, aby obsahoval pouze dva řádky kódu:

```
require_once 'Zend/Controller/Front.php';
Zend_Controller_Front::run('/path/to/app/controllers');
```

Tento kód inicializuje *Front controller*, který bude předávat požadavky příslušným kontrolerům akcí (*action controllers*).

2.6.3 Jádro Zend Frameworku

[13] Jádrem ZF je `Zend_Controller`. Tento balíček má na starost obsloužení požadavku, vytvoření modelu a zobrazení pohledu (*view*). Je založen na MVC. Obsluha požadavku probíhá následujícím způsobem:

1. V *bootstrap souboru* dojde k zavolání metody `run`, která začne obsluhovat požadavek.

2. Následuje proces *routování*. Z předané URL se zjistí název kontroléru (*controller*) a akce (*action*), která požadavek obslouží. V ZF jsou používány tzv. pěkné URL (*nice URLs*) ve tvaru http://example.com/controller/action/parameter/value. Tedy například adresa http://blog.com/article/edit/id/5 by mohla na fiktivním blogu sloužit k editaci článku s ID 5. Kontrolérem by byl `article`, akci by odpovídalo `edit`.

3. Nyní `Zend_Controller_Dispatcher` na základě názvu kontroléru a akce vytvoří instanci kontroléru a zavolá příslušnou metodu. V příkladu z předchozího bodu by se použil kontrolér se jménem `ArticleController` a akce se jménem `EditAction`. Tato třída musí být uložena v příslušném adresáři pod názvem `ArticleController.php`

4. V příslušné metodě kontroléru se model naplní daty.

5. Následuje zobrazení pohledu (*view*), nastavení případných hlaviček a odeslání odpovědi klientovi.

2.6.4 Proč použít Zend Framework

Někdo by si mohl říci, proč vlastně používat nějaký framework třetí strany. Účelnější by přece mohlo být napsání svého vlastního frameworku, který by splňoval všechny mé požadavky, neobsahoval spoustu věcí navíc a nenutil programátora uchylovat se k různým kompromisům.

Konkrétně u ZF patří mezi pádné argumenty, proč nevyvíjet vlastní framework, komunita vyskytující se kolem tohoto projektu. Za dobu vývoje už vytvořila obrovský kus práce. Tomu nemůže jednotlivec konkurovat. Pokud vývojáři některá část ZF nevyhovuje, stále zůstává prostor na práci s kódem a přizpůsobení si různých částí frameworku pro své potřeby.

Dalšími nesnázemi při použití vlastního frameworku může být nedostatečná dokumentace. Pravděpodobně se nikdy nedostanete k sepsání podobně kvalitní dokumentace jako je u komerčních frameworků. S tím souvisí další obtíž, a to když práci na projektu převezme někdo jiný. Asi nebude nadšený z toho, že se musí učit práci s nástrojem, který nikde jinde nepoužije.

Kapitola 3

Specifikace a analýza požadavků

Pěvecký sbor VUT v Brně Vox Iuvenalis má požadavek vytvořit informační systém, který umožní lepší komunikaci mezi členy sboru.

Systém bude sloužit pro komunikaci mezi sbormistrem a zpěváky. Uživatelem systému bude člen sboru. Všichni uživatelé mohou sdílet s ostatními informace a kontakty o sobě. Systém jim umožní uveřejňovat fotografie, prohlížet informace jiných uživatelů, zanechávat vzkazy nebo vkládat zajímavé tipy a akce.

Administrátor vstupuje do systému v tzv. administrátorském režimu, jenž mu umožňuje vkládat termíny koncertů a zkoušek, přidávat aktuality ze sboru, přidávat nahrávky z koncertů, provádět správu databáze a nastavovat přístupová oprávnění uživatelům.

Spolu s nasazením systému nesmí vznikat další výdaje, jako například výdaje na nákup softwaru apod. Další podmínkou je jednoduché a intuitivní ovládání aplikace.

