UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Platforma pro týmový vývoj softwarových děl v PHP

Bc. Petr Vacek

Diplomová práce
2012

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Vacek**
Osobní číslo:	**I09390**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Platforma pro týmový vývoj softwarových děl v PHP**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V úvodní části práce bude představena problematika týmového vývoje softwarových děl se zaměřením na zachycení a popsání platforem, či jednotlivých částí, které svojí funkcionalitou pokrývají komplexně vývoj softwaru v jazyce PHP. Budou analyzovány dostupné produkty s důrazem kladeným na open source a budou analyzovány a zhodnoceny možnosti jejich vzájemné integrace.
V aplikační části práce budou konkrétně rozebrány a implementovány jednotlivé systémy, které jsou běžně využívány při vývoji SW a to minimálně v rozsahu: systém správy verzí, centrální úložiště identit, systém pro správu chyb a požadavků, dokument management systém, management znalostí, integrované vývojové prostředí (IDE).
Výsledkem diplomové práce bude integrovaná platforma, která rozsahem své funkcionality bude pokrývat základní požadavky na týmový vývoj softwarových děl a to v minimálním rozsahu výše popsaných prvků (jak serverová tak i klientská část). K vytvořené platformě budou zpracovány metodické pokyny pro administrátory a programátory.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Gutmans, Andi, Rethans, Derick a Bakken, Stig. 2007. Mistrovství v PHP 5. Praha : Computer press, 2007. ISBN: 978-80-251-1519-0.
2. Price, Brad. 2005. Active Directory. Praha : Computer press, 2005. ISBN: 80-251-0602-0.
3. Schroder, Carla. 2009. Linux. Praha : Computer press, 2009. ISBN: 978-80-251-2407-9.
4. Štědroň, Bohumír. 2009. Open Source software. Praha : Grada, 2009. ISBN: 978-80-247-3047-9.

Vedoucí diplomové práce: **Ing. Lukáš Čegan, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2011**
Termín odevzdání diplomové práce: **18. května 2012**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2011

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 30. 08. 2012 Bc. Petr Vacek

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Lukáši Čeganovi, Ph.D. za vedení, rady a připomínky při tvorbě této práce.

Anotace

Práce popisuje problematiku týmového vývoje se zaměřením a popsáním jednotlivých částí platformy, která svoji funkcionalitou komplexně pokrývá vývoj softwarových děl v jazyce PHP. Aplikační část obsahuje popis a implementaci jednotlivých systémů využívaných při vývoji: systém správy verzí, centrální úložiště identit, systém pro správu chyb a požadavků, dokument management systém, management znalostí a integrované vývojové prostředí. Vytvořená platforma obsahuje metodické pokyny pro administrátory a programátory.

Klíčová slova

Týmový vývoj, php, platforma, software, UML

Title

Platform for team development software in PHP.

Annotation

This work describes the issue of team development with focus on and describing various parts of platform that comprehensively covers the functionality of software development in PHP language. Application section contains a description and implementation of software used in development: control version system, bug tracking system, document management system, knowledge management and integrated development environment. The developed platform providing guidelines for administrators and programmers.

Keywords

Team development, php, platform, software, UML

Obsah

Seznam zkratek

API	Application Programming Interface
APT	Advanced Packaging Tool
CMM	Capability Maturity Model
CVS	Concurrent Version System
DIT	Directory Information Tree
DMS	Dokument Management Systém
EDM	Electronic Document management
ERD	Entity-Relationship Diagram
GPL	General Public License
HTML	HyperText Markup Language
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IS	Information System
LDAP	Lightweight Directory Access Protocol
OSS	Open-Source Software
PHP	Hypertext Preprocessor
RDN	Relative Distinguished Name
RUP	Rational Unified Process
SDP	Solution Deployment Process
SEI	Software Engineering Institute
SQL	Structured Query Language
SVN	Subversion
UML	Unified Modeling Language
UP	Unified Process

Seznam obrázků

Úvod

Dnešní vývoj menších i rozsáhlých aplikací vyžaduje součinnost více lidí. Tuto součinnost nazýváme týmový vývoj a přináší nám mnoho výhod, které vedou softwarové dílo k úspěšnému dokončení a předání zákazníkovi.

Úmyslem týmového vývoje není pouze rozložení mnoho programátorské práce na více vývojářů, ale především i specializaci každého jednotlivce na konkrétní část vývoje. Týmová spolupráce přináší výhody, které rozvíjejí jednotlivce týmu. Je to díky sdílení informací, poznatků a zkušeností, které tým získal na předcházejících projektech. Ostatním toto umožní snadnější integraci do procesu vývoje softwarového díla.

Spolupráce týmu se ale neobejde bez dalších nástrojů (softwarů), které slouží pro správu všech informací na projektu. Jedná se např. o zmiňované poznatky a zkušenosti programátorů. Nástrojů na trhu je celá řada. Nevýhodu však přináší fakt, že některé přináší vysoké náklady a to si ne každá firma může dovolit. Potom přichází zajímavé ohlédnout se na tzv. open source produkty, jejichž většina bývá svobodně šiřitelná s možností využití jak v komerčním tak v nekomerčním prostředí. Díky dostupnému zdrojovému kódu je potom možné nástroje přizpůsobit dle firemních požadavků a dále dle potřeby i vzájemně propojit, integrovat atd. Sdílení dat mezi integrovanými produkty potom přináší úsporu času a snadnější správu aplikací.

1 Vývoj softwaru

Tato kapitola obsahuje úvod do softwarového inženýrství, popis jednotlivých metodik a fází vývoje a poukazuje na smysl týmového vývoje při vývoji softwaru.

1.1 Softwarové inženýrství

Softwarové inženýrství můžeme charakterizovat jako činnost, která zahrnuje inženýrství, informatiku a management. Zabývá se problematikou vývoje rozsáhlých softwarových děl.

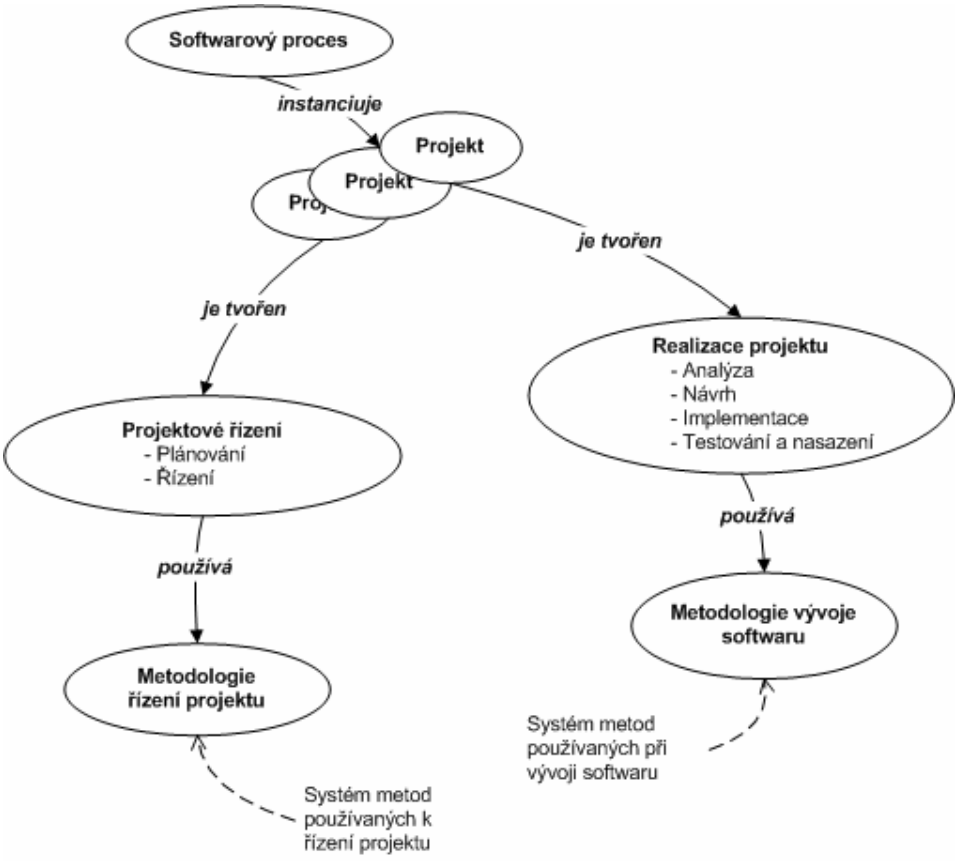
Vývoj softwarových systémů zahrnuje mnoho faktorů, které musí být splněny, abychom dosáhli požadovaného produktu:

- odpovídající hardwarové a softwarové vybavení,
- netechnické aspekty dané organizační strukturou a ekonomické možnosti organizace vyvíjející produkt,
- znalost z oblasti specifikace požadavků na produkt,
- analýza produktu, návrh, implementace, testování a nasazení u zákazníka,
- kvalitní lidské zdroje (analytici, programátoři, testeři atd.), kteří jsou schopni efektivně aplikovat znalosti problematiky oblasti produktu při samotné realizaci tohoto produktu,
- projektové řízení vývoje samotného produktu, které umožňuje efektivní využití všech jmenovaných faktorů, s cílem vytvořit produkt požadované kvality. [1]

1.1.1 Softwarový produkt

Vývoj softwarového produktu vyžaduje tzv. *softwarový proces*, ten zahrnuje řadu činností a metod, které byly jmenovány v předcházející kapitole. Softwarový proces, který nám říká jak vytvořit výsledný produkt, definuje následně projekty, které se nazývají *instance procesu*. Jednotlivé instance jsou pak vztažené k zakázkám. Dále je každá instance procesu tvořena *realizací* – činnosti závislé na vývoji samotného produktu a *řízením* – projektové řízení. Technická i netechnická část vývoje vyžaduje svou *metodologii*. V případě řízení projektu hovoříme *o metodologii projektového řízení*, která představuje metody využívané v projektovém řízení, zatímco *metodologie vývoje softwarového systému* obsahuje systém metod využívaných při vývoji softwaru. [1]

Schéma vývoje softwarového produktu je znázorněno pomocí sémantického grafu, viz Obrázek 1.



Obrázek 1 – Sémantický graf vývoje softwarového produktu, zdroj [1]

1.1.2 Softwarový proces

Softwarový proces představuje uspořádanou množinu kroků, které směřují k vytvoření nebo modifikaci softwarového díla. Krok potom představuje nějakou *aktivitu* nebo *podproces*, které mohou probíhat v čase souběžně – vyžadováno řízení.

Pro každý použitý softwarový proces je důležité, aby se nechal využít na více projektech – *znovupoužitelnost*. Cílem tohoto využití je maximální efektivita a výsledky vysoké kvality.

Pomocí stupnice SEI (Software Engineering Institute) můžeme dělit úroveň a využití samotného softwarového procesu. Stupnice je definována body 1 – 5, které vyjadřují kvalitu firmy nebo organizace v konkrétním měřítku. Tento model hodnocení vyspělosti a schopností dodavatele softwarového produktu se nazývá CMM (Capability Maturity Model). [1]

Jednotlivé úrovně úrovně stupnice SEI můžeme charakterizovat takto:

1. *Počáteční (Initial)* – není definován softwarový proces, jednotlivé projekty jsou řešeny případ od případu (ad hoc).
2. *Opakovatelná (Repeatable)* – identifikace opakovatelných postupů v jednotlivých projektech, následná reprodukce v novém projektu.
3. *Definovaná (Defined)* – softwarový proces je definován a dokumentován na základě integrace dříve identifikovaných opakovatelných kroků.

4. *Řízená (Managed)* – schopnost řízení a monitorování na základě definovaného softwarového procesu.
5. *Optimalizovaná (Optimized)* – informace získané dlouhodobým procesem monitorování softwarového procesu jsou využity ve prospěch optimalizace softwarového procesu. [1]

1.2 Metodiky využívané při vývoji softwaru

Proces vývoje softwaru (SDP), znám i jako metodika tvorby softwarového vybavení (SEP) definuje hlavní otázky vývoje – *kdo, co, kdy* a *jak* [2]. Můžeme jej také popsat jako proces, ve kterém jsou uživatelské požadavky realizovány vytvořeným softwarem.

Metodika vývoje softwaru definuje přístup k procesu vývoje, použité nástroje, modely a používané postupy. Postupem doby bylo vyvinuto velké množství různých metodik, ne každá je však vhodná pro každý projekt.

