VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

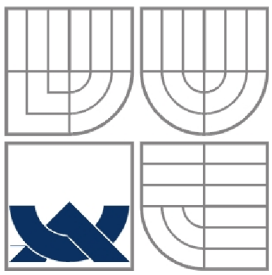
SNADNÁ INSTALACE A KONFIGURACE INFORMAČNÍHO SYSTÉMU PRO SPRÁVU PROJEKTŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

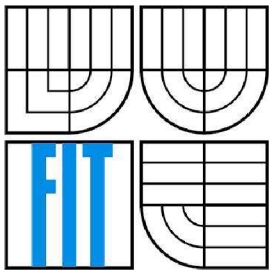
AUTOR PRÁCE
AUTHOR

BC. DUŠAN HOSKOVEC

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SNADNÁ INSTALACE A KONFIGURACE INFORMAČNÍHO SYSTÉMU PRO SPRÁVU PROJEKTŮ

SIMPLE INSTALLATION AND CONFIGURATION OF INFORMATION SYSTEM FOR PROJECT MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. DUŠAN HOSKOVEC

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ALEŠ SMRČKA

BRNO 2009

Abstrakt

Tato diplomová práce pojednává o vývoji nástroje pro snadné vytvoření domácích stránek, které slouží ke správě projektu. Práce je členěna do čtyř klíčových kapitol. První kapitola se zabývá studiem technologií, které se používají pro tvorbu domácích stránek projektů. Druhá kapitola je zaměřena na analýzu požadavků, kladených na informační systém. Ve třetí kapitole nalezneme návrh implementace analyzovaného systému. Čtvrtá kapitola se věnuje implementačním detailům vyvinutého systému.

Abstract

This master's thesis is focused on a development of tool for simple installation and configuration of information system for project management. This piece of work consists of four chapters. First part implies a technology study, used for making project home websites. The second chapter gives an analysis of requirements for an information system. There is a suggestion of implementation of analyzed system in the third chapter. The fourth chapter deals with implementation details of developed system.

Klíčová slova

informační systém, správa projektů, instalace, konfigurace, MoinMoin, Bugzilla, subversion

Keywords

information system, project management, installation, configuration, MoinMoin, Bugzilla, subversion

Citace

Hoskovec Dušan: Snadná instalace a konfigurace informačního systému pro správu projektů, diplomová práce, Brno, FIT VUT v Brně, 2009

Snadná instalace a konfigurace informačního systému pro správu projektů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Aleše Smrčky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Dušan Hoskovec
Datum 26. 5. 2009

Poděkování

Rád bych poděkoval vedoucímu diplomové práce, panu ing. Aleši Smrčkovi, za vstřícnost a dobré rady, které vedly k řešení této diplomové práce.

Obsah

1 Úvod

V současné době je používání počítačů a počítačových programů běžnou součástí každodenního života. Počítače se nachází v mnoha sférách lidského života, aniž si to možná neuvědomujeme. Počítače řídí výrobu aut, lékařské přístroje, předpovídají počasí, podílejí se na řízení raketoplánů, vyhledávají nové látky, analyzují DNA. Čím více je využití počítačů různorodější, tím více je třeba počítačových programů, které se zaměřují na konkrétní problematiku.

Vývoj nových programů je tedy každodenní záležitostí. Vyvíjejí se nové programy, řešící úplně novou oblast problémů. Vylepšují se zastaralé programy, aby pokryly širší záběr problematiky, nebo naopak, aby se hlouběji specializovaly při řešení některého konkrétního problému. V neposlední řadě se udržují programy, které existují.

Během vývoje vyžaduje každý poněkud rozsáhlejší program určitou míru řízení. Tato míra řízení je tím větší, čím dlouhodobější vývoj projektu je předpokládán, čím více lidí se má na projektu podílet a čím vyšší kvality výsledného programu se má docílit. Dnes již existují metody, jak sledovat a plánovat postup vývoje, jak předcházet rizikům, jak sledovat a dodržovat rozpočet. Pro tyto metodiky již vznikla podpora v podobě počítačových programů zabývajících se správou projektu.

Některé z těchto programů se zabývají pouze určitými oblastmi správy projektů. Jedná se například o systémy, které se starají o podporu správy chyb, dále o systémy, které se zabývají správou zdrojových kódů nebo správou procesu vývoje projektu.

Jiné programy zabývající se správou projektů jsou rozsáhlejší a zabývají se více oblastmi problematiky řízení projektů. Takové programy jsou však pro menší projekty příliš komplexní a někdy dokonce zbytečně složité na nastavení.

Tato diplomová práce si klade za cíl vytvořit nástroj, který pokrývá základní požadavky kladené na správu projektu a je jednoduše instalovatelný a konfigurovatelný, přičemž tento nástroj bude složen z několika existujících nástrojů zabývajících se různými částmi správy projektu. Výsledkem této práce je instalační balíček jednoduše instalovatelného a konfigurovatelného informačního systému pro správu projektů.

Diplomová práce je strukturovaná do čtyř hlavních kapitol, kterým předchází tento úvod. První z nich se zabývá studiem stávajících technologií pro správu projektu a popisuje, jak se s takovými technologiemi pracuje.

Druhá kapitola se zaměřuje na analýzu požadavků, které jsou kladeny na výslednou aplikaci. Dále se věnuje výběru vhodných technologií, které budou použity při řešení, výběrem implementačního jazyka a výběrem linuxové distribuce, na níž se bude aplikace vyvíjet.

Třetí kapitola je věnována návrhu výsledné aplikace, zejména pak návrhem tříd objektů, které budou v aplikaci implementovány. V této kapitole jsou také naznačeny vztahy objektů při jejich vzájemné interakci.

Kapitola čtvrtá popisuje výslednou implementaci aplikace, přičemž podrobněji rozebírá některé její části.

Kapitola týkající se stávajících technologií pro správu projektu je částečně přejata ze semestrálního projektu. Tato kapitola je v diplomové práci rozšířena.

2 Studijní fáze

2.1 Technologie pro tvorbu webových stránek

Tento odstavec byl přejat z [1]. V současné době se pro tvorbu statických webových stránek používá v drtivé většině jazyk HTML. Tento jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Jak uvádí Simpson v [2], je v současné době tendence nahradit jazyk HTML jazykem XML (eXtensible Markup Language, česky rozšiřitelný značkovací jazyk), což se do jisté míry daří díky jazyku XHMTL. XHTML je jistým mezistupněm mezi jazykem XML a HTML, neboť z jazyka XML přebírá pravidla stukturování tagů a z jazyka HTML jejich sémantický význam. Přesto však bohužel podpora XML zatím není na takové úrovni, aby mohl být nasazen při vývoji rozsáhlých webových aplikací a tvorbě webových stránek.

Pro oddělení datové části od popisu vzhledu vznikly tabulky kaskádových stylů – CSS. Z čehož plyne, že prostou informaci v podobě HTML kódu můžeme stylizovat do uživatelsky přívětivého tvaru. Další výhodou je, že styly máme definovány na jednom místě a jejich změna se projeví v celém výsledném dokumentu. Bohužel kvůli konkurenčnímu boji mezi předními tvůrci webových prohlížečů není interpretace CSS jednotná, a je tedy nutné kontrolovat výslednou podobu dokumentu ve více druzích internetových prohlížečů.

Tento odstavec byl přejat z [3]. Pokud chceme tvořit dynamické webové stránky, použijeme k tomu například skriptovací jazyk PHP. PHP je jazyk interpretovaný, jeho skripty se začleňují přímo do struktury jazyka HTML a jsou prováděny na straně serveru. Syntaxe kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP se stalo velmi oblíbeným především díky jednoduchosti použití a také díky tomu, že kombinuje vlastnosti více programovacích jazyků, a nechává tak vývojáři částečnou svobodu v syntaxi. Bohužel se při provádění implementace ukázalo, že tato vlastnost má i druhou stánku věci a sice, že když dojde k překlepu nebo zapomenutí uvozujícího znaku při použití proměnné, je obvykle těžké dohledat chybu. S verzí PHP 5 se výrazně zlepšil přístup k objektově orientovanému programování. Od roku 1994 je PHP jedním z nejpoužívanějších způsobů tvorby dynamicky generovaných WWW stránek. Jeho tvůrce (Rasmus Lerdorf) jej vytvořil pro svou osobní potřebu přepsáním z Perlu do jazyka C.

Při zmínce o programovacím jazyku PHP je také nutno se zmínit o některých jeho rozšířeních. Jedním z nich je například funkční a dobře vybavený šablonovací systém Smarty[1]. Tento šablonovací systém má plnit podobnou funkci jako tabulky kaskádových stylů, a sice oddělení aplikační

[1] Domovské stránky projektu Smarty dostupné na: <http://www.smarty.net/>

a zobrazovací logiky. Oproti tabulkám kaskádových stylů nabízí Smarty například možnost validace formulářů a podobných vylepšeních, která mají programátorovi ulehčit práci.

Tento odstavec čerpá z domovských stránek projektu webového serveru Apache2[2]. Skriptovací jazyk PHP je prováděn na straně webového serveru. Nejpoužívanějším webovým serverem je v současné době webový server Apache a jeho nástupce Apache 2, což lze tvdit na základě analýzy uvedené na serveru Netcraft, která je dostupná v [4]. Apache 2 je webový server dostupný zdarma jako open source software. Webové servery všeobecně fungují tak, že naslouchají požadavkům ze sítě a vracejí výsledky dotazu v podobě HTML kódu. Webový server tvoří ve skutečnosti několik procesů, z nichž jeden kontroluje činnost ostatních. Tento řídící proces běží s právy administrátora systému a sám o sobě žádné požadavky nevyřizuje. Ostatní procesy zpravidla nemají taková oprávnění jako proces řídící a svoji činnost oznamují procesu řídícímu. Způsob naslouchání serveru Apache v síti a obsluhy příchozích požadavků řídí jeho konfigurační soubor. Chování serveru je takto možno ovlivnit více jak 150 různými direktivami. Vhodné nastavení těchto parametrů je důležité pro optimální výkonnost serveru.

Tento odstavec čerpá z domovských stránek projektu databázového systému MySQL[3]. Při dalším zkoumání tvorby dynamických stránek zjišťujeme, že mnoho webových stránek pracuje s databázemi. Nejpopulárnější verzí databáze je MySQL. MySQL je databázový systém vytvořený švédskou firmou MYSQL AB. Jeho hlavními autory jsou Michael "Monty" Widenius a David Axmark. Tento systém je dostupný jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Databázi si můžeme představit jako prostor, do kterého se ukládají všechny potřebné údaje. Zpracováním a přístupem údajů v databázi bývá pověřen program, kterému se říká DBMS (Databáze Management System), nebo-li česky SŘBD (Systém Řízení Báze Dat). Celé je to tedy zařízeno tak, že veškeré zpracování, ukládání a získávání dat je prováděno SŘBD programem. Jakákoli aplikace, která používá databázi, k ní vždy přistupuje přes SŘBD program. Naprostá většina dnešních SŘBD je založena na tzv. relačním modelu dat, i když v poslední době je možno sledovat trend posunu od relačních databází k objektovým. V relačním modelu jsou veškerá data uspořádána do databázových tabulek. Každá tabulka zpravidla uchovává údaje o určitém objektu. Databázová tabulka je uspořádána do řádků a sloupců. Relační model klade velký důraz na zachování integrity dat. Zavádí pojmy referenční integrita, cizí klíč, primární klíč, normální tvar apod. Referenční integrita je nástroj databázového stroje, který pomáhá udržovat vztahy v relačně propojených databázových tabulkách. Referenční integrita se definuje cizím klíčem, a to vždy pro dvojici tabulek. Tabulka, v níž je pravidlo uvedeno, se nazývá podřízená tabulka (používá se také anglický termín slave). Tabulka, jejíž jméno je v omezení uvedeno, je nadřízená tabulka (master). Pravidlo referenční integrity vyžaduje, aby každý záznam použitý v podřízené tabulce existoval v nadřízené tabulce. Primární klíč je pole nebo

[2] Domovské stránky webové serveru Apache2 dostupné na: < http://httpd.apache.org/>

[3] Domovské stránky databázového systému MySQL dostupné na: < http://www.mysql.com/>

kombinace polí, jednoznačně identifikující každý záznam v databázové tabulce. Žádné pole, které je součástí primárního klíče, nesmí obsahovat hodnotu `NULL`. Každá tabulka může mít definovaný pouze jeden primární klíč. Cizí klíč je sloupec databázové tabulky, který odkazuje na jiný sloupec (jiné tabulky). Hodnoty takovéhoto sloupce musí být shodné s některou z hodnot v sloupci, ke kterému je klíčem. Vytváří se tak reference – odkaz. MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. MySQL byla od počátku optimalizována především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování a až donedávna nepodporovala pohledy, spouštěče (triggery) a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu – programátorům webových stránek – již poněkud scházet.

2.2 Práce s nástroji pro správu verzí projektu

V současné době existují dva majoritní druhy systémů pro správu verzí projektu. Prvním je Concurrent Version System (dále jen CVS) a druhým je Subversion. Jako první prostudujeme systém CVS. Nejdříve si definujeme pár základních pojmů:

- **Repozitář** – je strukturou, do které jsou ukládány všechny verze souboru. Pro práci s ním stačí znát jeho přesnou adresu.
- **Větve** (branches) – umožňují vytvořit více vývojových verzí programu.
- **Značky** (tags) – slouží k označení souborů, které budou vydány jako verze systému. Každý si pak může nahrát označenou verzi a poté se k ní vrátit.
- **Moduly** – slouží k tomu, aby se často používané části kódu stále neopakovaly.

K práci s CVS nám bude stačit pracovat s příkazovou řádkou programu CVS, neboť práce s grafickými nadstavbami tohoto systému pro správu verzí jsou principielně stejné. Příkaz pro CVS má pevnou strukturu. Vždy se skládá z obecné volby, názvu příkazu, volby příkazu a jména adresáře, modulu či souboru. Obecnými volbami jsou komprese a určení repozitáře. Přepokládejme, že máme CVS nainstalované.

1. Nejdříve si založíme repozitář příkazem: `cvs -d repository_root_directory init`
2. Poté si nadefinuje, že budeme s tímto repozitářem pracovat, což udělám definicí proměnné `CVSROOT` a parametrem `-d`. Například: `export CVSROOT=/usr/local/cvsroot`
3. Do repozitáře naimportujeme náš projekt, přičemž předpokládáme, že se nacházíme v pracovním adresáři projektu: `cvs import projectname projectname start`
4. Nyní si potřebujeme vytvořit kopii souborů z repozitáře, se kterou budeme pracovat: `cvs \ checkout projectname -r1.2` Pokud bychom vynechali část s parametrem `-r`, nakopírujeme si aktuální revizi.

5. Informaci o kopii, s kterou právě pracujeme, získáme příkazem: `cvs status`
6. Příkazem: `cvs update` získáme aktuální kopii souborů na serveru. Během přepisování souborů pracovní kopie jsme informováni o průběhu této akce.
7. Konečně příkazem: `cvs commit -m 'Komentar ke zmenam'` nahrajeme námi upravené soubory na server. Parametr `-m` označuje komentář, identifikující změny, které byly na souboru provedeny.

Zde byl nastíněn pouze základní způsob práce s CVS. Více podrobností lze zjistit v dokumentaci projektu [5], ze které byl přejat i tento návod.

Nyní nastíníme práci s verzovacím systémem subversion. K již zmíněným termínům definujeme termín konflikt verzí.

- **Konflikt verzí** – je stav, k němuž dojde, pokud dva nebo více vývojářů commitovalo (ukládalo svou verzi) soubor, na němž oba pracovali, do úložiště.

Při práci se Subversion si podobně jako u CVS vystačíme si s příkazovým řádkem.

1. Nejdříve si vytvořme vlastní repozitář příkazem: `svnadmin create \ /home/user/repository --fs-type fsfs`. Takto vytvořený repozitář bude ukládat informace do souborů na serveru místo do databáze.
2. Dále naimportujeme náš projekt do vytvořeného repozitáře: `svn import \ /home/user/projekt/ file:///home/user/repository / -m "Import projektu"`
3. Provedeme stažení souborů z repozitáře do vlastního adresáře, kde budeme pracovat na úpravách: `svn co file:///home/user/repository/ /home/user/pracovni/`
4. Nyní vykonáme příkaz pro commit námi upraveného souboru do repozitáře: `svn commit \ /home/user/pracovni -m "Komentar"`

Stejně jako u CVS, slouží příkaz `svn update` ke stažení aktuální kopie souborů do adresáře, kde budeme pracovat s vlastní pracovní kopií.

Při popisu práce s verzovacím systémem subversion se musíme také zmínit o dvou význačných vlastnostech tohoto systému. Jedná se o takzvané větvení a značkování.

Větvení, tedy vytvoření takzvané větve projektu, použijeme v případě, že chceme vytvořit mírně nebo zcela odlišnou verzi projektu, která je však založena na stejné výchozí linii. Klasickým případem je vytvoření dokumentace pro jiné oddělení pracovníků, pro které chceme vytvořit dokumentaci, která poukazuje na jiné aspekty produktu.

K vytvoření nové větvu projektu se používá příkaz:

```
svn http://svn.example.com/repos/calc/trunk \
http://svn.example.com/repos/calc/branches/my-calc-branch \
-m "Vyvori novou verzi s mou kalkulackou"
```

Jak již napovídá samotná syntaxe příkazu, vykoná tento příkaz zkopírování všech částí projektu spravovaných v svn repositáři do nového adresáře v adresáři branches. Práce s větvemi produktu je pak velice podobná obvyklé práci s produktem, jen je udána úplná cesta k větvi, tedy do adresáře

branches, a nikoliv do adresáře trunk. Z výše zmíněného důvodu se důrazně doporučuje udržovat strukturu repositáře s následujícími adresáři v kořeni repositáře: trunk, branches, tags.

Dalším nutným příkazem pro práci s větvemi svn repositáře je příkaz `switch`. Příkaz `switch` se používá k aktualizaci stávající pracovní kopie tak, aby odrážela změny v jiném adresáři repositáře. V našem případě se bude jednat o repositář námi vytvořené větve projektu. Pro náš nedávno zmíněný příklad s verzí naší kalkulačky, může být použití tohoto příkazu následovné. Nejdříve si otevřeme adresář s naší pracovní kopií, například takto: `cd my-calc-branch`, my-calc-branch je název adresáře s naší pracovní kopií. Dále následuje příkaz pro provázání adresáře v repositáři s adresářem v pracovní kopii: `svn switch http://svn.example.com/repos/calc/branches/my-calc-branch`.

Neméně potřebným příkazem pro práci s větvemi projektu může být příkaz `merge`, který slouží k synchronizaci mezi jednotlivými větvemi projektu. Příkaz `merge` použijeme tak, že jej zavoláme z naší pracovní kopie na tu část repositáře subversion, kterou chceme synchronizovat. Příkaz pro synchonizaci s hlavní větví projektu vypadá například takto: `svn merge \ http://svn.example.com/repos/calc/trunk`. Tento příkaz zajistí, že pracovní kopie přijme změny z hlavní větve projektu. Pokud však chceme námi vyvíjenou větev reintegrovat zpět do hlavní větve, použijeme parametru `reintegrate`. Příkaz pro zpětnou integraci poté vypadá například takto: `svn merge –reintegrate http://svn.example.com/repos/calc/branches/my-calc-branch`. Tento příkaz musí být následován příkazem `svn commit –m "Sloučí my-calc-branch back zpět do adresáře trunk"`. Pokud by tento příkaz nebyl zadán, nebylo by sloučení kompletní.

