VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

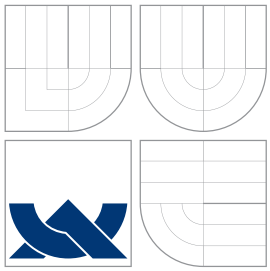
PLATFORMA PRO AUTOMATIZOVANÉ GENEROVÁNÍ INFORMAČNÍCH SYSTÉMŮ - GENERÁTOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

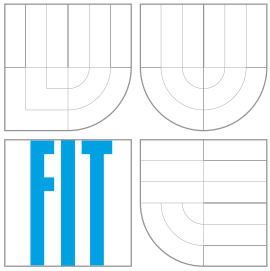
AUTOR PRÁCE
AUTHOR

IVETA ŠENFELDOVÁ

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

PLATFORMA PRO AUTOMATIZOVANÉ GENEROVÁNÍ INFORMAČNÍCH SYSTÉMŮ - GENERÁTOR

THE BASE FOR AUTOMATIC GENERATING THE INFORMATION SYSTEMS - THE GENERATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — IVETA ŠENFELDOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ GRULICH
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Šenfeldová Iveta**
Obor: Informační technologie
Téma: **Platforma pro automatizované generování informačních systémů - Generátor**
Kategorie: Web
Pokyny:

1. Prostudujte existující implementace generických IS určených pro webové aplikace.
2. Navrhněte generátorovou část pro generický IS(specifikátor) tj. konfigurační informační systém pro zadání zakázky na nový IS. Uživatel (zákazník) definuje své požadavky na IS, který je následně systémem generován. Ten je pak možné modifikovat manuálně. Všechny IS lze automatizovaně povýšit na novou verzi. Vyřešte bezproblémové povyšování na novou verzi ve spojení s testovacím systémem.
3. Zvolte vhodné programátorské prostředí a systém implementujte.
4. Systém testujte, zvláště pak kooperaci s produktem.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Grulich Lukáš, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Slečna

Jméno a příjmení: **Iveta Šenfeldová**
Id studenta: 78734
Bytem: Pohoří 328, 742 85 Vřesina
Narozena: 12. 06. 1986, Ostrava-Vítkovice
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Platforma pro automatizované generování informačních systémů - Generátor
Vedoucí/školitel VŠKP: Grulich Lukáš, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

..... Šmídová

Nabyvatel Autor

Abstrakt

Největší podíl rozpočtu na vývoj informačního systému tvoří lidské zdroje. Cílem této práce je snížit tyto náklady na minimum pomocí automatizovanosti, ale se zachováním možnosti upravit každý projekt individuálně. Práce je rozdělena do dvou částí (generátor a produkt), každá je řešena v rámci jiné práce. Tato práce popisuje generátor, který je zodpovědný za vytváření a správu produktů.

Klíčová slova

generátor, informační systém, SVN, ActiveRecord, webový server, Apache, databáze, Linux, Ruby, HTML, CSS

Abstract

The cost of human resource is the biggest part of a budget in case of developing an information system. The goal of this project is to lower this cost as much as possible using automatization, but still keeping the possibility to handle each project individualy. The work is separated into two parts (generator and product), each described in a different thesis. This thesis describes the generator, whitch is responsible for creating and managing products.

Keywords

generator, information system, SVN, ActiveRecord, web server, Apache, database, Linux, Ruby, HTML, CSS

Citace

Iveta Šenfeldová: Platforma pro automatizované generování informačních systémů - Generátor, bakalářská práce, Brno, FIT VUT v Brně, 2008

Platforma pro automatizované generování informačních systémů - Generátor

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Lukáše Grulicha

.....
Iveta Šenfeldová
14. května 2008

Poděkování

Ráda bych poděkovala vedoucímu práce, panu Ing. Lukáši Grulichovi, za odbornou pomoc a konzultace.

Obsah

Kapitola 1

Úvod

Tato práce společně s prací „Platforma pro automatizované generování informačních systémů - produkt“ má za úkol vytořit informační systém, který radikálně snižuje nároky na lidské zdroje maximální automatizovaností. Přístup snižování nákladů se stává trendem v oblasti webových technologií, kde je motivací především vzrůstající poptávka po jednoduchých informačních systémech. Tato práce se bude výhradně zabývat automatizovaným generováním takových systémů. Generátor bude vytvářet produkty, které se skládají z modulů. Základní verze systému s vybranými moduly, které jsou již v nabídce, je určená k okamžitému použití. Avšak v případě, kdy má zákazník nějaké speciální přání týkající se modulu nebo designu, je možné modul (design) doprogramovat. Systém je proto zaměřen na maximální znovupoužitelnost kódu se zachováním možnosti specifických úprav pro zákazníky.

Generátor se zabývá převedením konkrétních požadavků zákazníka na hotový produkt. Zároveň nabízí systém zákazníkovi webový hosting - data budou uložena na serveru společně s produktem a generátorem. Doména je libovolná a tu si volí a zařizuje zákazník sám. Pro dosažení správné funkčnosti je třeba zvolit vhodné nástroje a tím se zabývá kapitola Potřebné nástroje (viz kapitola 2). Kapitola Návrh řešení (viz kapitola 3), detailněji popisuje původní a následně současný návrh. V kapitole Ladění a testování aplikace (viz kapitola 4) je nastíněna důležitost ladění a testování veškerých vyvíjených aplikací. Kapitola Možná rozšíření (viz kapitola 5) pojednává o částech systému, které lze nějakým vhodným způsobem rozšířit na základě praktických zkušeností.

Kapitola 2

Potřebné nástroje

Pro hladký průběh vývoje jakékoliv aplikace je důležité zvolit správné nástroje, které budou k dokončení práce potřeba. V případě informačních systémů a požadavků s nimi spojenými, se nabízí mnoho použitelných nástrojů, které lze často různě kombinovat, ne však vždy je taková kombinace vhodná. Zásadními nástroji, které ovlivňují vývoj a běh této vytvářené aplikace, jsou především:

- webový server,
- databáze,
- správa revizí,
- skriptovací jazyk.

Pro správnou kooperaci mezi těmito nástroji musí existovat další nástroje, které nejsou tak viditelné, avšak nejsou zanedbatelné. Jsou to například knihovny ve zvoleném jazyce, které mimo jiné řeší připojování k databázi, správa revizí pro týmovou spolupráci a další. V souvislosti s knihovnami se nabízí také možnost použití frameworku. Ten ale nebyl v této práci využit, protože nebyl nalezen takový ucelený framework, který by byl dostatečně modulární, aby byl vhodný pro tento případ. V současné době se při vývoji webových aplikací používá nejčastěji technologií PHP (PHP: Hypertext Preprocessor, skriptovací jazyk) a MySQL (databáze)[8]. Pro dynamický obsah na klientu se také často využívá technologie JavaScript a Ajax. Vybrané technologie však nebyly zvoleny na bázi populárnosti, a tak se použité technologie liší od nejpoužívanějších.

2.1 Operační systém

Práce předpokládá použití operačního systému Linux, distribuce Ubuntu. Tato distribuce je vhodným operačním systémem pro server vzhledem k tomu, že vychází periodicky (co půl roku) nová stabilní verze. Takto lze jednoduše držet krok s nejnovějšími technologiemi, aniž bychom používali nedostatečně testovaný software.

2.2 Webový server

Webový server je software, který má za úkol zpřístupnit svůj obsah klientům. V případě požadavku na statický obsah, tj. stránka HTML, je beze změny poskytnuta klientovi, v

případě dotazu na dynamický obsah, je potřeba data nejprve získat, zformátovat a připravit k prezentaci v (X)HTML. Pro dosažení požadované funkčnosti webového serveru je jej třeba správně nakonfigurovat. Různé webové servery se velmi liší a každá konfigurace je jiná. Jak vypadá konfigurace Apache 2, je popsáno v následující kapitole.