Při výběru správné metodiky je dobré vycházet ze zdrojů, které jsou k dispozici a cílů, k jakým má sloužit. Základními faktory při výběru vhodné metodiky jsou:

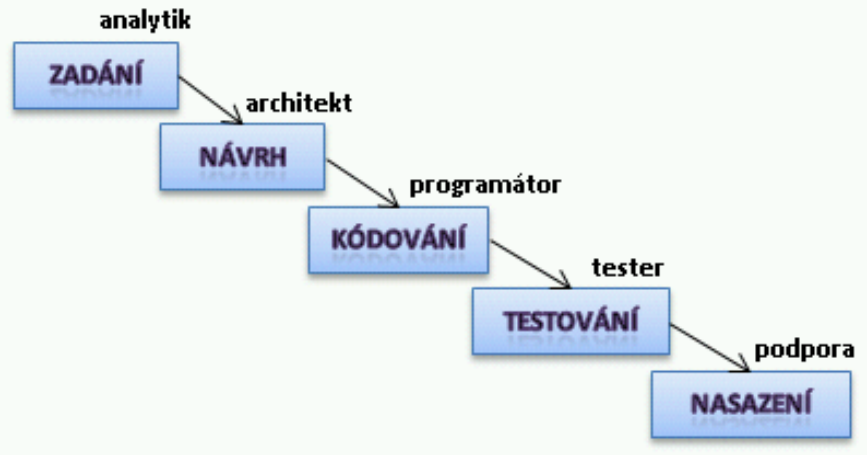
- rozpočet,
- rozsah týmu,
- citlivost projektu,
- používaná technologie,
- dokumentace,
- znalost týmu,
- dosavadní procesy. [3]

1.2.1 Vodopádový přístup

Vodopádový model představuje sekvenční vývojový přístup, ve kterém je na vývoj nahlíženo jako na neustále se svažující tok („tok vodopádu“) fázemi požadavků, návrhů, implementace, testování, integrace a údržby. Jako první formální popis vodopádového modelu je často citován článek z roku 1970, který publikoval Winstron W. Royce (1929-1995), ačkoliv pojem „vodopádový přístup“ ve článku použit přímo nebyl. Royce představil tento model jako příklad chybného a nefungujícího se modelu. [4]

Hlavní principy vodopádového přístupu:

- Projekt je rozdělen na fáze, které na sebe postupně navazují, některé se mohou překrývat.
- Důraz je kladen na plánování, časové rozvrhy, termíny, rozpočty a realizace celého systému najednou.
- Po celou dobu životnosti projektu je dodržována přísná kontrola. Je využito rozsáhlých dokumentací, revizí, schvalování uživatelem (signoff) na konci jednotlivých fází a vstupů od managementu informačních technologií před začátkem následující fáze. [4]



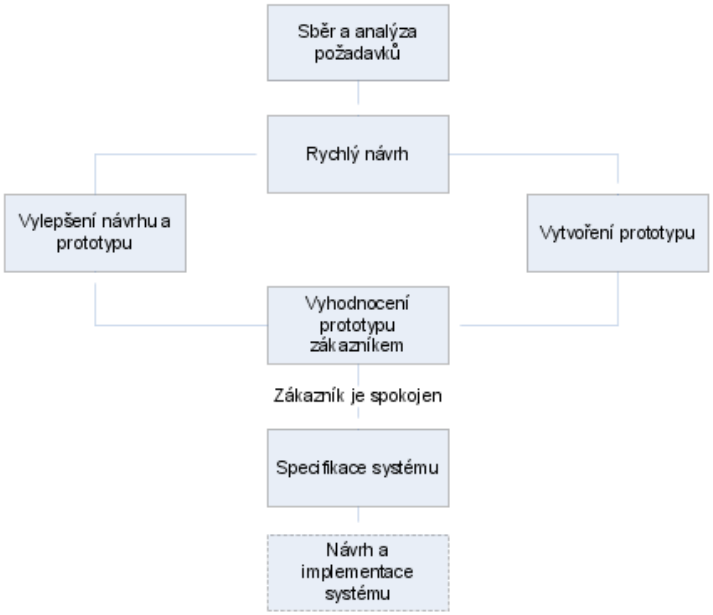
Obrázek 2 – Vodopádový model, zdroj [25]

1.2.2 Prototypový přístup

Při prototypovém přístupu vývoji softwaru dochází k vývoji neúplných verzí, kterým se říká tzv. prototypy. Prototypování se rozlišuje na dva základní přístupy: jednorázové (Throwaway, Close-ended, Rapid) a vývojové prototypování (Evolutionary, Beadboar). [4]

Základní principy prototypového přístupu jsou:

- Vývojové prototypování staví na strukturovaném vývoji funkční verze systému, které zahrnuje hlavní část uživatelských požadavků a která je základem pro finální produkt.
- Nejedná se o samostatný a kompletní přístup metodiky vývoje, ale o přístup k jednotlivým částem větších metodik vývoje (např. přírůstková nebo spirálová metoda).
- Projekt je rozdělen na menší části z důvodů snížení rizik v celém vývoji a zjednodušení úprav projektu v průběhu procesu vývoje.
- Uživatel je zapojen v celém procesu vývoje, což přináší velkou pravděpodobnost přijetí konečné implementace uživatelem.
- Dokud se prototyp nevyvine tak, že splňuje požadavky uživatele, jsou iterativním způsobem vyvíjeny jen malé ukázky systému.
- Většina prototypů je vytvářena s tím, že budou vyřazeny, ale v v některých případech je možné pokročit od prototypu k funkčnímu systému. [4]



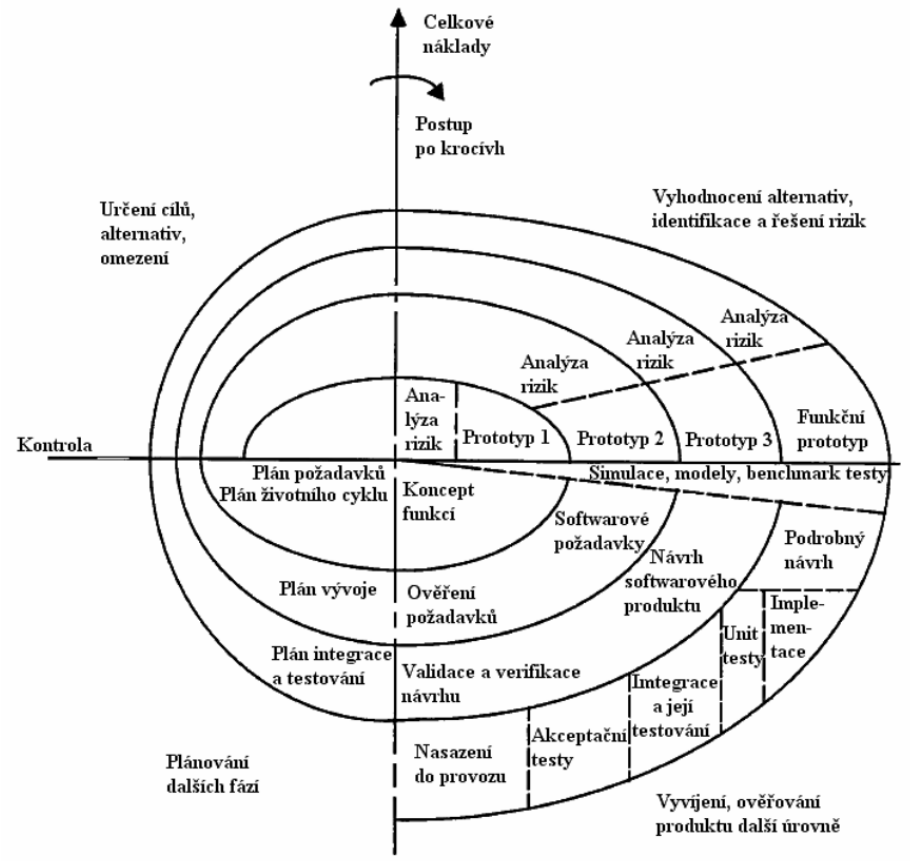
Obrázek 3 – Model prototypování, zdroj [26]

1.2.3 Spirálový přístup

Spirálový přístup vývoje formuloval v roce 1986 profesor Barry Boehm jako kombinaci prvků designového přístupu a prototypového přístupu. Kombinuje výhody obou těchto konceptů – shora-dolů (prototypování) a zdola-nahoru (designování). Spirálový přístup je určený pro rozsáhlé projekty. [4]

Hlavní principy spirálového přístupu:

- Zaměření na analýzu rizik a minimalizace projektových rizik rozdělením projektu na menší segmenty a umožnění změn během procesu vývoje. Během vývoje softwaru je také možno vyhodnocovat rizika a zvažovat další pokračování projektu v průběhu životního cyklu.
- Jednotlivé cykly spirály spouští stejný sled kroků pro každou část produktu a pro každou úroveň zpracování.
- Každý cyklus spirály obsahuje čtyřmi základními fázemi (kvadranty):
  - *Analýza* – stanovení cílů, alternativ a rozsahu iterace.
  - *Vyhodnocení* – zhodnocení alternativ, definice a řešení rizik.
  - *Vývoj* – vývoj produktu a validace očekávaných výsledků.
  - *Plánování* – plán pro příští iteraci. [4]
- Na počátku každého cyklu se definují zúčastnění subjekty a jejich podmínky kladené na úspěch iterace. Na konci každého cyklu se provádí revize a předání. [4]



Obrázek 4 – Spirálový přístup vývoje, zdroj [27]

1.2.4 Inkrementální přístup - Unifikovaný proces

Průmyslovým standardem SEP je metodika USDP, jejímiž autory jsou autoři UML. Běžně je však v literatuře označována zkráceně UP - Unifikovaný proces. Metodika UP je iterativní a inkrementální metodika, která umožňuje systematický přístup k tvorbě softwaru. Metodika UP je procesní částí projektu, oproti UML, který si lze představit, jako jazykovou část projektu. [2]

Metodika UP je založena metodách Eriscon (Eriscson approach, 1967), Rational (Rational Objectory Process, 1966-1997) a na dalších metodách, které vycházejí z nejlepších postupů. UP je pragmatická, ověřená metoda vývoje softwaru, která zahrnuje nejlepší ověřené postupy z praxe. [2]

Aplikace metodiky UP v projektu

Metodika UP je obecnou metodou tvorby softwaru, každý projekt vyžaduje vytvořit novou instanci. Tímto postupem se od sebe odlišuje každý softwarový projekt. Proces konkrétní aplikace metodiky obsahuje rovněž definici a začleňování:

- vnitropodnikových standardů,
- šablon dokumentů,
- nástrojů – např. nástroje pro správu konfigurace,
- databází – např. sledování chyb, sledování stavu projektu,
- úprav životnosti – např. úprava měřítek, vylepšení, pro kontrolu kvality používaných zabezpečovacích systémů (změna použité kryptografie atd). [2]

Pravidla metodiky UP

Metodika UP obsahuje tři základní pravidla:

- zásada řízená případem užití a rizikem,
- zásada soustředění se na architekturu,
- zásada iterace a přírůstku (inkrementu). [2]

Metodika UP využívána pro tvorbu softwarových produktů je založena na návrhu a postupném vývoji robustní architektury konkrétního systému. Architektura nám popisuje nejen strategické aspekty s možností rozkladu systému na jednotlivé komponenty, ale současně i způsob, jakým se tyto komponenty vzájemně ovlivňují a jakým jsou nasazeny na příslušném hardwaru. Ke vzniku kvalitního produktu vede spíše kvalitní architektura bez minimální úvahy, jak bude napsaný kód vypadat. [2]

UP je iterativní a přírustková (inkrementační). Iterativní vlastnost představuje rozklad projektu na menší podprojekty (iterace), které výslednému produktu postupně přináší nové funkce dávkově, nebo na přírustky (inkrementy), které vedou k tvorbě plně funkčního produktu. Software je tedy tvořený v procesu postupných upřesňování našeho konečného záměru. Tento postup se také zásadně liší od kaskádové tvorby softwaru, která byla založena v důsledné posloupnosti *analýzy, návrhu* a *tvorby*. V případě UP není posloupnost příliš omezující. Ke klíčovým pracovním postupům metodiky UP, jako je analýza, se vrací ve skutečnosti v průběhu projektu vícekrát. [2]

Iterativní a přírustkový proces vývoje metodiky UP

Během vývoje velkého softwaru je snaha o rozdělení do více tzv. „miniprojektů“. To nám přináší snazší správu a úspěšné dokončení. Každý z těchto „miniprojektů“ představuje iteraci. Hlavním východiskem je skutečnost, že iterace obsahuje všechny prvky normálního softwarového projektu:

- plánování,
- analýza a návrh,
- tvorba,
- integrace a testování,
- interní nebo externí uvedení. [2]

Každá iterace představuje vlastní základní linii, která se skládá z částečně kompletní verze finálního systému a z veškeré přidružené projektové dokumentace. Základní linie jsou postupně během vývoje vrstveny tak dlouho, dokud není dosažena konečná podoba vytvářeného systému.

Rozdíl mezi dvěma základními liniemi je nazýván přírůstek (inkrement). Životní cyklus projektů podle metodiky UP je proto označován jako iterativní a přírůstkový (inkrementační). [2]

Iterace jsou dále seskupovány do fází, které vytvářejí makrostrukturu metodiky UP.

Pracovní postupy iterace

Každá iterace obsahuje pět základních pracovních postupů (workflows), které definují, co je třeba udělat a také, jak toho dosáhnout. Jednotlivé iterace obsahují zpravidla i další postupy, jako jsou plánování, časový odhad a vše co se dané iterace týká. Tyto pracovní postupy však v metodice UP obsaženy nejsou.

Základní pracovní postupy definované metodikou UP jsou:

- *požadavky* – definují, co by měl systém dělat,
- *analýza* – odladění požadavků a jejich strukturování,
- *návrh* – realizace požadavků v architektuře systému,
- *implementace* – tvorba softwaru,
- *testování* – ověření implementace, zdali odpovídá požadavkům a funguje, jak se očekává. [2]

Rozložení projektu na sled iterací přináší flexibilní přístup k plánování projektu. Nejčastěji se využívá časová posloupnost iterací, kdy dokončení jedné iterace vede k zahájení další. Iterace však můžeme vykonávat i paralelně, výhodou pak je možnost zkrácení celého projektu a efektivní využití vývojového týmu. Základem však musí být velmi důsledné plánování.

Jednotlivé iterace v metodice UP generují tzv. základní linii (baseline). Tato základní linie představuje konkrétní verzi množiny revidovaných a schválených artefaktů generovaných příslušnou iterací. [2]

Rozdíly mezi jednotlivými liniemi potom nazýváme přírůstky (inkrementy). Přírůstky jsou jednotlivé dílčí kroky, které směřují k finální verzi systému. [2]

Struktura metodiky UP

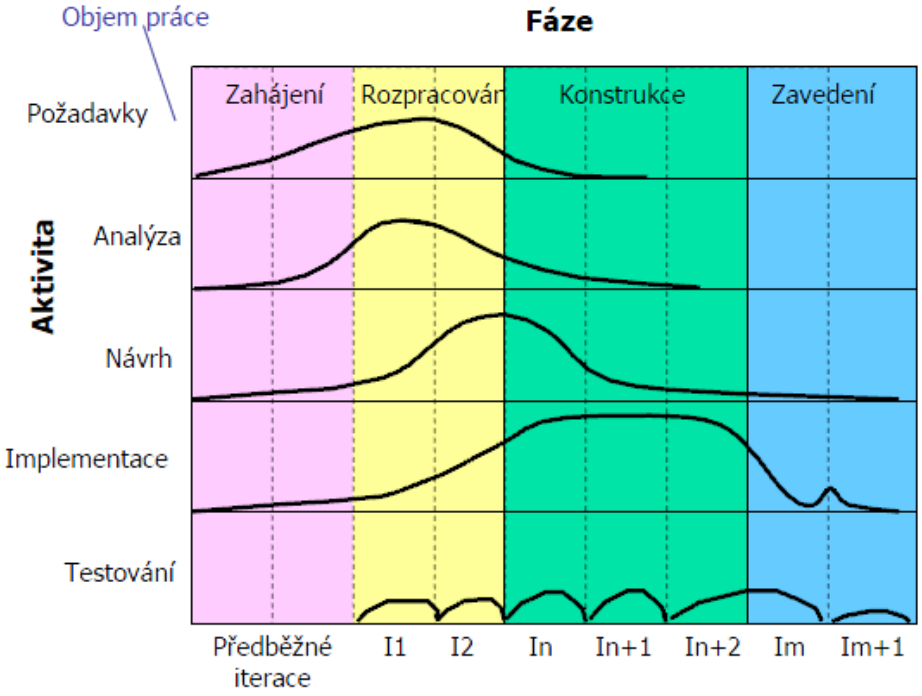
Životní cyklus projektu je rozdělen do několika fází, které končí definovanými hlavními milníky. Milník představuje indikátor pokroku v projektu. V každé fázi projektu můžeme realizovat pět základních pracovních postupů (požadavky, analýza, návrh, implementace, testování) a libovolný počet dodatečných pracovních postupů. Konkrétní počet iterací v jednotlivých fázích závisí na velikosti projektu. Jednotlivé iterace by však neměli trvat déle než 2-3 měsíce. [2]

Metodika UP se skládá ze čtyř po sobě následujících fází, z nich každá končí hlavním milníkem:

- *zahájení* (inception) – období plánování,
- *rozpracování* (elaboration) – období architektury,
- *konstrukce* (construction) – počátky provozuschopnosti,
- *zavedení* (transitiv) – nasazení produktu do uživatelského prostředí. [2]

Každá fáze má určitý cíl, na nějž jsou soustředěny aktivity jednoho nebo více pracovních postupů a jenž je důležitým milníkem v životě projektu.