Příkaz `tag` slouží k zachycení současného stavu adresáře spravovaného subversion repositářem. Příkaz `tag` může vypadat takto:

```
svn copy http://svn.example.com/repos/calc/trunk \
http://svn.example.com/repos/calc/tags/release-1.0\
 –m “Tagování projektu”.
```

Značkování je tedy v praxi pouhým kopírováním adresáře v subversion repositáři.

Podrobnější popis můžeme získat například v dokumentaci pro subversion [6], z níž byl přejat i tento návod.

2.3 Problematika řízení projektů

Tato podkapitola je přejata z [7]. Pokud se chceme zabývat problematikou řízení, je nutné nejdříve specifikovat, co rozumíme termínem projekt a co je definováno termínem řízení projektu.

Pojem projekt má více definic, uvedeme si některé z nich. První definice říká, že projekt je řízený proces aplikace úkolů a zdrojů s definovaným cílem v určeném časovém rámci. Další definice tvrdí, že projekt je dočasně vyvinuté úsilí, vynaložené na vytvoření jedinečného produktu nebo

služby. Projekt může být i síť činností mající formální začátek a konec, přidělené zdroje a směřující k vytvoření určitého produktu. Často má také stanoven rozpočet, v rámci kterého musí být stanovených cílů dosaženo. S vytvořením tohoto produktu je vždy spojeno určité riziko. Projekt může být také jedinečný proces sestávající se z řady koordinovaných a řazených činností s daty zahájení a ukončení, prováděných k dosažení cíle, který vyhovuje specifickým požadavkům, včetně omezení daným časem, náklady a zdroji[4].

Lze tedy tvrdit, že projekt je unikátní dílo, které se v některém ohledu odlišuje od podobných projektů a můžeme tvrdit, že aktivity projektu se týkají alespoň dvou osob a vyžadují více jak dva týdny úsilí. Dále lze tvrdit, že budou vyžadovány nové postupy nebo nové technologie a také, že rozpočet je těsný. Projekt má typicky více úkolů a předpokládá se, že tyto úkoly jsou závislé na dokončení jiných úkolů. Také se předpokládá, že vývoj produktu je několikafázový a tyto fáze je nutno koordinovat. V neposlední řadě je nutno dodat produkt v požadované kvalitě.

Nyní definujeme termín řízení projektů. Řízením projektu se rozumí uplatnění znalostí, dovedností, nástrojů a technik v projektových činnostech s cílem splnit nebo překročit potřeby zájmových skupin a jejich očekávání od projektu. Plnění nebo překračování těchto potřeb a očekávání vždy vyžaduje vyrovnávat proti sobě stojící požadavky mezi rozsahem prací, časem, náklady a kvalitou, zájmovými skupinami s různými potřebami a očekáváními a stanovenými a nestanovanými požadavky. Řízení projektu zahrnuje tedy plánování, organizování, sledování a řízení všech hledisek projektu v nepřetržitém procesu tak, aby se dosáhly cíle projektu.

Každý projekt prochází několika fázemi. Rozpoznáváme zejména tyto fáze projektu:

- **definování** – v této fázi se analyzují požadavky, studuje proveditelnost, vyvíjí se první scénáře, analyzují se přínosy a náklady, stanovují se cíle a vyhodnocují se alternativy.
- **plánování** – je fází projektu, kdy se identifikují úkoly a kritické aktivity, odhaduje se čas a začíná se s náborem personálu.
- **organizování** – tato fáze se zaobírá sestavováním týmu nebo týmů, dokončováním definice posloupnosti úkolů a jejich přiřazováním týmům a také se zde stanovují kontrolní nástroje.
- **provádění/kontrola** – v této fázi se posuzuje stav projektu, napravují se nalezené chyby, dochází zde k řízení změn, sepisují se zprávy o postupu, sleduje se rozvrh, rozpočet a kontroluje se kvalita.
- **uzavření** – je poslední fází projektu, kdy dochází k převzetí projektu zákazníkem, předání dokumentace, zabezpečením dodávky, závěrečnému bilancování a udržování projektu.

Pří řízení větších projektů se můžeme setkat s problémy spojených s velikostí zásahu do rutinní práce uživatele a nutnou mírou jeho přizpůsobení se, dále s problémy spojenými s prostředím, ve kterém je projekt realizován, například kdy projekt nemá dostatečný počet vývojářů s požadovanými znalostmi. Také se můžeme setkat s problémy technického rázu, kdy jsou technické požadavky na

[4] ČSN ISO 10006: Management jakosti – Směrnice jakosti v managementu projektu. 1999.

splnění termínu velmi vysoké, například výsledný software má být napsán v unikátním programovacím jazyce, ke kterému navíc neexistuje dokumentace. Dalším problémem může být problém věcného rámce, kdy se nedodržuje rozpočet projektu. A v neposlední řadě se můžeme setkat i s problém externí závislosti, který nastává v momentě, kdy externí dodavatel nedodal objednávku včas.

V současné době však existují techniky a nástroje, jak tyto problémy řešit nebo alespoň předvídat.

3 Požadavky kladené na systém a jejich analýza

3.1 Požadavky kladené na systém

3.1.1 Neformální požadavky

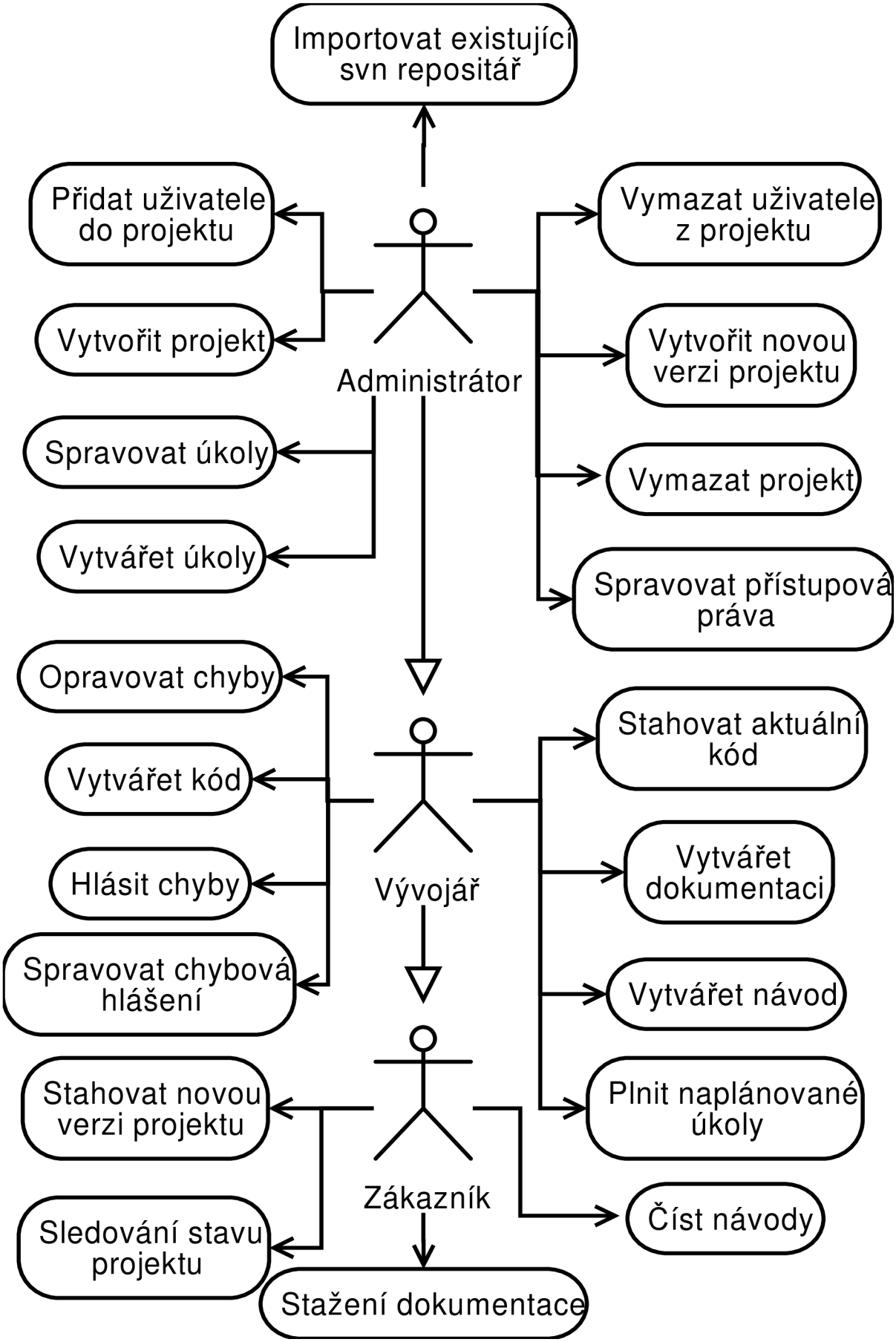
Systém by měl být navržený pro menší až střední skupiny programátorů pracujících na menším až středním projektu, kde je potřeba pouze nízká až střední úroveň řízení projektu. Administrátor projektu vytváří nový projekt, na kterém pak pracují vývojáři. Vývojáři si stahují ze systému aktuální verzi zdrojových kódů projektu a nahrávají své soubory se zdrojovými kódy. Vývojáři do systému vkládají návody, jak použít jimi vytvořené zdrojové kódy. Vývojáři do systému vkládají dokumentaci. Vývojáři čtou a upravují dokumentaci a návody. Vývojáři a administrátor mají možnost debatovat o směru a stavu vývoje projektu a mohou jeho stav plánovat. Vývojáři zde řeší vzniklé problémy a budují si bázi znalostí, které využijí nebo využili při řešení projektu. Administrátor může být totožný s vlastníkem projektu. Systém bude poskytovat podporu pro oznamování chyb, kdy bude oznamovat o jakou chybu se jedná, kdy a kde vznikla a kdo ji oznámil, zda je řešena a kým. Systém musí podporovat správu více projektů, každý projekt má svého administrátora a své vývojáře a je zcela nezávislý na ostatních projektech. Administrátor domovských stránek projektu konfiguruje systém, tak aby co nejlépe odpovídal požadavkům většiny vývojářů a ostatních uživatelů, kteří mají mít přístup do systému. Systém musí být zabezpečený tak, aby k němu neměl přístup neoprávněný uživatel, zejména pak ke zdrojovým částem vyvíjeného projektu.

3.1.2 Funkční požadavky

- Informační systém musí být jednoduše instalovatelný a konfigurovatelný.
- Informační systém slouží pro tvorbu domovských stránek projektů.
- Informační systém zajišťuje správu verzí.
- Informační systém zajišťuje hlášení chyb.
- Informační systém zajišťuje podporu pro správu a tvorbu dokumentace.
- Informační systém zajišťuje podporu budování báze znalostí.
- Informační systém zajišťuje podporu řízení projektu.

3.1.3 Nefunkční požadavky

- Informační systém musí běžet v prostředí operačního systému Linux.
- Se systémem může v jedné chvíli pracovat více uživatelů.
- Systém musí podporovat více jazyků.
- K výslednému informačnímu systému bude sepsána dokumentace a jednoduchý uživatelský manuál.
- Uživatelské rozhraní bude jednoduché a přehledné.
- Informační systém kompletuje existující řešení.



Obrázek 3-1: Diagram případů použití

3.2 Diagram případů použití

V informačním systému vystupují tři aktéři:

- **Vývojář** – za vývojáře považujeme každého programátora, designéra, analytika, který se podílí na vývoji projektu.
- **Administrátor** – je osoba, která projekt založila. Administrátor je osobou, která se stará o správu uživatelů, správu přístupu k jednotlivým částem projektu. Dále se stará o domovské stránky projektu. Administrátor může být přímo vlastník projektu nebo jakýkoliv vývojář.
- **Zákazník** – je aktér, který zastupuje všechny ostatní uživatele, které lze považovat za potencionální zákazníky projektu. Pokud se tedy jedná například o projekt distribuovaný pod licencí freeware, může být zákazník téměř kdokoliv pohybující se na internetu.

V analyzovaném informačním systému rozpoznáváme tyto případy použití:

- **Spravovat přístupová práva** – tento případ použití zahrnuje změnu přístupových práv uživatelů k přístupu jednotlivých částí systému. Zejména se jedná a manuální nastavení přístupu k jednotlivým domovským stránkám projektu. Přístupová práva mají výchozí nastavení ihned po vytvoření nového projektu.
- **Vytvořit projekt** – tento případ použití je vždy použit jako první. Tento případ použití zahrnuje vytvoření nového projektu v informační systému, nastavení přístupových práv k jednotlivým částem informačního systému a nastavení pracovního prostředí podle potřeb osob, která jej použijí. Každý projekt je samostatný a nezávislý na ostatních projektech, jež informační systém spravuje.
- **Vymazat projekt** – tento případ použití zahrnuje smazání projektu včetně všech jeho součástí. Za součást projektu považujeme také všechny uživatelské účty příslušející k danému projektu, taktéž všechny zdrojové kódy projektu.
- **Přidat uživatele do projektu** – umožňuje přidání nového uživatelského účtu do projektu, může se jednat o libovolný uživatelský účet.
- **Smazat uživatele z projektu** – je opačným případem k přidání uživatele do projektu. V tomto případě bude libovolný uživatelský účet smazán z projektu.
- **Vytvářet úkoly** – je případem použití zahrnujícím vkládání nových úkolů pro uživatele informačního systému.
- **Spravovat úkoly** – tento případ užití zahrnuje aktualizování, změny a případně i vymazání jednotlivých případů použití. Taktéž zahrnuje přiřazování zdrojů jednotlivým úkolům, jejich odebírání a přiřazování osob, jež se mají daném úkolu podílet.

- **Importovat existující svn repositáře** – je případem použití, který může a nemusí být použit. Jedná se, jak název napovídá, o importaci existujícího svn repositáře do projektu. Tento případ je použit pokud chceme začít spravovat již existující zdrojový kód.
- **Vytvořit novou verzi projektu** – tento případ použití vyznačuje, že administrátor je schopen jediným příkazem připravit z určené verze zdrojových kódů novou verzi projektu a zveřejnit ji na stránkách projektu.
- **Vytvářet kód –** je případem, kdy vývojář vyvájí kód projektu a později jej ukládá na server projektu.
- **Strahovat aktuální kód** – je případem použití, kdy vývojář chce pracovat s aktuální verzí zdrojových kódů a musí si je tedy opatřit stažením ze severu.
- **Hlásit chyby** – je případem použití, kdy vývojář, který zjistil chybu ve zdrojovém kódu má možnost ji nahlásit do systému, kde s ní bude možno pracovat. Taktéž zahrnuje případ, že chyba je následně někomu přidělena na vyřešení.
- **Opravovat chyby** – tento případ použití indikuje, že vývojář opraví chybu, která byla předtím nalezena a jemu přidělena, hotovou opravu nahlásí do systému.
- **Spravovat chybová hlášení** – je případem použití, kdy osoba, jež opravenou chybu zadala, zkontroluje, zda byla chyba opravena správným způsobem. V případě, že oprava byla správná, je opravování této chyby uzavřeno, v opačném případě je chyba znovu otevřena a celý proces opravování chyby se opakuje.
- **Vytvářet návod** – je případem použití, kdy vývojář má možnost do informačního systému psát návod k použití vyvíjeného projektu. Taktéž má možnost tento návod upravovat.
- **Číst návody** – je případem použití souvisejícím s vytvářením návodu. S tím rozdílem, že číst návody může i aktér zákazník, narozdíl od jeho psaní.
- **Plnit plánované úkoly** – je případem použití, jež indikuje stav, kdy vývojář dokončil úkol a má možnost oznámit tuto skutečnost systému.
- **Vytvářet dokumentaci** – je dalším případem použití aktéra vývojář. Znamená, že vývojář má možnost on-line vytvářet dokumentaci systému, případně zaznaménavat si poznámky ohledně své práci na vývoji projektu.
- **Stahovat novou verzi projektu** – je případ použití indikující, že libovolný aktér zákazník má možnost si z informačního systému projektu stáhnout novou zveřejněnou verzi projektu.
- **Stažení dokumentace** - tento případ použití poukazuje na skutečnost, že aktér zákazník má mít možnost si z informačního systému stáhnout aktuální verzi dokumentace projektu.
- **Sledování stavu projektu** – je případem použití, který vyznačuje možnost sledování průběžného stavu projektu aktérem zákazník. Jedná se zejména o plánování sledování časového plánu projektu a sledování modifikacího vyvíjeného projektu.

3.2.1 Seznam akcí aktéra administrátor

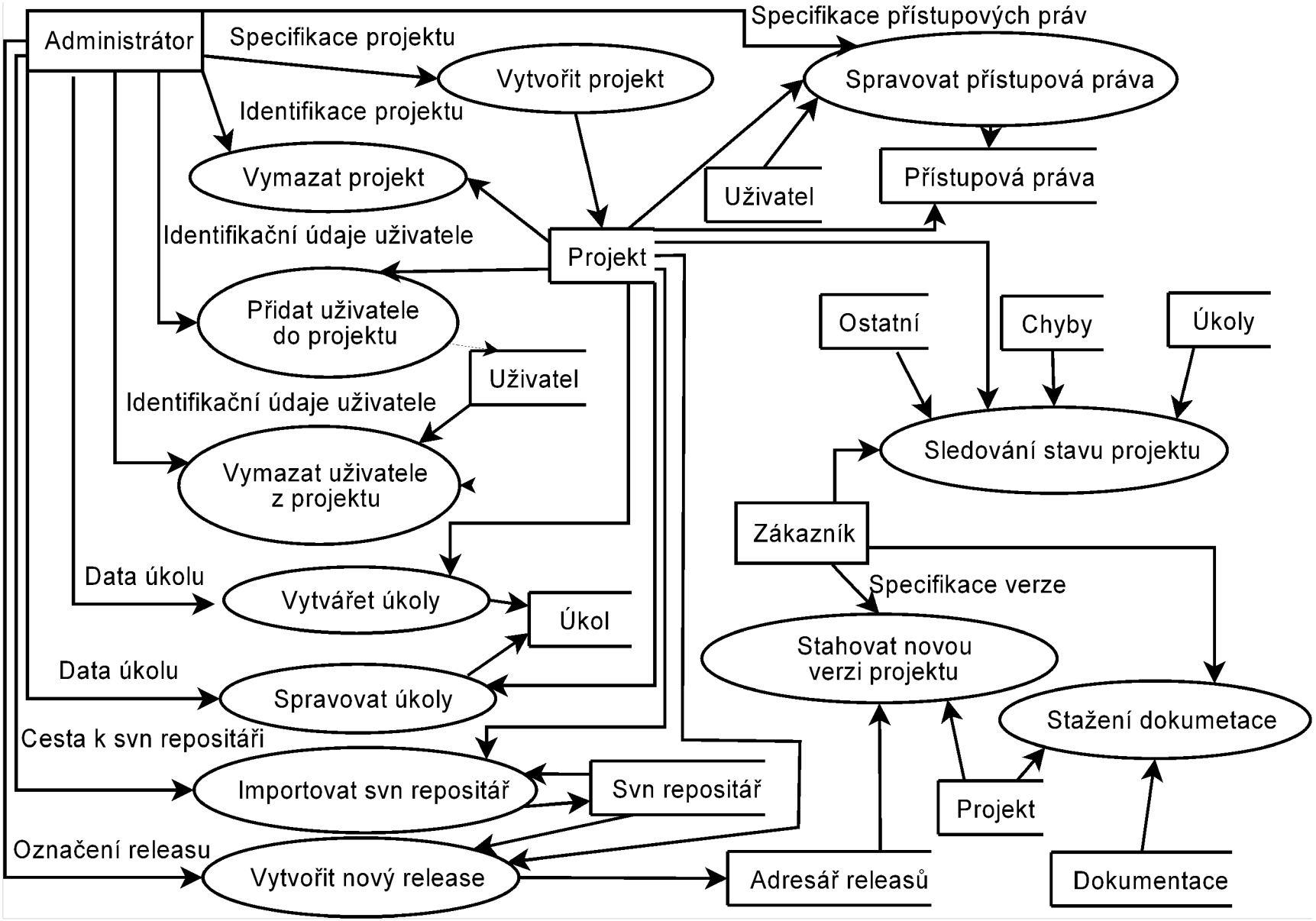
- Importovat existující svn repositáře
- Přidat uživatele do projektu
- Vymazat uživatele z projektu
- Vytvořit projekt
- Vymazat projekt
- Vytvářet úkoly
- Spravovat úkoly
- Spravovat přístupová práva
- Vytvořit novou verzi projektu
- Všechny akce aktéra vývojář