2.2.1 Apache

Apache (v současné době verze 2) je jedním z nejlepších open source webových serverů používaných více než 60% všech správců webových aplikací.[3] Je vhodnou volbou i pro tuto práci již jen díky předešlým zkušenostem. Apache má velmi dobré vlastnosti:

- velmi dobře konfigurovatelný s modulárním návrhem,

  je velmi snadné jeho schopnosti rozšířit. Každý, kdo má do jisté míry dostatečné znalosti s programováním v jazycích C nebo Perl může napsat modul, který provádí funkce, které Apache standardně nemá. Z toho plyne, že je k dispozici mnoho modulů, které se dají využít,[3]

- výborně podporuje různé skriptovací jazyky (např. PHP, Perl a jiné),

- aplikace je multiplatformní a vzhledem k tomu, že jsou uvolněny zdrojové kódy v jazyce C, není většinou problém portovat Apache na jinou architekturu,

- podporuje verzi protokolu http 1.1,

  Apache je jedním z prvních webových serverů, které v sobě integrují protokol HTTP 1.1. Je plně v souladu s novým standardem HTTP 1.1 a je zpětně kompatibilní s protokolem HTTP 1.0. Před protokolem HTTP 1.1 musel webový prohlížeč čekat na odpověď ze serveru předtím, než mohl odeslat další požadavek. U protokolu HTTP 1.1 to už neplatí. Prohlížeč může zasílat požadavky souběžně. To bude pro uživatele znamenat zřejmě zlepšení výkonu, protože paralelně zobrazované soubory se v prohlížeči zobrazují rychleji,[3]

- podporuje rozhraní CGI (Common Gateway Interface),

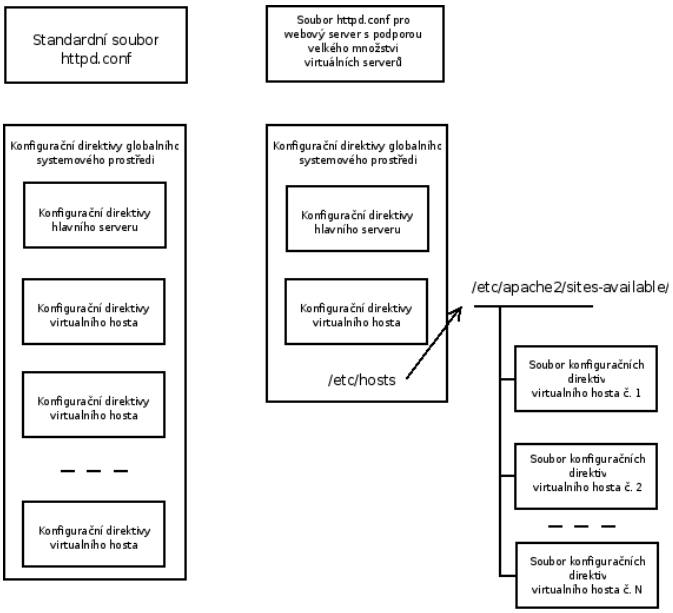
  Apache podporuje rozhraní CGI pomocí modulů mod_cgi a mod_cgid. Jedná se o verzi rozhraní CGI 1.1, která nabízí rozšířené vlastnosti, jako jsou uživatelské systémové proměnné a podpora pro ladění skriptů, jež stěží nalezneme na jiných webových servrech,[3]

- podpora systému FastCGI.

Pro správnou funkčnost je třeba editovat konfigurační soubor httpd.conf (viz obrázek 2.1). Apache pro tuto konfiguraci nemá žádné grafické uživatelské rozhraní, takže je zapotřebí tento soubor otevřít pro zápis pod uživatelem root v textovém editoru a doplnit potřebné vlastnosti.

Pro spuštění, zastavení nebo restartování Apache slouží příkazy: start, stop a restart, které lze užít v kombinaci s cestou k ovládacímu scriptu, která se pro různé distribuce (popř. operační systémy) liší. Pro náš případ (Linux, Ubuntu) je to: /etc/init.d/apache2 start. Pokud chceme, aby Apache načetl novou konfiguraci, použijeme příkaz reload.

Pro to, aby byl Apache schopný zpracovávat skripty napsané v jazyce Ruby, bylo třeba nalézt vhodné řešení. Prvně byl zaveden do Apache modul s názvem mod_ruby, který měl



Obrázek 2.1: Segmenty konfiguračního souboru httpd.conf

být schopný řešit tento problém. Po zavedení se zdálo, že je vše v pořádku, ale po čase se ukázalo, že volba nebyla vhodnou variantou, protože se aplikace začala chovat náhodně. Verze mod_ruby byla zřejmě již zastaralá a vývoj nejspíš dále nepokračuje. Další možností, která se ukázala jako spolehlivá, pak bylo nakonfigurovat Apache tak, aby byl schopný zpracovávat skripty skrz CGI (viz obrázek 2.2).

CGI je jazykově nezávislá specifikace portálového rozhraní, které se dá realizovat téměř jakýmkoli jazykem pro vývoj aplikací , včetně jazyků C, C++, Perl, skriptovacích jazyků příkazových procesorů a jazyka Java.[3]

Na následujících řádcích je uvedena konfigurace Apache v souboru httpd.conf:

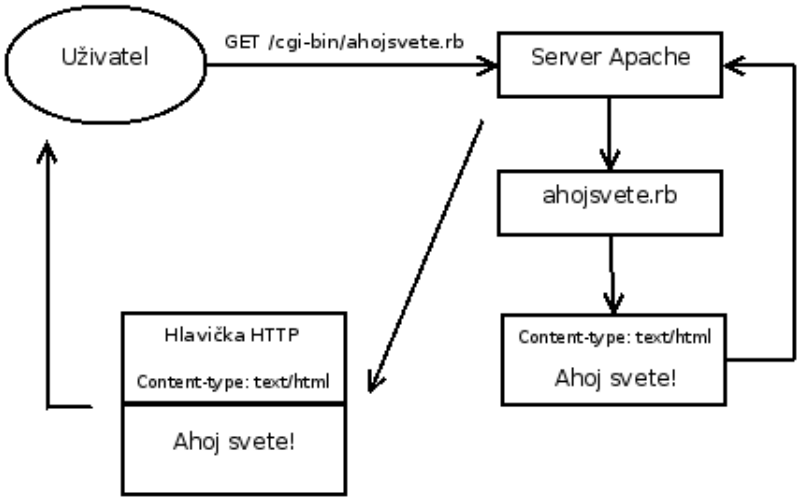
```
<Directory /work/generator/>
   Options ExecCGI
   AddHandler cgi-script .cgi .rb
</Directory>
```

Adresářový kontejner `<Directory>` Apachi říká, že v tomto adresáři budou CGI skripty uloženy a direktiva Options udává, že má Apache v daném adresáři povolit spouštění skriptů. Direktiva AddHandler nastavuje obslužný program skriptů pro daný seznam souborových přípon (v tomto případě .cgi a .rb).

Pro správu všech vytvořených stránek se pak musí vytvořit tzv. virtual hosty. Více o tomto tématu pojednává kapitola 3.2.3. Apache byl vybrán jako webový server, další možnosti byly:

2.2.2 Mongrel

Mongrel je open source knihovna a webový server pro Ruby. Instalace je jednoduchá přes RubyGems (sudo gem install mongrel) a následně se spustí v podobě daemona (mongrel_rails start -d). Mongrel však ke své činnosti potřebuje framework Ruby on Rails, který



Obrázek 2.2: Jak pracuje program CGI

je pro řešení této práce nevhodný.[6] Další nevýhodou v tomto případě je to, že pro každé vytvořené stránky by bylo potřeba spustit další Mongrel (Mongrel nepodporuje virtual hosty), což je naprosto nevýhodné.

2.2.3 WEBrick

WEBrick je další možností webového serveru. Je zabudován přímo ve frameworku Ruby on Rails viz kapitola 2.5.1 . Je to knihovna napsaná v jazyce Ruby. WEBrick také dokáže zpracovávat CGI skripty a erb šablony. Umí také pracovat s Java servlety, které umožňují vytvářet dynamický obsah stránky za pomocí Java platformy.

2.3 Správa revizí

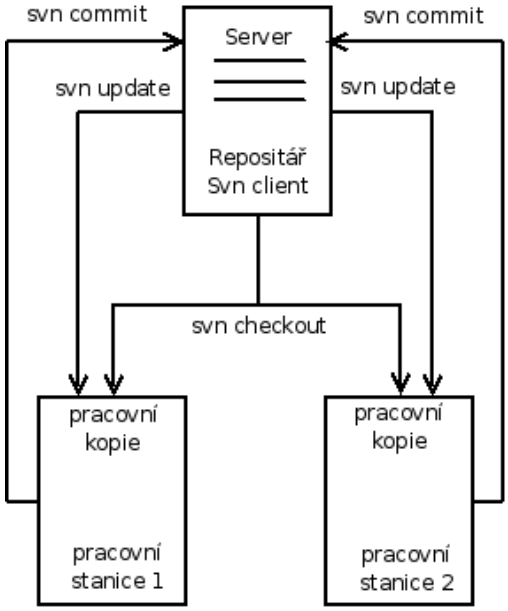
Při řešení větších projektů, jako je například tato práce, je velmi výhodné používat nástroj pro správu revizí. Správa revizí je téměř nutností při vyvíjení aplikace, na které se podílí dvě a více osob, nemůže tak dojít k přepsání kódu. Každá revize obsahuje určité změny provedeny na projektu. V případě, že je repozitář uložen na serveru a ne na pracovní stanici, na které je aplikace vyvíjená, slouží repozitář i jako záloha. Veškeré vzniklé revize, ať už s většími či menšími úpravami, jsou uchovávány a lze se k nim v budoucnu vrátit, což je nesporná výhoda. Chce-li se na projektu podílet další osoba, stačí pouze jeden příkaz k tomu, aby si na své pracovní stanici vytvořil tzv. working copy (pracovní kopii), která obsahuje nejnovější provedené změny. Počet pracovních kopií není omezen. Na výběr je hned z několika aplikací poskytujících správu revizí. V Linuxu existuje GUI nadstavba na Subversion (SVN) s názvem Rapidsvn a v některých programech, jako je třeba Netbeans, je práce s SVN možná právě přes GUI v rámci Netbeans.