Základní pochopení metodiky UP je popisuje Obrázek 5. Podél horního okraje jsou znázorněny jednotlivé fáze a pod nimi jsou uvedeny základní pracovní postupy. Dále jsou podél dolního okraje znázorněny konkrétní iterace fází. Křivka potom znázorňuje relativní objem práce vykonané v jednotlivých základních pracovních postupech v příslušných fázích projektu. [2]



Obrázek 5 – Fáze podle metodiky UP, zdroj [28]

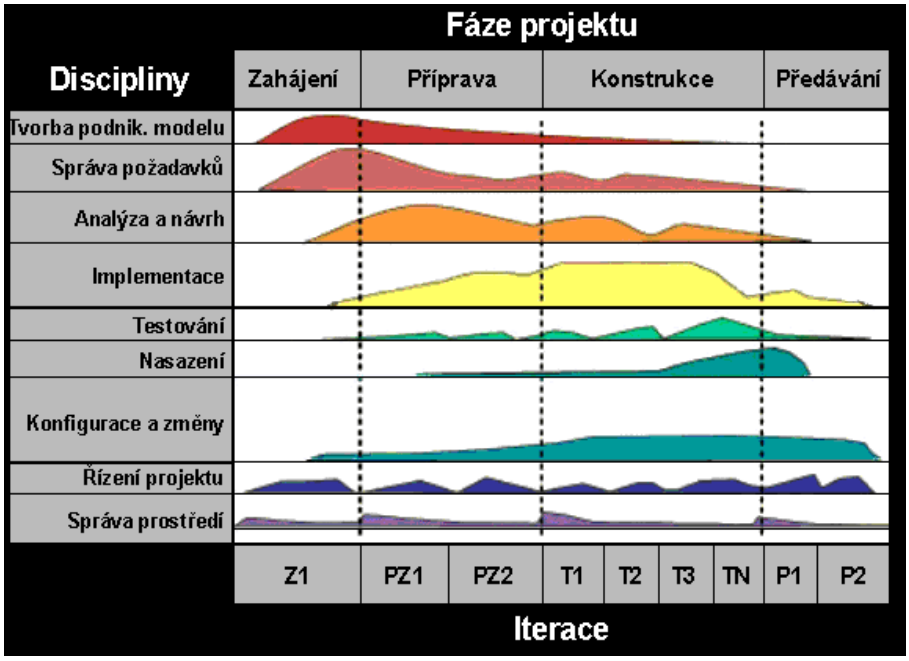
Z obrázku vyplývá, že nejvíce času je věnováno definování požadavků a analýze. Ve fázi rozpracování je důraz kladen na požadavky, analýzu a částečně na návrh. Během fáze stavby je důraz kladen především na návrh a implementaci. V konečné fázi zavedení je důraz kladen na implementaci a testování.

Nejdůležitější vlastností metodiky UP je to, že se soustřeďuje na cíl, nikoli na výsledek. Každá z fází je zakončena milníkem, jenž je definován podmínkami, které musí splnit. Jednou z podmínek může být i fakt, že milník zajišťuje dílčí výsledek projektu.

1.2.5 RUP

Metodika Rational Unified Process (RUP) je komerční verzí metodiky UP. RUP dodává společnost IBM, která ji v roce 2003 převzala od společnosti Rational Corporation. V současné době se o RUP nemluví jako o komerční verzi, ale jako o rozšíření metodiky UP mnoha významnými způsoby. UP bychom měli považovat za otevřený standart a RUP za specifickou komerční podtřídu, která některé vlastnosti a funkce bázové třídy (UP) překrývá a určité rozšiřuje. [2]

Metodiky UP i RUP jsou postaveny na stejném základu, obě modelují aspekty *kdo, kdy* a *co*. Rozdílem však je, že metodika RUP je propracována více do detailů a v některých případech se nepatrně od UC liší syntaxí. Metodika RUP obsahuje veškeré standardy, nástroje a další nezbytnosti, které již nejsou součástí UP, ale museli bychom je při vývoji stejně opatřit. Navíc je dodávána s bohatým uživatelským prostředím včetně dokumentace a popisu jednotlivých nástrojů. [2]



Obrázek 6 – Schéma specifikace RUP, zdroj [29]

Obrázek 6 je znázorňuje schéma RUP, stejně jako u UP, horizontální osa představuje čas a na vertikální ose jsou naneseny jednotlivé discipliny (aktivity), které jsou během projektu aplikovány.

Základní filosofií metodiky RUP tvoří šest základních pravidel, tzv. "nejlepších praktik" používaných při vývoji software:

1. *Iterativní vývoj software* – rozdělení celého projektu na čtyři fáze, z nichž každá se skládá z několika iterací se životním cyklem typu „vodopád". Výsledkem každé této iterace je spustitelná verze.
2. *Správa požadavků* – využívá koncepce „aktivní správy požadavků" spočívající ve stálém kontaktu zadavatele s dodavatelem a možnosti zpřesňování a úprav požadavků v průběhu projektu.
3. *Architektura založená na komponentách* – využití architektury systému založené na komponentách z důvodů snadného rozdělení mezi několik vývojových týmů, snazšího zapracování změn a možnosti postupné tvorby softwaru.
4. *Vizuální modelování* – pro modelování se v RUP využívá prostředků jazyka UML (možnost vygenerování kódu z analytického návrhu).
5. *Ověřování kvality software* – výsledný produkt lze předat do provozu pouze v případě, že splňuje požadovaná kritéria. Jsou to funkcionalita, spolehlivost a výkon.
6. *Řízení změn software* – možnost provádění změn i v pozdějších stádiích. [6]

1.3 Fáze podle metodiky UP

Tato kapitola popisuje jednotlivé fáze vývoje podle metodiky UP.

1.3.1 Fáze zahájení

Cílem fáze zahájení je „odstartování“ samotného projektu. Je zde kladen důraz na pracovní postupy zabývající se definicí požadavků a jejich analýzou. V zahájení však mou být obsaženy i návrhářské nebo implementační činnosti, pokud je třeba vytvořit technický prototyp, který má většinou za úkol potvrdit správnost koncepce. V této fázi ve většině případů nedochází k pracovnímu postupu testování.

1.3.2 Fáze rozpracování

Cílem fáze rozpracování je tvorba spustitelné architektonického základu. Jedná se o spustitelný systém, sestavený na základě specifikované architektury. Nejedná se již o prototyp, který by se mohl „zahodit“, ale jedná se o „první náznak“ vyvíjeného systému. Tento spustitelný základ je postupně během dalších fází rozšiřován a postupně se vyvine v konečnou verzi výsledného systému. Jelikož následující fáze vychází z výsledků fáze rozpracování, lze ji hodnotit jako kritickou a je na ni třeba dbát důraz.

Ve fázi rozpracování je důraz kladen na následující aktivity:

- *požadavky* – upřesnění rozsahu systému a požadavků na něj kladených,
- *analýza* – definování toho, co budeme tvořit,
- *návrh* – tvorba stabilní architektury,
- *implementace* – tvorba spustitelného základu systému,
- *testování* – testování základu systému. [2]

1.3.3 Fáze konstrukce

Účelem fáze konstrukce je splnit všechny požadavky analýzy a návrhu a následně vyvinout ze spustitelného základu konečnou verzi systému. Hlavním zadáním této fáze je zachování integrity architektury vytvářeného systému. Jakmile začneme klást důraz na kód systému, může nás začít „tlačit“ čas a potom může dojít k narušení původní vize. Toto potom vede ke snížení kvality výsledného systému a následnému nárostu nákladů na jeho údržbu.

Popis aktivit ve fázi konstrukce:

- *požadavky* – odhalit všechny požadavky, na které bylo v předchozích fázích zapomenuto,
- *analýza* – dokončení analytického modelu,
- *návrh* – dokončení modelu návrhu,
- *implementace* – zajištění počáteční provozní způsobilosti, jedná se o hlavní aktivitu,
- *testování* – testování počátečního funkčního systému. [2]

1.3.4 Fáze zavedení

Fáze zavedení začíná v okamžiku, kdy je dokončeno testování a konečné nasazení systému. Je zde zahrnuta oprava všech nalezených chyb v beta-verzi a příprava nasazení výsledného systému.

Cíle této fáze lze shrnout do následujících bodů:

- oprava chyb,
- příprava uživatelského pracoviště na přijetí nového softwaru,
- přizpůsobené softwaru, aby fungoval na pracovišti uživatele,
- úprava softwaru v případě vzniku nepředvídatelných problémů,
- tvorba manuálů a další dokumentace,
- školení uživatelů, konzultace,
- finální revize. [2]

Popis aktivit ve fázi konstrukce:

- *požadavky* – nevyužitelné,
- *analýza* – nevyužitelné,
- *návrh* – úprava návrhu, pokud během testování byly nalezeny chyby,
- *implementace* – přizpůsobení softwaru pracovišti uživatele a oprava chyb, které nebyly při testování nalezeny,
- *testování* – beta-testy a přejímací testy na pracovišti uživatele. [2]

1.4 Týmový vývoj

Týmová spolupráce má mnoho výhod, které přináší efektivitu při vývoji softwaru. Mezi hlavní výhody patří určitě to, že skupina udělá více než pouhý jednotlivec, tzv. synergický efekt.

Členové týmu mezi sebou sdílí informace, poznatky a zkušenosti. V týmu se podporuje kreativita a tvořivost, noví členové se mohou učit rychleji, než kdyby pracovali samostatně, přítomnost ostatních členů pak může i jedince motivovat k vyšším výkonům. Dobrá atmosféra celého týmu je motivující a podporující k práci.

Při vývoji v týmu je kritickým faktorem úspěchu sdílení informací mezi členy vývojového týmu – zdrojových kódů, evidence chyb, sledování postupu implementace, rozdělení práce a řady dalších věcí.

Nejdůležitějším výhodou je samotné rozdělení práce. To přináší při týmovém vývoji možnost specializace členů týmu na konkrétní činnost. Každý dělá to, co umí nejlépe. Specializace každého člena týmu je většinou svázána konkrétním pracovním postupem vycházejícím z metodiky UP (požadavky, analýza, návrh, implementace, testování). Z těchto pracovních postupů nám potom vychází několik základních pracovních pozic, které spolupracují na projektu:

- vedoucí projektu,
- analytik,
- programátor,
- tester.

Při společném vývoji je nutné mít vždy stejné a dostupné informace týkající se projektu, na kterém tým spolupracuje. Jedná se například o základní požadavky na software zpracované vedoucím projektu, diagramy vytvořené analytikem a v neposlední řadě reporty testerů vzniklé z testování softwaru. Důležité je i sdílení samotných zkušeností mezi programátory z jiných projektů (využití know-how kolegy při řešení stejného problému, který řešil nedávno).

Dalším důležitým požadavkem při týmovém vývoji je možnost spolupráce programátorů na stejném kódu. Je nutné, aby vždy programátor pracoval na svém kódu a neměl do něj v daný okamžik přístup nikdo jiný. Následně je třeba, aby se práce programátora mohla sloučit se zdrojovými kódy ostatních a tím se postupně rozvíjel společný projekt.

Na programátory je při týmové spolupráci kladen i důraz na dodržování stejných metodik, při vývoji se jedná především o dodržení tzv. štábní kultury vyvíjeného softwaru. Hlavní výhoda je určitě v tom, že se tímto velmi usnadní samotné čtení zdrojového kódu, pokud všichni dodržují stejná pravidla. Do metodik pro programátory dále patří i používání stejného vývojového prostředí (IDE), v kterém lze nadefinovat šablony pro dodrží zmiňované štábní kultury. IDE potom přináší mnoho dalších výhod (např. doplnění kódu při zápisu zkratky, automatické formátování kódu, přehledné zobrazení struktury projektu, možnost integrace dalších pluginů pro využití frameworku a mnoho dalších).

Požadavků kladených na týmový vývoj, který bude dobře fungovat, je velké množství. Přichází potom v úvahu společná platforma se svými metodikami, která nám toto může velmi ulehčit. Základem této platformy musí být centrální správa identit – každý uživatel, který se podílí na týmové spolupráci, zde musí být založen a přiřazen ke své roli, případně konkrétnímu projektu. Platforma dále řeší evidenci veškerých požadavků na vyvíjený software, připomínky, reporty z testování, sdílení nových poznatků atd. Uživatelé mohou kdykoliv přistoupit k těmto informacím a mohou na ně i reagovat. Společné metodiky potom zajistí administrátorovi platformy, včetně jejích uživatelů (programátoři, testeři atd.), sdílet základní postupy při práci s touto platformou. Metodika má velký význam při přijetí nového člena do týmu, který se takto snadno a rychle dozví všechny potřebné informace.

2 Platforma pro týmový vývoj

Tato kapitola popisuje základní informace o PHP, vlastnostech open source aplikací, zhodnocuje úvodní rešerši se zaměřením na potřeby platformy při vývoji v PHP a představuje základní softwarové prostředky pro využití při týmovém vývoji.