3.1 Uživatelé systému

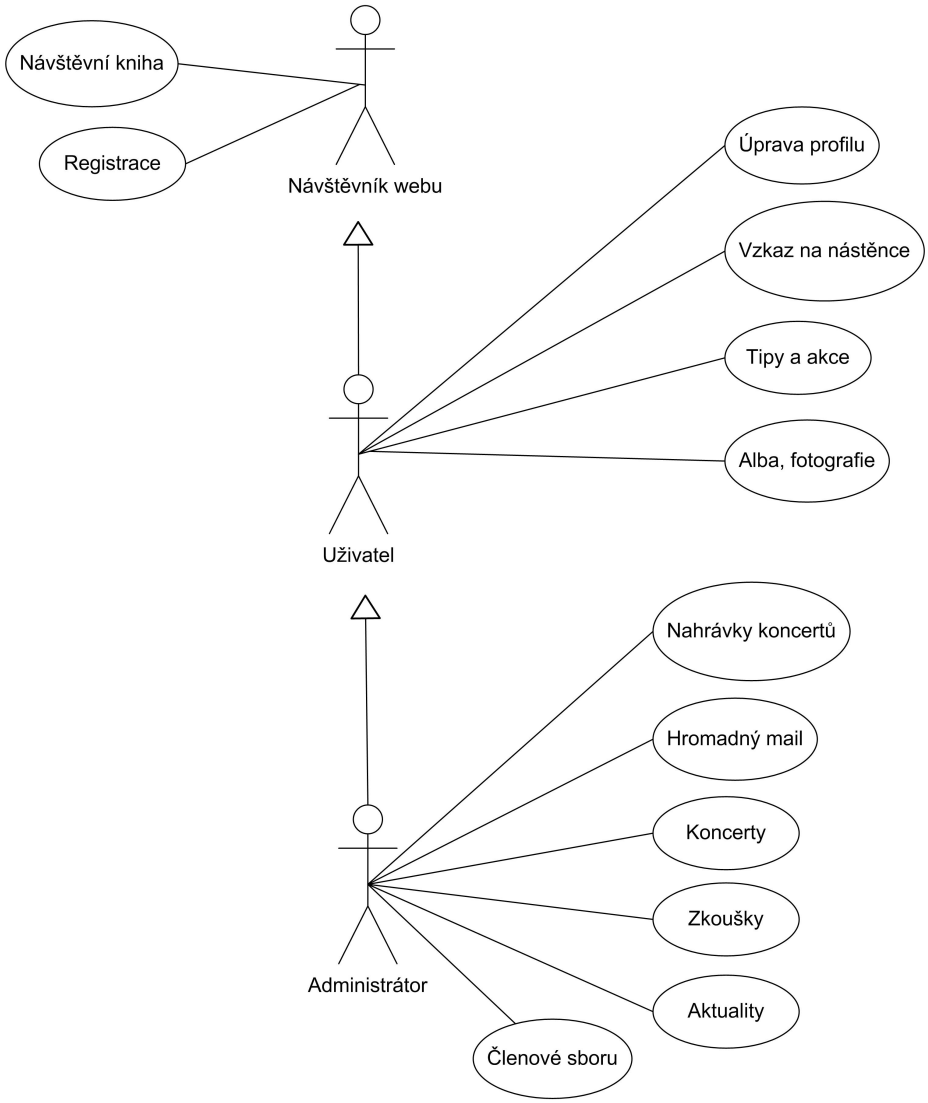
Hlavním uživatelem systému jsou členové sboru, dále označováni pod pojmem *uživatel*. Uživatel pracuje se systémem ze svého počítače a má přístup pouze k některým funkcím. Správu uživatelských účtů, zadávání termínů zkoušek a koncertů, přidávání aktualit a nahrávání záznamů koncertů může provádět pouze osoba s administrátorským účtem – *administrátor*.

3.2 Diagram případů užití

Diagram případů užití je zobrazen na obrázku 3.1. Každý případ užití je dále podrobněji rozepsán.

3.2.1 Popis případů užití

Pro přesnější specifikaci je u každého případu užití uveden podrobný popis požadavků. U každé části systému platí, že zatímco administrátor může mazat a editovat veškerý obsah, uživatelé mohou takto měnit pouze své vlastní příspěvky. Jedinou výjimkou jsou přihlašovací údaje uživatelů. Ty nemůže ostatním uživatelům měnit ani administrátor.



Obrázek 3.1: Diagram případů užití

Autentizace a autorizace

Uživatel i administrátor se přihlašují přes společnou obrazovku. Pro přihlášení je nutné zadat uživatelské jméno a heslo. Na základě přihlášení systém rozpozná uživatelova přístupová práva a případně mu zakáže přístup do sekcí, pro které nemá dostatečné oprávnění.

Bez úspěšného přihlášení se uživatel do systému vůbec nedostane. Bude mít přístup pouze na přihlašovací obrazovku nebo k formuláři pro registraci nového uživatele.

Návštěvní kniha

Návštěvní kniha je součástí stávajících veřejných webových stránek Vox Iuvenalis. Příspěvky může přidávat kdokoliv, včetně návštěvníků webu. Po autorovi příspěvku je požadována jeho přezdívka a samozřejmě text příspěvku.

Úprava profilu

Přihlášený uživatel může měnit informace pouze ve svém profilu. Profil každého uživatele je rozdělen do dvou samostatných částí. První z nich je *uživatelský účet*. Tady si může uživatel změnit své přihlašovací údaje a e-mailovou adresu. Druhou částí jsou *osobní informace*. Povinné jsou položky jméno a příjmení. Ostatní údaje o jeho osobě může uživatel nechat prázdné.

Vzkaz na nástěnce

Nástěnka slouží jako prostředek rychlé a snadné komunikace mezi členy sboru. Přidávat příspěvky mohou všichni uživatelé. Záznamy obsahují pouze text příspěvku.

Tipy a akce

Toto místo slouží pro zveřejnění informací o zajímavých tipech nebo akcích, které by jinak mohly zapadnout mezi příspěvky v diskuzním fóru nebo na nástěnce. Každý příspěvek se skládá z titulku a vlastního textu sdělení. Do této sekce mohou přispívat všichni uživatelé.

Alba, fotografie

Součástí systému je i fotogalerie umožňující členění fotografií do alb. Při zakládání nového alba je po uživateli požadován pouze název alba. Při nahrávání fotografií si uživatel vybere, do kterého alba bude fotografie zařazena a volitelně zadá popisek fotografie. Z důvodu šetření místem na serveru je nezbytné u nahrávaných fotografií zmenšit jejich rozlišení.

Nahrávky koncertů

Nahrávky koncertů budou sloužit jako výukový materiál pro členy sboru. Povolenými formáty nahrávek jsou mp3, wma a zip. Každá nahrávka obsahuje krátký titulek a prostor pro podrobnější popis. Uživatel má možnost seřadit si seznam nahrávek podle data vložení, titulku nebo názvu souboru. Občas je zapotřebí nahrát archiv s více nahrávkami. Systém musí umožnit nahrání souboru až do velikosti 10 MB. Nahrávky může vkládat jen administrátor.