3.2.2 Seznam akcí aktéra vývojář

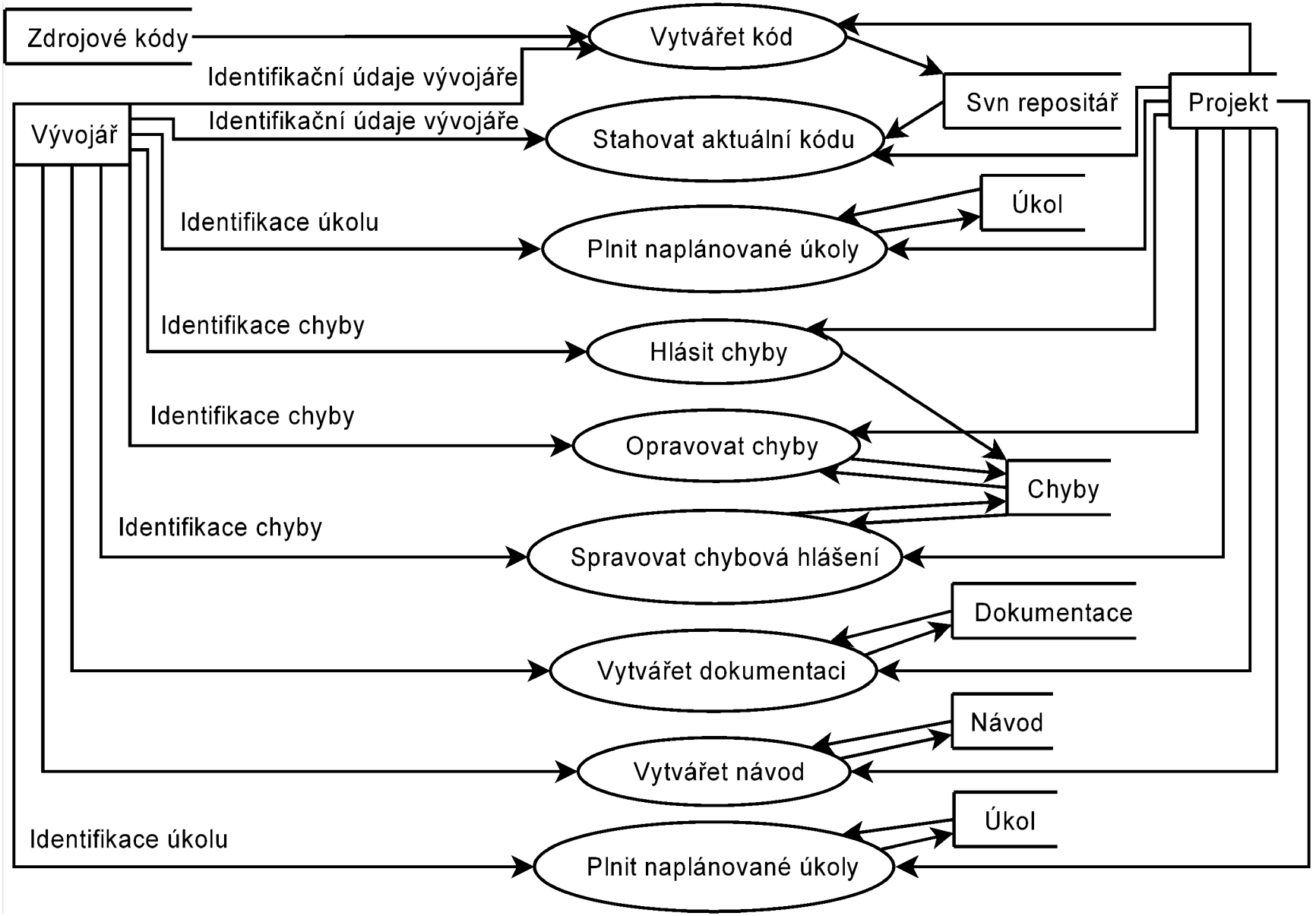
- Všechny akce aktéra zákazník
- Opravovat chyby
- Vytvářet kód
- Stahovat aktuální kód
- Vytvářet návod
- Opravovat chyby
- Hlásit chyby
- Spravovat chybová hlášení
- Plnit plánované úkoly

3.2.3 Seznam akcí aktéra zákazník

- Číst návody
- Stahovat novou verzi projektu
- Stažení dokumentace
- Sledování stavu projektu



Obrázek 3-2: Diagram datových toků - 1. část



Obrázek 3-3: Diagram datových toků - 2. část

3.3 Analýza funkčních a nefunkčních požadavků

Z analýzy funkčních požadavků vyplývá, že bude vhodné do informačního systému zabudovat technologii pro správu souborů, které se v čase mění. Analyzujeme tedy systémy pro správu verzí. Dále bude vhodné zabudovat do systému software pro tvorbu a manipulaci s texty. Analyzujeme tedy wiki systémy, které umožňují snadné vytváření, upravování a distibuci textů. Pro oznamování chyb analyzujeme existující bug-tracking systémy.

3.3.1 Analýza současných systémů pro správu verzí

Obsah této podkapitoly byl přejat z internetové encyklopedie wikipedia[5].

CVS – Concurrental Version System je systémem s otevřeným kódem. Spravuje několik skupin souborů sdružených do repositářů, kde každý repositář má vlastní řízení přístupu a je dělen na moduly, které jsou reprezentovány stromovou strukturou. Repositář je pak celý uložen v souborovém systému serveru. Změny jsou sledovány a uchovávány, takže v každém bodě je možno se vrátit k první verzi projektu. Skupinu souborů lze označit nálepkou, a tím pak vydávat samostatné verze projektu. V kterémkoliv okamžiku je možné vytvořit novou větev a tu poté rozvíjet samostatně. Do tohoto systému lze doinstalovat podporu na webový server apache2 zvanou viewVC [8]. Po této úpravě lze repozitář procházet online.

Subversion – je systém vycházející ze staršího CVS, zachovává podobný způsob i styl práce a odstraňuje nedostatky CVS, kterým je například nemožnost přesunu či kopírování adresářů nebo časová a prostorová náročnost větvení, tagování a podobně. K Subversion systému existuje dokumentace zvaná Version Control with Subversion [6], která je volně dostupná. Jedná se o systém typu klient server, kdy na serveru (v centrálním úložišti) existují repositáře. K repositářům lze přistupovat jak lokálně, tak pomocí protokolu svn. K tomuto systému lze doinstalovat podpora na webový server apache2 zvaná viewVC. Repozitář pak lze procházet online. Nebo lze řešit podporu zobrazování obsahu repositáře implementovat i pomocí webDAV [9]. Systém SVN je možno provozovat společně s SSH zabezpečením.

Mercurial – je multi platformní distibuovaný systém správy verzí. Tento systém je převážně implementován v jazyce python. Mercurial je primárně programem založeným na příkazové řádce. Systém Mercurial je vysoce výkonným, škálovatelným, decentralizovaným, plně distribuovaným systémem, který je schopen pracovat jak s textovými, tak s binárními soubory. Podobně jako subversion umožňuje větvení a značkování. Systému Mercurial využívají v současnosti významné projekty, například Mozilla nebo OpenSolaris.

[5] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

3.3.2 Analýza současných wiki systémů

Wiki je označení webů, umožňující uživatelům přidávat a upravovat obsah, podobně jako v internetových diskusích. Slabinou všech wiki systémů je nedostatek ochrany před trvalými následky způsobených vandaly. Mnoho wiki systému se vyhýbá povinným registračním procedurám. Existuje možnost blokovat IP adresu nebo login uživatele, avšak ani tato ochrana není zcela spolehlivá. Nejlepší možností se jeví přepnutí databáze do read-only módu, nebo povolit zápis do wiki pouze zavedeným uživatelům. Škody způsobené vandaly lze poměrně snadno a rychle napravit. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[6].

MediaWiki – je svobodným softwarem pro provoz wiki. Je snadno rozšiřitelná pomocí modulů. Lze ji provozovat v mnoha jazycích. MediaWiki dokáže vykreslovat matematické vzorce pomocí systému Latex. MediaWiki nemá žádnou nativní WYSIWYG podporu. MediaWiki je přizpůsobitelná, což znamená, že si lze jednoduše změnit předpisy stylů. Je závislá na webovém serveru (například apache2), PHP verze 5.1 a vyšší a databázového serveru MySQL 4.0 a vyšší, nebo Postgre SQL 8.1 a nejnovější s plpgsql a tsearch2. Tato verze wiki je široce rozšířena. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[6].

MoinMoin – další wiki systém, který je však závislý pouze na jazyce Python a lze jej tedy provozovat na webovém serveru podporujícím python. Není tedy zapotřebí mít nainstalované PHP nebo jakoukoliv databázi. Může být umístěna na webovém serveru Apache2 s nainstalovaným rozšířením mod_python verze alespoň 3.1.3. Další příjemnou vlastností MoinMoin je, že podporuje běh více wiki systému na jednom serveru. Dále má MoinMoin WYSIWYG editor. MoinMoin je sice navržena pro menší počet stránek, avšak tato kapacita se počítá v řádu tisíců, což je dostatečné. Tato verze wiki je zabudována například v domovských stránkách softwaru Apache, Debian, Fedora, či Ubuntu. Tento odstavec přejímá informace z domovských stránek projektu, dostupných na [10].

DokuWiki – je jednoduše použitelný wiki systém, který je určen převážně pro tvorbu strukturovaných dokumentů, především dokumentace. Je navržen pro menší společnosti a jednotlivce. Data jsou ukládána v textových souborech na serveru. Ke svému běhu potřebuje webový server (preferován je webový server Apache), dále je vyžadováno PHP 4.3.10, avšak doporučováno je PHP verze 5 a vyšší. K instalaci není vyžadována práva roota. DokuWiki nemá WYSIWYG editor. Tento wiki systém podporuje vícejazyčnost. Tento odstavec přejímá informace z domovských stránek projektu, dostupných na [11].

TWiki – je mocná, bezpečná, flexibilní platforma, která je webově založena. Ke svému provozu vyžaduje systém TWiki webový server s interpretem pro jazyk Perl verze 5, Revision Control Systém verze 5.7 a vyšší, dále také program diff a cron. Tento odstavec přejímá informace z domovských stránek projektu, které jsou dostupné na [12].

---

[6] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

dotProject – je profesionální groupware systémem pro správu projektů obsahující moduly pro společnosti, projekty, úkoly implementovanými Ganttovými diagramy, fóra, kalendář, a další. Dokáže zpracovávat úlohy, společenskou hierarchii, seznam věcí, které je potřeba vykonat k dosažení cíle a zdroje, mimo jiné také obsahuje kalendář. Jedná se tedy o velice komplexní systém se širokou podporou. Manuál k tomuto systém je vhodně zpracován a existují i zkušební verze. Tento systém využívá jazyka PHP a databáze MySQL. Celý systém je kompletně přeložen do češtiny. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[7].

3.3.3 Analýza současných bug tracking systémů

Bugzilla – je webovou aplikací pro sledování chyb. Pro svou činnost vyžaduje webový server (například Apache) s podporou pro interpretaci skriptů napsaných v programovacím jazyce Perl (alespoň verze 5) a databázový systém MySQL nebo PostgreSQL. Chybu může vložit kdokoliv. Chyba je pak přiřazena vývojáři. Tento systém pro oznamování chyb lze použít nejen pro oznamování chyb, ale také jako návrhy pro vylepšení a požadavky nových funkcí. Pro Bugzillu existuje rozšíření do MediaWiki – BugSquish [13] – které zkontroluje instalaci Bugzilly a do MediaWiki přidá informaci o uzavřené chybě. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[7].

Mantis Bug Tracker – je populárním webově založeným bug trackingovým systémem, který je napsán v programovacím jazyce PHP a ke svému běhu vyžaduje přítomnost webového serveru (například Apache 2.0.54) s interpretem jazyka PHP (alespoň verze 4.3.11), databázový systém MySQL (minimální verze 4.0) nebo PostgreSQL (verze 8.0). Tento odstavec čerpá z domovských stránek projektu dostupných na [14].

phpBugTracker – je systém pro bug tracking, který má stejnou funkčnost jako Bugzilla. Oproti Bugzille je tento systém navržen tak, aby se oddělily presentační, aplikační a databázové vrstvy. Pro svůj běh vyžaduje phpBugTracker webový server s interpretem jazyka PHP a modulem PEAR a databázovým systémem MySQL, PostgreSQL nebo Oracle. Tento odstavec čerpá z domovských stránek projektu dostupných na [15].

Scarab – je vysoce konfigurovatelný issue tracking systém. Výhodou i nevýhodou tohoto systému je právě jeho vysoká konfiguratelnost. Na jedné straně mu tato vlasnost dovoluje být nasazen na libovlný projekt, ať již technického, nebo netechnického rázu. Na druhé straně je tato vysoká konfigurovatelnost nevýhodou, neboť zde nejsou pevně určena pravidla, co se má stát, když se stav chyby změní z otevřené na uzavřenou, či jiný. Tento systém umožňuje exportovat a importovat chyby z XML souboru. Systém Scarab ke svému běhu potřebu platformu Java a jednu z následujících databází: MySql, PostgeSQL, Oracle. Tento odstavec čerpá z domovských stránek projektu dostupných na [16].

[7] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

3.3.4 Vybrané existující systémy

Ze dvou nabízených systémů pro správu verzí je vhodnější systém Subversion, neboť odstraňuje některé nedostatky CVS, dále je možno jej provozovat přes SSH a v neposlední řadě s ním lze pracovat přes webové rozhraní, například prostřednictvím modulu serveru apache2 WebDAV (Web-based distributed Authoring and Versioning).

Z analyzovaných wiki systémů musíme zamítnout systém dotProject, protože jeho podpora je příliš komplexní a neodpovídá nefunkčnímu požadavku jednoduchosti. Ze zbylých tří wiki systému je nejvhodnějším kandidátem wiki systém MoinMoin. Systém MoinMoin nemá sice WYSIWYG editor, ale zato pro jeho běh není potřeba žádného databázového systému, pouze webového serveru s nainstalovaným interpretem jazyka python. Hlavním důvodem pro výběr systému MoinMoin je ta skutečnost, že je schopen na jednom serveru podporovat paralelně více wiki systémů.

Při volbě bugtrackingového systému vyloučíme systém Scarab, neboť jeho vysoká konfigurovatelnost je z hlediska nefunkčního požadavku jednoduchosti nevyhovující. Dále vyřadíme phpBugTracker, neboť oproti ostatním poskytuje méně funkcionality. Ze zbylých dvou bug trackingových systémů zvolíme systém Bugzilla, neboť se jedná o bug trackingový systém, který je používán širokou veřejností.

Tabulka 3-1: Tabulka přístupových práv odvozená ze znalosti diagramu datových toků, diagramu případů použití a znalosti využitého softwaru

AKCE	ROLE	PŘIDÁNÍ	ČTENÍ	ZMĚNA	RUŠENÍ
Příspěvek ve wiki	Vývojář	Ano	Ano	Ano	Autor/moderátor
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Ano	Ano	Autor	Autor
Soubor v SVN	Vývojář	Ano	Ano	Ano	Ano
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Ne	Je možné	Ne	Ne
Dokumentace	Vývojář	Ano	Ano	Ano	Ano
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Ne	Ano	Ne	Ne
Návod „how to“	Vývojář	Ano	Ano	Autor/moderátor	Autor/moderátor
	Administrátor	Ano	Ano	Autor/moderátor	Autor/moderátor
	Zákazník	Ano	Ano	Ano	Autor/moderátor
Budovat knowledge base	Vývojář	Ano	Ano	Ano	Ano
	Administrátor	Ano	Ano	Autor/moderátor	Autor/moderátor
	Zákazník	Ano	Ano	Autor/moderátor	Ne

Oznamovat chyby	Vývojář	Ano	Ano	Ano	Ano
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Je možné	Ano	Ne	Ne
Spravovat přístupová práva	Vývojář	Ne	Ne	Ne	Ne
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Ne	Ne	Ne	Ne
Konfigurovat pracovní prostředí	Vývojář	Ne	Ne	Ne	Ne
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Ne	Ne	Ne	Ne
Vytvářet nový projekt	Vývojář	Ne	Ne	Ne	Ne
	Administrátor	Ano	Ano	Ano	Ano
	Zákazník	Ne	Ne	Ne	Ne

4 Návrh aplikace založený na analýze požadavků

4.1 Volba implementačního jazyka

Volba implementačního jazyka je pro vývoj aplikace bws velice význačná, neboť nevhodně zvolený implementační jazyk může vývoj velice znesnadnit, na druhou stranu pak volba vhodného implementačního jazyka může přinést při implementaci mnohé výhody.

Před samotným výběrem jazyka, v němž bude aplikace implementována, si musíme uvědomit, jaké obraty se budou při implementaci vyskytovat nejčastěji a také jaký programovací princip je nám bližší (objektový nebo procedurální). V potaz bychom měli také vzít další vlastnosti jazyka, například, zda se vytváří nějaký mezikód, zda lze předpokládat jeho nasazení na více platformách, složitost syntaxe, případně jak je náročné napsaný kód číst, opravovat či jej udržovat. Důležitým faktem při rozhodování může být také existence, nebo absence různých knihoven, které nesou moduly, třídy nebo jiné objekty, pro snadnější práci při implementaci aplikace `bws`. V neposlední řadě můžeme při výběru implementačního jazyka přihlédnout k faktu, zda existují nějaké podpůrné nástroje, které dokáží například samy vygenerovat dokumentaci ze zdrojových kódů.

Jaké jsou tedy předpoklady práce při implementaci projektu? Jelikož již víme, že budeme pracovat s několika aplikacemi, se kterými se většinou pracuje skrze příkazový řádek, předpokládáme, že budeme poměrně často potřebovat volat příkazy příkazového řádku, které budeme také muset vyhodnocovat.

Jedním z požadavků je snadná konfigurace, předpokládejme tedy, že budeme vytvářet nebo měnit konfigurační soubory jednotlivých aplikací. Z tohoto důvodu tedy bude třeba, aby zvolený implementační jazyk dovedl pohodlně pracovat se soubory. Neméně podstatné bude obsah těchto souborů číst po řádcích a nalézt v nich tu část, kterou budeme potřebovat, potažmo tedy vhodně zvolený implementační jazyk musí poskytovat prostředky pro parsování vybraného textu.

Dalším neméně podstatným požadavkem je podpora spolupráce s databáhí MySQL, neboť aplikace Bugzilla ke svému fungování využívá právě tuto databázi. Bylo by tedy vhodné vybrat si takový jazyk, který umí obsluhovat příkazy psané v SQL dialektu.