2.1 Vývoj v PHP

PHP je skriptovací programovací jazyk, který je především určený pro programování dynamických stránek a webových aplikací.

Při využití PHP pro dynamické stránky jsou skripty prováděny na straně serveru, k uživateli je následně přenášen až výsledek jejich zpracování. Interpret PHP skriptu je možné volat pomocí příkazového řádku, dotazovacích metod HTTP nebo pomocí webových služeb. PHP je nezávislý na platformě a podporuje mnoho knihoven pro různé potřeby. Např. zpracování textu, grafiky, práci se soubory, přístup k databázovým systémům (MySQL, Oracle, PostgreSQ a další), podporuje řady internetových protokolů (HTTP, SMTP, FTP, IMAP, POP3, LDAP a další).

PHP patří mezi nejoblíbenější skriptovací jazyky pro web a je v něm napsáno mnoho velkých aplikací, které denně na Internetu využívá každý uživatel.

2.2 Open Source aplikace

Open source, v překladu otevřený software, je software s otevřeným zdrojovým kódem. Otevřenost představuj jak technickou dostupnost zdrojového kódu, tak legální dostupnost – licenci software, která dovoluje, při dodržení podmínek, uživatelům kód využívat a i modifikovat.

V běžném popisu se dnes termín open source používá i pro mnoho dalších vlastností, které s otevřeností zdrojového kódu úplně nesouvisí, ale vyskytuje se u mnoha open source programů. Jedná se např. o bezplatnou licenci softwaru nebo vývoj dobrovolnou komunitou programátorů.

Z hlediska bezpečnosti má open source software výhody i nevýhody. Pokud se vyskytne chyba v softwaru, může ji najít a opravit daleko větší skupina programátorů. Na druhou stranu toto přináší i možnou zranitelnost, chyby mohou využít útočníci. Proto je potřeba software pravidelně aktualizovat.

2.3 Zhodnocení úvodní rešerše pro potřeby platformy

Na základě úvodní rešerše a potřeb při vývoji v jazyce v PHP byly navrženy základní produkty pro fungování platformy.

Z pohledu uživatele, především programátora, je nejdůležitější kvalitní vývojové prostředí, které mu maximálně dokáže zjednodušit a urychlit práci. Se samotným prostředím souvisí i jednoduchá práce s verzemi vyvíjeného softwaru, je tedy třeba aby měl programátor k dispozici klienta pro práci s repozitářem. Posledním zásadním produktem klienta je webový server, na kterém programátor může ladit a testovat aplikace. Databázi potom může využívat společnou, která bývá nainstalována na serveru, může ji mít však i lokální stanici.

Základem serverové části je operační systém, je třeba vybrat takový, aby měl podporu a k dispozici pravidelné updaty, samozřejmostí je jeho možnost podpory pro využití coby webového a databázového serveru.

Na serveru budou nainstalovány potřebné produkty, které musí být napojené na centrální úložiště identit, kdy jeden uživatel bude moci využívat, dle svých práv, více produktů aniž by ho administrátor musel do každého zvlášť zakládat.

Z pohledu nainstalovaných produktů, které budou využívat jednotliví uživatelé, je důležitý systém pro správu verzí, který je základem týmovém vývoje, kdy více programátorů pracuje na stejném projektu. Pro podporu znalostního managamentu je třeba systém pro uchování znalostí (např. postup při nasazení produktu), ke kterému mají přístup všichni uživatelé, kteří zde mohou vkládat nové znalosti a sdílet je tak s ostatními. Podobným je i systém pro uchování a správu dokumentů, ten pomůže spravovat veškerou dokumentaci, hledat v ní konkrétní informace, sdílet ji atd. Dokumentace tak dostane řád a každý člen týmu si vždy najde, co potřebuje. Posledním nainstalovaným produktem, který nelze v platformě postrádat je systém pro správu chyb a požadavků. Ten musí být k dispozici rovněž všem, včetně zákazníků. Bude sloužit pro evidenci veškerých chyb, připomínek a požadavků k vývoji. Systém musí umět pracovat s různými stavy požadavků, které půjdou zařadit po daný projekt. Je nutné, aby byl systém přehledný podporoval českou jazykovou mutaci, která může být důležitá pro zákazníky.

2.4 Základní kritéria pro výběr softwaru

Základními kritérii pro výběr vhodných produktů platformy jsou:

- open source řešení,
- bezplatná licence,
- možnost napojení na centrální správu identit,
- podporovaný shodný operační systém,
- snadné a intuitivní uživatelské ovládání.

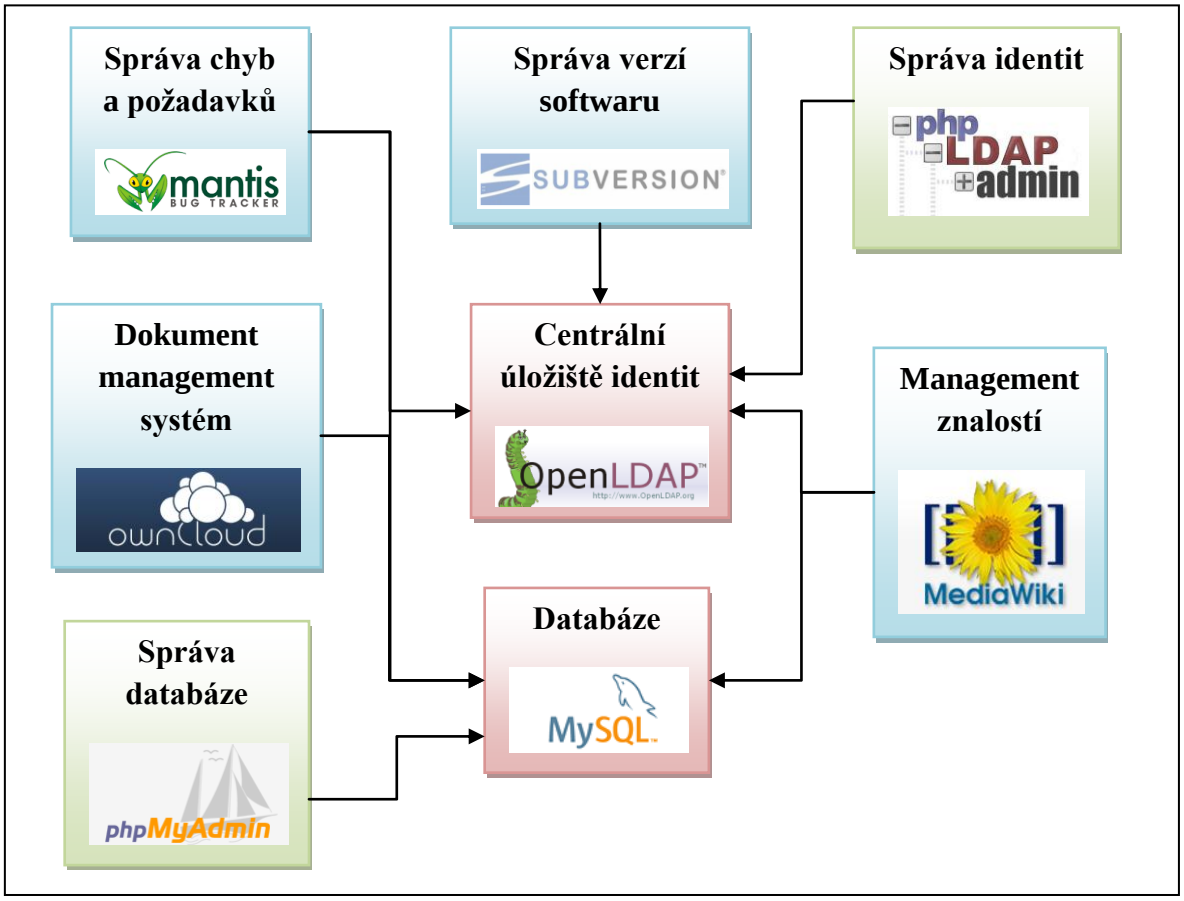
Produkty musí být vybrány dle těchto parametrů, které zajistí kvalitní a dostupnou platformu pro týmový vývoj.

2.5 Výsledná platforma, vybrané produkty

Tato kapitola popisuje výslednou platformu serverové a klientské části s vybranými produkty a důvody jejich použití. Samotné produkty jsou dále podrobně rozebrány v kapitole Softwarové prostředky platformy.

2.5.1 Serverová část

Všechny produkty serverové části popisuje Obrázek 7 a následně jsou produkty rozebrány pod obrázkem.



Obrázek 7 – Schéma softwarových prostředků serverové části

Jako operační systém platformy byl zvolen Linux, jeho distribuce *Debian*. Tato distribuce přináší vynikající stabilitu a jednoduchou údržbu, výhodou je i velké množství dokumentace a návodů.

Databáze byla vybrána *MySQL*, jedná se o jednu z neoblíbenějších a nejvíce využívaných relačních databází s bezplatnou licencí. Její největší výhodou je, že všechny produkty, které byly testovány, tuto databázi podporují. Stejně jako pro zvolený operační systém, i pro ni existuje velké množství dokumentace. Jako klient databáze byl použit oblíbený nástroj *phpMyAdmin*, který nabízí přehlednou grafickou správu jednotlivých databází, včetně správy práv uživatelů k jednotlivým databázím.

Pro potřeby zavedení centrálního úložiště identit byl použit *LDAP* – distribuce *OpenLDAP*. Distribuce nabízí mnoho dokumentace a nabízí vše, co je pro úložiště identit vyžadováno. LDAP byl vybrán pro snadnou implementaci, rozšíření, širokou podporu a rychlost vyhledávání v DIT. Jako klient byl zvolen *phpLDAPadmin*, který nabízí jednoduché, grafické prostředí pro správu uživatelů a dalších objektů.

Jako verzovací sytém byl zvolen *Subversion*, který je stále velmi využívaný. Jeho implementace i ovládání je jednoduché a umožňuje snadnou práci s repozitářem. Nabízí mnoho klientů, z nichž byl vybrán *Tortoise SVN*. Ten nabízí i českou jazykovou mutaci.

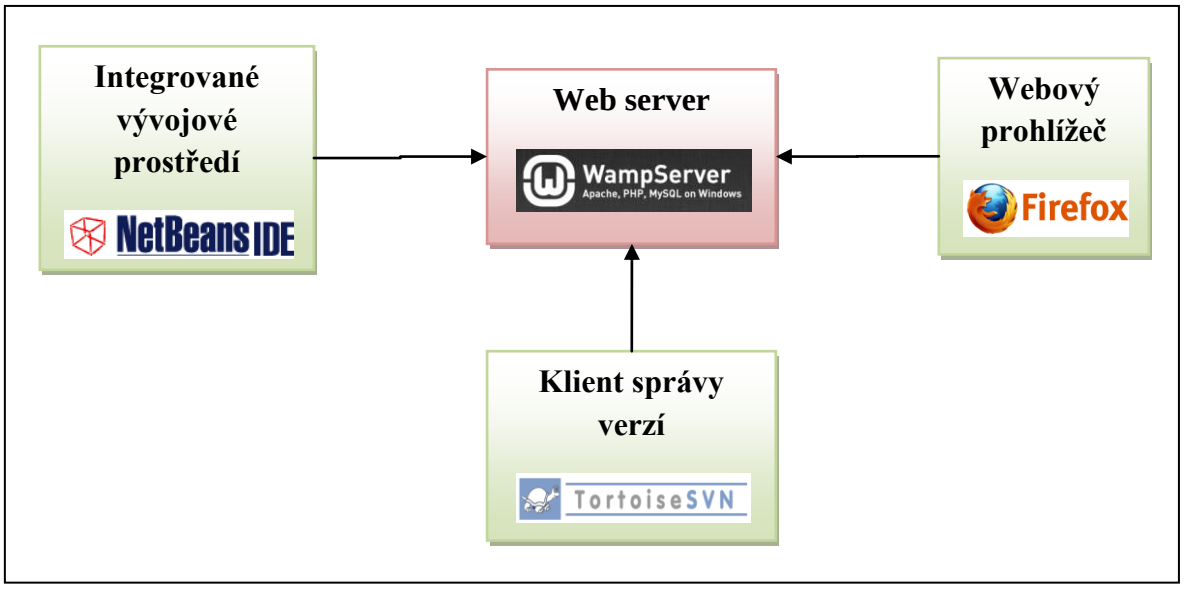
Systém pro správu chyb a požadavků byl zvolen *Mantis*. Disponuje jednoduchým a intuitivním ovládání v české jazykové mutaci. Nabízí přehlednou úvodní nástěnku, kde jsou zobrazeny úkoly a požadavky v různých stavech. *Mantis* nabízí mnoho uživatelských rolí, které se uplatní napříč celým týmem.

Produkt *OwnCloud* představuje v platformě systém pro správu dokumentů. Jeho široká funkcionalita nabízí většinu požadavků na kvalitní DMS a přináší další výhody disponující cloudovým úložištěm.

Jako management znalostí, který je rovněž ve výsledné platformě nepostradatelný, byl zvolen oblíbený a známý wiki systém *MediaWiki*. Důvodem pro volbu bylo velké rozšíření, snadná instalace a konfigurace, intuitivní ovládání pro uživatele a i samotný výkon aplikace. Nabízí i českou jazykovou mutaci.

2.5.2 Klientská část

Všechny produkty klientské části popisuje Obrázek 8 a následně jsou produkty rozebrány pod obrázkem.



Obrázek 8 – Schéma softwarových prostředků klientské části

Jako integrované vývojové prostředí byl vybrán *NetBeans IDE*, který nabízí propracované a jednoduché vývojové prostředí, včetně užitečných funkcí pro vývojáře. Jedná se o multiplatformní aplikaci s možností doinstalování velkého množství pluginů, např. pro práci s SVN.

Jak již bylo zmíněno v serverové části, jako klient SVN byl zvolen *TortoiseSVN*. Nabízí integraci do průzkumníka a velice přehledné menu pro práci s repozitářem. Pro práci s repozitářem lze však i využít integrovaného klienta v *NetBeans IDE*.

Dále byl vybrán webový server *Wamp*, který nabízí snadnou správu a konfiguraci webového serveru v grafickém režimu.