Hromadný mail

Jedná se o další prvek usnadňují sbormistrovi kontakt se zpěváky. Pomocí jednoduchého formuláře může snadno odeslat e-mail všem členům sboru. E-mailová adresa je povinnou součástí každé registrace. Odpadá tak možnost, že někteří uživatelé nebudou mít e-mailovou adresu zadanou. Hromadný e-mail může odesílat pouze administrátor.

Koncerty

Díky této funkci budou v systému uvedeny termíny všech koncertů. Každý záznam se skládá z termínu koncertu, místa konání a doplňujícího textu. S koncerty může pracovat pouze administrátor.

Zkoušky

Tato sekce slouží ke shromáždění všech termínů a informací o zkouškách. Každý záznam se skládá z termínu zkoušky a doplňujícího textu. Přidávat nové termíny může pouze administrátor.

Aktuality

Aktuality jsou určeny zpěvákům jako zdroj informací o aktuálním dění ve sboru. Mohou být vkládány pouze administrátorem. Aktualita se skládá z nadpisu a doplňujícího textu.

Členové sboru

Nové členy sboru může přidávat pouze administrátor. U každého člena musí zadat jeho jméno a příjmení. Volitelnou položkou je členova přezdívka.

Kapitola 4

Návrh systému

Po dokončení specifikace požadavků, jejich zpracování a analýze jsem přešel k návrhu systému. Na základě výše uvedených požadavků jsem začal vytvářet model systému a navrhnul jsem uživatelské rozhraní.

4.1 Normalizace

Pojem *normalizace* můžeme chápat jako proces, při kterém jsou data uložená v tabulkách relační databáze upravována do takové podoby, aby odpovídala určitým požadavkům – tzv. *normálním formám*. Těchto forem existuje šest. Čím vyšší je dosažená norma, tím kvalitnější je návrh databáze.

4.1.1 Normální formy

V každé normální formě je definováno, jaké podmínky s ohledem na datové nebo funkční závislosti mezi jednotlivými atributy musí tabulka splňovat. Zároveň musí být splněny také podmínky všech nižších normálních forem. Tzn. tabulka ve třetí normální formě splňuje všechny podmínky formy druhé a zároveň splňuje ještě některé další vlastnosti.

Ve většíně případů postačuje normalizace databáze do třetí normální formy, kdy už dochází k odstranění většiny anomálií. Z tohoto důvodu zde uvádím pouze tyto první tři stupně normalizace.

1. normální forma (1NF)

Relace je v první normální formě, právě když všechny její jednoduché domény obsahují pouze atomické (dále nedělitelné) hodnoty.

2. normální forma (2NF)

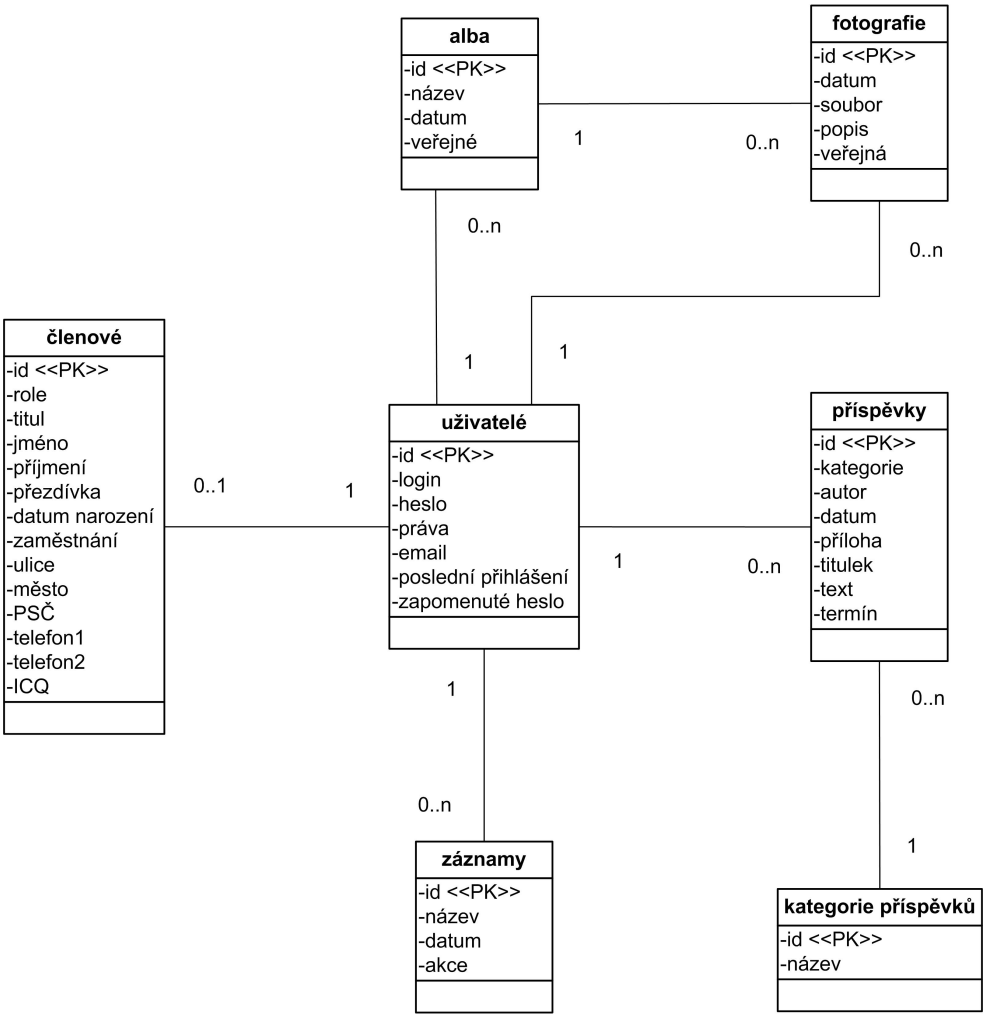
Ve druhé normální formě se nachází relace splňující všechny podmínky 1NF a zároveň každý její atribut, který není primárním klíčem, je plně funkčně závislý na každém kandidátním klíči.