Nyní již známe klíčové požadavky, na jejichž základně zvolíme jeden z následujících jazyků:

python je interpretovaný objektově orientovaný programovací jazyk, který je víceparadigmatický. Termín víceparadigmatický znamená, že je navržen tak, aby v něm mohlo být programováno objektově, procedurálně, ale také funkcionálně. Silnou stránkou jazyka Python je, že kód napsaný v programovacím jazyce Python je krátký a dobře čitelný. Čitelnost jazyka je důsledkem

jednoduché syntaxe jazyka a také fakt, že jazyk Python dbá na psaní čitelného kódu, neboť interpret jazyka hlídá i správné odsazování, díky tomu se nikdy neztratíte v bludišti různých bloků a závorek. Díky těmto vlastnostem je jazyk python doporučen k výuce. Významnou vlastností jazyka Python je vysoká produktivnost z hlediska rychlosti psaní programů. Jazyk Python má přirozenou podporu jmenných prostorů a výjimek. Pro jazyk Python je napsáno mnoho modulů, které řeší spoustu problémů, mimo jiné pro práci se soubory, regulárními jazyky a obsluhou příkazů z příkazové řádky. Tento odstavec čerpá z internetové encyklopedie Wikipedia[8].

bash je skriptovacím jazykem, který používá linuxová příkazová řádka. Termínem skriptovací jazyk rozumíme takový jazyk, kdy jsou instrukce jazyka čteny ze zdrojového kódu a následně ihned vykonány, bohužel toto má za následek, že zpracování jednoho takového skriptu je pak mnohem pomalejší než u jazyků komplikovaných. Výhodou skriptovacího jazyka je jeho přenositelnost. Psaní v shellu umoňuje automatizovat mnoho úloh, které by si vyžadovaly psát mnoho příkazů na příkazové řádce. Výhodou jazyka bash je tedy jeho vysoká přenositelnost mezi linuxovými operačními systémy. Přesto však jazyk bash není pro implementaci aplikaci bws úplně vhodný, neboť manipulace se soubory je poněkud obtížnější. Tento odstavec je přejat z internetové encyklopedie Wikipedia[8].

perl je interpretovaným programovacím jazykem, který se stal populárním nástrojem pro tvorbu CGI skript. Programovací jazyk perl je vhodný pro psaní krátkých jednoduchých programů, avšak také je možné v něm tvořit dlouhá programová díla. Stejně jako u jazyku bash je jeho výhodou snadná přenositelnost mezi různými operačními systémy linux. V jazyce perl jsou implementovány mnohé moduly třetích stran, mezi nimi také modul pro práci s různými databázemi a taktéž formuláři. K jazyku perl existuje také široká podpora, počínajíce dokumentací, kvalitní literaturou a všeobecně širokou odbornou veřejností. Jazyk perl umí dynamicky pracovat s pamětí, což znamená, že není třeba destuktorů objektů, taktéž poskytuje pokročilé datové typy. V jazyku perl lze pracovat jak procedurálně, tak objektově, taktéž v něm lze snadno pracovat se soubory. Jazyk perl má také několik nevýhod. Nedisciplinovaný programátor může snadno vytvářet nesrozumitelný kód, neboť jazyk perl to umožňuje. Oproti některým jiným jazykům spotřebovává jazyk perl více paměti než ostatní. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[8].

Z výše zmíněných jazyků zvolíme jazyk Python, nebuď tento jazyk je vhodný pro práci se soubory, což je ve skriptovacím jazyce bash poněkud obtížnější. Další předností tohota jazyka je fakt, že nutí programátora vytvářet lehce udržovatelný a čitelný zdrojový kód, což je výhoda oproti programovacímu jazyku Perl. Jazyk Python má stejně jako jazyk Perl mnoho knihoven pro práci s různými oblastmi problémů. Pro implementaci aplikace `bws` tedy zvolíme víceparadigmatický jazyk Python.

[8] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

4.2 Volba vývojového prostředí

Jedním z požadavků implementace projektu je, aby výsledný produkt byl kompatibilní s operačním systémem linux. Operačních systémů, které jsou založeny na jádře linux je však mnoho a jsou označovány jako distribuce. Tyto distibuce lze rozdělit dle toho, jaký používají balíčkovací systém na tři velké skupiny.

4.2.1 Distribuce používající balíčkovací systém Debian

Debian je distribuce GNU/Linuxu, který je vyvíjen velkým množstvím dobrovolníků z celého světa. Jedná se o jednu z nejrozšířenějších linuxových distribucí na světě. Z desktopových stanic je v poslední době vytlačován distribucí Ubuntu, která vychází právě ze systému Debian. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[9].

Greenie je mladá linuxová distribuce ze Slovenska. Po technické stránce vychází z distribuce Ubuntu, avšak má odlišnou filozii. Greenie Linux je určen především pro začátečníky. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[9].

Knoppix je jednou z nejznámějších live CD distribucí. Knoppix je znám především pro svou propracovanou detekci hardware a snadnou použitelnost. Obdobně jako ostatní operační systémy s debianovským balíčkovacím systém i Knoppix vychází ze systému Debian. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[9].

Ubuntu[10] je linuxovou distribucí, která je navržena primárně pro pracovní stanice. Distribuce Ubuntu vychází ze systému Debian a má mnoho derivátů, mezi nimiž je například Kubuntu, kde je pracovní prostředí GNOME nahrazeno pracovním prostředím KDE. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[9].

4.2.2 Distribuce využívající balíčkovacího systému RPM

CentOs je volně dostupná linuxová distribuce, která vychází ze systému Red Hat Enterprise Linux a je s ním maximálně kompatibilní. Oproti distribuci RHEL jsou aktualizace vydávány se zpožděním. Jedná se tedy o plně hodnotný systém, vycházející ze systému Red Hat Enterprise Linux, který je spravován především komunitou vývojářů. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[9].

Fedora je další linuxovou distribucí založenou na balíčkovacím systému RPM, je vyvíjena komunitou vývojářů soustřeďujících se kolem Fedora Projectu, sponzorovaného společností Red Hat, od jejíž distribuce se projekt oddělil. Cílem projektu bylo všeobecné použití open source softwaru. Fedora tedy vznikla jako reakce na obchodní strategii Rad Hatu, který se v roce 2003 zaměřil na

[9] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

[10] Česká domácí stránka projektu Ubuntu je dostupná na: <http://www.ubuntu.cz/>

komerční sféru. Fedora, podobně jako Ubuntu, je vyvíjena s důrazem na použití na desktopových počítačích, avšak umožňuje i využívání na serverech. Výhodou operačního systému Fedora je její vysoká kompatibilita s novým hardwarem. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[11].

Mandriva Linux je linuxovou distribucí zaměřenou především na snadnost instalace a použití. První verze Mandriva Linuxu byla založena na Red Hat Linuxu 5.1. Systém Mandriva Linux se vyznačuje velkým množstvím nástrojů pro snadnou konfiguraci systému. Od systému RHEL se odlišuje tím, že své balíčky kompiluje taktéž pro procesory třídy Pentium a vyšší. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[11].

Red Hat Enterprise Linux je přímým nástupcem operačního systému Red Hat. Od systému CentOs se odlišuje především faktem, že je orientován na komerční sféru včetně mainframů. Placená je pouze podpora a servis včetně přístupu k webové službě s názvem Red Hat Network. Zdrojové kódy programů systému RHEL jsou volně přístupné včetně průběžných aktualizací a je na nich založeno mnoho odvozených distribucí, které odstranily ochranné známky a loga. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[11].

4.2.3 Distribuce používající jiný balíčkovací systém

Slackware je jedna z nejstarších z doposud aktivně vyvíjených linuxových distribucí. Systém Slackware obsahuje pouze stabilní a prověřené verze programů. Klade důraz na jednoduchost, stabilitu a konfigurovatelnost. Operační systém Slackware používá starší balíčkovací systém pkgtolls, který je postavený nad tar balík zkomprimovaný gzipem. Přestože tento balíčkovací systém podporuje závislosti, dodávaný sofware pro správu tuto možnost nevyužívá. Avšak existují nástroje, které závislosti balíčků řeší. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[11].

Arch Linux je nezávislá linuxová distribuce, která je vyvíjena jako nenáročný odlehčený a snadno přizpůsobitelný systém. Arch Linux je vhodný pro zkušenější uživatele, kterým umožňuje připravit systém přizpůsobený podle specefických potřeb a bez zbytečných součástí. Instalace a konfigurace probíhá v textovém režimu. Arch Linux používá vlastní systém binárních balíčků, které jsou spravovány Pacmanem. Arch Linux kromě pěti oficiálních repozitářů nabízá také repositář, do něhož si mohou uživatelé a vývojáři přidávat další software, který chybí v oficiálních zdrojích. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[11].

Gentoo je distribucí operačního systému linux vyvíjenou podobně jako Debian komunitou. Hlavním rozdílem oproti ostatním distribucím je skutečnost, že je založena přímo na zdrojových kódech. Každý si tedy přeloží svůj unikátní systém dle svých požadavků. Překládání, ale není nutností. Balíčkovací systém této distribuce se nazývá Portage a řeší závislosti a předání parametrů pro překlad a díky tomu zvládne instalaci a správu systému i poučený začátečník. Nespornou

[11] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

výhodou systému Gentoo Linux je maximální možnost nastavení jednotlivých aplikací, přehledná konfigurace a především konfigurace pro konkrétní hardware Naopak nevýhodou je náročnost na výpočetní výkon během instalace a také délka kompilace. Tento odstavec byl přejat z internetové encyklopedie Wikipedia[12].

4.2.4 Shrnutí a samotný výběr

Nyní zbývá z navržených distribucí vybrat tu, která bude pro implementaci nejvýhodnější, a to jak z hlediska jednoduchosti implementace, tak z hlediska míry použitelnost ve výsledného produktu v dané distribuci. V potaz vezmeme tedy především míru oblíbenosti distribuce u uživatelů začátečníků nebo mírně pokročilých, neboť ti jsou cílovou skupinou, pro niž se aplikace vyvíjí. Dalším kritériem pro výběr distribuce bude, zda má ve svých repositářích, které spravují jednotlivé balíčkovací systémy všechny balíčky, které bude výsledná aplikace potřebovat, jedná se zejména o balíčky aplikací Bugzilla, MoinMoin, Svn, Apache 2 a Python. Dalším ukazatelem vhodnosti distribuce je snadnost a intuitivnost použití distribuce. A v neposlední řadě přehledneme k té distribuci, se kterou máme nejvíce zkušeností.

Jelikož již známe kritéria, která by měla vybraná linuxová distribuce splňovat, můžeme začít vyřazovací metodou hledat vhodné distribuce pro vývoj aplikace. Z distribucí s balíčkovacím systémem Debian musíme vyloučit distribuci Knoppix, která je především Live distribucí a tedy nelze předpokládat, že na této distribuci poběží vývoj několika projektů. Z distribucí s balíčkovacím systémem RPM vyloučíme distribuci Red Hat Enterprise Linux, neboť ta je vyvíjena pro komerční sektor, kde se dají předpoklád spíše projekty středního až velkého rozsahu dlouhodobějšího charakteru, pro které aplikace bws není primárně určena. Z distribucí, které využívají jiný balíčkovací systém vyloučíme distribuci Arch Linux, neboť se jedná o distribuci, která je vhodná především pro zkušenější uživatele. Ze stejného důvodu vyloučíme taktéž distribuci Gentoo, neboť i ta je navržena především pro pokročilejší uživatele. Z distribucí používající jiný balíčkovací systém musíme vyloučit i distribuci Slackware, neboť ta není defaultně dodávána s balíčkovacím systémem, který umí efektivně řešit závislosti mezi jednotlivými balíčky.

Stále ještě zbývá poměrně mnoho distribucí, mezi kterými se musíme rozhodnout. Nyní zkusíme vyřadit distribuce na základě oblíbenosti. Z internetového zdroje distrowach dostupného na adrese http://distrowatch.com/index.php se dozvídáme, že nejnavštěvovanějšími stránkami distribucí jsou internetové stránky distribuce Ubuntu, a to téměř dvojnásobně než u ostatních stránek, lze tedy předpokládat, že distribuce Ubuntu je u uživatelů oblíbená a tudíž budeme uvažovat o distribuci Ubuntu jako o distribuci, na které by bylo vhodné vyvýjet. Nyní zbývá zjistit, zda splňuje i ostatní výše zmíněná kritéria.

[12] Internetová encyklopedie Wikipedia, dostupná na: <http://en.wikipedia.org/wiki/>

Po instalaci systému Ubuntu zjišťujeme, že všechny výše zmíněné balíčky jsou přítomné v balíčkovacím systému a tudíž tato podmínka je splněna. Uživatelské prostředí v distribuci Ubuntu je příjemné a celkem intuitivní, avšak já jsem zvyklý pracovat v prostředí KDE, a proto tedy volím derivát distribuce Ubuntu – Kubuntu, kde je nativním prostředím právě prostředí KDE.

Výslednou distribucí, na které proběhne vývoj aplikace bws je tedy distribuce Kubuntu.

4.3 Popis systému linux pro potřeby aplikace bws

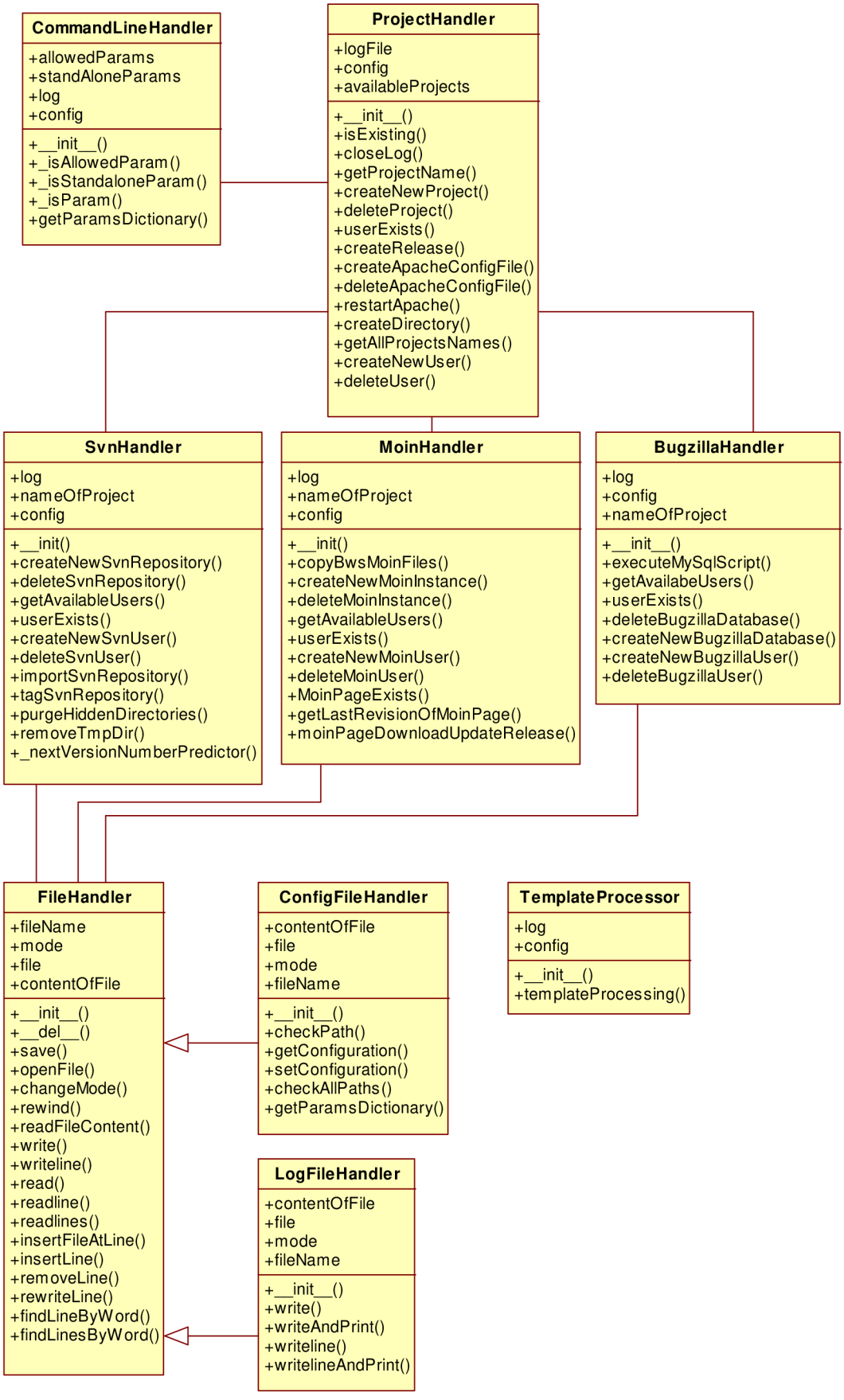
Před samotným začátkem implementace je vhodné si nastudovat alespoň základní strukturu linuxového operačního systému, neboť tato bývá v mnoha distrubucích velice podobná, ne-li stejná. Pro adresářovou strukturu operačního systému linux existuje FSSTND standard (Filesystem standard), který si nyní přiblížíme.

Tabulka 4-1: Výňatek ze standardu FSSTND [17]

Adresář	Popis
/	Primární, nebo též kořenový adresář celé hierarchické struktury souborů.
/bin/	V tomto adresáři se nacházejí základní programy.
/boot/	Je adresářem pro soubory pro zavedení systému.
/dev/	Adresář, který obsahuje speciální soubory zařízení.
/etc/	Je adresářem obsahujícím konfigurační soubory pro jeden počítač.
/home/	V tomto adresáře se nacházejí všechny domovské adresáře uživatelů.
/lib/	Tento adresář nese sdílené knihovny a moduly potřebné pro start a programy z /bin a /sbin.
/mnt/	Je adresářem, kde jsou dočasně připojené svazky.
/opt/	V tomto adresáři jsou volitelné programové balíky.
/root/	Je domovským adresářem pro uživatele root.
/sbin/	Zde se nacházejí základní programy vyjma těch, které jsou pro běžné uživatele.
/tmp/	Tento adresář slouží k uchovávání dočasných souborů.
/usr/	Je sekundární hierarchií.
/usr/bin/	Adresář s uživatelskými programy.
/usr/lib/	Adresář s knihovnami.
/usr/sbin/	Je adresářem pro systémové programy, které běžní uživatelé nepoužívají.
/usr/share/	Tento adresář obsahuje architekturově nezávislé soubory.
/var/	Adresář obsahující proměnná data.
/var/cache/	Je adresářem sloužícím pro uchovávání cache pro aplikace.
/var/lock/	Tento adresář obsahuje zámky.

/var/log/	Slouží pro uchovávání logovacích záznamů a žurnálových souborů.
/var/mail/	Je adresářem pro uživatelské poštovní schránky.
/var/spool/	Do tohoto adresáře se ukládají fronty nezpracovaných dat pro aplikace.
/var/tmp/	Je adresářem pro pomocné soubory.