2.3.1 CVS - Concurrent Versions System

CVS je open source aplikace určená pro správu revizí. Dnes už relativně zastaralá a na její místo nastoupila aplikace Subversion (SVN) viz kapitola 2.3.2. Oproti SVN není schopno CVS uzamykat soubory proti jejich přepsání. V SVN platí, že např. při ukládání změn na server se soubory uzamknou, modifikují a následně odemknou opět pro zápis.

2.3.2 SVN - Subversion

Vlastnosti, popsané v kapitole Správa revizí, vychází především ze zkušeností s aplikací SVN (viz obrázek 2.3), která byla v rámci této práce využita. Důležitými vlastnostmi SVN je hlavně to, že v případě, že by mělo dojít k přepisu části kódu, nahlásí SVN konflikt a v daném souboru vyznačí obě provedené změny. Takový konflikt musí uživatel vyřešit ručně a pomocí příkazů dát SVN vědět, že konflikt byl vyřešen. Další vlastností a výhodou je to, že při ukládání změn na server je možné zapsat k revizi poznámku a tu lze v budoucnu vyhledat v záznamu. Repozitář lze kdykoli přesunout na jiný server, je pak ale nutné znovu vytvořit pracovní kopii na pracovní stanici pomocí příkazu svn checkout path. SVN server lze spouštět samostatně nebo jako součástí Apache. Pro jednoduché použití stači samostatný server svnserve. SVN server se spouští v podobě daemona svnserve -r path -d, kde path je cesta s uloženými repozitáři. Vypnutí SVN se pak v linuxu provede pomocí příkazu killall svnserve.



Obrázek 2.3: Schéma práce s SVN

Adresářová struktura pracovní kopie

Adresářová struktura pracovní kopie je rozdělena do tří hlavních částí a těmi jsou adresáře: branches, tags a trunk. Adresář branches obsahuje podadresář sites a v něm jsou uloženy další podadresáře vygenerovaných stránek různých uživatelů. Adresář tags pak obsahuje podadresáře, které mají v názvu číslo stabilní verze. Čím vyšší označení, tím novější verze.

Obsahem posledního adresáře trunk je vývoj nejnovější verze, která, když je prohlášena za stabilní a dostatečně otestována, může být nakopírována do adresáře tags pod nejnovějším označením čísla verze.

Nejčastěji používané SVN příkazy a jejich význam

- svn cat [PATH] - výpis obsahu souboru na stdin, parametr PATH je nepovinný v případě, že se soudbor nachází v daném adresáři,
- svn checkout path1 path2 - stáhnutí pracovní kopie z adresy path1 a uložení na adresu path2,
- svn commit [PATH] -m popis - uložení změn na server, parametr PATH je nepovinný v případě, že se nacházíme v pracovní kopii, -m popis slouží k popsání provedených změn v projektu pro případné budoucí dohledání,
- svn copy path1 path2 -m popis - kopírování adresářů/souborů v rámci pracovní kopie a po zkopírování se změny pošlou na server, proto je zde stejný parametr -m popis jako u svn commit,
- svn export path1 path2 - export celé pracovní kopie nebo části z adresy path1 na libovolné místo path2,
- svn info [PATH] - získání informací o daném repozitáři, parametr PATH je nepovinný v případě, že se nacházíme v pracovní kopii,
- svn list [PATH] - vrátí adresářovou strukturu, parametr PATH je nepovinný v případě, že se nacházíme v pracovní kopii,
- svn log [PATH] - jako příkaz svn list, pouze vrací výpis logu, tzn. veškeré popisy daných revizí,
- svn merge -r číslo_dřívější_revize číslo_novější/nejnovější_revize path1 path2 - tento příkaz slouží ke sloučení změn dvou pracovních kopií, rozsah je určen čísly revizí, path1 představuje jednu pracovní kopii a path2 druhou,
- svn resolved [-R] [PATH] - tento příkaz SVN říká, že konflikty v souborech byly vyřešeny a může je tak uložit na server. Parametr -R je nepovinný a udává, že se příkaz svn resolved projde celou pracovní kopii rekurzívně a aplikuje tak změny na všechny soubory v konfliktu. PATH použijeme v případě, že se nenacházíme v pracovní kopii,
- svn status [PATH] - tento příkaz zobrazí, zda jsou některé soubory v konfliktu, modifikované, v SVN nepřidané a podobně. Parametr PATH je opět nepovinný, pokud se nacházíme v pracovní kopii,
- svn update [PATH] - příkaz sloužící k aktualizaci pracovní kopie na pracovní stanici. Stáhnou se veškeré změny ze serveru. Parametr PATH opět nepovinný.

Dalšími podobnými aplikacemi jako Subversion jsou Git, Bazaar, CVS a jiné. Mají odlišné některé vlastnosti, ve své podstatě ale splňují stejný účel.

2.3.3 Git

Git je opět open source aplikace pro správu revizí. Jeho použití je výhodné u velkých a náročných projektů, nicméně jej lze použít i na malé projekty. Git se zaměřuje především na rychlost a efektivitu. Stejně jako SVN má každý vývojář svou pracovní kopii, která je ale zároveň celým repozitářem a změny mezi ostatními vývojáři se sloučí. Není tak potřeba mít neustálý přístup k centrálnímu serveru, na kterém může být repozitář také uložen.
Lokální repozitář může obsahovat i kopie jiných repozitářů. Není proto třeba následovat pouze základní línii projektu, ale i jiné dočasné větve apod.[5]

2.3.4 Bazaar

Další nabízející se distribuovaný nástroj pro správu revizí s názvem Bazaar, který se řadí mezi GNU programy. Bazaar se zaměřuje především na použitelnost a výkonnost. Má podporu refaktoringu a proto v budoucnu nejsou problémy s přejmenováním souborů či adresářů.[4]

2.4 Databáze

Databáze se používají k uložení různých dat. Obsahují různé softwarové prostředky pro manipulaci s daty a tomuto systému se říká systém řízení báze dat (SŘBD). Obvykle jdou ruku v ruce s informačním systémem, který ji využívá ke své funkčnosti. Dnes jsou nejrozšířenějšími databázemi tzv. relační databáze. Relační proto, že jsou založeny na relačním modelu dat a relační algebře. Databáze obsahuje tabulky (relace), nad kterými jsou definovány přípustné operace jako např. výběr dat, uložení dat, vymazání dat apod.

2.4.1 MySQL

Open source databáze, velmi rozšířená a populární co se týče webových technologií. V případě nadměrné zátěže lze využít následující technologie: MySQL cluster - více počítačů se chová jako jeden server. Load balancing - dynamické rozdělení zátěže na více počítačů podle zátížení. Replikace - data mohou být načítána z pomocných serverů, úpravy lze ale stále provádět jen na jednom hlavním serveru.

2.5 Skriptovací jazyk

Pro to, aby stránky nebyly pouze statické a mohly vykonávat složitější operace, je třeba zvolit skriptovací jazyk, který toho bude schopný, a se kterým bude umět webový server (v tomto případě Apache2) pracovat. Jazyků je na výběr mnoho, v současné době je asi nejpoužívanější PHP (původně Personal Home Page, nyní PHP: Hypertext Preprocessor). Novým jazykem, který se stává pomalu populárním, a který není přímo pro webové technologie určený, ale ne však nepoužitelný, je Ruby. Ten byl také v této práci využit především díky snadné rozšířitelnosti, pro získání nových zkušeností a také pro možnost porovnat s jinými jazyky svého druhu.
Skriptovací jazyky si jsou svou syntaxí relativně podobny, ne však už přístupem k různým vlastnostem programovacích jazyků jako je např. objektovost.