3. normální forma (3NF)

Relace se nachází ve třetí normální formě, právě když splňuje všechny podmínky 2NF a navíc každý neklíčový atribut je plně závislý na primárním klíči.

4.2 ER diagram

Model uložení dat v systému je zobrazen v ER (*Entity Relationship*) diagramu na obrázku 4.1. Návrh databáze se nachází ve třetí normální formě.



Obrázek 4.1: ER diagram

4.3 Uživatelské rozhraní

Uživatelské rozhraní poskytuje uživateli přístup k aplikaci a zprostředkovává mu přístup k implementovaným případům použití.

4.3.1 Jednoduchý a přehledný vzhled

O uživatelském rozhraní se dá říci, že je jednou z nejdůležitějších součástí aplikace. Je totiž jedinou částí programu, se kterou přijde uživatel do styku. Pokud bude uživatelské rozhraní nepřehledné a zmatené, pak si ani sebelepší program oblibu u uživatelů nezíská. Dokonalé

uživatelské rozhraní není samozřejmě jen otázkou grafického zpracování. Velmi záleží také na intuitivnosti, přehlednosti a jednoduchosti ovládání.

Protože s aplikací budou pracovat i lidé s malým množstvím zkušeností v oblasti práce s výpočetní technikou, bylo jednoduché ovládání jedním z hlavních požadavků při specifikaci systému.

Do návrhu uživatelského rozhraní jsem zapracoval připomínky uživatelů během testování aplikace. Mou hlavní snahou bylo dosažení jednoduchého a intuitivního ovládání. Z tohoto důvodu jsem se při vývoji držel následujících zásad:

Přehledná navigace

K hlavní navigaci v aplikaci slouží jednotné menu dostupné na každé obrazovce. V tomto menu je vždy zvýrazněna aktuálně prohlížená položka. Uživatel má tedy přehled, na kterém místě systému se právě nachází.

Snadné přechody mezi stránkami

Každá obrazovka obsahuje na jednotném místě odkazy umožňující uživateli provádět další akce s aktuálním objektem. Např. při prohlížení fotografií je zobrazen odkaz na nahrání další fotografie do právě prohlíženého alba.

Jednotný vzhled

Všechny odkazy jsou formátovány jednotným způsobem. Je tedy na první pohled patrné, co je odkaz a co obyčejný text.

Dále mají odkazy s podobnou funkcí text doplněn stejným piktogramem. Např. všechny odkazy pro vkládání příspěvků, fotografií apod. obsahují kromě textu také ikonku „plus“.

Stránkování dat

Stránka obsahující desítky příspěvků nebo fotografií by byla značně nepřehledná. Navíc by se přenášelo větší množství dat a docházalo by ke zpomalení práce s aplikací. Z tohoto důvodu je na stránkách zobrazován pouze omezený počet záznamů.

Limit počtu záznamů zobrazených na stránce je možné nastavit v konfiguraci systému. Přesáhne-li počet požadovaných záznamů tento limit, bude zobrazen omezený počet záznamů a do výstupu bude vložena jednoduchá navigace umožňující snadné procházení mezi jednotlivými stránkami se záznamy.

4.3.2 Omezení chyb uživatelů

Při návrhu uživatelského rozhraní je zapotřebí brát ohled na skutečnost, že uživatelé nejsou stroje a někdy udělají chybu. Jejich chyba nesmí mít dopad na data uložená v systému ani na stabilitu celé aplikace. Častým jevem je uživatelovo „překliknutí“. Např. místo kliknutí na odkaz pro úpravu záznamu vybere omylem jeho odstranění.

Abych zabránil takovýmto chybám, je v aplikaci každý odkaz pro odstranění záznamu doplněn o ověřovací mechanismus. Uživateli se zobrazí okno s otázkou, jestli chce vybraný záznam opravdu odstranit. K definitivnímu smazání záznamu z databáze dojde, až pokud je vybrána možnost „ano“. Vybere-li uživatel možnost „ne“, potvrzovací okno se uzavře a žádná akce provedena nebude.

4.4 Zpomalení vlivem načítání stránek ze serveru

Tento informační systém je navržen jako webová aplikace. Proto dochází při jeho používání k větším časovým odezvám, než je tomu v případě kompilovaných aplikací. Za toto zpomalení může odesílání dat na server, zpracování PHP skriptu serverem a následné odeslání výsledku zpět k uživateli.