Aplikace bws bude používat adresář /etc, ve kterém vytváří adresář /bws a do něho ukládá soubor config.txt, který nese konfigurační údaje aplikace bws. Dále je použit adresář /usr/bin, do kterého se nahrávají všechny skripty, které pak může uživatel může používat. Taktéž aplikace pracuje s adresářem /usr/share. V tomto adresáři je vytvořen adresář /bws, do kterého jsou nahrány všechny zdrojové části aplikace. V adresáře /usr/share/ se ještě navíc pracuje s jazykovými soubory, které jsou uloženy v podadresáři /locale. V podarsáři /locale se nacházejí další podadresáře, které jsou pojménovány dle zkratky jazyka, jenž reprezentují. V těchto adresáří ještě mohou být adresáře začínající prefixem LC, například LC_MESSAGES, které označují skupinu překladu. Pro konkrétní případ LC_MESSAGES se jedná o skupinu zpráv, které jsou obvykle sdělovány uživateli. Aplikace bws je zatím přeložena do dvou jazyků, a sice do češtiny a do jazyka anglického. Dalším adresářem, který je využit aplikací bws je adresář /var/lib. Tento adresář slouží pro ukládání projektů, tedy jejich webových stránek, repositářů a dalších vitálních dat. Posledním adresářem, který používá aplikace bws je adresář /var/log, do kterého je uložen soubor nesoucí data z logu databáze, bez tohoto souboru, stejně jako bez souboru konfiguračního je aplikace bws nefunkční.



Obrázek 4-1: Diagram tříd

4.4 Diagram tříd

První třídou v diagramu tříd je třída pojmenovaná **CommandLineHandler**. Jak napovídá její název, objekt této třídy je zodpovědný za zpracování příkazu příkazové řádky. Objekt této třídy využívá k obsluze příkazů objekty dalších tříd. Zejména pak objektu třídy ProjectHandler.

Objekt třídy **ProjectHandler** je volán pokaždé, když je vyžadována jakákoliv manipulace s projektem, zejména pak, jedná-li se o vytvoření či smazání projektu. Objekt třídy ProjectHandler obvykle existuje pouze jeden. Objekt třídy ProjectHandler zastřešuje funkcionalitu jednotlivých zastoupených použitých technologií.

Objekt třídy **SvnHandler** je většinou podřízen a řízen objektu třídy ProjectHandler. Třída SvnHandler disponuje metodami pro obsluhu části projektu, jenž je zodpovědný za správu verzí. Tato třída má implementovány metody pro vytváření, mazání a manipulaci s repositáři projektu. Jednou z těchto metod je metoda, která zajišťuje vytvoření tagu (uložení aktuálního stavu repositáře projektu). Dále také implementuje metodu pro snadné importování již existujícího repositáře do projektu.

Objekt třídy **BugzillaHandler** je taktéž podřízen a řízen objektem třídy ProjectHandler. Třída BugzillaHandler je vybavena metodami pro vytváření nové databáze pro bugtracingovou aplikaci bugzilla. Objekt této aplikace je z důvodu práce s MySQL databází taktéž vybaven metodami pro práci s příkazy psané v SQL dialektu.

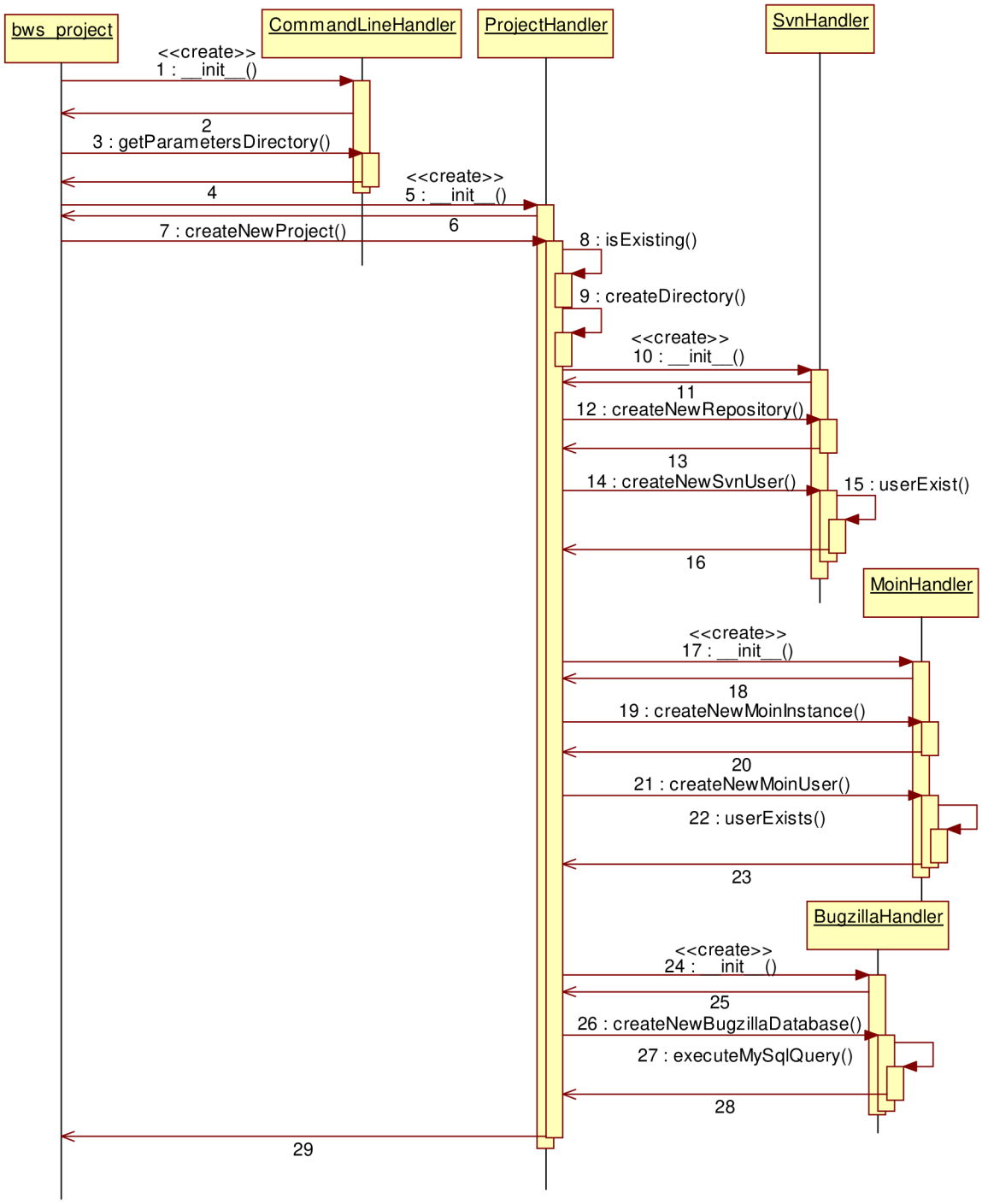
Objekt třídy **MoinHandler** je rovněž podřízen a řízen objektem třídy ProjectHandler. Třída MoinHandler poskytuje prostředky pro vytváření nových instancí aplikace MoinMoin, taktéž jejich mazání, ale především implementuje metody pro práci s jednotlivými stránkami aplikace MoinMoin, zejména pak implementuje metodu pro aktualizaci stránek s novými verzemi projektu.

FileHandler je třída, jejíž objekt je použit kdykoliv je potřeba obsluhovat jakýkoliv soubor, víceméně se jedná o rozšíření systémového modulu. Oproti systémovému modulu jej tato třída rozšiřuje o možnost přepisování souboru a dále také vyhledávání v souboru na základě zadaného výrazu.

LogFileHandler je třída dědící chování z třídy FileHandler. Objekt třídy LogFileHandler je použit k obsluze (převážně zápisu) logovacího souboru aplikace. Třída LogFileHandler upravuje zejména zápisové metody třídy FileHandler. Položky do logovacího souboru jsou ukládány v pevném formátu. Zápis do logovacího souboru je tedy vždy uvozený časovým údajem.

ConfigFileHandler je druhou třídou, jež rozšiřuje třídu FileHandler. Objekt třídy ConfigFileHandler je použit pro obsluhu konfiguračního souboru aplikace. Objekt třídy ConfigFileHandler slouží zejména ke čtení hodnot z konfiguračního souboru, taktéž je schopen hodnoty do konfiguračního souboru zapisovat. Rovněž je schopen zkontrolovat hodnoty v konfiguračním souboru.

Objekt třídy **TemplateProcessor** je používán k obsluze vytváření nových konfiguračních souborů pro použité aplikace, především se jedná o vytváření konfiguračních souborů pro server Apache2, taktéž pro konfiguraci zpřístupnění svn repositáře, dále pro nastavení přístupu k databázy pro aplikaci bugzilla a v neposlední řadě také pro vytvoření konfiguračních souborů potřebných pro vytvoření nové instance wiki systému MoinMoin.



Obrázek 4-2: Diagram interakce - vytvoření projektu

4.5 Vytvoření projektu

Diagram uvedený níže popisuje interakci mezi jednotlivými objekty, které se podílejí na vytváření nového projektu aplikace.

Vytvoření projektu probíhá tak, že uživatel se superuživatelskými právy zavolá z konzole příkaz pro vytvoření projektu (`bws_project`). Tento příkaz může být volán s parametry pro název projektu a údaje pro prvního uživatele projektu, detailnější informace o příkazu `bws_project` lze nalézt na přiloženém CD v dokumentaci.

Ihned po zavolání příkazu je vytvořen objekt třídy `CommandLineHandler`. Tento objekt se postará o korektní zpracování parametrů příkazu, které mohly, ale nemusely být zadány z příkazové řádky. Pokud byly na příkazové řádce parametry nesprávně zadány, nebo použity, bude tato skutečnost oznámena uživateli a příkaz pro provedení vytvoření nového projektu se neprovede.

Pokud byly všechny parametry zadány správně, nebo nebyly zadány vůbec, je vytvořen objekt třídy `ProjectHandler`. Dále je zavolána metoda `createNewProject`, které jsou předány zpracované parametry formou slovníku a je jí svěřeno řízení dalších akcí.

Objekt `ProjectHandler` nejdříve požaduje zadání názvu projektu, pokud se tento nenacházel mezi parametry, které byly zadány z příkazové řádky, byl-li parametr zadán, je krok zadávání názvu projektu přeskočen. Pokud parametr názvu projektu nebyl zadán, bude nyní uživatel, který spustil skript, dotázán na název projektu. Po zadání názvu projektu je volána metoda `isExisting` objektu třídy `ProjectHandler`, která zkontroluje zda zadaný název projektu již neexistuje. Pokud existuje je tato skutečnost oznámena uživateli a ten pak musí zadat jiné jméno, nebo ukončit skript.

Bylo-li zadáno jméno projektu správně, je uživatel dotázán na zadánání údajů identifikujících prvního uživatele projektu. Pro vytvoření uživatele je požadováno jeho jméno, email a heslo. Pokud jsou tyto údaje zadány nesprávně, je tato skutečnost uživateli oznámena a uživatel je nucen zadat nové údaje, nebo ukončit vytváření projektu. Pokud však byly údaje zadány úspěšně, je několikrát volána metoda `createDirectory`, která má za úkol vytvořit všechny adresáře, které bude projekt potřebovat. Metoda `createDirectory` je volána pro každý adresář projektu zvlášť. Pokud byly všechny adresáře úspěšně vytvořeny, pokračuje se dalším krokem. Pokud však všechny adresáře nebyly úspěšně vytvořeny, je vytváření projektu ukončeno a všechny doposud vytvořené součásti jsou vymazány.

Dalším krokem při vytváření je vytvoření nového objektu třídy `SvnHandler`, který bude použit pro manipulaci se subversion repositářem. Po vytvoření objektu třídy `SvnHandler` je tomuto objektu poslána zpráva `createNewRepository`. Po zavolání této metody je vytvořen nový subversion repositář. Pokud není repositář správně vytvořen, je opět vše dosud vytvořené skriptem smazáno.

V případě, že vytváření projektu proběhlo úspěšně, je zavolána metoda `createNewSvnUser`, tato metoda má na starost vytvoření přístupu do subversion repositáře pro uživatele, který byl definován. Při ověřování správnosti jména uživatele pro vytvoření subversion repositáře je podobně jako u `ProjectHandler` volána metoda `userExist` objektu třídy `SvnHandler`. Pokud uživatel subversion repositáře již existuje, nebude vytvořen a dosavadní vytvořené útvary programu budou vymazány.

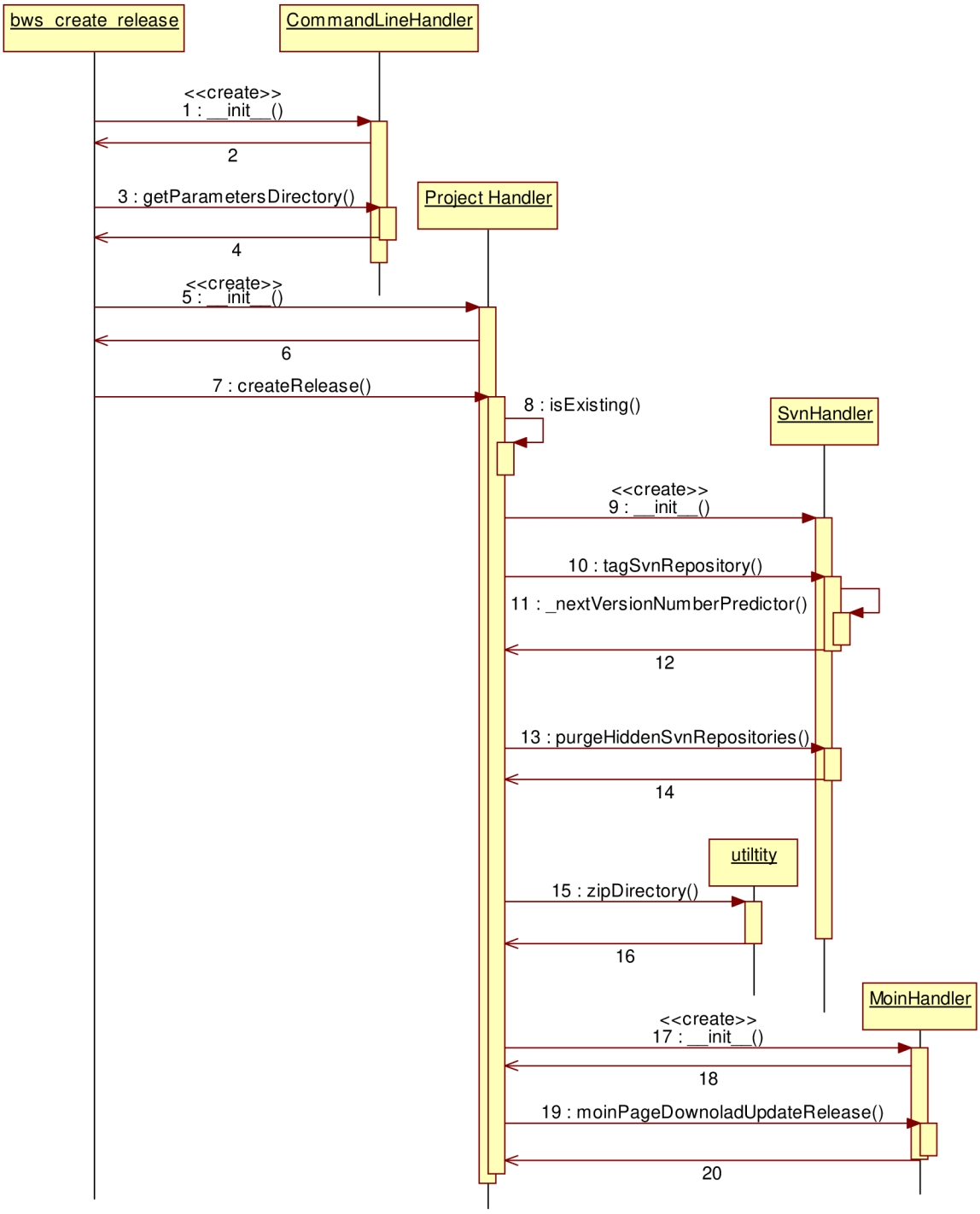
Pokud vytvoření subversion repositáře i konfigurace přístupu do subversion repositáře pro specifického uživatele proběhlo v pořádku, je toto oznámeno zpět objektu třídy `ProjectHandler`, který přebírá řízení projektu. Je vytvořen objekt třídy `MoinHandler`. Objekt `ProjectHandler` volá metodu `createNewMoinInstances` objektu `MoinHandler`, která jak napovídá její název je zodpovědná za korektní vytvoření a nakonfigurování nové instance aplikace MoinMoin. V případě, že byla instance aplikace MoinMoin úspěšně vytvořena, je tato skutečnost oznámena zpět objektu třídy `ProjectHandler`, pokud však při vytváření instance došlo k chybě, budou všechny vytvořené objekty projektu smazány a tato skutečnost bude oznámena uživateli.

Dalším krokem je vytvoření prvního uživatele aplikace MoinMoin. To je realizováno metodou `createNewMoinUser`. Obdobně jako při vytváření přístupu pro prvního uživatele, je nutno zkontrolovat data, která byla předána k identifikaci nového uživatele. Pro kontrolu, zda již uživatelský účet s tímto jménem náhodou neexistuje, je zavolána metoda `userExist`, která vrací pravdivostní hodnotu indikující existence uživatelského účtu. Pokud účet se uživatelským jménem již existuje budou všechny doposud vytvořené objekty vzniklé při instalaci vymazány.

Za předpokladu, že vytvoření uživatele proběhlo bez problémů, pokračuje vytváření projektu dalším krokem, kterým je inicializace nové databáze pro aplikaci Bugzilla. Práci s aplikací Bugzilla má na starost objekt třídy `BugzillaHandler`. Tento objekt je vytvořen na žádost objektu třídy `ProjectHandler`. Ihned po vytvoření tohoto objektu je volána metoda tohoto objektu `createNewBugzillaDatabase`, která má na starost právě vytvoření databáze aplikace Bugzilla a její správnou konfiguraci. Při vytváření nové databáze se několikrát využívá metody `executeMySqlQuery`, případně metody `executeMySqlScript`, které jsou schopné zpracovávat příkazy psané v jazyce sql, případně i dávkové soubory psané v jazyce sql. Tyto metody pro práci s databází MySQL vždy hlídají korektní zpracování sql příkazu a pokud by tento z nějakého důvodu nebyl korektně zpracován, dojde k ukončení instalace a smazání všech doposud vytvořených součástí projektu, tedy i dosud vytvořených součástí pro MoinMoin, svn a všech vytvořených adresářů, které byly v projektu vytvořeny.

Během konfigurace databáze pro její použití aplikací bugzilla je taktéž vytvořen první uživatel, který bude mít přístupová práva pro práci s touto databází. Vytvoření uživatele je realizováno zápisem jediné řádky do správné relační tabulky právě vytvořené databáze. Pokud by během procesu vytváření uživatele došlo k jakékoliv chybě, zejména k chybě týkající se nespojení s databázovým

serverem, je tato chyba zachycena a program postupuje tak, že dojde k vymazání veškeré dosavadní práce na vytváření projektu, tedy k vymazání všech dosud vzniklých součástí projektu.



Obrázek 4-3: Diagram interakce - vytvoření nové verze projektu

4.6 Vytvoření nové verze projektu

Obdobně jako u vytváření nového projektu se vytváří nová verze projektu pomocí zavolání příkazu `bws_create_release` z příkazové řádky. Taktéž lze příkazu zadat parametry z příkazové řádky, a tím obejít pozdější zadávání údajů nutných pro identifikaci projektu a prvního uživatele projektu. Podrobnější informace o parametrech příkazu lze nalézt v dokumentaci aplikace přiložené na CD.

Stejně jako v předchozím případě se pro obsluhu příkazu a především pro zpracování případných parametrů, které mohly být zadány z příkazové řádky, využije objektu třídy `CommandLineHandler`, který byl vytvořen skriptem `bws_create_release` jako jeden z prvních příkazů tohoto skriptu.