3 Softwarové prostředky platformy

Tato kapitola popisuje detailní popis open source softwarových prostředků, které svojí funkcionalitou pokrývají komplexně vývoj softwaru v jazyce PHP.

3.1 Operační systém

Operační systém je základní softwarové vybavení počítače, které je zavedeno do paměti počítače při jeho startu a zůstává v činnosti až do vypnutí počítače. Jedná se o velmi komplexní software, jehož vývoj je mnohem složitější a náročnější, než vývoj obyčejných softwarů. Skládá se z jádra (kernel) a pomocných systémových nástrojů. Hlavním úkolem operačního systému je zajistit uživateli možnost ovládat počítač, vytvořit pro procesy stabilní aplikační rozhraní (API) a přidělovat jim systémové prostředky.

3.1.1 Linux

V současné době je jedním z nejprogresivnějších operačních systém Linux, ten je postaven na principech systému UNIX. Linux je moderní operační systémem splňující náročné požadavky sféry podnikových serverů a v neposlední řadě i oblasti desktopových aplikací, mezi něž spadají vlastnosti jako preemptivní multitasking, paralelismus a běh vícevláknových aplikací. Je zároveň víceuživatelský, víceprocesorový, modulární a dobře škálovatelný. [19]

Základní charakteristika operačních systému Linux:

- open source řešení,
- moderní a bezpečný operační systém,
- vysoká stabilita a dostupnost,
- možnost přívětivého ovládání v grafickém režimu,
- více uživatelský a více úlohový systém,
- velké množství open source aplikací,
- velké množství manuálů,
- mnoho distribucí dostupných v češtině,
- a další.

Operační systém Linux je ideální pro použití jako serverové řešení v implementované platformě.

3.1.2 Windows

Další z možných řešení je operační systém postavený na platformě Microsoft Windows. Jedná se však o čistě komerční řešení. Tento systém přináší homogenní prostředí celé počítačové sítě. Škála produktů, které tato společnost nabízí, pokrývá potřeby jak malých firem a domácností, tak i požadavky velkých společností s mnoha pobočkami.

Základní charakteristika operačních systému Windows:

- komerční řešení,
- globálně nejpoužívanější operační systém,
- silná podpora ze strany vývojářů softwaru a hardwaru,
- relativně stabilní operační systém,
- přívětivé prostředí v grafickém režimu,
- snadné nasazení,
- mnoho nástrojů pro snadnou správu,
- a další.

Debian

„Projekt debian[1] je sdružení dobrovolníků, kteří sdílejí myšlenku na vytvoření svobodného operačního systému. Tento systém se nazývá Debian GNU/Linux, nebo krátce Debian.“ [20]

Debian není závislý na konkrétním jádře. V současné době využívá jádro Linux, které je svobodným softwarem, jehož vývoj byl zahájen Linusem Torvaldsem a je nyní podporováno více než tisícem programátorů z celého světa. V současnosti se pracuje na úpravě Debianu i pro jiná jádra, obzvláště Hurd. Hurd je kolekce serverů běžících nad mikrojádrem (např. Match neboL4). [20]

Debian oproti „komerčním distribucím“ (Red Hat, SUSE) nevyužívá balíčkový systém postavený na RPM (RPM Package Manager), ale má vlastní balíčkový systém, tzv. deb-balíčků, který je velice propracovaný a umožňuje provádět jednoduchou správu balíčků z různých zdrojů, nazývá se APT (Advanced Packaging Tool). APT usnadňuje správu softwaru, např. vyhledávání konfiguraci a instalaci balíčků. [21]

Zhodnocení

Jako operační systém platformy pro vývoj byla nainstalována poslední verze distribuce Debian 6.0 „Squeeze“. Debian byl vybrán pro svoji vynikající stabilitu a jednoduchou údržbu. V dnešní době se jedná o hojně využívaně a prověřené serverové řešení, z kterého vychází také distribuce Ubuntu, jenž je oblíbená jako desktopový operační systém.

3.2 Databáze

Databáze je nástroj pro shromažďování jakýkoliv informací, které mají své logické uspořádání. Můžeme si ji představit jako soubor dat, kterými slouží pro popis reálného světa (např. evidence zaměstnanců ve firmě).

Zavádějí se zde pojmy jako je entita a atribut. Entita je prvek reálného světa (např. zaměstnanec) a je popsána svými vlastnostmi, ty se nazývají atributy (např. jméno,

[1] http://www.opensource.org/osd.html

příjmení, telefon). Mezi jednotlivými entitami jsou vazby, zaměstnanec může pracovat v konkrétním oddělení – vazba mezi zaměstnancem a oddělením (zaměstnanec pracuje v oddělení. Vazby mají různé typy:

- *1:1* – jedna entita má vazbu právě pouze s jednou entitou (např. každý člověk vlastní pouze svůj občanský průkaz)
- *1: N* – jedna entita má vazbu s více dalšími entitami (např. jeden člověk vlastní více aut)
- *M:N* – bez omezení (např. student má zapsáno více předmětů, ale tento předmět má zapsáno více studentů). V praxi se tato vazba využívá jako tzv. vazební tabulka.

3.2.1 Relační databáze

Relační databáze jsou v dnešní době nejvíce využívané databáze. Vychází z tzv. relačního modelu, jehož základem je relace. Tu si můžeme představit jako tabulku, která se skládá ze sloupců a řádků. Sloupce odpovídají jednotlivým vlastnostem (atributům) entit. Data v jednom řádku tabulky zobrazují aktuální stav entity. [23]

Tabulka je základním stavebním kamenem při budování celé databáze. Relace tedy představuje celou tabulku a prvky relace odpovídají konkrétnímu řádku tabulky, který se označuje jako databázový záznam. Soubor jednotlivých tabulek (relací) tvoří databázi (relační schéma). [23]

MySQL

MySQL je multiplatformní relační databázový systém. Je k dispozici pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

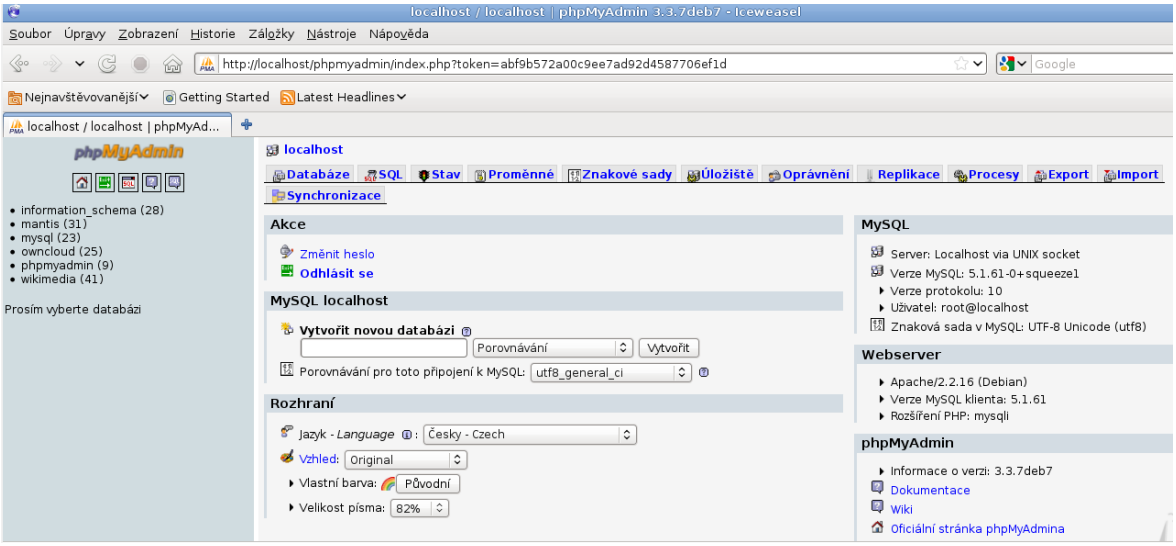
Každá databáze v MySQL se skládá z jedné nebo více tabulek, které mají své řádky a sloupce. V řádcích rozeznáváme jednotlivé záznamy (řádek odpovídá záznamu). Sloupce mají svůj název a datový typ, který odpovídá jednotlivým polím záznamu (sloupec odpovídá poli záznamu).

phpMyAdmin

Webová aplikace phpMyAdmin slouží k jednoduché správě obsahu databáze MySQL. Jedná se o nejvíce využívanou aplikaci pro správu databáze MySQL, která je poskytována jako open source produkt.

PhpMyAdmin podporuje vytvářet/odstraňovat databáze, vytvářet/upravovat/odstraňovat tabulky, spravovat uživatele a jejich oprávnění k jednotlivým databázím, sledovat statistiky vytížení databází, exportovat a importovat data, spravovat další databázové objekty, a v neposlední řadě provádět SQL dotazy.

Široká funkčnost aplikace je výhodou pro uživatele. Mezi další paří i mnoho překladů, mezi kterými nechybí čeština.



Obrázek 9 – Základní obrazovka aplikace phpMyAdmin

3.3 Systém správy verzí

Systém správy verzí, zkráceně verzovací systém, je systém k uchovávání historie jakýkoliv dat v digitální podobě. Při vývoji softwaru, je tento systém použit pro verzování zdrojových kódů, kdy se evidují jednotlivé verze během stádia vývoje softwaru.

Verzovací systém eviduje uživatele, čas a změny zdrojového kódu, jaké uživatel v softwaru udělal. Toto slouží nejen k přehledu všech změn (verzí), ale také možnosti vidět stav dat kdykoliv v minulosti a možnosti se k těmto změnám vrátit (např. z důvodů chyby v některé následující verzi). Každé změně, resp. verzi, je vždy přiděleno unikátní číslo v rámci celého projektu, které je označováno jako číslo revize. [17]

Smysl verzování je spolupráce velkého množství programátorů na jednom softwarovém projektu. Verzovací systémy hlídají a řeší případné kolize. Kolize je situace, kdy dva programátoři upravují stejnou část zdrojového kódu. Bez využití verzovacího systému si nelze týmový vývoj představit.

Systémy správy verzí se dělí na dvě základní skupiny:

- *centralizované* – data jsou ukládána na jediný server a ke všem funkcím je vyžadováno stále připojení k tomuto serveru,
- *distribuované* – data jsou ukládána ke každému vývojáři lokálně, přičemž má na svém disku celou historii, to umožňuje rychlejší práci.

Mezi centralizované open source verzovací systémy patří CVS a Subversion a mezi distribuované můžeme řadit např. Git, který je v současné době na vzestupu. [17]

3.3.1 Subversion

Subversion, zkráceně SVN, patří mezi centralizovaný open source nástroj pro správu obsahu zdrojových kódů (SCM – Source Content Management). Suversion je open source řešení, které slouží ke správě a verzování zdrojových kódů napsaných v libovolném programovacím jazyce. [18]

Subversion vychází ze staršího CVS a řeší jeho nedostatky, jako je např. nemožnost přesunu nebo kopírování adresářů. Snaží se však zachovat podobný způsob a styl práce jako byl u CVS.

Subversion se skládá ze dvou částí:

- *klientská* – nástroje pro práci s verzemi přímo v pracovním adresáři,
- *serverová* – centrální úložiště verzí (repository).

Základní terminologie a postupy v subversion:

- *Repository* (centrální úložiště, repozitář) – stará se o organizaci projektu a správu jeho verzí. Správa se provádí s klientskými nástroji.
- *Branch* (větev) – slouží k organizaci repozitáře podobnou adresářové struktuře. Pokud si klient z repozitáře „vyzvedne“ větev, vznikne mu adresářová struktura, která odpovídá větvím v repozitáři.
- *Revison* (revize) – jedná se o pořadové číslo každé změny v repozitáři. Každá změna v libovolné větvi vytvoří novou revizi v rámci projektu. Revize obsahuje informace o tom, co bylo změněno, uživateli který změnu provedl, času provedení revize a poznámku. Pomocí revize můžeme sledovat změny ve větvích v čase.
- *Working copy* (pracovní kopie) – kopie dat z konkrétní větve repozitáře, která je uložena na lokálním disku klienta. V pracovní kopii klient provádí různé změny, které jsou následné commitem uloženy zpět do repozitáře.
- *Commit* (odevzdání) – odeslání a nahrání všech změn provedených od posledního commitu na repozitář. Commit je nejčastější operace při práci s repozitářem. Pokud se provádí commit celé pracovní kopie, jedná se o atomickou operaci (musí být dokončená celá, nebo nic). Jsou odeslány veškeré změny ve všech objektech ve správě verzí. Pokud však dojde k jakékoliv chybě během přenosu (odevzdání), není vytvořena žádná nová verze.
- *Konflikt* – stav, kdy stejný objekt, který má být commitován, byl změněn někým jiným a již se v repozitáři v aktuální verzi nachází v jiné podobě, než jaký má uživatel ve své pracovní kopii. SVN nedovolí provést commit celé pracovní kopie, pokud se v ní nachází jakýkoliv konflikt.
- *Merge* (sloučení) – sloučení změn z větve v repozitáři do pracovní kopie. Je možné určit rozsah změn pomocí intervalu revizí.