Abych maximálně zkrátil dobu mezi odesláním požadavků a přijetím výsledků, snažil jsem se v aplikaci časově náročné akce neprovádět zbytečně opakovaně. Např. souhrn všech informací o přihlášeném uživateli není nutné zjišťovat z databáze při načtení každé stránky. Tyto informace se získají pouze při přihlášení uživatele a nadále zůstavají uloženy v *sessions*. Odpadá tak vyhledávání dat v databázi a zkracuje se doba běhu skriptu.

Dalším řešením tohoto problému by mohlo být naistalování aplikací Zend Optimizer a Zend Accelerator na webový server. Došlo by tak k podstatnému zkrácení doby potřebné pro vykonání PHP skriptu. Toto řešení však nebylo v praxi možné uplatnit, protože jsem nemohl ovlivnit nastavení webového serveru, na kterém aplikace běží.

Kapitola 5

Implementace systému

Po návrhu datového modelu a vzhledu uživatelského rozhraní jsem začal implementovat systém.

Pro implementaci systému byly použity nástroje uvedené v kapitole 2. Výsledné HTML stránky jsou validní podle standardů konsorcia *W3C*, čímž je zajištěno dosažení správného vzhledu na většině zobrazovacích zařízení.

5.1 Konfigurace aplikace

Konfigurace aplikace se provádí úpravou souboru *config.ini*. Jsou zde uloženy údaje potřebné pro připojení k databázi a různé parametry ovlivňující vzhled uživatelského rozhraní (počet příspěvků zobrazovaných na úvodní stránce apod.).

5.2 Použití cizího kódu

Při implementaci jsem se rozhodl v některých částech systému využít volně dostupné aplikace. Jejich představení a popis licencí, pod kterými jsou šířeny, následuje.

MooTools

MooTools je kompaktní, modulární, objektově orientovaný framework, na jehož základě jsou vytvořeny některé další aplikace použité v systému. Mootools je dostupný online na adrese `http://mootools.net/`.

Framework MooTools se šíří pod licencí *MIT*. Tato licence umožňuje se softwarem nakládat téměř libovolně. Uživatelé jej mohou používat, kopírovat, modifikovat i prodávat. Jedinou podmínkou zůstává zahrnutí textu licence do všech kopií a odvozenin softwaru.

DatePicker

Tato jednoduchá aplikace umožňuje uživatelům zadávat do formuláře datum výběrem z kalendáře. Pro svůj provoz potřebuje MooTools a spadá pod licencí MIT. DatePicker je dostupný online na adrese `http://www.styledisplay.com/mootoolsdatepicker/`.

Slimbox

Slimbox slouží ke snadnému prohlížení obrázků na webu. Stejně jako DatePicker vychází také z frameworku Mootools a vstahuje se na něj licence MIT. Slimbox je dostupný online na adrese `http://www.digitalia.be/software/slimbox`.

TinyMCE

Jedná se o platformně nezávislý JavaScriptový *WYSIWYG* HTML editor šířený pod licencí *LGPL*. TinyMCE je dostupný online na adrese `http://tinymce.moxiecode.com/`.

Licence LGPL (*Lesser General Public License*) je variantou licence *GPL*. Ta umožňuje libovolné upravování a používání zdrojových kódů. LGPL na rozdíl od GPL umožňuje spojení s kódem, který není šířen pod GPL.

5.3 Zpracování formulářů

O všechny vstupy uživatelů pocházející z webových formulářů se stará *Zend_Form* – jeden z balíčků Zend Frameworku.

Každá z hodnot získaných z formuláře je pomoci filtrů *StripTags* a *StripTrim* zbavena nepovolených HTML značek a přebytečných bílých znaků. Validátor *StringLenght* kontroluje, jestli délka zadaného textu je v povolených mezích.

Zend_Form v případě chybně vyplněho formuláře zajistí jeho opětovné zobrazení uživateli s tím, že korektně zadané položky se automaticky znovu vyplní a uživatel je tak nemusí zbytečně podruhé vypisovat.

5.4 Registrace uživatelů

Do systému mají přístup pouze registrovaní uživatelé. Při registraci je nutné zadat uživatelské jméno, přístupové heslo a e-mailovou adresu. Všechny tyto údaje jsou povinné. Uživatelské jméno a e-mailová adresa musí být unikátní. V systému nemohou existovat dva uživatelé se stejným uživatelským jménem nebo heslem.

Z důvodu zabezpečení uživatelských účtů jsou do databáze hesla ukládána v zakódované podobě. Pro zakódování je použita hashovací funkce *SHA1* vracející řetězec o délce 40 znaků.

Další bezpečnostní problém představuje kolize hesel. Pokud mají dva uživatelé stejné heslo, bude stejný i jeho hash. Jestliže tuto skutečnost jeden z uživatelů zjistí, může se přihlásit i k účtu druhého uživatele. Z tohoto důvodu je při registraci každému uživateli vygenerován náhodny řetězec, tzv. *sůl* (*salt*), který se s heslem vhodně propojí a následně zakóduje funkcí SHA1.

5.5 Zapomenuté heslo

Pokud uživatel zapomene své přístupové heslo, nemusí si zakládat nový účet, aby opět získal přístup do systému. Na stránce určené pro obnovu hesla zadá do formuláře svou e-mailovou adresu zadanou při registraci. Na tuto adresu mu bude zasláno URL, na kterém najde nové heslo.