Porovnání výhod a nevýhod známých skriptovacích jazyků:

PHP
+ velké rozšíření po světě
+ dostupné různé pomocné nástroje
+ dostupné knihovny
+ velmi populární v poslední době
- není příliš rychlý

Ruby
+ získává na populárnosti v poslední době
+ existující framework Ruby on Rails
+ snadná rozšířitelnost
+ podpora modulování
- není příliš rychlý

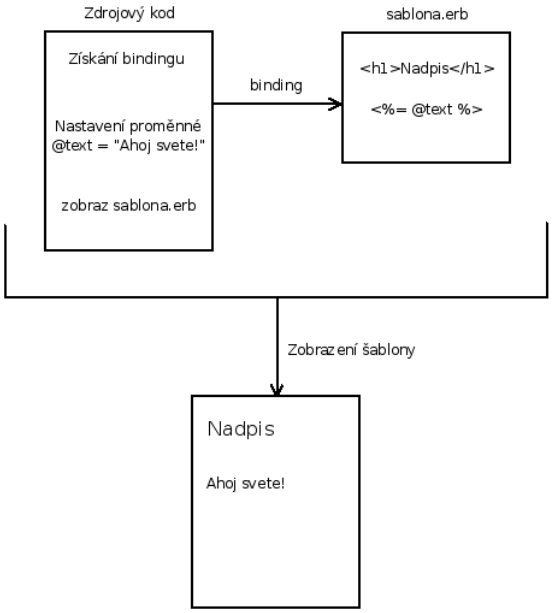
Python
+ rychlost
- složitost
- malé zastoupení na trhu

2.5.1 Ruby

Interpretovaný skriptovací programovací jazyk, který byl zvolen pro řešení této práce, je plně objektově orientovaný, poměrně nový jazyk. Má jednoduchou syntaxi, kterou se lze snadno naučit a přitom je dostatečně výkonný a schopný konkurovat již "zaběhlým" jazykům, jako je například Perl nebo Python. Autorem jazyka je Japonec Yukihiro Matsumoto a vzhledem k tomu, že ještě donedávna nebyla k dispozici dostatečná dokumentace v angličtině, se jazyk Ruby rozšířil především v zemi svého původu - v Japonsku. K hlavním výhodám Ruby patří zejména přenositelnost kódu mezi platformami, možnost spuštění v interaktivním módu, podpora regulárních výrazů - z toho důvodu se také Ruby hodí na práci s texty, jednoduchá a snadno naučitelná syntaxe a další. Jako nevýhody můžeme považovat závislost na interpretru, málo české dokumentace a také třeba malou rozšířenost oproti jiným skriptovacím jazykům.

Šablonovací systém Ruby

Pro oddělení logiky kódu a formátování stránky existuje v Ruby (i v jiných jazycích, např. v PHP existuje Smarty) šablonovací systém, díky kterému je možno vytvořit zvláštní soubor, který řeší pouze zobrazování HTML. Taková šablona je svázána se zdrojovým kódem pomocí tzv. bindingu a pracuje se speciálními proměnnými (název začínající znakem @), které se ve zdrojovém kódu naplní a v šabloně je lze pak zobrazit nebo dále upravovat díky značkám `<\% \%>`, mezi které se píše kód v Ruby, který se pak vykoná. Taková šablona má příponu .erb a je třeba ji ve zdrojovém kódu načíst a zobrazit (viz obrázek 2.4).



Obrázek 2.4: Šablonovací systém v Ruby

Framework Ruby on Rails

Pojem framework je sada programů nebo knihoven, kde jsou již předpřipravené třídy a funkce k použití a tím je ulehčena práce vývojáři. K tomu, aby vývojář dosáhl požadované činnosti své aplikace, je už jen zapotřebí naprogramovat své příliš specifické požadavky. Skriptovací jazyk Ruby takový framework má s názvem Ruby on Rails (RoR).

Framework je znovupoužitelný a jakmile je jednou nainstalovaný, lze jej využít pro jakýkoliv počet aplikací.Potom už není potřeba psát znovu knihovny, které jsou ve frameworku obsaženy.[1]

RoR obsahuje vše potřebné k vytvoření webové aplikace založené na databázi. V současné době jsou podporovanými databázemi tyto: MySQL, PostgreSQL, SQLite, SQL Server.[2]

Active record
Active record je součástí RoR a slouží k automatickému mapování tříd mezi tabulkami databáze. Pro práci s Active Record není potřebná znalost jazyka SQL, je ale třeba znát funkčnost relačních databází.

Technologie YAML

YAML A'int Markup Language - je to standard pro serializaci dat určen pro všechny programovací jazyky.[7] Ruby YAML podporuje nativně a tak je spolupráce bezproblémová.Tato technologie je určená pro uložení objektu do souboru a také pro deserializaci, čili získání dat ze souboru zpět. V této práci je využito především deserializace ve specifikátoru, kde každý modul má svůj soubor module.yaml a v něm uloženy důležité informace, zda je modul veřejný (pblic) a také, na kterých dalších modulech závisí a podle toho, vybere-li zákazník modul, na kterém závisí další moduly, se označí automaticky moduly k němu přidružené.

2.6 HyperText Markup Language (HTML) a Cascading Style Sheets (CSS)

Pro zobrazení základních statických prvků na webové stránce je základem jazyk HTML. Tento značkovací jazyk pochází z rodiny jazyků SGML. V současné době je používaná verze 4.01 a připravuje se verze 5.0, která by měla odstranit nepoužívané nebo zastaralé vlastnosti předchozí verze. Jazyk CSS byl vyvinut především pro definici designu stránek HTML. V poslední době se s vývojem jazyka HTML vypouští veškeré tagy a atributy sloužící k definici barvy písma, barvy či obrázku pozadí stránky apod. a vše se řeší pomocí kaskádových stylů, jež lze zadat různými způsoby, ne jen přilinkováním externího souboru, ale třeba také vkládanými styly přímo do kódu stránky.

Kapitola 3

Návrh řešení

Před tím než začne programátor psát kód, je vhodné si rozmyslet, jak by daná aplikace měla vypadat a co bude používat za nástroje a u větších projektů je výhodné udělat si návrh, který lze v průběhu prací na projektu do jisté míry upravovat. Důležité je vytyčit si dosažitelné cíle a rozebrat vlastnosti různých nástrojů tak, aby spolu byly schopny navzájem spolupracovat. Tento informační systém (generátor a produkt) je plně modulární, tj. skládá se z modulů, a je zde kladen důraz na znovupoužitelnost kódu, proto je většina tříd psána co nejvíce obecně. Pro vygenerování nového produktu slouží uživateli tzv. specifikátor, ve kterém si může v několika krocích navolit požadované vlastnosti svého budoucího systému. První a poslední zobrazená stránka je pouze informativní. Na úvod je uživatel lehce seznámen s vlastnostmi produktu a poslední stránka pak informuje o úspěšném vygenerování systému. V mezikrocích si uživatel vybere moduly, které chce ve svém systému mít a dále je uživatel tázán na název jeho domény a na správné číslo v antispamové ochraně podle obrázku. Na této stránce jsou zobrazeny vybrané moduly uživatelem a v případě špatně opsaného čísla je uživatel informován, že nastala chyba v antispamové ochraně. V takovém případě je údaj nutno opsat znovu.

3.1 Původní návrh

Tato práce je pouze jednou částí velkého projektu. Druhá část (produkt) je řešena samostatně a v konečném důsledku mají tyto dvě práce navzájem kooperovat. Původní návrh byl tedy takový, že částí budou naprosto odděleny, každá bude mít zpočátku svůj repozitář a poté se zavede pouze jeden repozitář, aby spolu mohly práce spolupracovat. Práce původně obsahovala knihovny navíc, které jsou nyní součástí produktu. Jsou to: log a settings. Log slouží k zápisu důležitých událostí, zejména pak výjimek, které se udály za běhu aplikace. V settings jsou uloženy především cesty k repozitáři a k serveru.

3.2 Současné řešení

Po konzultaci s vedoucím bakalářské práce, panem Ing. Lukášem Grulichem, byla část generátor zakomponována v podobě modulů do práce řešící produkt. Produkt je postaven na vlastním frameworku, který je jednoduchý a přehledný, na rozdíl od RoR viz kapitola 2.5.1, který je pro tento případ příliš komplexní a jeho nastudování by tak vyžadovalo mnoho času. Pro správnou funkčnost bylo třeba část generátor do určité míry upravit.

Příkladem může být odstranění přímého načítání knihoven. Tuto záležitost nyní řeší modul Loader, který má za úkol vše potřebné načíst při spuštění.