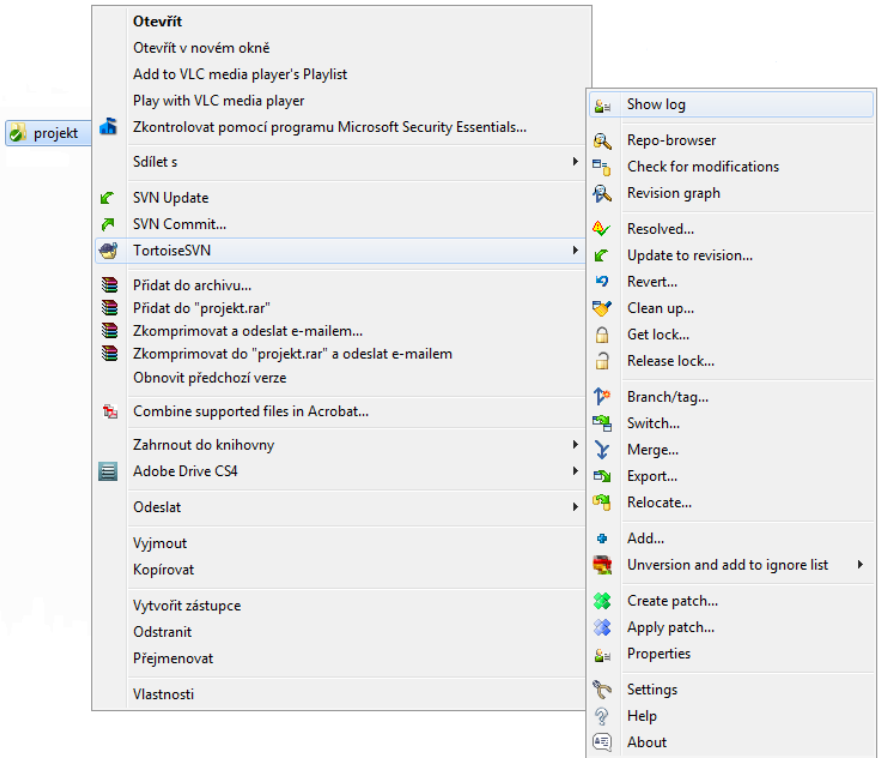
- *Changeset* (sada změn) – sada změn z pracovní kopie uživatele, která se odesílá při commitu do repozitáře. Může se také jednat o sadu změn provedených přímo v repozitáři. Výhodou subversion je, že takto ukládá pouze rozdíly mezi jednotlivými revizemi. Tímto se snižuje objem přenášených dat z klienta na server a ušetří se i místo na disku.
- *Cheap-copy* (odkazy v repozitáři) – realizace kopie v rámci repozitáře. Objekty nejsou fyzicky duplikovány, ale jsou pouze vytvořeny tzv. odkazy na kopírované objekty. Díky tomu není Subversion náročné na datový prostor. [18]

Klient – TortosiseSVN

TortoiseSVN je bezplatná open source aplikace pro Apache Subversion. S využitím této aplikace můžeme pohodlně spravovat soubory a adresáře v centrálním úložišti Subversion. [22]

Základní funkce Tortoise SVN jsou:

- *Integrace do rozhraní* – integrace do operačního systému, např. do průzkumníka v operačním systému Windows.
- *Překryv ikon* – jednotlivé stavy všech souborů a složek jsou indikovány překryvnými ikonami. Tím snadno na první pohled vidíme, v jakém stavu se naše pracovní kopie nachází.
- *Snadný přístup k příkazům* – všechny příkazy SVN jsou k dispozici z vlastního kontextového menu, které Tortoise SVN přináší s integrací do systému, viz Obrázek 10.



Obrázek 10 – Základní menu Tortoise SVN

Aplikace TortosieSVN ve svém menu příkazů zahrnuje všechny dostupné příkazy popsané v předchozí kapitole věnované Subversion.

Pokud nevyužijeme integrovaného SVN přímo ve vývojovém prostředí (IDE), pak TortoiseSVN je velmi jednoduchý a šikovný nástroj pro správu Subversion.

Zhodnocení

Subversion je efektivní a velmi používaný nástroj pro verzování zdrojových kódů. Je k dispozici pro různé operační systémy včetně Linuxu. Jeho implementace a ovládání je jednoduché a uživatelé mohou s repozitářem snadno pracovat pomocí popisované aplikace Tortoise SVN, která obsahuje i českou jazykovou mutaci. SVN lze také snadno implementovat přímo do vývojového prostředí, v popisované platformě je implementováno do NetBeans IDE. Více je popsáno v samostatné kapitole věnované prostředí IDE.

3.4 Centrální úložiště identit

Centrální správa identit představuje správu individuálních identit (uživatelských účtů) v rámci konkrétního systému, např. informačního systému firmy. V podnikovém informačním systému se správa identit týká zavedení a správy jednotlivých rolí i přístupových práv pro jednotlivé uživatele firemní sítě. Centrální úložiště identit dává managementu a správcům IT nástroje a technologie, které jsou potřebné pro kontrolu přístupu uživatelů k podnikovým datům a aplikacím.

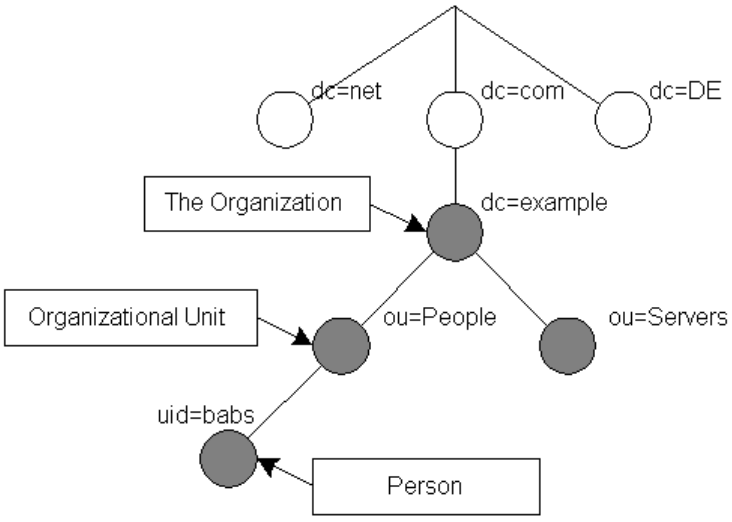
3.4.1 LDAP

LDAP (*Lightweight Directory Access Protocol*) je standardní protokol pro ukládání a přístup k datům na adresářovém serveru, funguje na principu klient-server a v komunikaci využívá jak synchronní tak asynchronní mód. LDAP byl založen na protokolu X.500 (X.500 - Mezinárodní standard, vyvinutý spolkem International Consultative Commitee of Telephony and Telegraphy, pro formátování elektronických zpráv přenášených mezi počítačovými sítěmi), který byl vyvinut ve světe ISO/OSI, ale již se nedostal do praxe. Bylo to zejména pro jeho „velikost" a následnou „těžkopádnost". LDAP z protokolu X.500 obsahuje většinu z jeho důležitých funkcí, ale jak již z názvů napovídá, jedná se o „odlehčenou" (lightweight) verzi. [7]

LDAP je vhodný pro udržování adresářů a práci s informacemi o uživatelích. Podle tohoto protokolu jsou jednotlivé položky na serveru ukládány formou záznamů a uspořádány do stromové struktury (jako ve skutečné adresářové architektuře). Lze ho využít např. pro správu elektronické pošty, telefonních čísel, konfigurace aplikací, samby serveru atd. Součástí LDAP je také autentizace klienta.

DIT

LDAP pro záznam dat využívá strom objektů – DIT (Directory Information Tree). Strom je tvořen jednotlivými záznamy. Každý záznam obsahuje své unikátní jméno DN (Distinguished Name) skládající ze svého relativního jména RDN (Relative Distinguished Name) a RDN předchůdce. Struktura popisuje Obrázek 11.



Obrázek 11 – LDAP adresářový strom, zdroj [30]

LDIF

LDIF (Data Interchange Format) je speciální standardizovaný formát, ve kterém se uchovávají záznamy a jejich atributy.

Příklad:

```
dn: uid=jnovak,dc=ldap
objectClass: person
cn: Jan Novák
sn: Novák
userPassword: {SSHA}JkvL8jDBU4dJ1hnPol48XO8G5eOLeKLi
telephoneNumber: 587697
description: Testovací uživatel
```

Schéma

Schéma představuje soubor definic (pravidel) nad adresářovým stromem. Definuje množinu objektů a jejich atributy – povinné a nepovinné.

Grafická nadstavba administrace

phpLDAPadmin

Pro snadnou administraci LDAP stromu lze využít mnoha webových aplikací. V implementaci byl vybrán phpLDAPadmin. Jedná se rovněž o open source řešení s bohatou dokumentací, výhodou je česká jazyková mutace.

Obrázek 12 – Základní obrazovka aplikace phpLDAPadmin

Jeho hlavní funkce jsou:

- prohledávání LDAP stromu,
- prohlížení LDAP schémat,
- vytváření, editace, mazání, kopírování a vyhledávání záznamů.

Zhodnocení

Jako implementace LDAP byla vybrána distribuce OpenLDAP jako možné open source řešení, které je hojně využívané a existuje k němu mnoho dokumentace.

Mezi hlavní výhody LDAPu patří především snadná implementace, rozšíření a široká podpora ze strany vývojářů. Nelze také opomíjet vysokou rychlost vyhledávání v DIT.

3.4.2 Active Directory

Další možným řešením pro centrální uložení identit je Active Directory. To je implementováno pomocí adresářových služeb LDAP firmou Microsoft, jedná se však o čistě komerční řešení pro použití v prostředí systému Microsoft Windows. Active Directory umožňuje administrátorům nastavovat politiku, instalovat programy na mnoho počítačů nebo aplikovat kritické aktualizace v celé organizační struktuře. Informace a nastavení Active Directory jsou uchovávány v centrální organizované databázi. [8]

3.5 Systém pro správu chyb a požadavků

Systém pro správu chyb a požadavků, v praxi nazývaný bug tracking systém, je speciální aplikace, která je cenným nástrojem při vývoji a testování softwaru. Umožňují vývojářům softwaru sledovat chyby, které se na projektu objevily. Uživatelé, testeři a další mohou pomocí těchto systémů zakládat hlášení o chybách, na které během užívání nebo testování narazili. Tyto chyby jsou pak evidovány a slouží pro identifikaci problému, který je následně řešen.

Bug tracking systém umožňuje tyto základní funkce:

- evidenci chyb a jejich závažnosti,
- přidělení chyby ke konkrétnímu projektu,
- evidence podrobností o chybném chování softwaru a jak k chybě dochází,
- evidence osoby, která na chybu narazila a zaevidovala ji,
- nastavení stavu chyby nebo požadavku,
- další stavy jako zdali je chyba/požadavek připraven k otestování,
- informace o nasazení o nahrání na ostrou verzi.

V podnikovém prostředí může být bug tracking system použit i jako generátor sestav produktivity programátorů u dané chyby. V praxi dále bývá úzký vztah mezi těmito systémy a helpdesky, kdy tým aplikační podpory z těchto helpdesků vytváří tzv. tikety (požadavek na řešení problému) a programátoři tento problém řeší. Pro management má systém sledování chyb a požadavků i význam pro sledování samotných projektů, uživatelů, kteří chyby zakládají a rychlosti řešení programátorů.

3.5.1 Mantis

Mantis je webový systém pro evidenci chyb v software (bug tracking). Je napsán v jazyce v PHP a ke své funkčnosti vyžaduje databázi (MySQL, MS SQL nebo PostgreSQL) a webový server. [9]

Mantis umožňuje vývojářům mít přehled o všech chybách, které se vyskytly během vývoje. Uživatelské rozhraní Mantisu obsahuje barevně rozlišený seznam chyb, které upozorňují uživatele na aktuální stav různých chyb. Záznamy o chybách do něj může vkládat více uživatelů v různých rolích. Chyby jsou v Mantisu evidovány k projektům, kterých může být více. Ty na sobě nejsou závislé a na každém z nich může pracovat jiná skupina uživatelů.

Webové rozhraní obsahuje mnoho užitečných funkcí, které mohou napomoci při identifikaci a rychlé opravě problému.

Příjemný je i fakt, že lze uživatele snadno autorizovat pomocí LDAP serveru bez většího zásahu do kódu aplikace. Mantis má v sobě již integrovánu tuto možnost ověřování, kdy stačí jednoduše nakonfigurovat soubor s konfigurací. Velkou výhodou je česká jazyková mutace, která je obsažena ve standardní instalaci.

Mantis je zdarma pro použití a upravování. Je povoleno ho šířit, tak jak je poskytován v rámci licence GPL.

Základní funkce aplikace Mantis:

- evidence chyb v projektech,
- možnost reakce na danou chybu,
- možnost přiložení libovolného souboru k založeným chybám,
- monitorování konkrétních chyb,
- možnost exportu do CSV, Microsoft Word/Excel,
- velká variabilita nastavení formuláře pro evidenci chyb,
- automatické generování dokončených změn na projektech,
- RSS čtečka,
- a další.

Obrázek 13 – Základní obrazovka aplikace Mantis

Zhodnocení

Mantis byl vybrán jako efektivní open source nástroj pro evidenci chyb. Jeho velké klady popisuje seznam základní funkcí. Disponuje jednoduchým a intuitivním ovládání v české jazykové mutaci. Snadnou práci ocení i zákazník, který se může aktivně podílet na testování softwaru a případném hlášení chyb.

3.6 Dokument management systém

Dokument management systém (DMS), nebo také systém pro správu dokumentů (EDM - Electronic Document management), je systém určený ke správě elektronických dokumentů nebo digitalizovaných papírových dokumentů. [13]

Dokument management systém řeší tyto základní požadavky:

- Začleňování dokumentů:
  - *Vkládání souborů* – hromadné nebo přímé vkládání dokumentů.
  - *Meta popis dokumentů* – popis jednotlivých dokumentů a jejich revizí, který nemusí být součásti obsahu dokumentu. Metapopis je využíván pro rychlejší vyhledávání konkrétního dokumentu. Součástí metapopisu je i kategorie, do které je dokument zařazen. Dále sem patří autor, rešerše atd.
  - *Správa metapopisu* – administrace metapopisů, jaké mohou dokumenty mít.
- Správa verzí dokumentu:
  - *Přidělování identifikátorů* – dokumentům a revizím přináší jednoznačnost v rámci systému správy dokumentů.
  - *Verzování* – verze (revize) slouží pro rozlišení různých stavů, ve kterých se daný dokument během svého života nacházel. Ideální je, pokud systém uchovává rozdíly pouze jako binární změnu souboru a soubor lze zrekonstruovat do požadovaného funkčního stavu.
- Dostupnost dokumentace (centralizovaný přístup):
  - *Centrální zdroj* – dokumenty jsou poskytovány z jediného zdroje a jsou vždy aktuální.
  - *Snížení redundance dat* – díky centralizování dat není potřeba vyměňovat data mezi uživateli, stačí pouze poskytovat odkaz na daný soubor (místo, kde se nachází).
- Dohledatelnost:
  - *Fultextový index* – vyhledávání možné nejen pomocí metapopisů, ale také přímo v dokumentech.
  - *Kategorizace* – zobrazování dokumentů dle jejich kategorií, které by měly být organizované do hierarchických struktur, kde nadřazená kategorie autoamticky zahrnuje své podřízené (stromová struktura).
- Přístupová práva – blokování přístupu neautorizovaným uživatelům.
- Archivace – archivace pomocí export dokumentů včetně jejich adresářové struktury (využití např. XML).
- Workflow – nástroje pro sledování „toků“ dokumentů, umožnění uživatelům vidět pohromadě dokumenty k vyřízení, např. seznámení s novými interními pravidly ve firmě, dokumenty připravené na schválení atd.