Poté, co je vrácen slovník parametrů (v případě, že nebyly zadány žádné parametry je navrácena jen chybová zpráva, že žádné parametry nebyly nalezeny a délka slovníku, tedy 0). Dále je vytvořen objekt třídy `ProjectHandler`, kterému jsou předány parametry ze slovníku. Po vytvoření objektu je zavolána metoda `createRelease` objektu třídy `ProjectHandler`. Tato metoda má za úkol vytvořit novou verzi projektu, nejdříve je však pomocí metody `isExisting` objektu třídy `ProjectHandler` ověřeno, že tento projekt skutečně existuje. Pokud tento objekt neexistoval, byl by proces vytváření projektu zastaven, dále by tato skutečnost byla zaznamenána v logovacím souboru projektu a také oznámena uživateli.

V případě, že projekt se zadaným jménem existuje, je potom objektem třídy `ProjectHandler` vytvořen objekt třídy `SvnHandler`. Dalším krokem je zavolání metody `tagSvnRepository`, která označkuje větev projektu. Značkování v praxi znamená zkopírování současného stavu repositáře do jiného adresáře v repositáři. Metoda `tagSvnRepository` dále zavolá metodu `_nextVersionPredictor`, která se pokusí najít v projektu adresář s poslední verzí projektu a zjistit její číslo. Číslo verze projektu je obvykle zapsána v tečkové notaci (například verze 0.0.1). Metoda `_nextVersionPredictor` zjistí číslo této verze, zvýší její nejnižší řád o jedna a tuto hodnotu vrátí zpět. Dále je metodou `tagSvnRepository` vytvořen dočasný adresář, který nese ve svém názvu číslo verze. Do tohoto adresáře je nakopírován obsah označkovaného adresáře. Zavoláním metody `purgeSvnHiddenDirectories` se z dočasného adresáře odstraní všechny skryté adresáře, které nesou informace pro verzovací systém (obvykle se jedná o adresáře s názvem `.svn`). Plná cesta k tomuto pročištěnému dočasnému adresáři se vrátí objektu třídy `ProjectHandler`.

Dalším krokem při vytváření nové verze projektu je zavolání funkce `zipDirectory` z modulu `utility`, tato funkce přijímá jak vstupní parametr plnou cestu k adresáři, který má být zkomprimován metodou zip. Tato funkce tedy zkomprimuje daný adresář a vrátí zpět celou cestu ke komprimovanému souboru.

Objekt třídy `ProjectHandler` vytvoří objekt třídy `MoinHandler`, aby bylo možno vykonat metodu `moinPageUpdateDownloadRelease` třídy `MoinHandler`. Tato metoda má jako vstupní parametr právě komprimovaný soubor s novou verzí projektu. Nedříve se zjistí, zda existuje stránka, která nabízí stažení nových verzí projektu. Pokud tato stránka nebyla nalezena, je proces zveřejňování nové verze ukončen chybovým hlášením uživateli. Existuje-li tato stránka, nahraje se komprimovaný soubor do příloh k dané stránce nabízející stažení nových verzí projektu. Původní komprimovaný soubor je smazán, stejně tak jako dočasný adresář.

Posledním krokem při publikování nové verze projektu je zveřejnění této nové verze projektu. Nyní již stačí vložit odkaz na přílohu stránky aplikace MoinMoin na příslušnou stránku. Toto je realizováno parsováním stránky. Celý obsah dané stránky je načten a poté se v něm hledá příslušné místo, kam má být odkaz na přílohu vložen, taktéž je nutné přesunout původní odkaz na nyní již starší verzi projektu k ostatním starším verzím projektu. Aplikace je naimplementována tak, že na dané stránce aplikace MoinMoin je nalezen výraz `=== Latest release ===` a `=== Previous releases ===`, kdy první z nich je nadpisem pro poslední a tedy nejaktuálnější verze projektu a druhý je nadpis pro všechny nebo některé předchozí verze projektu. Po nalezení těchto míst je ze stránky vyjmut odkaz na poslední verzi projektu a je vložen ihned pod nadpis uvozující odkazy na předchozí verze projektu. Všechny odkazy na předchozí verze projektu jsou tedy zachovány a umístěny pod nadpis previous releases. Nyní již tedy pouze zbývá vložit na stránku odkaz na poslední aktuální verzi projektu. Jelikož je vše již připraveno, stačí pod nadpis uvozující poslední verzi vložit odkaz a přílohu, ve které se nachází poslední verze projektu.

5 Implementace navržené aplikace

5.1 Způsoby obsluhy požadavků aplikace bws

Aplikace `bws` funguje na podobném principu, jako většina informačních systémů, a to díky tomu, že zastřešuje práci s aplikacemi, které typicky fungují na třívrstvém principu: klient – server – databáze.

Představme si nejdříve zpracování společných požadavků aktérů zákazník, vývojář a administrátor. Všichni tři aktéři používají jako klienta internetový prohlížeč, skrze něhož odesílají na server požadavky, které mají být zpracovány. Tyto požadavky se týkají zobrazení některé HTML stránky, kterou obsluhuje aplikace MoinMoin nebo aplikace Bugzilla. Požadavek je tedy odeslán internetovým prohlížečem na server, který jej zpracuje a odpověď odešle zpět jako HTML kód. Zpracování odpovědi se provádí jinak pro aplikaci MoinMoin a jinak pro aplikaci Bugzilla.

Pro požadavek zobrazení nebo jiné manipulace se stránkami aplikace MoinMoin je na serveru vyhledána aplikace MoinMoin, která zpracuje požadavek, který je pak odeslán zpět. Stránky aplikace MoinMoin jsou nastaveny jako domovské nebo výchozí stránky každého projektu, pokud je administrátor nepřenastavil jinak.

Pro požadavek zobrazení stránek aplikace Bugzilla nebo manipulace s aplikací Bugzilla. Aplikace Bugzilla pracuje s MySQL databází a je vysoce pravděpodobné, že pro zpracování požadavku se s ní bude muset spojit. Aplikace Bugzilla se tedy skrze systém řízení báze dat spojí s MySQL databází. Z databáze obdrží data, o něž si zažádal, a ty pak pak zapracuje do výsledku, který odešle zpět uživateli, jenž si je vyžádal.

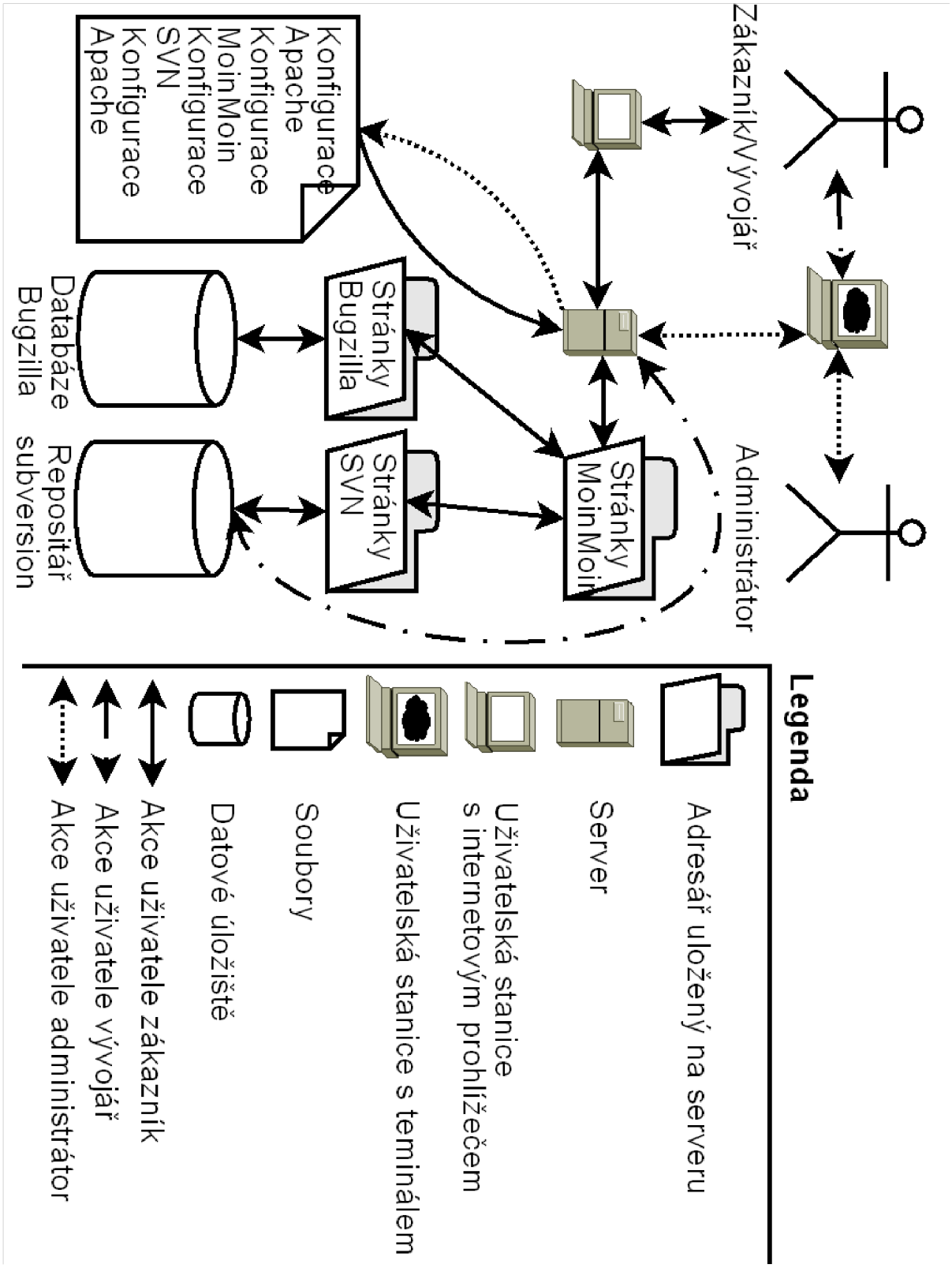
Poslední druh požadavku, který mohou mít všichni uživatelé společný, je zobrazení subversion repositáře, které zprostředkovávají subversion stránky aplikace, potažmo modul serveru Apache2 nazvaný web_dav. Modul web_dav je schopen zpracovat obsah subversion repositráře do HTML stránky, která je pak vrácena jako výsledek uživatelova požadavku.

Nyní si popíšeme ještě dva speciální druhy požadavků, které mohou nastat při práci s aplikací `bws`.

Prvním případem je situace, kdy vývojář chce nějakým způsobem manipulovat se subversion repositářem, například, pokud se snaží commitovat nebo aktualizovat verzi kódu z repositáři. Vývojář pro tento případ nepoužívá internetový prohlížeč, nýbrž některou z aplikací, která pracuje se subversion repositáři a pracovními kopiemi, nebo přímo příkazovou řádku. Aplikace pro práci s pracovními verzemi a repositáři zajišťuje komunikaci a přenos sama, nicméně je potřeba mít přístupové údaje do repositáře. S příkazovou řádkou je to velice podobné a je třeba znát příkazy aplikace subversion a taktéž přístupové údaje do repositáře.

Posledním případem je, když chce administrátor spouštět skripty aplikace `bws`, které jsou uloženy na serveru. K tomu je potřeba, aby se administrátor na tomto serveru přihlásil se

superuživatelskými právy (lokálně, nebo vzdáleně) a spustil některý z požadovaných příkazů. Tento je pak vykonán na straně serveru.



Obrázek 5-1: Schéma aplikace bws

5.2 Práce s aplikací bugzilla

5.2.1 Vytvoření nové instance aplikace bugzilla

Vytvoření nové instance pro aplikace bugzilla se skládá ze dvou kroků. Prvním z nich je správné nakonfigurování aplikace a druhým je samotné vytvoření databáze.

Pro správnou konfiguraci je potřeba získat jisté údaje od uživatele. Především se jedná o název projektu, a dále pak o identifikační údaje prvního uživatele (jméno, email) a především jeho heslo. Po té, co jsou tyto údaje korektně zadány a validovány, jsou tyto údaje připsány do konfiguračního skriptu aplikace bugzilla. Tento skript v sobě nese předdefinované značky s prefixem a suffixem `$$`, které jsou pak nahrazeny správnými hodnotami. Transformaci z těchto značek na hodnoty provádí objekt třídy `TemplateProcessor`, který je součástí modulu utility.

Vytvoření databáze se všemi tabulkami, jež jsou potřebné pro správný běh aplikace bugzilla, je v tomto případě jednodušším krokem. Pro vytvoření nové databáze stačí nastavit příkazem: `env PROJECT='bws_nazevProjektu'` proměnnou pracovního prostředí. Pak ihned následuje zavolání skriptu `checksetup.pl`, který již sám vytvoří novou databázi a nakonfiguruje ji tak, že je okamžitě použitelná.

Po úspěšném vytvoření nové databáze jsou ze skriptu vymazány pomocné údaje, které byly potřebné pro vytvoření prvního uživatele.

5.2.2 Vytvoření uživatele aplikace bugzilla

Vytvoření uživatele aplikace bugzilla je implementováno přímým zápisem dat do databáze prostřednictvím příkazu jazyka SQL. Data jsou zapsána do tabulky profiles, jejíž struktura je zobrazena níže.

Jako hodnotu do sloupce userid můžeme zadat hodnotu `NULL`, neboť databáze přiřazuje toto číslo automaticky a přitom zaručuje, že je unikátní a tedy vhodná jako primární klíč tabulky, kterým je. Hodnotou kterou zadáme do sloupce `login_name` je emailová adresa uživatele, jehož účet chceme vytvořit. Sloupec `cryptpassword` nyní přeskočíme a vrátíme se k němu později, neboť tato hodnota se získává složitějším postupem. Do sloupce `realname` zadáváme skutečné jméno uživatele, lze zadávat libovolný řetězec znaků. Do sloupce `disabledtext` zadáme prázdný řetězec. Sloupec `mybugslink` vyplníme hodnotou `1`. Do sloupce resfreshed_when je zadána hodnota aktuálního času. Sloupec `exter_id` vyplníme hodnotou `NULL`.

Nyní zbývá ještě získat hodnotu do sloupce `cryptpassword`. Hodnotou pro tento sloupec je jednostranně kryptovaná hodnota hesla, kterým bude uživatel přistupovat ke svému účtu aplikace Bugzilla. Nejlepší metodou pro získání kryptovaného hesla je přímo použít metodu, která je

implementována přímo v aplikaci Bugzilla. Pro zavolání této metody je implementován skript v jazyce Perl. Tento skript nejdříve zpracuje parametry, které byly zadány na příkazové řádce. Pak je volána metoda `bz_crypt`, která zpracuje svůj jediný parametr, kterým je nekryptované heslo. Metoda `bz_crypt` pak vrací šifrované heslo, se kterým pak umí aplikace Bugzilla pracovat.

Tabulka 5-1: Tabulka profile aplikace bugzilla verze 2.22

profiles	
userid	mediumint(9)
login_name	VARCHAR(255)
cryptpassword	VARCHAR(64)
realname	VARCHAR(255)
disabledtext	mediumtext
mybugslink	tinyint(4)
refreshed_when	DATE
extern_id	mediumint(9)

5.3 Práce s aplikací MoinMoin

5.3.1 Vytvoření nové instance MoinMoin

Pro vytvoření nové instance aplikace MoinMoin je třeba nejdříve od uživatele získat název projektu. Podmínkou je, že jméno projektu musí být v celé aplikaci unikátní. Pokud bylo jméno projektu korektně zadáno, pak se přikročí k dalším krokům vytváření nové instance.

První krok při vytváření nové instance aplikace MoinMoin je vytvoření všech potřebných adresářů. Pak dojde ke zkopírování původní instance MoinMoin do nové lokace. Dalším krokem je změna vlastníka souborů této nově vytvořené instance. Novým vlastníkem je uživatel, pod kterým se skrývá server Apache2 (v operačním systému Kubuntu se jedná o uživatele `www-data`, který patří do skupiny `www-data`). Když byl uživatel úspěšně změněn, je ještě třeba změnit přístupová práva uživatelů.

Nyní je potřeba ještě upravit konfigurační soubor famrconig. Tento konfigurační soubor obsahuje informace o tom, pro které URL adresy se má použít patřičná instance aplikace MoinMoin. Úprava konfiguračního souboru spočívá v přidání jediného řádku s regulárním výrazem, který označuje, pro které URL adresy se má použít tato instance aplikace MoinMoin. Tato akce je automatická.

Poslední akcí, kterou je třeba vykonat, je zkopírování stránek a šablon stránek, které jsou součástí aplikace `bws` a rozšiřují instanci aplikace MoinMoin. Toto kopírování zajišťuje metoda `copyBwsMoinFiles` objektu třídy `MoinHandler`.

5.3.2 Vytvoření nového uživatele aplikace MoinMoin

Pro vytvoření nového uživatele aplikace MoinMoin potřebujeme nějdříve získat jeho identifikační údaje. Těmito údaji rozumíme uživatelovo skutečné jméno, jeho emailovou adresu, heslo a případně uživatelův alias. Zvýšené požadavky jsou kladeny především na jméno uživatele, které musí být v celé aplikaci MoinMoin unikátní. Emailová adresa je kontrolována regulérním výrazem. Uživatelské heslo musí být minimálně pět znaků dlouhé.

Samotné vytvoření uživatele je realizováno zavoláním skriptu aplikace MoinMoin, který je pro tyto účely přímo vytvořen. Tento skript je pojmenován `moin.py` a v operačním systému Kubuntu je lokalizován zde: `/usr/share/python-support/python-moinmoin/MoinMoin/script/`. Pro vytvoření uživatele pak stačí zavolat tento skript následovně: `python moin.py \ --config-dir='/etc/moin/mywiki.py' --wiki-url='localhost/bws/projektName'\ account create --name='nazev uzivatele' --email='email uzivatele' \ --password='heslo' --alias='alias uzivatele'`. Jelikož však tento skipt voláme jako uživatel root, musíme ještě zajistit, aby k těmto souborům měl přístup server Apache2. Toto provedeme změnou vlastníky souborů (příkaz `chown`), které byly vytvořeny. Apache2 je v systému Kubuntu veden jako uživatel `www-data` patřící do skupiny `www-data`.