3.2.1 Adresářová a souborová struktura

Generátor je součástí produktu v podobě několika modulů a těmi jsou: Antispam, Product Customizer, Product Generator a SVN. Všechny tyto moduly obsahují minimálně dva soubory s názvem init.rb a lib.rb. Soubor init.rb pouze inicializuje potřebné věci, u některých modulů není potřeba a soubor lib.rb řeší funkční části projektu. V každém z modulů se ještě podle potřeby může nacházet podadresář templates, ve kterém jsou připraveny šablony erb pro zobrazení obsahu stránky. Všechny tyto moduly se nacházejí v adresáři modules, který sdružuje moduly jak produktu, tak generátoru. Nadřazená složka potom může být buď trunk nebo číslo stabilní verze v adresáři tags. Vše lépe naznačuje obrázek 3.1. Ostatní adresáře v trunk jsou loader, themes, var a vital. V těchto adresářích jsou obsaženy soubory řešící speciální požadavky systému, kterými může být např. přilinkování knihoven a další. Tyto adresáře resp. soubory jsou nutné pro správný chod produktu.

Loader

Slouží pro přilinkování veškerých potřebných knihoven pro vývoj. Pro přilinkování knihovny k souboru slouží direktiva „require“, která se obvykle nachází na začátku každého souboru se zdrojovým kódem. V tomto případě to tak není, loader načítá všechny knihovny pro potřeby vývojáře v jednom souboru a po načtení knihoven inicializuje povolene moduly podle modmanageru.

Themes

Sdružuje veškeré dostupné grafické reprezentace aplikace, mezi kterými lze přepínat. Design je první věc, kterou uživatel vidí, a proto hraje také důležitou roli v rozhodování zákazníka a v rozlišení od konkurence.

Var

V adresáři var se nachází pouze jeden soubor a to system.log. Tento soubor slouží pro zapisování důležitých událostí za běhu aplikace. Další funkcí souboru je možnost ladění aplikace v průběhu vývoje. Více o system.log viz 4.1.2

Vital

Zde jsou sdruženy ty části systému, které jsou životně důležité pro chod produktu.

3.2.2 Funkce jednotlivých modulů

Jednotlivé moduly jsou mezi sebou provázány a jsou součástí produktu. Některé jsou vhodné pro obecné použití, jiné jsou příliš specifické vzhledem k tomuto řešení. Dva hlavní moduly jsou Product Generator a Product Customizer. Pomocné moduly jsou AntiSpam a SVN. Veškeré moduly se nachází ve složce components/, více o adresářové struktuře viz předchozí kapitola . Popis modulů:

Antispam

Slouží k ochraně proti spamu a robotům, aby tak nedocházelo k případným vznikům nechtěných stránek. Řešení je pouze ilustrativní, obrázek s číselným kódem není obměňován a toto může být podnět k budoucímu rozšíření. Ukázka antispamové ochrany - v tomto případě je číslo zadáno správně a stránky se vygenerují (viz obrázek 3.2). Zobrazí se třetí krok, ve kterém bude odkaz vedoucí na právě vygenerované stránky. Ukázka antispamové

ochrany - případ, kdy uživatel zadá špatné číslo (viz obrázek 3.3).

Dnes jsou různé antispamové ochrany nutností. Díky skriptům vytvořeným lidmi, kteří se snaží například nabídnout svůj produkt, který bývá buď nežádoucí nebo nevýhodný či pochybného původu, je nutné provádět různá ošetření, zejména pak u stránek s formuláři. Antispamová ochrana by měla mít logický a proměnlivý charakter tak, aby ji nedokázal prolomit žádný skript. V minulosti se např. používaly obrázky s čísly bez žádného nebo s jednobarevným pozadím, což bylo prolomeno. Skript dokázal „přečíst“ čísla z obrázku a doplnit je do formulářového pole, a proto jsou dnes často vidět obrázky s čísly a pozadí má různobarevné mřížky či jiné obrazce. Dalším možným opatřením jsou potom věty typu: „Nebe je modré. Jaké je nebe?“. Uživatel má potom za úkol logicky doplnit odpověď do formulářového pole. Problémem tady může být to, že uživatel může napsat „modré“, „modre“, „Modré“ apod. a vše bude správně. Proto je třeba mít uložené (např. v databázi) veškeré správné možnosti odpovědí.

Product Customizer

Je modul, který řeší zobrazení a funkčnost specifikátoru, díky kterému lze vytvořit informační systém dle požadavků uživatele. V rámci tohoto modulu je také v administraci vytvořena stránka s názvem Povýšení verze stránek, kde lze povýšit na nejnovější dostupnou stabilní verzi. Více o tomto tématu pojednává kapitola . Jako jediný modul z generátoru je pro uživatele „viditelný“, protože pracuje s šablonami, které zobrazují obsah na stránkách. Tento modul využívá ve velké míře metody z ostatních modulů, především pak svn příkazy a získávání verzí pomocí Product Generatoru. Specifikátor dále implementuje předchozí zmíněný modul - Antispam. V předposledním kroku při vytváření stránek je uživatel tázán na číslo z obrázku. Spolupráce s Apachem je uživateli skrytá, avšak je zásadní. V předposledním kroku, stejně jako u antispamu, se zpracovává šablona pro vytvoření virtual hostů. Je proveden zápis údajů dodaných uživatelem do souborů, které se postarají o zobrazení stránek na localhostu. Více o tématu kapitola .

Product Generator

Zajišťuje generování nových stránek, zjišťování nejnovější verze stránek a zjišťování čísel revizí pro sloučení dvou verzí. Spolu s modulem Product Customizer řeší nejdůležitější operace celého generátoru. V hojné míře využívá modulu SVN pro získávání dat ze souborů.

SVN

Jak už z názvu plyne, je modul, který slučuje příkazy svn v jedné třídě. Příkazy jsou napsány obecně tak, aby tento modul byl dobře znovupoužitelný. Modul neobsahuje veškeré svn příkazy, ale pouze ty, které jsou v rámci této práce zapotřebí. Jsou popsány v kapitole 2.3.2.Většina ostatních modulů pracuje s příkazy svn respektive kooperují s modulem SVN. Tento modul je velmi důležitou součástí tohoto systému. Zejména pak při generování nových stránek a povyšování verze, kde se vykoná hned několik po sobě jdoucích svn příkazů. Je důležitým nástrojem pro správu veškerých verzí a také v neposlední řadě zálohou.

3.2.3 Virtual hosty

Pro správu většího počtu stránek je třeba zadat do Apache konfigurace tzv. virtual hostů tak, aby byl schopen Apache po zadání url pro konkrétní stránky je zobrazit. Při generování nových stránek se proto vykoná připravená šablona, která vytvoří nový soubor s názvem domény a zapíše do něj požadovanou konfiguraci. Šablona neobsahuje žádné prvky HTML, pro modul render je celý obsah šablony jeden string, který se vykoná. Celá konfigurace spočívá ve vytvoření souboru, který se uloží do adresáře sites-available (celá cesta: /etc/apache2/sites-availalble/) s obsahem podobným následující ukázce:

```
<VirtualHost *>
   DirectoryIndex index.rb
    ServerAdmin webmaster@host.foo.com
    DocumentRoot /data/work/generator
    ServerName generator
   <Directory <%=Settings.instance.httpd[:document_root]%>>
     Options Indexes FollowSymLinks +ExecCGI
     AddHandler cgi-script cgi rb rbx
     AllowOverride All
     Order allow,deny
     allow from all
   </Directory>
</VirtualHost>
```

Direktiva DirectoryIndex udává název souboru, který se spouští, pokud je ze serveru požadován adresář. Pokud tedy uživatel požaduje například kořenový adresář, je spuštěn soubor index.rb. Pomocí ServerAdmin je nastavena emailová adresu na administrátora, která se uživateli zobrazí v případě potíží. DocumentRoot je cesta k adresáři, ze kterého se budou stránky spouštět. ServerName udává, pro který hostname bude tento virtual host aktivní. Adresářový kontejner Directory pak znovu definuje nastavení Apache v případě, že by nebyl zapnutý default a neexistovala by tak dostupná konfigurace.

Pro zajištění funkčnosti se ještě musí vykonat další příkazy: a2ensite a reload Apache. Příkaz a2ensite vytvoří symbolický link do adresáře sites-enabled (celá cesta: /etc/apache2/sites-enabled/) pro povolení stránek. Pro načtení nové konfigurace je zapotřebí restart Apache, v tomto případě je použit příkaz reload. Rozdíl mezi příkazem restart a reload je v tom, že restart Apache vypne a znovu zapne, kdežto reload Apache neukončí a jen načte znova konfiguraci. Pokud chceme nějaké stránky zakázat (zrušit), slouží k tomu příkaz a2dissite, který je, už podle názvu, opakem příkazu a2ensite. Ten zruší symbolický link na dané stránky.