URL obsahuje parametr ID složený z deseti náhodně vygenerovaných znaků a časového razítka (*timestamp*). Tyto dvě hodnoty zaručí, že žádní dva uživatelé nebudou mít vygenerované ID stejné. Hodnota ID se uloží do sloupce `forgotten_passw` v databázi.

Po otevření URL zaslané uživateli e-mailem, vyhledá systém v tabulce `users` záznam obsahující ve sloucpi `forgotten_passw` hodnotu parametru ID. Vygeneruje se náhodný řetězec o délce 6 znaků, který je následně patričně zakódován a uložen do databáze jako uživatelovo nové heslo. Toto heslo je spolu s pokyny pro další postup zobrazeno uživateli.

Současně se vymaže hodnota sloupce `forgotten_passw`, čímž dojde ke zneplatnění použité URL. Po přihlášení s novým heslem si může uživatel heslo změnit na libovolné jiné.

5.6 Autentizace

Autentizaci (proces ověření identity) uživatele má na starost balíček Zend_Auth. Zadané hodnoty uživatelského jména a hesla ověří proti sloupcům `username` a `passw_sha1` v databázi. Pokud je přihlášení úspěšné, uloží se informace o uživateli (jméno, e-mail apod.) do sessions. Tím odpadá opětovné zjišťování těchto informaci při načtení každé stránky.

Po úspěšném prvním přihlášení do systému musí dojít k propojení uživatelského účtu s konkrétním členem sboru. Uživateli je zobrazen formulář se seznamem všech členů sboru, kteří ještě nejsou přiřazeni k žádnému uživatelskému účtu. Uživatel v seznamu vybere své jméno a po potvrzení volby bude propojení dokončeno.

Ze systému se může uživatel kdykoliv odhlásit. K tomu slouží odkaz zobrazený na každé obrazovce aplikace.

Při každé akci provedené uživatelem se zaznamená čas jejího provedení. Pokud uživatel nebude se systémem delší dobu pracovat, bude automaticky odhlášen. Doba, po které dojde k automatickému odhlášení, se získává z konfiguračního souboru aplikace. Ve výchozím nastavení dojde k automatickému odhlášení za 30 minut.

5.7 Úvodní obrazovka

Úvodní obrazovka aplikace slouží jako přehled všech nových příspěvků v systému. Informace jsou na stránce rozděleny do několika částí. Toto rozdělení je patrné na obrázku 7.2.

oblast č. 1 Práce s uživatelským účtem

oblast č. 2 Hlavní menu

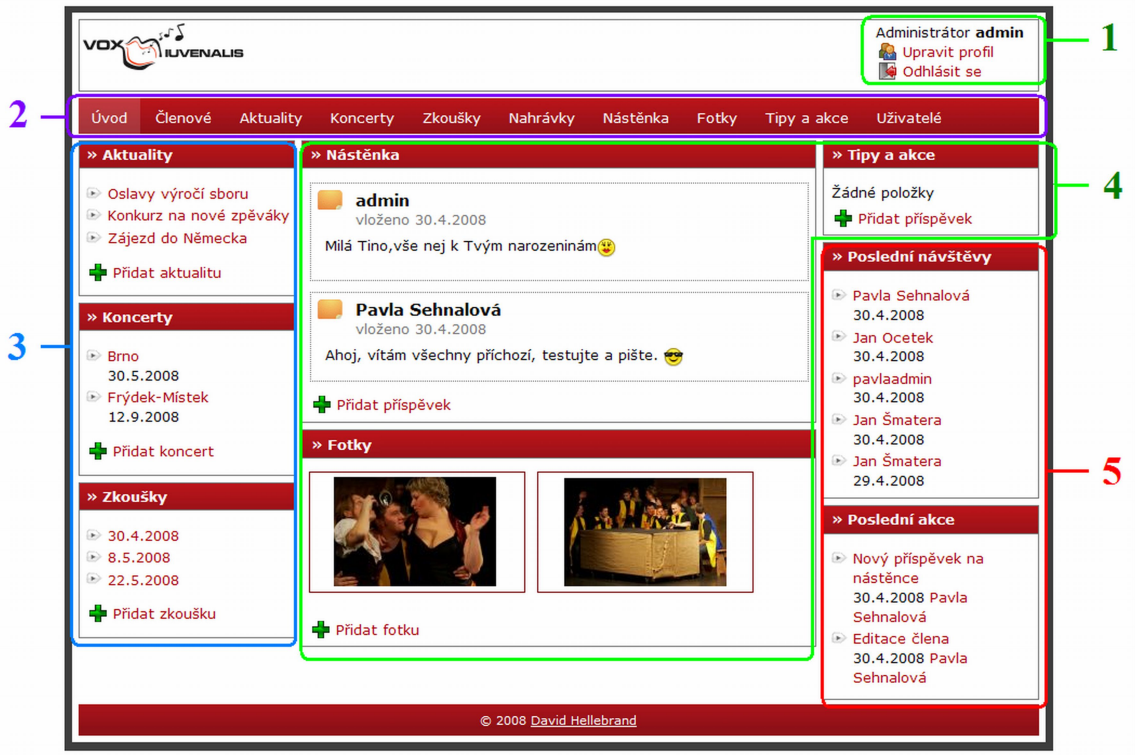
oblast č. 3 Informace zadávané administrátorem

oblast č. 4 Příspěvky vkládané uživateli

oblast č. 5 Výpisy posledních akcí v systému

Oblasti č. 1 a 2 se nacházejí na všech obrazovkách aplikace. Obsahují jméno přihlášeného uživatele, jeho uživatelská práva, odkaz na odhlášení ze systému a hlavní menu celého systému.