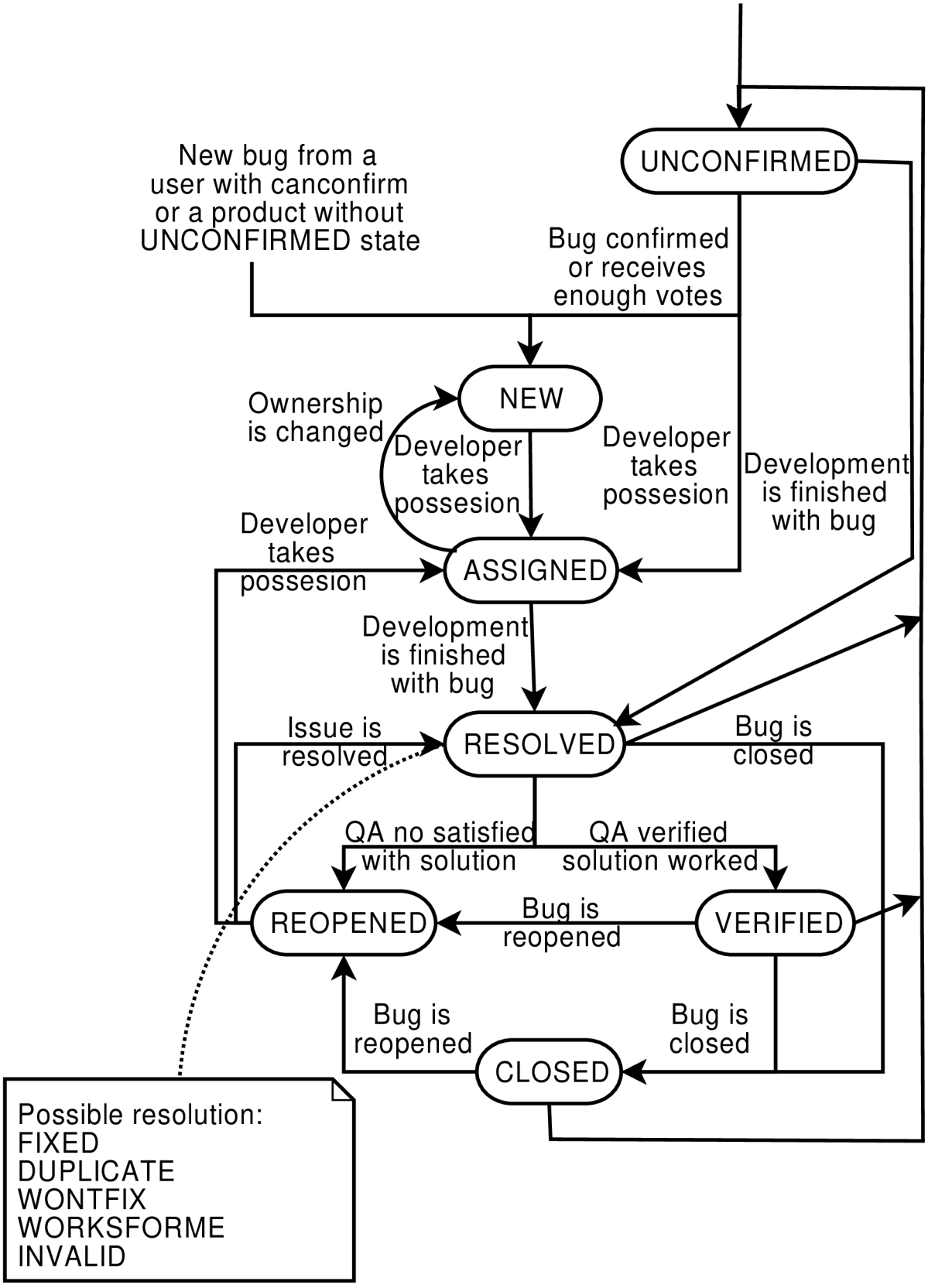
5.4 Bugzilla popis stavů jednotlivých chyb

Řešení chyby prochází v aplikaci několika stavy, které dokumentuje níže zmíněný obrázek.

Stav UNCONFIRMED je stav chyby ihned po objevení. Z tohoto stavu lze přejít do stavu NEW, pokud byla tato chyba potvrzena, nebo pokud dostala dostatek hlasů. Pokud se některý z vývojářů rozhodl touto chybou zabývat, když je tato chyba teprve ve stavu UNCONFIRMED je stav řešení této chyby změněn na ASSIGNED. Z tohoto stavu lze také přejít do stavu RESOLVED, pokud byla chyba během vývoje vyřešena ještě než byla schválena, nebo někomu přidělena.

Řešení chyby se může do stavu NEW dostat také tak, že chybu oznámil uživatel, který ji může prohlásit za potvrzenou. Taktéž se do tohoto stavu může dostat, pokud spravovaný projekt nemá stav UNCONFIRMED. Ze stavu NEW lze přejít do stavu ASSIGNED, pokud byla chyba přiřazena některému vývojáři, nebo pokud se touto chybou začne některý z vývojářů zabývat sám.

Jak již bylo zmíněno dříve, ve stavu ASSIGNED se nalézá chyba, která je přiřazena, nebo řešena některým z vývojářů. Pokud byl během řešení změněn vývojář, který chybu řeší, prochází chyba přes stav NEW zpět do toho stavu. Ze stavu ASSIGNED pak lze přejít do stavu RESOLVED pokud bylo řešení chyby uzavřeno.



Obrázek 5-2: Stavy bugů a jejich posloupnost[13]

[13] Tento obrázek je přejat a transformován, pro potřeby diplomové práce. Originál je dostupný na: <http://upload.wikimedia.org/wikipedia/commons/3/31/Bugzilla_Lifecycle_color-aqua.svg>

Ve stavu RESOLVED se nachází chyba, kterou vývojář považuje za vyřešenou. Chyba, která je označena jako vyřešená, pak může být označena jako FIXED, pokud byla opravována vývojářem a opravena. Dále může být označena jako DUPLICATE, pokud se jedná o duplikát již některé nahlášené chyby, která byla, či nebyla vyřešena. Dalším stavem vyřešené chyby je WONTFIX, tedy, že není opravena, ale od její opravy se z nějakého důvodu upustilo. Stav řešení chyby WORKSFORME indikuje, že vývojář tuto chybu nedokáže reprodukovat a tedy se o žádnou chybu nejedná. Dalším stavem je stav INVALID, který je použit, pokud zadaná chyba nenese dostatečné označení a k jejímu řešení je potřeba více informací. Stavy REMIND a LATER jsou již zastaralé a neměly by se používat. Ze stavu RESOLVED lze přejít do stavu CLOSED, pokud oprava této chyby není posuzována oddělením pro zajištění kvality. Pokud projekt používá oddělení pro zajištění kvality, může přejít stav řešení této chyby do stavu REOPENED, pokud nebyla dosažena požadovaná kvalita opravy, nebo do stavu VERIFIED, pokud tato kvalita dosažena byla.

Ve stavu REOPENED se nalézá chyba, jejíž řešení bylo provedeno vývojářem, avšak nebylo schváleno oddělením pro údržbu kvality projektu. Z tohoto stavu lze přejít do stavu RESOLVED, pokud byla chyba opravena, nebo do stavu ASSIGNED, pokud se dalšího řešení chyby ujal některý vývojář (nebo byl její opravou pověřen).

Do stavu VERIFIED se dostane řešení chyby, která byla opravena a kvalita opravy je schválena oddělením pro záruku kvality. Z tohoto stavu lze přít do stavu CLOSED, pokud bylo řešení chyby uzavřeno, nebo do stavu REOPEN, pokud bylo řešení chyby znovu otevřeno, nebo do stavu UNCONFIRMED pokud bylo řešení chyby znovu otevřeno, přičemž nebylo nikdo potvrzeno.

Stav řešení chyby CLOSED značí, že řešení chyby bylo dokončeno, z tohoto stavu lze přejít pouze do stavu UNCONFIRMED, pokud dojde z nějakého důvodu k jejímu znovuotevření.

5.5 Způsob uložení stránek aplikace MoinMoin

Aplikace MoinMoin může ukládat své stránky dvěma způsoby. Buď do databáze, nebo do souborového systému. V případě aplikace `bws` využíváme způsobu uložení do souborového systému. Stránky apliace MoinMoin se ukládají do dvou hlavních adresářů, prvním je adresář `data` a druhým je `underlay`.

Do adresáře `underlay` se ukládají stránky, které jsou nedílnou součástí aplikace MoinMoin a nebudou se měnit. Jedná se především o stránky nápovědy, které začínají prefixem `Help`, dále prezentací, které jsou uvozeny prefixem `WikiCourse` a šablon pro stránky, které končí sufixem `Template`. Šablony pro tvorbu stránek lze použít vždy při vytváření nové stránky.

V adresáři data se ukládají stránky, které vytvořil nebo upravil některý z uživatelů alikace MoinMoin.

Každá stránka aplikace MoinMoin je uložena ve svém vlastním adresáři, který nese stejný název jako je název stránky. Pokud se stránka nachází v hierarchické struktuře (je podstránkou nějaké

stránky), pak je název adresářem popisem celé cesty k dané podstránce, kde jsou úrovně odlišeny znaky (2f). Každý adresář pak obsahuje podadresář nazvaný revisions a soubor current. V podadresáři revisions jsou uloženy textové soubory pojmenovaná vždy číslem revize, které je nulami zepředu doplněno na osm znaků (tedy například 00000001). Soubor current pak obsahuje jedinou textovou hodnotu, kterou je číslo aktuální revize stránky, která má být zobrazena.

5.6 Vícejazyčná podpora

Jedním z požadavků na aplikace bws je i vícejazyčná podpora, tato vícejazyčná podpora je implementová využitím nástroje gettext.

Nástroj gettext poskytuje principy a metody právě pro vývoj vícejazyčných aplikací. Pro využití služeb nástroje gettext je potřeba nejdříve v každém modulu importovat modul gettext a dále je potřeba označit, který text, spíše textový řetězec, jenž se bude překládat. Zpravidla se jedná o zprávy, která budou zobrazeny uživateli. Označení řetězce, který bude překládán, je vlastně voláním metody modulu gettext, která implementuje překládání. Řetězec, který má být přeložen, je předán metodě _() jako parametr, který je typu řetězec znaků.Metoda _() tento řetězec přeloží a vrátí přeložený řetězec znaků.

V každé třídě je jednou z prvních věcí, které se vykonávají, zavolání nastavení jazyka, do kterého se má překládat. V aplikaci bws se překládá do jazyka, který je v systému nastaven jako lokální.

Před samotným použitím je nejdříve potřeba získat řetěce, které mají být přeloženy. Tyto získáme z aplikace pomocí příkazu xgettext s příslušnými parametry, kterými jsou především soubory, potažmo moduly, z nichž se mají tyto řetězce získat. Příkaz gettext vrací vrátí soubor s pevnou strukturou, která je složena z hlavičky, dále pak následuje seznam dvojic překládaného řetězce a jeho překladu, jelikož však překlad zatím ještě neexistuje, je druhá hodnota z dvojice prázdným řetězcem. Vygenerovaný soubor obvykle nese příponu .pot.

Z tohoto soubor je potřeba vygenerovat jazykové šablony pro překlad do určitého jazyka. Tyto šablony obvykle nesou příponu .po a je potřeba do nich doplnit překlad toho kterého řetězce. Přeložené soubory .po zbývá transformovat do strojově čitelného překladu (který je obvykle komprimován). Tento soubor pak nese příponu .mo a jeho název by měl být shodný s názvem aplikace, v našem případě tedy bws.mo.

Soubory .mo již stačí pouze nahrát na určené pozice. V operačním systému Kubuntu jsou tyto pozice /usr/share/bws/locale/en/LC_MESSAGES/, kde adresář en nahradíme adresářem, který je pojmenován dle jazyka, jehož překlad chceme nahrát do systému.

5.7 Závislost na technologiích

Jelikož je aplikace bws zaštřešujícím balíčkem, který sdružuje již existující technologie do jediného uživatelského rozhraní, je tedy na místě předpokládat, že je na těchto aplikacích silně závislý.

Aplikace bws je závislá zejména na změně databázového schématu aplikace bugzilla, dále pak na změně šifrování hesla uživatele aplikace bugzilla. Změna aplikační logiky aplikace bugzilla by se neměla projevit.

Závislost bws na aplikaci MoinMoin se může projevit při změně syntaxe pro tvorbu vlastních stránek, dále při změně způsobu ukládání stránky. Naopak změna způsobu vytváření uživatele by se neměla projevit, pokud zůstane stejná syntaxe volání skriptu, který má vytvoření uživatele za úkol.

Na aplikaci subversion je aplikace bws závislá víceméně pouze na syntaxi příkazů aplikace subversion.

Následující tabulka zachycuje parametry operačního systému a softwarových balíčků, na kterých byla aplikace bws vyvinuta. V případě, že bude aplikace bws používána v systému s jinými parametry, není zaručeno, že bude pracovat správně.

Tabulka 5-2: Parametry vývojového prostředí

Název	Označení
Operační systém	Kubuntu 8.04
Linuxové jádro	linux-generic_2.6.24.24.26_i386.deb
Programovací jazyk	python_2.5.2-0ubuntu1_all.deb
Bug trackingový systém Bugzilla	bugzilla_2.22.1-2.2ubuntu1.8.4.1_all.deb
Wiki systém MoinMoin	python-moinmoin_1.5.8-5.1ubuntu2.2_all.deb
Webový server Apache2	apache2_2.2.8-1ubuntu0.5_all.deb
Modul web_dav	libapache2-svn_1.4.6dfsg1-2ubuntu1_i386.deb
Databázový server	mysql-server_5.0.51a-3ubuntu5.4_all.deb
Verzovací systém	subversion_1.4.6dfsg1-2ubuntu1_i386.deb

5.8 Struktura implementovaného debianovského balíčku

Debianovský balíček je jediný soubor, který je archivem, ve kterém se nacházejí dva adresáře:

Control – obsahem tohoto adresáře jsou řídící skripty balíčku, které jsou použity při instalaci, odinstalaci a konfiguraci. Tyto skripty lze obecně psát v libovolném skriptovacím jazyce, kterým však musí být server, na nějž se aplikace instaluje, vybaven. V debianovském balíčku rozeznáváme tyto

významné skripty (jejich přesný název musí být dodržen). Tyto skripty mohou, ale nemusí být obsaženy v balíčku.

- **preinst** – je skript, který je volán před zahájením instalace, v projektu není použit.
- **postint** – je skript, který je volán po úspěšném ukončení instalace balíčku do systému, v projektu není použit.
- **prerm** – je skriptem, který je volán před zahájením odinstalace. V aplikaci je tento skript použit pro smazání všech aplikací vytvořených projektů. Skript je napsán v programovacím jazyce Python.
- **postrm** - je skriptem, který je volán po úspěšné odinstalaci, a není v projektu použit.

Dále adresář control obsahuje dva neméně důležité soubory. Prvním z nich je soubor md5sum, který obsahuje kontrolní součty všech souborů, které jsou obsaženy v datové části balíčku. Kontrolní součty jsou použity správcem balíčků pro kontrolu, zda je obsah balíčku konzistentní.

Nejdůležitějším souborem v adresáři control je pak právě soubor se shodným názvem – control. Tento soubor musí být vždy v každém debianovském balíčku obsažen. Tento soubor totiž obsahuje vitální údaje pro balíčkovací systém, který podle nich ví, jak s balíčkem zacházet. Zejména se jedná o informace názvu aplikace, verze balíčku a architektuře, pro niž je balíček naprogramován (v našem případě je jednu o architekturu i386). Dále obsahuje infomační data jako jméno správce balíčku (ten se může lišit od tvůrce), velikosti, kterou aplikace zabere po instalaci a popis balíčku. Avšak nejdůležitějšími informacemi, které nese soubor control, jsou informace o vztahu našeho balíčku k jiným softwarovým balíčkům. Nejdůležitějším vztahem je závislost na jiných balíčcích, neboť bez těchto by aplikace nemohla správně fungovat. Uvádíme seznam balíčků, na nichž je aplikace závislá a bez nichž nebude správně fungovat:

- bugzilla
- python-moinmoin
- apache2
- libapache2-svn
- mysql-server
- subversion

Pro lepší fungování aplikace je také navržen balíček kdesvn.

Balíčkovací systém pak sám na základě závislostí navrhne a pohlídá správnost postupu instalace aplikace.

Data – obsahem tohoto adresáře jsou výsledné zdrojové soubory aplikace. Obsah tohoto adresáře musí přesně kopírovat strukturu kořenového adresáře, do něhož bude aplikace instalována. Tato podkapitola čerpá z instruktážního videa, dostupného na serveru YouTube [18].

5.9 Struktura archivu tar.gz

Výsledná aplikace je dostupná také v podobě tar gzipovaného archivu, který je strukturou velice podobný debianovskému balíčku. Následuje popis této struktury:

files – obsah tohoto adresáře je prakticky totožný s obsahem adresáře obsahující data z debianovského balíčku. Obsah adresáře files se bude stejně jak je tomu u debianovského balíčku nakopírován na patřičná místa.

install – je instalační skript aplikace. Tento skript se stará právě o výše zmíněné nakopírování souborů na příslušná místa.

prerm – je skriptem, který je totožný se skriptem z debianovského balíčku a má také stejnou funkcionalitu a sice před instalací vymazat všechny projekty aplikace. Fakt, že je volán před zahájením odinstalace aplikace ze systému, je implementován tak, že níže zmíněný skript uninstall volá skript prerm jako první příkaz.

uninstall – je skriptem, který se stará o kompletní odinstalaci aplikace. Stejně jako skript install je i tento napsán v programovacím jazyce Python.

Tento archív je skutečně jen komprimovaným archívem souborů a nutné závislosti nijak neřeší a je pouze na uživateli, který aplikaci instaluje, aby se postaral o naplnění závislostních požadavků. Postup instalace je uveden v návodů, který je přiložen jako příloha na konci diplomové práce.

6 Závěr

Tato diplomová práce popisuje analýzu, návrh a vývoj aplikace pro snadnou instalaci a konfiguraci informačního systému pro správu projektů a nenavazuje na žádnou jinou letos dokončovanou diplomovou práci. Cílem diplomové práce bylo vytvořit nástroj, který umožňuje vytvářet informační systém, který umožňuje spravovat libovolný počet projektů. Tento systém měl být jednoduše instalovatelný a konfigurovatelný.

Výsledkem diplomové práce je informační systém, který pokrývá všechny základní požadavky, které souvisí se správou informačního systému a které byly specifikovány v kapitole analýzy. Výhodou implementovaného řešení je ta skutečnost, že při vytvoření nového projektu dochází k vytvoření pouze datové části a programová logika je stále jen jedna. Další implementované řešení je, to, že vyškeré uživatelské příkazy aplikace bws jsou parametrizovatelné, a tudíž lze vše vykonat jediným příkazem volaným z příkazové řádky, z čehož plyne, že implementovaný systém lze snadno použít jako vestavěný systém pro jiné aplikace, které by ho chtěly případně používat.

Dalším směrem pro rozšíření je implementace grafického rozhraní, které by práci s aplikací bws ještě více zjednodušilo a zpříjemnilo. Aplikace bws lze také dále rozšířit o používání technologií, které pokrývají další aspekty správy projektu nebo další technologie, které pokrývají stejnou oblast správy projektu. Do aplikace bws lze také implementovat více šablon pro práci se správou projektu a tím uživateli ještě více zjednodušit správu projektu.

Dalším možným vývojem může být rozšiřování projektu tak, aby byl schopen spravovat i rozsáhlé projekty, avšak vždy musí být na výběr volba správy jednoduchého projektu s minimálními požadavky.