Protože se příkazy a2ensite a reload Apache smí spouštět pouze pod uživatelem root, bylo třeba nalézt řešení této skutečnosti. Byla vytvořena nová skupina apache-control a změněn vlastník příkazu a2ensite. Díky tomu lze tento příkaz spouštět bez zadávání rootovského hesla. To ale nefunguje pro reload Apache. Pro tento případ je vytvořen program v jazyce C s názvem reload.c, ve kterém se získají práva k vykonávání takovýchto příkazů a následně se reload Apache provede, což je poslením krokem ke korektnímu zobrazení obsahu právě vytvořených stránek.

Pro prezentaci funkčnosti virtual hostů je po vygenerování nových stránek zobrazen v po-

sledním kroku specifikátoru odkaz vedoucí na vytvořené stránky. Stránky se budou zobrazovat na localhostu a adresou bude název zadané domény uživatelem. Pro zobrazení stránek na localhostu je ještě nutno editovat soubor /etc/hosts, do kterého se za adresu 127.0.0.1 zapíší postupně veškeré generované názvy stránek.

3.2.4 Výběr modulů ve specifikátoru

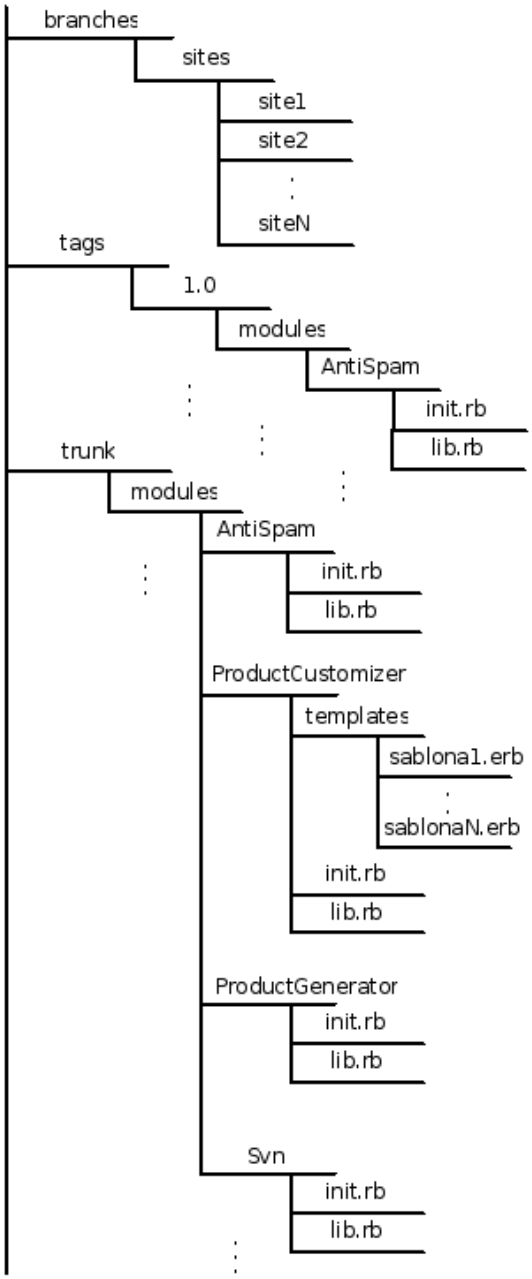
Jakmile se uživatel dostane do druhého kroku při vytváření svých stránek, získá seznam všech dostupných modulů, které může pro své stránky využít. Název každého modulu je doplněn o popis tak, aby bylo zřetelné, k čemu daný modul slouží. Při výběru modulu, na kterém závisí další moduly, se pomocí javascriptu automaticky označí tak, aby byla zajištěna správná funkčnost. U vybraných modulů se pak zobrazí jejich cena. Zobrazeny jsou na stránce pouze ty moduly, které mají vlastnost public, čili jsou veřejné. Tato nastavení jsou uložena v souboru module.yaml u každého modulu. Jak už z přípony souboru vypovídá, je využito technologie YAML pro serializaci dat viz kapitola 2.5.1. Z tohoto formátu pak lze poměrně jednoduše získávat potřebná data.

3.2.5 Správa stránek uživatelů

V administraci je možnost správy stránek uživatelů. Uživatel má možnost ovlivňovat své stránky na základě odebírání či přidávání modulů. Uloženou změnu konfigurace si pak stránky musí načíst z modmanageru. Ostatní nastavení si ale uchovávají ve své databázi (viz obrázek 3.4).

3.2.6 Povýšení verze stránek

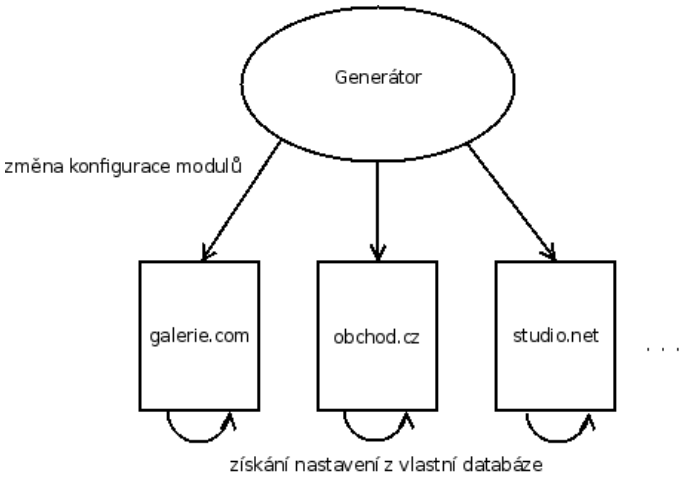
Sekce povýšení verze stránek v administraci je přístupná pouze administrátorům systému nebo uživatelům k tomu určeným. Na stránce je seznam veškerých stránek s uvedenou verzí a pokud je verze stránek rovna verzi nejnovější, je checkbox nastaven na hodnotu DISABLED, tzn. stránky nelze povýšit. Po označení vybraných stránek a odeslání hodnot se provádí následující algoritmus: jsou vytvořeny pracovní kopie (svn checkout) všech označených stránek v adresáři /tmp/, názvy stránek slouží zároveň i jako názvy adresářů. Dále se provede sloučení dvou verzí (svn merge), tj. starší verze a nejnovější, v případě, že nenastanou žádné konflikty, uloží se změny na server (svn commit), stránky se exportují (svn export) na server a smaže se pracovní kopie. Na stránce se opět zobrazí seznam všech dostupných stránek. Nově povýšené stránky mají potom checkbox nastaven na hodnotu DISABLED. Nastanou-li konflikty, zobrazí se na další stránce editovatelné oblasti (HTML tag textarea) tolikrát, v kolika souborech nastal konflikt a administrátor tak může konflikty pohodlně vyřešit a po odeslání formuláře se zavolá metoda pro vyřešení konfliktů, která si nejdříve získá potřebná data nejen z POSTu a nový obsah souborů zapíše zpět do daného souboru. Poté se pro všechny stránky s konflikty vykoná posloupnost příkazů svn resolved - soubory jsou tímto bez konfliktů, svn commit - uložení změn na server a svn export - export povýšených stránek na server. Nakonec se zobrazí stránka opět s nabídkou veškerých vytvořených stránek. Nově povýšené stránky už nemají možnost povýšení do vytvoření další nové stabilní verze.



Obrázek 3.1: Adresářová a souborová struktura

Obrázek 3.2: Zadání korektních údajů do antispamové ochrany

Obrázek 3.3: Zadání nekorektních údajů do antispamové ochrany



Obrázek 3.4: Čtení konfigurace z databází

Kapitola 4

Ladění a testování aplikace

Pro zajištění správné funkčnosti aplikace je nutné ladění (debuggování) a testování. Díky ladění lze zjistit, co se přesně nachází v proměnných, jakého typu proměnné jsou apod. Pro ladění aplikací existují různé nástroje, většinou jsou zabudovány ladící prostředky přímo v aplikaci, ve které je psán zdrojový kód. Příkladem takových aplikací je NetBeans, Eclipse a jiné. Ladění, na rozdíl od testování, probíhá v průběhu vývoje aplikace. Testováním lze získat výsledné hodnoty, které se musí shodovat s požadovanými výsledky. Probíhá především v závěrečném stadiu vývoje aplikace a v případě nalezených chyb je nutno je odstranit a znova otestovat funkčnost.