3.6.1 ownCloud

Produkt ownCloud, jak už z názvu napovídá, je cloudové úložiště, které zahrnuje potřeby dokument management systému a zároveň nabízí další služby, které uživatel určitě ocení.

Cloudové úložiště je obdobou služeb virtuálního úložiště, které je určeno k ukládání a sdílení dat. Hlavním smyslem použití cloudu je potřeba uživatelů mít svá data přístupna na více zařízení, to souvisí s oblibou využívání mobilních zařízení. Místem pro osobní či firemní data se tak stává cloudové úložiště, které můžeme rovněž nazvat online diskem. Cloudové úložiště řeší synchronizaci dokumentů mezi různými zařízeními uživatele. Např. pokud pracujeme v práci na nějakém dokumentu, nemusíme si jej přenášet na flash disku domů, kde ho najdeme ve stejném stavu, jako jsme ho v práci dokončili. Pokud nemáme nainstalován klient cloudu, pak můžeme soubor editovat přes webový prohlížeč. [14]

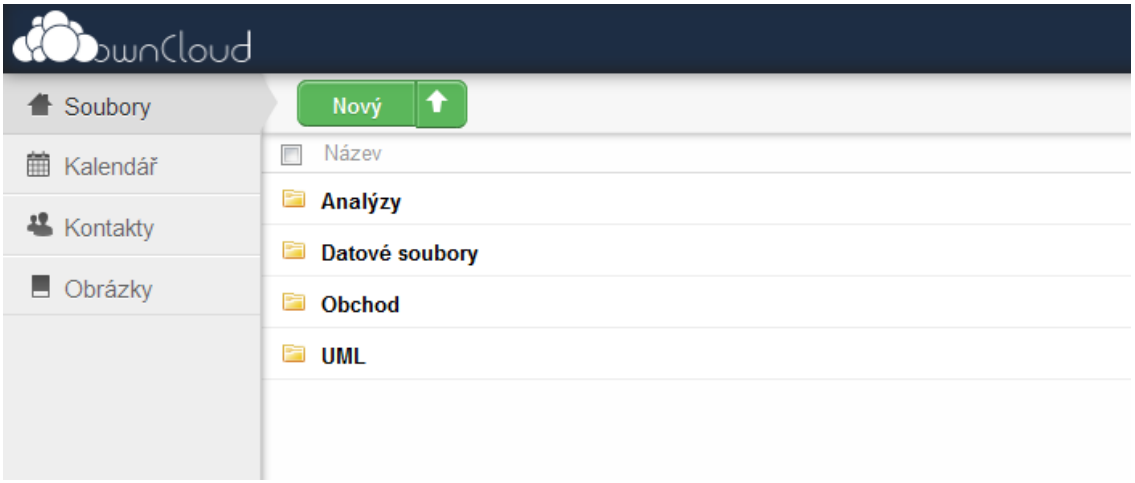
OwnCloud je bezplatný open source systém, který svou funkcionalitou konkuruje např. Dropboxu[2] nebo Google Drive[3]. Lze ho zprovoznit na vlastním serveru nebo je možnost použití placeného hostingu.

Aplikace má jednoduché, intuitivní uživatelské prostředí, přičemž instalace je také snadná. Správa systému nabízí registraci libovolného počtu uživatelů, kteří se zařazují do skupin s různým oprávněním. Aplikaci je možné přepnout do českého prostředí. Systém nabízí dva základní typy uživatelů: User a Admin.

Hlavní funkce a výhody produktu ownCloud:

- dostupnost dat z libovolných zařízení,
- umístění libovolného formátu dat,
- nastavení práv k souborům nebo skupinám,
- verzování dat,
- vyhledávání v datech,
- snadná úprava dokumentů na disku zařízení nebo přímo z webového prohlížeče,
- textový editor pro editaci souborů,
- libovolné zakládání skupin pro data,
- sdílení mezi dalšími uživateli,
- synchronizační klient pro Linux/Windows/Mac OS X,
- podpora aplikací třetích stran,
- další možnosti jako kontakty, galerie, kalendář a mnoho dalších. [14]

[2] https://www.dropbox.com
[3] https://drive.google.com



Obrázek 14 – Základní obrazovka aplikace ownCloud

Zhodnocení

OwnCloud byl vybrán jako efektivní open source nástroj pro správu dokumentů. Z výčtu jeho funkcionality vyplývá, že nabízí většinu požadavků na kvalitní DMS a přináší další zmiňované výhody cloudového úložiště.

Výhodou je i snadná instalace a napojení na LDAP formou doinstalování dalšího modulu a nastavení parametrů ve formuláři. Jako nevýhodu můžeme zmínit však to, že se jedná poměrně o nový produkt, který zatím nebyl dlouhodobě testován. Při implementaci se vyskytl problém nekompatibility s některými webovými prohlížeči (nefunkční některá tlačítka).

3.7 Managament znalostí, wiki systémy

Managament znalostí, nebo také knowledge management, můžeme definovat jako tvorbu a údržbu znalostí. Hlavním důvodem využití managementu znalostí je identifikovat a uchopit konkrétní znalost, zkušenosti nebo jiné dovednosti a umožnit jejich prezentaci ostatním uživatelům. Využití znalostí přispívá k vyšší kvalitě, efektivitě a produktivity práce těchto uživatelů. [15]

Ke správě znalostí slouží speciální systémy, které umožňují sběr, sjednocení a šíření znalostí. Jedním z takových systémů je tzv. wiki systém. Wiki představuje označení webů, které umožňují uživatelům přidávat obsah podobně jako v internetových diskusích. Navíc ale dovolují tento obsah spravovat – editovat a mazat. Můžeme také říct, že wiki systém „vytváří weby“.

Wiki systém využívá jednoduchý značkovací jazyk, pomocí něhož vytváří jednotlivé dokumenty – webové stránky. K jejich interpretaci se pak používá webový prohlížeč. Jedná stránka ve wiki systému se nazývá „wiki stránka“ a skupina propojených stránek se nazývá zmiňované „wiki“ [16]. Snaha celého systému je, aby byl pro uživatele co nejjednodušší, zpravidla nebývá vyžadována ani registrace uživatelů, nicméně tyto systémy správu uživatelů nabízejí.

Wiki systémy reprezentují obsah třemi způsoby:

- kódem HTML,
- zobrazení kódu v prohlížeči – naformátování stránky,
- uživatelsky editovatelný zdrojový kód, ze kterého se vytvoří HTML kód (využiti tzv. wysiwyg editoru – editor pro úpravu HML v podobě, jaké je interpretován webovým prohlížečem). [16]

Hlavním důvodem využití zmiňovaného wysiwyg editoru je i fakt, že HMTL jazyk obsahuje velké množství tagů, jejichž použití může být komplikované pro uživatele.

Jedním z nejznámějších wiki systémů je MediaWiki, který i zastává management znalostí výsledné platformy a je podrobně popsán v následující kapitole.

3.7.1 MediaWiki

MediaWiki je open source software psaný v jazyce PHP a patří mezi profesionální wiki systémy.

Byl původně vytvořený pro potřeby webové encyklopedie Wikipedia[4]. Je navržený pro práci s velkým množstvím uživatelů, disponuje propracovanou strukturou stránek bez omezení výkonu a funkčnosti. MediaWiki je díky těmto vlastnostem velmi oblíbený wiki systém. [10]

Základní vlastnosti a funkce

MediaWiki je stavěna na velké množství uživatelů a každý z nich může mít vlastní, upravitelný skin. Aplikace plně podporuje UTF-8 a je přeložena do mnoha jazyků, při čemž nechybí ani čeština. Velké množství vlastností a chování MedieWiki lze nastavit pouze pro konkrétního uživatele. Je zde možnost i diskusí, které fungují formou speciálních článků.

V následujících odstavcích jsou dále rozebrány základní funkčnosti, které MediaWiki uživatelům nabízí.

Editace článků:

- panel nástrojů pro snadnou editaci,
- možnost editace pouze malých částí stránek,
- souhrnný seznam posledních editací daného článku,
- řešení konfliktů při editaci jednoho článku více uživateli.

[4] http://en.wikipedia.org

Struktura článků:

- klasifikace článků dle kategorií s možností hierarchického procházení článků,
- odkazy na konkrétní sekce článků,
- volitelného zobrazení obsahu pro dlouhé články,
- možnost odkazů na další stránky a tvorba tzv. pahýlů – odkaz na články pod určitou velikost zobrazený jinou barvou.
- mezi jazyčné odkazy pro odkazování na wiki v jiných jazycích,
- jmenné prostory pro rozlišení obsahu článků.

Syntaxe článků:

- úpravy syntaxe založeny na systému UseMod (využití HTML a vlastní wiki syntaxe),
- možnost vkládání informací s využitím LaTexové syntaxe (např. matematické vzorce),
- plná podpora UTF-8.

Sledování změn článků:

- grafické rozlišení změn v jednotlivých článcích,
- sledování změn konkrétního článku,
- sledování nedávných změn – rozdíl mezi různými verzemi článků, historie úprav článků, tvorba rychlých odkazů,
- zobrazení posledních článků, na kterých pracoval daný uživatel.

Multimediální data článků:

- možnost vkládání multimediálních souborů jako jsou obrázky a zvuky,
- automatická úprava rozlišení obrázku při nahrávání.

Bezpečnost aplikace:

- různé role oprávnění (anonymní uživatel, registrovaný uživatel, administrátor, vývojář atd.) a jejich předdefinována práva,
- volitelné schéma pro konkrétní roli uživatelů.

Zhodnocení

MediaWiki jakou open source řešení nabízí mnoho skvělých a nepostradatelných funkcí, které pomohou pomáhat vývojářům udržovat vlastí know-how vývoje. Jako jedna z mála obsahuje rozšíření pro ověřování uživatelů na LDAP serveru, což přináší i velké pohodlí pro administrátora. Důvodem pro vybrání bylo velké rozšíření, snadná instalace a konfigurace, intuitivní ovládání pro uživatele a i samotný výkon aplikace. Lze jej využít pro malé i velké projekty. Výhodou je i fakt, že nabízí české prostředí.

Jako negativa lze však jmenovat to, že MediaWiki není vhodná pro situace, kdy potřebujeme jednoznačně omezovat přístup na konkrétní částí obsahu (např. skupinu článků). Současně MediaWiki není dobré využívat jako specializované fórum pro vývojáře. Měla by sloužit pouze pro sdílení informací mezi vývojáři.

3.8 Integrované vývojové prostředí

Integrované vývojové prostředí (zkráceně IDE - Integrated Development Environment) představuje software, který usnadňuje programátorům vývoj v použitém programovacím jazyce. Je tvořen editorem zdrojového kódu, kompilátorem, případně interpretem a často také debuggrem. Některé IDE obsahují systém pro rychlý vývoj aplikací (RAD), který slouží pro vizuální návrh grafického uživatelského rozhraní. Pokud využíváme objektově orientované programování, můžeme v IDE použít tzv. object broker. [12]

Hlavním důvodem využití vývojového prostředí je zvýšení produktivity programátora. Prostředí je navrženo provázanými komponentami s podobným uživatelským rozhraním. To pro programátora znamená, že nemusí přepínat mezi jednotlivými módy jako při použití individuálních vývojových nástrojů. Abychom však dosáhli vysoké produktivity, je třeba IDE dobře znát a to často bývá až postupným načerpáním praktických zkušeností.

IDE je samostatný software, ve kterém probíhá celý vývoj aplikací. Tento software poskytuje mnoho nástrojů pro vývoj, úpravu, překlad i samotné ladění vyvíjené aplikace. Cílem IDE je shrnout vlastnosti a schopnosti nástrojů programovacího jazyka do ucelené podoby, která snižuje čas potřebný k pochopení samotného jazyka a tím zvyšuje produktivitu programátora. Úzká integrace činností může ještě více přispět ke zvýšení produktivity, např. lze zdrojový kód překládat ještě během psaní a k tomu navíc sledovat syntaktické chyby. [12]

Integrované vývojové prostředí bývá navržené pro konkrétní programovací jazyk, tak aby poskytovalo sadu vlastností, které se maximálně přizpůsobují konkrétním paradigmatům daného jazyka. Mezi rozšířené, komerční IDE můžeme jmenovat Microsoft Visual Studio a jako open source Eclips, NetBeans. Všechny tyto prostředí jsou tzv. více-jazyková. To znamená, že podporují vývoj ve více jazycích.

3.8.1 NetBeans IDE

NetBeans se řadí mezi úspěšná Open Source řešení, které disponuje velmi rozsáhlou uživatelskou základnou a širokou komunitou vývojářů. NetBeans vyvinula v roce 2000 firma Sun Microsystems, která toto vývojové prostředí zároveň financuje. Dnes existují dva produkty:

- NetBeans IDE – vývojové prostředí,
- NetBeans Platform – vývojovová platforma. [11]

Oba produkty, NetBeans IDE i Platform, jsou vyvíjeny pod licencí Open Source a je možné je zdarma využívat v komerčním i nekomerčním prostředí. Zdrojový kód je

k dispozici pod licencí GNU a GNL. V popisu platformy se však dále budu zabývat pouze posel NetBeans IDE.

NetBeans IDE je vývojové prostředí, které programátorům umožňuje psát, překládat, ladit a distribuovat aplikaci. Prostředí je naprogramováno v jazyce Java, avšak podporuje prakticky jakýkoliv programovací jazyk. Dále lze do NetBeans IDE doinstalovat mnoho modulů, které prostředí rozšiřují (např. modul pro práci s SVN, podporou konkrétního frameworku atd.)

Základem NetBeans IDE je propracovaný multi-jazykový editor, debugger, správce projektů, správce verzí souborů a jedinečné funkce pro týmovou spolupráci vývoje.

Hlavní okna a záložky NetBeans IDE

Hierarchie projektů

Přehlednost v otevřených projektech zabezpečuje stromová struktura. Nacházejí se tam zdrojové soubory projektů, ale i konfigurační soubory, knihovny či jiné potřebné soubory.

Přehled souborů v projektech

V této záložce se nachází stromová struktura adresářů otevřených projektů. Oproti hierarchii projektů se zde nacházejí i překompilované soubory a spustitelný soubor výsledné aplikace.