Literatura

[1] Neznámý kolektiv autorů: HTML [online]. Wikipedia, aktualizováno 2009-04-17 [cit. 2009-04-18]. Dostupné na URL : <http://cs.wikipedia.org/wiki/HTML>

[2] Simpson, John E.: Will XML replace HTML? [online]. O'Reily Media Inc., 2009, aktualizováno 2009-04-18 [cit. 2009-04-18]. Dostupné na URL: <http://www.xml.com/pub/a/2000/12/13/xmlhtml.html>

[3] Neznámý kolektiv autorů: PHP [online]. Wikipedia. aktualizováno 2009-04-15 [cit. 2009-04-18], Dostupné na URL: <http://cs.wikipedia.org/wiki/Php>

[4] Neznámý autor.: April 2009 Web Server Survey [online]. NETCRAFT LTD, 2009, aktualizováno 2009-04-06 [cit. 2009-04-18]. Dostupné na URL: <http://news.netcraft.com/archives/2009/04/index.html>
Dostupné na URL: <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>

[5] Blandy, J.: Introduction to CVS [online]. Ximbiot, 2007. [cit. 2009-04-22] Dostupné na URL: <http://ximbiot.com/cvs/cvshome/docs/blandy.html>

[6] Collins-Sussman, B., Fitzpatrick, B. W., Pilato, C. M:, Version Control with Subversion For Subversion 1.6. TBA, 2008, aktualizováno 2008 [cit. 2009-04-22] Dostupné na URL: < http://svnbook.red-bean.com/nightly/en/svn-book.pdf>

[7] Kreslíková, J., Martínek, Z.: Management projektů Studijní opora [online], 2007, [cit. 2009-04-22] Dostupné na URL: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/MPR-IT/texts/mpr-opora-v2.pdf>

[8] Neznámý autor. ViewVC — Web-based Version Control Repository Browsing [online]. Collaboration, 2008, Dostupné na URL: <http://viewvc.tigris.org/>

[9] Strmiska, L. Správa verzí [online]. Dostupný na URL: <http://www.fi.muni.cz/~kas/p090/referaty/2006-podzim/st/xstrmisk-verze.html>

[10] Waldman, T. a další: MoinMoinWiki [online]. aktualizováno 2009-04-27 [cit. 2009-04-27], Dostupné na URL: <http://moinmoin.wikiwikiweb.de/MoinMoinWiki>

[11] Gohr, A.: DokuWiki [online]. DokuWiki, 2007, aktualizováno 2009-03-08 [cit. 2009-04-27] Dostupné na URL: < http://www.dokuwiki.org/cs:dokuwiki>

[12] Neznámý kolektiv autorů: Twiki [online]. TWiki, 2007, aktualizováno 2005-07-23 [cit. 2009-04-27] Dostupné na URL: <http://twiki.org/cgi-bin/view/TWiki04x01/WebHom>

[13] Neznámý kolektiv autorů. Extension:BugSquish. MediaWiki, aktualizováno 2009-03-03 [cit. 2009-04-28] Dostupné na URL: <http://www.mediawiki.org/wiki/Extension:BugSquish>

[14] Chumakov, A. a kol.: Mantis [online]. Mantis, 2009, [cit. 2009-04-28] Dostupné na URL: <http://www.mantisbt.org/>

[15] Curtis, B.: phpBugTracker. phpBugTracker, 2009, [cit. 2009-04-28] Dostupné na URL: <http://phpbt.sourceforge.net/>

[16] The Scarab Development Team: Scarab – Administration Guide [online], [cit. 2009-05-10] Dostupné na URL: <http://www.tigris.org/files/documents/11/36536/scarab-admin-en-10425.pdf>

[17] Russel R., Quinlan D., Yeoh Ch.: File System Hierarchy Standard [online]. File System Hierarchy Standard Group, 2004, aktualizováno 2004-01-28 [cit. 2009-05-25], Dostupné na URL: <http://www.pathname.com/fhs/pub/fhs-2.3.pdf>

[18] Oxer, J.: Anatomy of a Debian Package [online]. 2006-07-21, [cit. 2009-05-04] Dostupné na URL: <http:// http://www.youtube.com/watch?v=lFzPrzY2KFM>

Seznam příloh

Uživatelský manuál

Co je to bws?

- Aplikace umožňující rychlejší, snadnější a přehlednější způsob vytváření domovských stránek projektů.
- Aplikace zastřešující používání obvyklých technologií použitých při vytváření domovských stránek projektů.
- Aplikace sjednocující rozhraní pro konfiguraci jednotlivých technologií do jednoho místa.
- Soubor návodů jak postupovat při vytváření a správě projektu.
- Pomocník nadšených začátečníků, kteří nechtějí trávit hodiny pročítáním manuálů a chtějí vyvíjet software.
- Pomocník zkušených programátorů, kteří nechtějí trávit čas při administraci projektu.

Aplikace bws klade důraz zejména na:
Jednoduchost použití / jeden příkaz nahrazuje sérii jednoduchým úkonů.

- Rozšiřitelnost / většinu obsahu si můžete přetvářet ke svému obrazu.

Použité technologie

- Apache2 / webový server. Rychlý, výkonný a spolehlivý. Je použit pro obsluhování uživatelských dotazů.
- Aplikace bws automaticky konfiguruje přístup serveru Apache2 k datům aplikace.
- Bugzilla / bug trackingový systém (systém pro hlášení a sledování stavů chyb produktu).
- Aplikace bws automaticky konfiguruje instanci aplikace bugzilla a přístup k ní.
- Subversion / verzovací systém, který umožňuje spravovat verze produktu, udržuje všechny změny a řeší paralelní přístup k vyvíjenému produktu.
- Aplikace bws zjednodušuje vytváření repozitáře a jeho provázání s internet.
- MoinMoin / souborový wiki systém.
- Aplikace bws tento systém připraví pro vaše použití, ať už jej naplníte jakýmkoliv obsahem.
- MySql / databázový systém využívaný bugzillou.
- Python / objektový programovací jazyk, který byl použit pro vytvoření aplikace bws.

První kroky

Před instalací

- Než začneme se samotnou instalací je nutno získat a zapamatovat či zapsat si některé údaje, které budeme potřebovat zadat po instalaci aplikace bws.
- Jedná se zejména o tyto informace:

- Přístupové heslo do databáze MySQL (pokud máte superuživatelská práva serveru, měli byste tento údaj znát, pokud ne, kontaktujte správce serveru).
- Uživatelské jméno, pod nímž pracuje server Apache2.
- Název uživatelské skupiny, po níž pracuje server Apache2.
- Název serveru, na kterém poběží aplikace bws. (pravděpodobně localhost)

- Dále je vhodné zjistit, zda jsou nainstalovány a nakonfigurovány všechny potřebné balíčky pro instalaci a správné použití aplikace bws.
- Jedná se zajména o tyto balíčky:
  - python
  - bugzilla
  - subversion
  - apache2
  - mysql/server
  - python/moinmoin

Instalace

5. Stáhneme balíček aplikace bws.(Použijte balík s příponou .deb pro systém s debianovským balíčkovacím systémem, nebo .tar.gz archiv)
6. V případě, že jste stáhli tar.gz archiv rozbalte jej pomocí přikazu: `tar .xvvzf \ Nazev_Balicku.tar.gz`, kde Nazev_Balicku je název balíčku aplikace bws.
7.
   - V případě, že používáte tar.gz archiv. Zkontrolujte, zda jsou splněny všechny závislosti pro instalaci balíčku a že všechny potřebné programy jsou správně nakonfigurovány. Vyhledejte rozbaleném adresáři soubor install.py a spuste jej z příkazové řádky pomocí příkazu: `sudo ./install.py`, ke spuštění tohoto příkazu budete potřebovat superuživatelská práva.
   - V případě, že používáte operační systém vycházející ze systému debian a stáhli jste si `.deb` balíček. Balíček nainstalujete pomocí příkazu: `sudo dpkg install \ Nazev_Balicku.deb`, kde Nazev_Balicku je název balíčku aplikace bws. V případě, že nejsou splněny závislosti, bude Vám tato skutečnost oznámena.
8. Aplikace je nainstalována a připravena k použití.

Po instalaci

Nyní je třeba dokončit konfiguraci aplikace bws.

1. V textovém editoru otevřeme soubor `/etc/bws/config.txt` (Nejjednodušším způsobem je spustit v příkazové řádce příkaz: sudo mc , který zobrazí midnight commandr, což je grafické prostředí pro správu souborů. Nalezneme daný soubor a stiskem klávesy F4 vyvoláme editační okno).

2. V konfiguračním souboru zeditujeme následující řádky:
   - `ApacheUser=`
   - `ApacheGroup=`
   - `DatabasePassword=`
   - `ServerHostName=localhost`
   - `SendmailServer=localhost`

Řádky `ServerHostName` a `SendmailServer` můžeme ponechat nastaveny na hodnotu localhost, pokud budeme provádět operace s aplikací bws přímo na serveru.
3. Uložíme změny a zavřeme editor.

První projekt

Vytvoření projektu

1. Pro vytvoření nového projektu je zapotřebí otevřít příkazovou řádku a zadatpříkaz:
   `sudo bws_project`
2. Při vytváření projektu bude uživatel vyzván k zadání názvu projektu. Název projektu musí být unikátní v celé aplikaci bws. Proto pokud bude zadán název již existujícího projektu, bude tato skutečnost oznámena uživateli.
3. V dalším čase může uživatel sledovat průběh instalace.
4. Poté bude uživatel dotázán, zda chce naimportovat existující repositář.
   - Zvolte ano pokud chcete naimportovat Váš již existující svn repositář obsahující váš projekt, můžete tak učinit není. Vybral/li jste možnost ano, pak budete dotázán na kompletní adresu k vašemu repozitáři (např. http://adresa.k.vašemu.repozitary). Naimportování projektu je možné později.. Pozor, aplikace bws umí pracovat pouze s repozitářem, který má strukturu v doporučené formě. Tedy trunk, tag, branches.
   - Zvolte ne pokud nechce naimportovat repositář, v takovém případě bude vytvořen nový. Taktéž lze naimportovat adresář později pomocí příkazu:
     `sudo bws_svn_import_repository`
5. Dále bude uživatel dotázán na informace týkající se vytvoření administrátorského účtu pro administrátora serveru Bugzilla:
   - skutečné jméno
   - email
   - heslo

Přidání uživatele

1. Pokud je již projekt úspěšně vytvořen, je možno vytvořit uživatele projektu. Uživatele projektu vytvoříte zadáním příkazu: `sudo bws_add_user`
2. V dalším kroku bude uživatel dotázán na název projektu, ke kterému má být vytvořen uživatelský účet. Při vytváření projektu bude uživatel vyzván k zadání názvu projektu. Název projektu musí

být unikátní v celé aplikaci bws. Proto pokud bude zadán název již existujícího projektu, bude tato skutečnost oznámena uživateli.

3. Po úspěšném zadání názvu projektu bude uživatel vyzván k zadání uživatelského jména, pod kterým má být vytvořen přístup k projektu. Pokud uživatel se zadaným jménem v projektu již existuje bude tato skutečnost ohlášena uživateli.
4. Dále bude uživatel vyzván k vložení emailu vytvářeného uživatele. (Tento údaj je použit při přístupu do aplikace Bugzilla).
5. Posledním krokem pro vytvoření uživatele je vložení hesla uživatele.

Základní práce s SVN

1. Pokud máte vytvořen projekt, můžete již nyní používat vytvořeného svn repositáře, pro správu zdrojového kódu Vašeho projektu. Repositář svn je umístěn na adrese: http://www.domena.cz/bws/nazevVasehoProjektu/svn
2. Zjistěte si od administrátora projektu uživatelské jméno, heslo a cestu k repositáři.
3. Nejdříve si na svém lokálním počítači vytvořte složku, ve které budete chtít mít uložen pracovní kopii Vašeho projektu. Pro účely ukázky budeme předpokládat, že repositář projektu se nachází na url adrese: http://www.domena.cz/bws/projekt2/svn. Námi vytvořený adresář se jmenuje: mojePracovniKopie a plná cesta k němu je: /home/Karel/mojePracovniKopie Jméno uživatele, pro přístup k repositáři je: Karel
4. Příkazem: `svn username Karel checkout \ http://www.domena.cz/bws/projekt2/svn /home/Karel/mojePracovniKopie` stáhneme ze serveru obsah repozitáře, do adresáře `/home/Karel/mojePracovniKopie`.
5. Všimneme si struktury adresářů v adresáři `mojePracovníKopie`. Vidíme adresáře trunk, branches a tags.
   - **trunk /** je adresářem, který slouží k ukládání zdrojového kódu a budete tedy pracovat především s tímto adresářem
   - **branches /** je adresářem, který slouží pro práci s větvemi projektu, pokud jste vývojářem projektu, bude Vám sděleno, na které větvi projektu máte pracovat a také jak dostat k datům v konkrétní větvi projektu
   - **tags /** je adresářem, kde se uchovávají tagy projektu, tedy stav v projektu v určitém čase. Tagování je aplikaci bws použito pro následné vytvoření releasu projektu.
6. Před každou započatou prací na nějakém logickém celku projektu provedeme příkazem: `svn \ username Karel update /home/Karel/mojePracovniKopie` aktualizaci naší pracovní kopie.
7. Změny kódu provádíme v adresáři trunk, případně v branchi (ve větvi projektu), která nám byla určena.

8. Vykonané změny uložíme na server příkazem: `svn --username Karel commit \ /home/Karel/mojePracovniKopie`. Příkaz svn commit ukládá na server jen změny v souborech, které již na serveru byli. Pokud chceme přidat do repozitáře soubory, provedeme tak příkazem: `svn add cestaKSouboru`, následované příkazem `commit`. V případě, že chceme naopak některé soubory z adresáře smazat, použijeme příkaz: `svn delete cestaKSouboru`, následované příkazem `commit`.
9. Vytvořené změny jsou uloženy v repositáři.

Pro práci se subversion je také doporučeno použít některého z grafických správců subversion, například programu RapidSVN, KdeSVN nebo podobných.

Anatomie bws

Před započetím zkoumání obsahu stránek je doporučeno důkladně prostudovat `HelpOnEditing` a `SyntaxReference`. Aplikace bws obsahuje šablonu pro snadnější vytváření stránek s úkoly. Doporučuje se všechny stránky s úkoly ukládat jako potomky stránky `Tasks`. Aplikace bws obsahuje šablonu pro snadnější vytváření stránek pro správu zdrojů. Doporučuje se všechny stránky s úkoly ukládat jako potomky stránky `Resources`. Aplikace bws obsahuje stránku `Download`, která se sama aktualizuje při vytvoření releasu projektu. Aplikace bws obsahuje stránku `FAQ`, pro otázky a odpovědi.

Použití šablony úkolů

1. Vytvoříme stránku úkolu pomocí šablony, kterou posléze zeditujeme.
2. V `== Name of task ==` nahradíme text Name of task názvem úkolu.
3. V části Planning nahradíme údaje `dd.mm.yyyy hh:mm` správnými časovými údaji ve formátu den.měsíc.rok hodina:minuta, případně vynecháme údaje hodina:minuta.
   - Údaj Planned start označuje plánovaný začátek řešení úkolu.
   - Údaj Planned end označuje plánovaný konec řešení úkolu.
   - Údaj Started označuje skutečný začátek řešení úkolu a může se lišit od plánovaného začátku, pokud je datum Started pozdější než datum Planned start, hrozí riziko, že může dojít ke zpoždění dodávky projektu.
   - Údaj Ended označuje skutečné ukončení řešení úkolu a může se lišit od plánovaného začátku, pokud je datum Ended pozdější než datum Planned end, hrozí zvýšené riziko, že dojde ke zpoždění dodávky projektu, zvláště pak, pokud je úkol na tzv. kritické cestě (kritická cesta je složena z úkolů, které nesmí být zpožděny, jinak dojde ke zpoždění dodávky projektu.
4. V části Dependencies nahradíme hodnoty: `Tasks/Task3: Task3 name`, tak že hodnota: `Tasks` představuje cestu ke všem úkolům a `Task3` představuje název stránky, kde je tento úkol, `Task3` name je skutečným názvem úkolu. Tato tabulka tedy obsahuje seznam všech úkolů, které musí být ukončeny před tím, než může začít tento úkol.

5. V části Resources nahradíme hodnoty: `Resources/Resource1name:Resource1 name`, tak že hodnota: `Resources` představuje cestu ke všem zdrojům a `Resource1name` představuje název stránky, kde je tento zdroj popsán, čast za druhou dvojtečkou, `Resource1 name`, je skutečný název zdroje. Analogicky je tomu: `Users/User1 name:User1 name`. Hodnoty `percent used` nahradíme procentuální hodnotou využití zdroje. Tedy například 200% představuje použití 2 jednotek zdroje.
6. Do části Description dopíšeme detailní popis úkolu.

Použití šablony zdrojů

1. Vytvoříme stránku zdroje pomocí šablony, kterou posléze zeditujeme.
2. V `== Name of resource ==` nahradíme text Name of resource skutečným názvem zdroje.
3. V části Counts nahradíme `aValue` celočíselnými nezápornými hodnotami, přičemž jednotlivé řádky mají následující význam.
   - **Free units / volné jednotky zdroje**, tedy počet jednotek zdroje, které nejsou nikým užívány (může se jednat například o dva nepoužívané počítače)
   - **Allocated units / momentálně využívané zdroje** (například dvě knihy, které jsou momentálně studovány vývojáři)
4. Část Allocations and Reservations vyplníme následovně:
   - V poli Start date nahradíme hodnotu `dd.mm.yyyy` datumem od kdy je jednotka (jednotky) zdroje zarezervována, nebo v užívání.
   - V poli End date nahradíme hodnotu `dd.mm.yyyy` datumem do kdy je jednotka zarezervována pro použití nebo používána.
   - V poli Count of units nahradíme text `aValue` hodnotou počtu zarezervovaných či používaných jednotek zdroje.
   - V poli Type si vybereme hodnotu `Reservation` nebo `Allocation` (druhou smažeme), dle toho, zda je zdroj zarezervován, nebo v používání.

Vytváření releasu

1. Vytvoření releasu provedeme zadáním příkazu: `sudo bws_create_release` do příkazové řádky.
2. Následuje dotaz na jméno projektu, zvolíme tedy název projektu.
3. Skript, jež jsme zavolali automaticky vytvoří tag a vyvolá commit tohoto tagu do repositáře projektu. Název releasu bude release/0.0.0, kde číslo 0.0.0 bude vždy o jedna zvýšeno oproti předchozí verzi. Takto vytvořený tag bude zabalen do adresáře pro releasy projektu a dále bude automaticky vložen jako příloha stránky `Download`.

Pokud chceme vytvořit novou verzi projektu, zavoláme příkaz: `sudo bws_create_release .v 0.0.0.0.1`, kde místo 0.0.0.1 vložíme číslo požadované verze projektu v tečkové notaci.

Používání bugzilly

1. Rozhraní aplikace bugzilla vašeho projektu lze najít na adrese: http://www.domena.com/bws/nazevVasehoProjektu/bugzilla
2. Po zadani této adresy dojde k přesměrování na domovské stránky aplikace bugzzila. Změnila se URL adresa.
3. V patičce stránky nalezneme odkaz Log in, na který klikneme a zobrazí se přihlašovací stránka aplikace bugzilla.
4. Zadáme přihlašovací údaje, které se skládají:
   - z uživatelského jména / které je emailem uživatele. Pokud jej nevíte, kontaktujte administrátora projektu.
   - hesla / které bylo zadáno při vytváření uživatele. Pokud jej nevíte, kontaktujte administrátora projektu.
5. Pokud jste administrátorem projektu a otevíráte Bugzillu poprvé, bude nutno vytvořit nový produkt, který bude bugzilla spravovat.
   - V patičce klikněte na odkaz Products.
   - Dále klikněte na odkaz Add.
   - V zobrazeném formuláři vyplňte hodnoty:
   - Product / názvem vašeho produktu (nebo projektu)
   - Description / popis vašeho produktu
   - Případně vyplňte další hodnoty.
6. Po stisknutí tlačítka Add, se zobrazí výstražná hláška: You will need to add at least one component before you can enter bugs against this product. Klikneme tedy na odkaz add at least one component.
7. V následujícím formuláři vyplníme hodnoty polí:
   - Component / názvem komponenty, která se bude vyskytovat v projektu.
   - Description / jednoduchým popisem komponenty
   - Default assigniee / emailovou adresou uživatele, který bude považován za výchozího uživatele jemuž se budou přiřazovat nalezené bugy (chyby).
   - Uživatel musí v aplikaci bugzilla existovat.
8. Nyní je možné vytvářet a vyhledávat bugy.