4.1 Ladící prostředky

Pro ladění aplikace byly využity zejména systémové záznamy, které zapisují události za běhu aplikace. Takový záznam má obvykle aplikace, u které lze očekávat, že dojde k tzv. výjimce (např. pád aplikace). Záznamy slouží především pro administrátory nebo pro pokročilé uživatele, kteří jsou schopni zapsané poznámce porozumět a případně i opravit chybu. Pro ladění aplikací existují i jiné, přímo k tomu určené, aplikace. Ty vypisují zprávy o tom, co by mohlo být příčinou pádu aplikace nebo pouze informují např. o nedealokované paměti (memory leaks) - příkladem je Valgrind. Co se spíše využije u této práce, jsou právě dodatečné nástroje zakomponované přímo v aplikaci sloužící pro psaní zdrojového kódu a tím je již zmiňovaný NetBeans nebo Eclipse. Do kódu lze vložit místo přerušení (breakpoint) a jakmile dojde program ve vykonávání až na breakpoint, vykonávání se zastaví a vývojář se může přehledně podívat do proměnných, jaké mají nastaveny hodnoty a na další vlastnosti.

4.1.1 Systémový záznam Apache

Webový server Apache má svůj vlastní soubor s názvem error.log pro zapisování výjimek a jiných dalších chyb či hlášení, které se udály za běhu aplikace. Tento soubor je jedním z možností, jak ladit chyby vzniklé v průběhu vývoje aplikace. Výpis je přehledný, označený datem a časem a často je uvedeno i číslo řádku, na kterém se chyba nachází v daném souboru. Ne všechny chyby však lze tímto způsobem odhalit. Je výhodné používat další podpůrné prostředky pro ladění. Error.log se nachází v operačním systému Linux (distribuce Ubuntu) ve /var/log/apache2/error.log

4.1.2 Systémový záznam pro produkt a generátor

System.log je soubor vytvořen v rámci bakalářské práce pro další možnosti ladění. Je to klasický soubor, do kterého lze zapisovat cokoliv. Při startu aplikace se původní obsah vymaže a nahradí se novým, který informuje o zavádění modulů do systému a dalších informacích. Na rozdíl od záznamu Apache tak neslouží vyloženě pro chyby nebo chybové hlášky, ale obecně pro logování událostí, které mají být podchyceny, nebo které si chce podchytit pro své účely vývojář. Třída log tak obsahuje pouze dvě metody: log a delete. Log je pro zápis a delete pro vymazání obsahu při startu aplikace, jak již bylo zmíněno. Metoda pro zápis (log) přijímá jeden parametr. Soubor je obvykle výhodný v případě, že potřebujeme zjistit, kterými větvemi program prochází a také jaký obsah je uložen v proměnných. System.log se nachází v adresáři trunk/var/system.log.

4.2 Testování

Testování je důležitou součástí vývoje aplikace. Probíhá zejména v konečném stadiu vývoje, ale do určité míry také v průběhu vývoje. Jakmile se zjistí případné chyby, je nutno je opravit a test vykonat znovu, tak aby bylo jisté, že je vše v pořádku.Testy by měly projít veškeré scénaře, které by mohly nastat. Je nezbytné testovat aplikaci na různé vstupy a porovnávat dosažené výsledky s očekávanými. V této práci je podstatná především kooperace generátoru s produktem. Poté, co se tyto dvě části spojily do jedné, bylo potřeba upravit soubory tak, aby byly schopny spolupracovat s frameworkem vytvořeným pro produkt. Po posledních úpravách bylo nezbytné otestovat spolupráci jednotlivých částí na korektnost. Bylo tak dosaženo požadované funkčnosti.

Kapitola 5

Možná rozšíření

Aplikace je možné většinou dále zlepšovat a přizpůsobovat požadavkům uživatelů. Téměř každá aplikace se dá vylepšit ať už jde o pouhou optimalizaci kódu nebo o úpravu grafického uživatelského rozhraní. Aby byl o aplikaci zájem a byla uživatelům prospěšná, musí jít s dobou, a proto je nutné sledovat moderní trendy a přizpůsobovat se. Důležité je i obměňovat vzhled, ale pokud se jedná o stránky (jako v tomto případě), musí zůstat stále přehledné a rozmístění prvků by mělo zůstat na původních pozicích nebo alespoň v podobné oblasti, v opačném případě by se mohlo stát, že se uživatel na stránkách přestane orientovat, což je nežádoucí. Co je podstatné pro tuto práci je, že vznikly určité nápady, které by mohly mít v budoucnu využití v podobě rozšíření této práce. Z časových důvodů nelze aplikaci vypracovat na úroveň dnes již existujících podobných systémů a toto může být podnětem k budoucímu rozšiřování. Dalším důvodem je možnost případného zájmu nějaké firmy o tento produkt, v takovém případě by nepochybně byla ze strany firmy další rozšíření vítaná či dokonce nutná.

5.1 Modul antispam

Vzhledem k tomu, že modul antispam v této práci nesplňuje plnohodnotně kritéria ochran, které jsou používány běžně v praxi, ale pouze naznačuje řešení a možnost využití, je tedy možné v budoucnu tento modul rozšířit do podoby, s jakou se dnes obvykle lze setkat. Pro dosažení úplné funkčnosti tohoto modulu by bylo zapotřebí náhodně získávat obrázky s různými posloupnostmi číslic, písmen nebo jejich kombinace. Po získání obrázku je nutno porovnat obsah textového pole na správný řetězec a v případě shody vykonat požadované operace. V případě neshody je důležité uživatele informovat, že nastala chyba, aby byl schopen se další chyby vyvarovat.

5.1.1 Různé typy antispamových ochran

Existuje více antispamových ochran a v budoucnu budou nejspíše vznikat další kvůli robotům šířícím spam, kteří se neustále „zdokonalují“ a dokáží tak prolomit stávající ochranu. Z tohoto důvodu je také více typů ochran. Jak už bylo zmíněno v kapitole 3.2.2, v této práci je snad nejčastější používaná ochrana a to přepis posloupnosti znaků z obrázku do textového pole. Jinou ochranou, objevující se spíše na zahraničních stránkách, je logické doplnění textu do formuláře pomocí předcházejícího textu. Existuje i placená ochrana, uživatel musí např. poslat SMS zprávu v určitém tvaru, aby mohl přispívat do diskusního fóra a jiné.

5.2 Ajaxové menu

Na webových stránkách se dnes velmi často objevují prvky asychronního javascriptu - AJAXu. Je to populární zejména z toho důvodu, že se nemusí obnovit znovu celá stránka, ale jen jeden její úsek, ve kterém má dojít ke změně. Populární je také přesun jednotlivých panelů na stránce pomocí drag&drop techniky, kdy stačí vybraný panel uchytit kursorem a přetáhnout na jiné místo. Pro příklad může posloužit dnes velmi známý portál Seznam.cz.

5.3 Dodatečné konfigurace modulů

V druhém kroku specifikátoru, při výběru modulů, může být rozšíření v podobě jakéhosi mezikroku, který by umožnil daný modul ještě dále specifikovat. Při označení checkboxu by se zobrazila nabídka pro další úpravy modulu, které by mohl zákazník v případě zájmu využít. Na stránce by pak byla možna úprava nabízených vlastností modulu. Stránka by měla umožňovat navrátit se zpět bez uložení pozměněného nastavení. Jako příklad může posloužit modul anketa, u které by si zákazník mohl vybrat mezi barevnou nebo černobílou anketou.

5.4 Odlišné distribuce (operační systémy) pro správu serveru

Dnes je mnoho existujících operačních systémů, ne všechny jsou však vhodné pro správu serveru. Ve světě se hojně využívá různých linuxových distribucí. V této práci byla využita distribuce Ubuntu založena na Debianu a systém je tomuto zcela přizpůsoben. V rámci dalšího rozšíření by stála za úvahu případná podpora více operačních systémů nebo distribucí. Systém by tak mohl automaticky detekovat použitý operační systém a na základě získaných informací by pak zvolil správné nastavení. Problém by nastal, kdyby na serveru běžel nepodporovaný operační systém. V takovém případě by musel zasáhnout administrátor. Návrh tohoto rozšíření by byl, v porovnání s dosud uvedenými, nejnáročnějším.

Kapitola 6

Závěr

Díky open source řešení bylo možné vytvořit takto komplexní systém ve školních podmínkách. Bez nástrojů jako Apache nebo SVN by to nebylo možné s dostupnými vstupními náklady. Systém je velmi lehce rozšířitelný a rozšířitelnost je jeho důležitá vlastnost, na kterou byl kladen důraz při návrhu. Podněty a připomínky k dalším úpravám by zákazníci mohli zadávat prostřednictvím dotazníku. Možnosti dalšího rozšiřování a případného směrování vývoje uvádí předchozí kapitola. Díky těmto úpravám by se práce stala atraktivnější pro potencionální klienty a konkurenceschopnejší stávajícím řešením.