Termíny koncertů a zkoušek v oblasti č. 5 jsou seřazeny vzestupně podle data jejich konání. Akce, které už proběhly, systém nezobrazuje. Ve všech ostatních kategoriích se na úvodní stránce zobrazuje posledních několik vložených záznamů. Jejich počet je možné nastavit v konfiguračním souboru aplikace.



Obrázek 5.1: Úvodní obrazovka aplikace

5.8 Systémové požadavky

Následující požadavky jsou nezbytně nutné pro funkčnost systému:

- Webový server Apache 2 s modulem mod_rewrite
- Skriptovací jazyk PHP 5.1.4 a vyšší
- Databázový systém MySQL 5.0 a vyšší

Kapitola 6

Testování systému

Testování softwaru před jeho nasazením do provozu je nezbytnou součástí vývoje. Testovat se dá mnoho vlastností softwaru. Některé oblasti testování jsou obsahem této kapitoly.

6.1 Typy testování

Testování funkčnosti

Nejdříve jsem se zaměřil na otestování správné funkčnosti systému. Tato fáze spočívá v provádění připravených scénářů, které mají modelovat různé stavy, do nichž se během provozu aplikace může dostat. Výstupy aplikace jsou porovnávány s požadavky uvedenými ve specifikaci.

Testování použitelnosti

Úkolem tohoto typu testování je odhalit, jakým problémům čelí uživatel při používání aplikace. Zjistí se, jestli orientace na stránkách nečiní pro uživatele problém a jestli jsou mu všechny ovládací prvky aplikace srozumitelné.

Testování probíhalo nejdříve na malé skupině mých přátel. V poslední fázi byl systém nasazen do testovacího provozu a testovali ho přímo budoucí uživatelé – členové sboru Vox Iuvenalis.

Testování bezpěčnosti

Při testování zabezpečení aplikace jsem se pokoušel odhalit bezpečnostní mezery, které by umožňovaly proniknutí neoprávněné osoby do systému, případně přístup do administrátorských sekcí osobám s nedostatečnými uživatelskými právy.

6.2 Výsledky testování

Testování funkčnosti systému jsem prováděl v průběhu implementace systému i po jejím dokončení. Během testování došlo k nalezení několika chyb v aplikaci. Všechny nalezené nedostatky jsem odstranil.

Na základě připomínek uživatelů během testování použitelnosti jsem do systému zapracoval některé změny. I díky těmto úpravám neobsahuje aplikace žádná problémová místa bránící uživatelům ve snadné práci se systémem.

V rámci testování bezpečnosti jsem se zaměřil na pokusy o neautorizovaný přístup do systému. Na základě použité bezpečností politiky (zakódování hesel, různé úrovně uživatelských oprávnění) by měl být systém dobře zabezpečen.

Platformy použité k testování

- MS Windows XP SP3
- MS Windows Vista Premium SP1 (32-bitů)
- Kubuntu 8.04 Hardy Heron

Webové prohlížeče použité k testování

- Internet Explorer verze 6 a 7
- Mozilla Firefox 2.0.0.14
- Opera 9.23
- Konqueror

Server použitý k testování

- Webový server Apache 2.2.6
- Skriptovací jazyk PHP 5.2.4
- Databázový systém MySQL 5.0.45

Kapitola 7

Závěr

Bakalářská práce se zabývá tvorbou informačního systému pro Vox Iuvenalis – pěvecký sbor VUT v Brně. Toto téma jsem si vybral, protože umožňovalo vytvoření produktu, který bude mít konkrétní využití a bude nasazen do praxe.

Práce na aplikaci začala zjištěním požadavků kladených na systém. Potřebné informace jsem získal z osobních konzultací se zástupci sboru. Na základě zjištěných požadavků jsem vytvořil diagram případů užití s podrobnějším popisem jednotlivých případů. Při návrhu systému byl podle specifikace vytvořen datový model databáze vyjádřený ER diagramem. Následoval návrh vzhledu uživatelského rozhraní a jeho schválení zástupcem sboru. V průběhu implementace jsem byl nucen provádět drobné změny v návrhu systému. V případech nejasností jsem se s dotazem obracel na zástupce sboru. Testování systému probíhalo během implementace i po jejím dokončení. Aplikace byla nasazena do testovacího provozu, během něhož se na testech podíleli přímo její budoucí uživatelé. Do výsledné podoby systému byly zapracovány i jejich připomínky.

Výsledný systém splňuje všechny požadavky na něj kladené a mohl tedy být nasazen do provozu. V budoucnu by mohl být rozšířen o některé další funkce. Např. doplnění fotogalerie o hromadný upload fotografií, případně o program, který umožní uživatelům pohodlněji vybírat fotografie určené pro nahrání do systému (zobrazí náhledy fotek, umožní jejich rotaci „na výšku“ nebo „na šířku“, zobrazí průběh uploadu apod.).

Dalším vhodným rozšířením by mohlo být propojení systému se stávajícími webovými stránkami sboru. Termíny koncertů nebo seznam členů sboru by se získávaly z databáze systému a vkládaly do veřejných stránek sboru. Tato funkce byla původně plánována už do stávající verze systému. Avšak server, na kterém běží webové stránky sboru, nesplňoval minimální systémové požadavky pro spuštění systému. Proto bylo od propojení upuštěno.