Původní představa této práce splňuje plánovaná očekávání. Systém jako celek by sice nebyl schopen po okamžitém nasazení konkurovat dostupným systémům, avšak toto nebylo cílem. Zásadní je správná kooperace generátoru s produktem a vytvoření platformy, na které se dále může stavět komerčně úspěšná aplikace. Mnoho vlastností systému je řešeno pouze náznakově jako ukázka schopnosti platformy.

Práce byla velkým přínosem díky zdokonalování se především v novém programovacím jazyce a pomocných aplikacích. Další zkušenosti pak byly nabyty také díky, do jisté míry, týmové práci vzhledem k tomu, že tato práce v konečném důsledku spolupracuje s jinou prací.

Dodatek A

Manuáky k jednotlivým sekcím generátoru

A.1 Vytvoření stránek

Pro vytvoření vlastních stránek musí být uživatel zaregistrován a přihlášen. Poté se uživateli zobrazí v sekci „Vytvořit nový systém“ další informace (viz obrázek A.1). V prvním kroku vytváření produktu pak má uživatel možnost zvolit si moduly, které bude jeho budoucí systém obsahovat (viz obrázek A.2) a vybere-li si také modul Změna tématu, má na výběr ze tří témat (viz obrázek A.3). V posledním kroku je uživatel tázán na název domény (viz obrázek A.4) a po opsání čísla z obrázku je systém generován a připraven k okamžitému použití (viz obrázek A.5).

A.2 Povýšení verze stránek

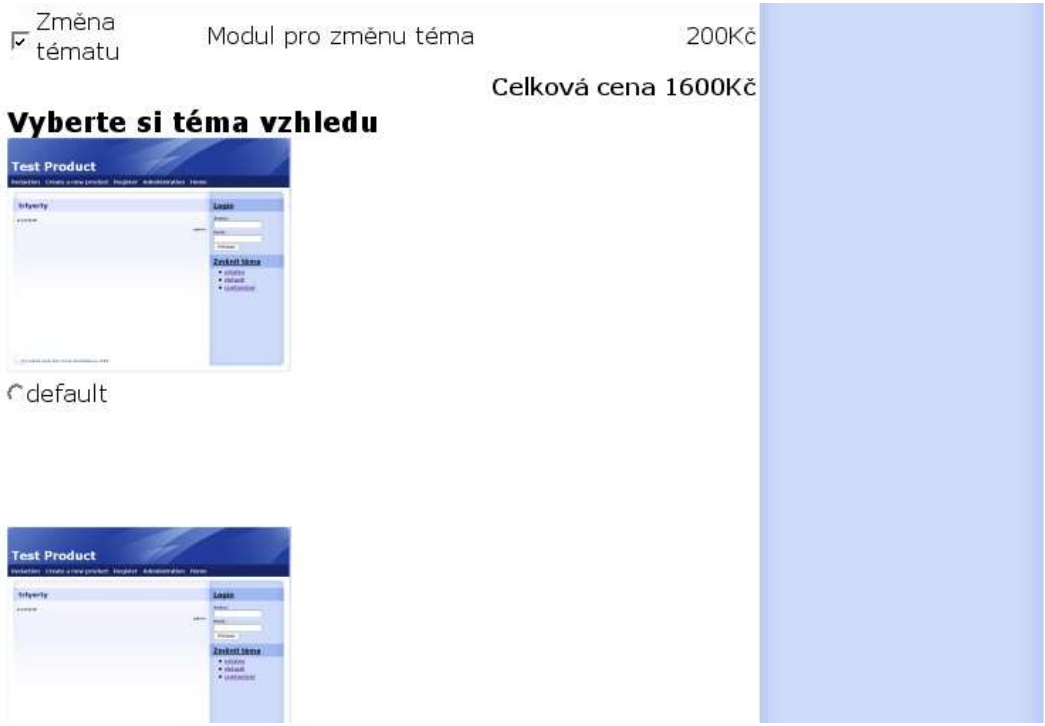
Povýšit verzi stránek je oprávněn pouze administrátor. V sekci „Administrace“ je odkaz „Povýšení verze stránek“, který vede na stránky s výpisem všech dosud vytvořených stránek (viz obrázek A.6). Po výběru stránek, které mají být povýšeny, se v případě bezkonfliktního povýšení stránka obnoví s upravenými verzemi stránek (viz obrázek A.7). Pokud ale nastanou při povyšování konflikty, jsou zobrazeny na následující stránce v textových polích, kde je administrátor může ihned editovat (viz obrázek A.8). Po vyřešení konfliktů se zobrazí původní stránka s upravenými verzemi stránek (viz obrázek A.9).

A.3 Správa stránek

Pro správu stránek se musí administrátor přihlásit a vstoupit do sekce „Administrace“. Zde po kliknutí na odkaz „Zákaznické stránky“ jsou zobrazeny veškeré stránky (viz obrázek A.10) a po kliknutí na název stránky se zobrazí veškeré moduly (viz obrázek A.11), které lze přidávat či odebírat. Při manipulaci s moduly se mění cena pomocí javascriptu.

Obrázek A.1: Úvodní informace

Obrázek A.2: Výběr modulů



Obrázek A.3: Výběr tématu

Obrázek A.4: Zadání domény

Obrázek A.5: Stránka s odkazem vedoucím na vytvořené stránky

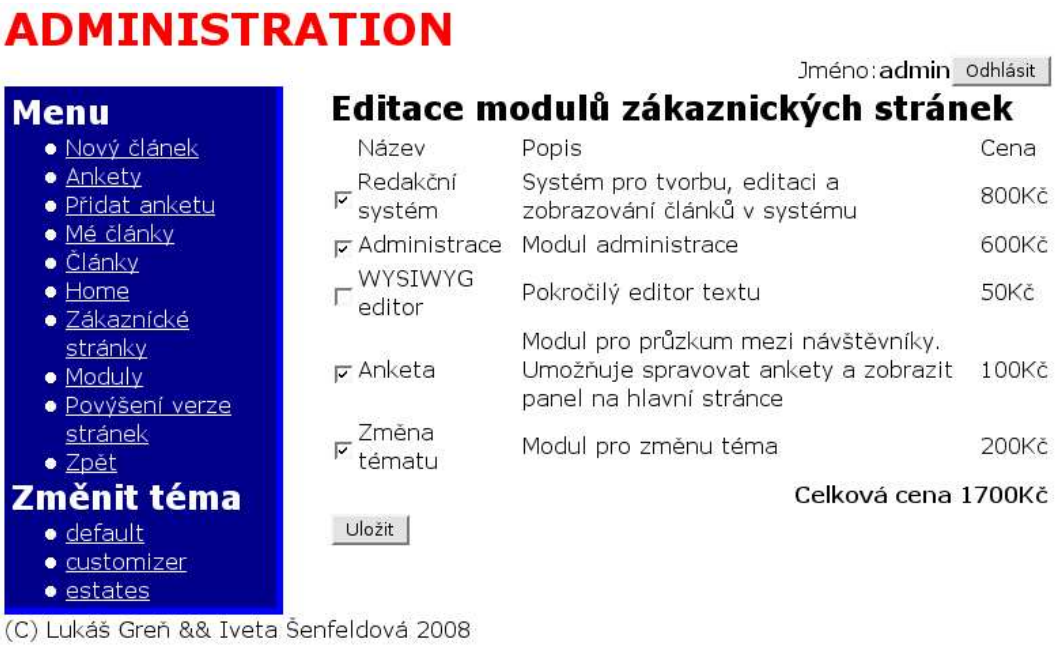
Obrázek A.6: Výpis existujících stránek

Obrázek A.7: Povýšené stránky

Obrázek A.8: Výpis vzniklých konfliktů

Obrázek A.9: Povýšené stránky po vyřešených konfliktech

Obrázek A.10: Výpis všech vytvořených stránek



Obrázek A.11: Výpis všech modulů daných stránek

Literatura

[1] David A. Black: *Ruby For Rails*. Manning publications, 2006. ISBN 1932394699.

[2] Curt Hibbs: *Ruby on Rails - a high-productivity web application framework*. Elektronický text v pdf.

[3] Mohammed J. Kabir: *Apache Server 2*. Computer Press, 2004. ISBN 80-251-0319-6.

[4] WWW stránky: Bazaar. `http://bazaar-vcs.org/`.

[5] WWW stránky: Git: distribuovaná správa revizí. `http://www.root.cz/clanky/git-distribuovana-sprava-revizi/`.

[6] WWW stránky: Mongrel. `http://mongrel.rubyforge.org/`.

[7] WWW stránky: The official yaml web site. `http://www.yaml.org/`.

[8] WWW stránky: Wikipedia lamp. `http://en.wikipedia.org/wiki/LAMP_(software_bundle)`.