Literatura

[1] *Cascading Style Sheets*. [online], [cit. 2008-04-15].
URL <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>

[2] *JavaScript*. [online], [cit. 2008-04-15].
URL <http://en.wikipedia.org/wiki/Javascript>

[3] *MySQL and mSQL : The History of MySQL*. [online], [cit. 2008-04-15].
URL <http://www.unix.org.ua/orelly/linux/sql/ch01_04.htm>

[4] *PHP*. [online], [cit. 2008-04-15].
URL <http://en.wikipedia.org/wiki/Php>

[5] *XHTML*. [online], [cit. 2008-04-15].
URL <http://en.wikipedia.org/wiki/Xhtml>

[6] ALLEN, R.: *Getting Started with the Zend Framework*. [online], [cit. 2008-04-15].
URL <http://akrabat.com/zend-framework-tutorial/>

[7] BURGET, R.; ZEMAN, D.: *Tvorba webových stránek studijní opora*. Brno, 2006.

[8] GILMORE, W. J.: *Velká kniha PHP5 a MySQL*. Brno: ZONER software s.r.o., 2005, ISBN 80-86815-20-X.

[9] HRUŠKA, T.; BURGET, R.: *Internetové aplikace (wap) II. část SGML, HTML, CSS, DOM studijní opora*. Brno, 2007.

[10] HRUŠKA, T.; BURGET, R.: *Internetové aplikace (WAP) VI. část Programování klienta (JavaScript) studijní opora*. Brno, 2007.

[11] MROZEK, J.: *Zend Framework - otázky a odpovědi*. [online], [cit. 2008-04-15].
URL <http://php.interval.cz/clanky/zend-framework-otazky-a-odpovedi/>

[12] MROZEK, J.: *Zend Framework: základní pojmy (MVC, TDD, ZF)*. [online], [cit. 2008-04-15].
URL <http://history.ronnieweb.net/?p=47>

[13] VÁVRŮ, V.: *Zend Framework v kostce*. [online], [cit. 2008-04-15].
URL <http://blog.php-group.cz/2007/11/10/zend-framework-v-kostce/>

[14] ZAJÍC, P.: *Seriál MySQL*. [online], [cit. 2008-04-15].
URL <http://www.linuxsoft.cz/article_list.php?id_kategory=232>

Seznam použitých zkratek

BSD Berkeley Software Distribution

CSS Cascading Style Sheets

DOM Document Object Model

ER Entity Relationship

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

MB Megabyte

MVC Model-View-Controller

PHP Hypertext Preprocesor

POP3 Post Office Protocol version 3

SGML Standard Generalized Markup Language

SMTP Simple Mail Transfer Protocol

SHA Secure Hash Algorithm

SQL Structured Query Language

URL Uniform Resource Locator

W3C World Wide Web Consortium

WWW World Wide Web

WYSIWYG What You See Is What You Get

XHTML Extensible Hypertext Markup Language

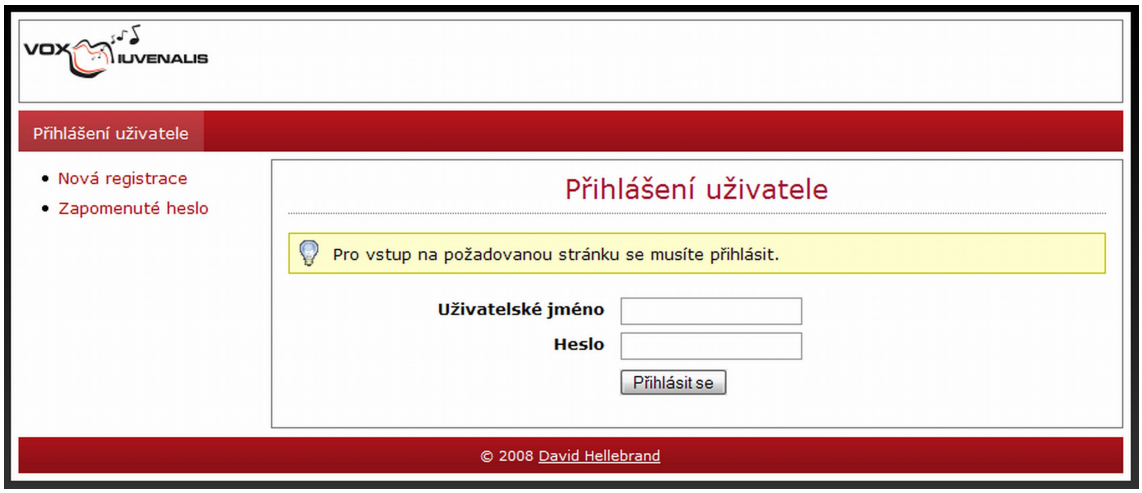
XML Extensible Markup Language

ZF Zend Framework

Seznam příloh

- Příloha A – některé obrazovky systému
- Příloha B – CD s kompletními zdrojovými kódy systému

Příloha A – obrazovky systému



Obrázek 7.1: Přihlašovací obrazovka

Obrázek 7.2: Fotogalerie

Příloha B – CD se zdrojovými kódy

Přiložené CD obsahuje následující soubory a adresáře:

`src`	adresář se všemi zdrojovými kódy aplikace
`readme.txt`	soubor se základními informacemi o aplikaci
`install.txt`	soubor s popisem instalace aplikace
`bachelors_thesis.pdf`	elektronická verze textu bakalářské práce