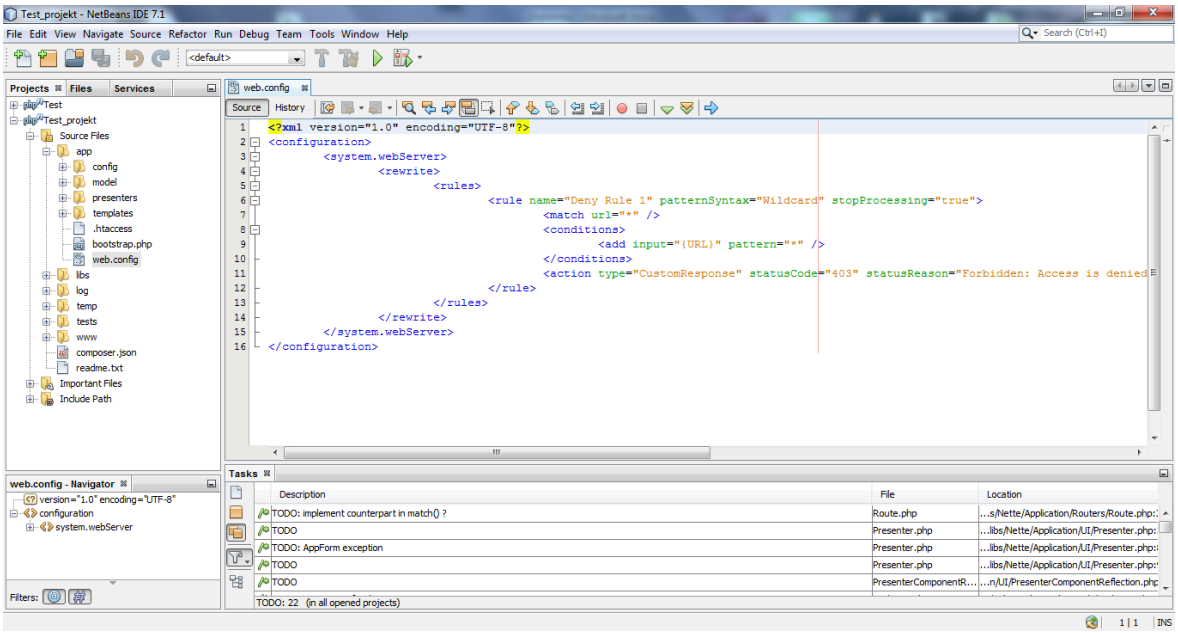
Služby

Mezi nejpoužívanější služby, které jsou k dispozici, patří správa serverů, databází a webových služeb. NetBeans IDE poskytuje ovladače na několik databází jako Java DB, MySQL, Oracle, PostgreSQL a jdalší. V případě potřeby lze vytvořit připojení k jiné databázi přidáním příslušného ovladače. Po vytvoření připojení k databázi je možné prohlížet data uložená v tabulkách, přidávat, editovat nebo mazat tabulky, sloupce atd.

Editor zdrojového kódu

Hlavní výhodou NetBeans IDE výborný editor, který urychluje psaní zdrojového kódu. Prostředí také výrazně napomáhá při přepisování kódu. Různé opravy kódu jsou tak nejen jednodušší a rychlejší, ale i bezpečnější, protože se snižuje výskyt chyb vyplývajících z nepozornosti.

Pomocníkem při psaní zdrojového kódu, bez kterého si už psaní programů nelze představit, je zvýrazňování syntaxe. Spolu s automatickým formátováním nám pomáhá lépe se orientovat ve vytvořených aplikacích. Pomocníkem, kterého si určitě oblíbí nejen začátečníci, ale i zkušení programátoři, je doplňování zdrojového kódu. NetBeans Pars kód při psaní a tak již při napsání prvního písmenka a po stisknutí klávesové zkratky Ctrl-Space nám nabídne možnosti, které se tam dají vložit. Na vrchu nabídky jsou potom nejčastěji používané výrazy. Tato nabídka neobsahuje pouze klíčová slova, ale i atributy, metody či třídy, které jsme si sami definovali. Editor navíc kontroluje i syntaktickou a sémantickou správnost a v případě chyby podtrhne nesprávný výraz a na levém okraji editoru se zobrazí vykřičník. Po kliknutí na tento vykřičník se zobrazí možné řešení problému. Toto řešení však nemusí být vždy dobré, protože editor není schopen rozeznat logiku programu. [24]



Obrázek 15 – Základní obrazovka aplikace NetBeans IDE

Zhodnocení

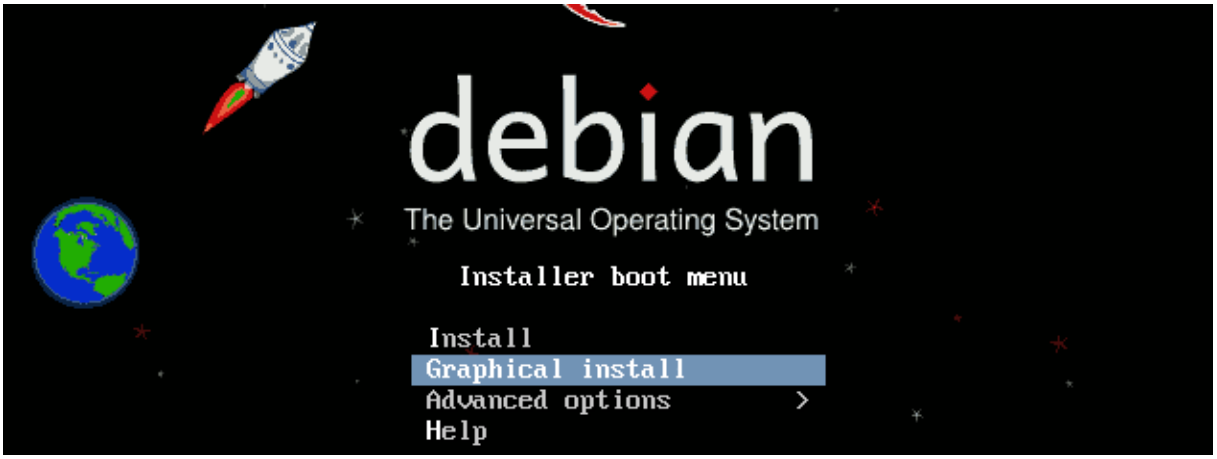
NetBeans IDE je velmi propracované a jednoduché vývojové prostředí, které nabízí mnoho užitečných funkcí pro vývojáře. Výhodou je i to, že se jedná o multiplatformní aplikaci a možností doinstalování velkého množství pluginů, např. pro práci s SVN. Velký konkurent představuje Eclips, který má podobné vlastnosti popsány výše. Z obou testovaných nelze říci, který je lepší, obsahuje více možností doinstalování dalších pluginů atd. Avšak se NetBeans IDE jevil jako jednoduší, k tomu pomohlo i mnoho online návodů a při prvních krokách s programováním bych jej doporučil více.

4 Instalace a konfigurace jednotlivých produktů serveru

Tato kapitola popisuje instalaci a konfiguraci všech softwarových produktů použitých v platformě.

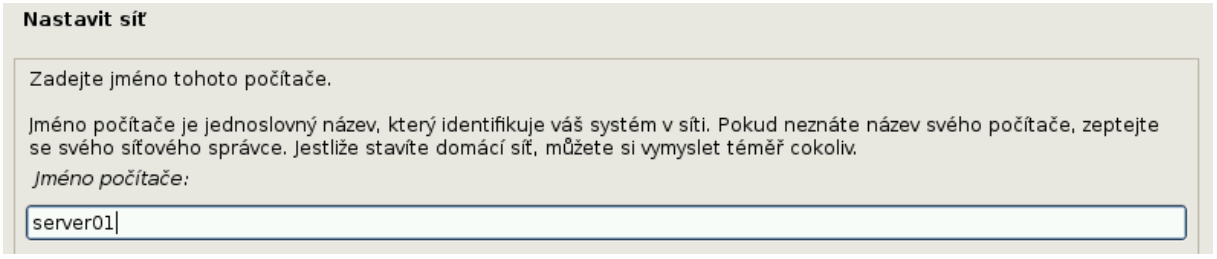
4.1 Instalace operačního systému DEBIAN

Ze stránek produktu debian[5] byla stažena poslední verze (v platformě použita a popsána verze 6.0.4 Squeeze). Instalace byla stažena jako image, která se spustí přes libovolný daemon. Instalace byla spuštěna v grafickém režimu.



Obrázek 16 – Výběr typu instalace

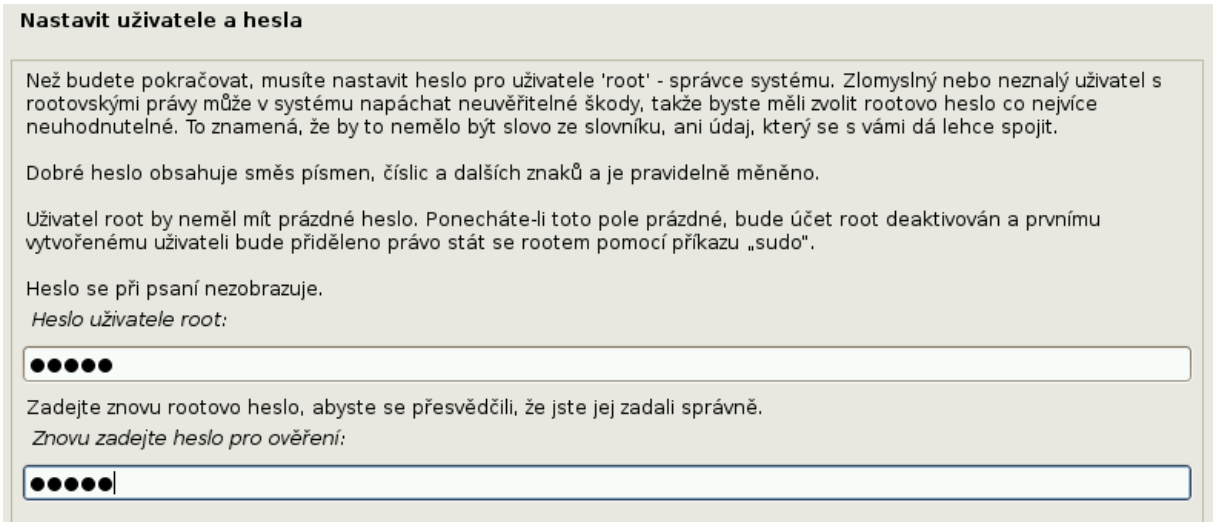
Po výběru instalace bylo zvoleno české prostředí a nastavena síť dle instrukcí.



Obrázek 17 – Název počítače

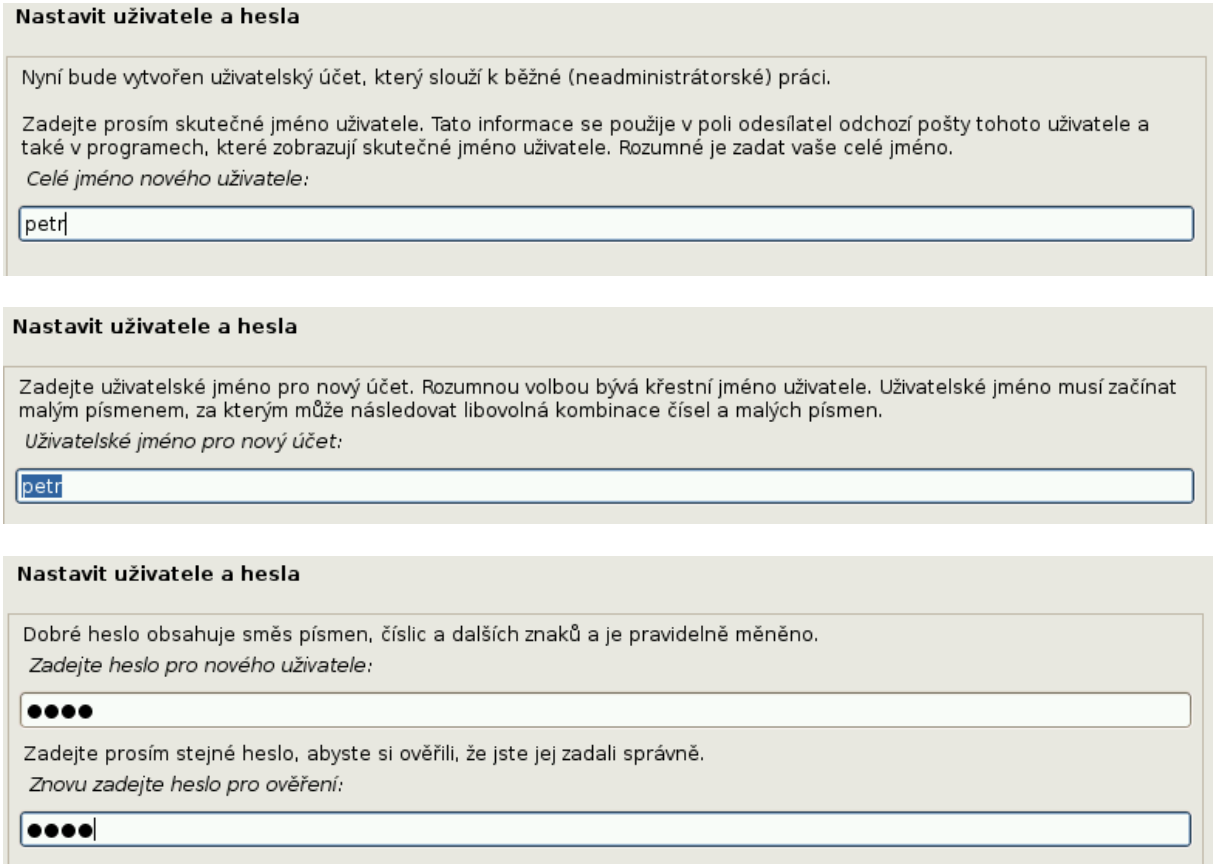
Následně bylo nastaveno heslo **root** uživatele – správce systému. Jako heslo byl zvolen řetězec: **asdfg**.

[5] http://www.debian.org



Obrázek 18 – Nastavení hesla root uživatele

Dále byl vytvořen uživatelský účet běžného uživatele. Uživatelské jméno bylo nastaveno na: **petr** a heslo: **petr**.

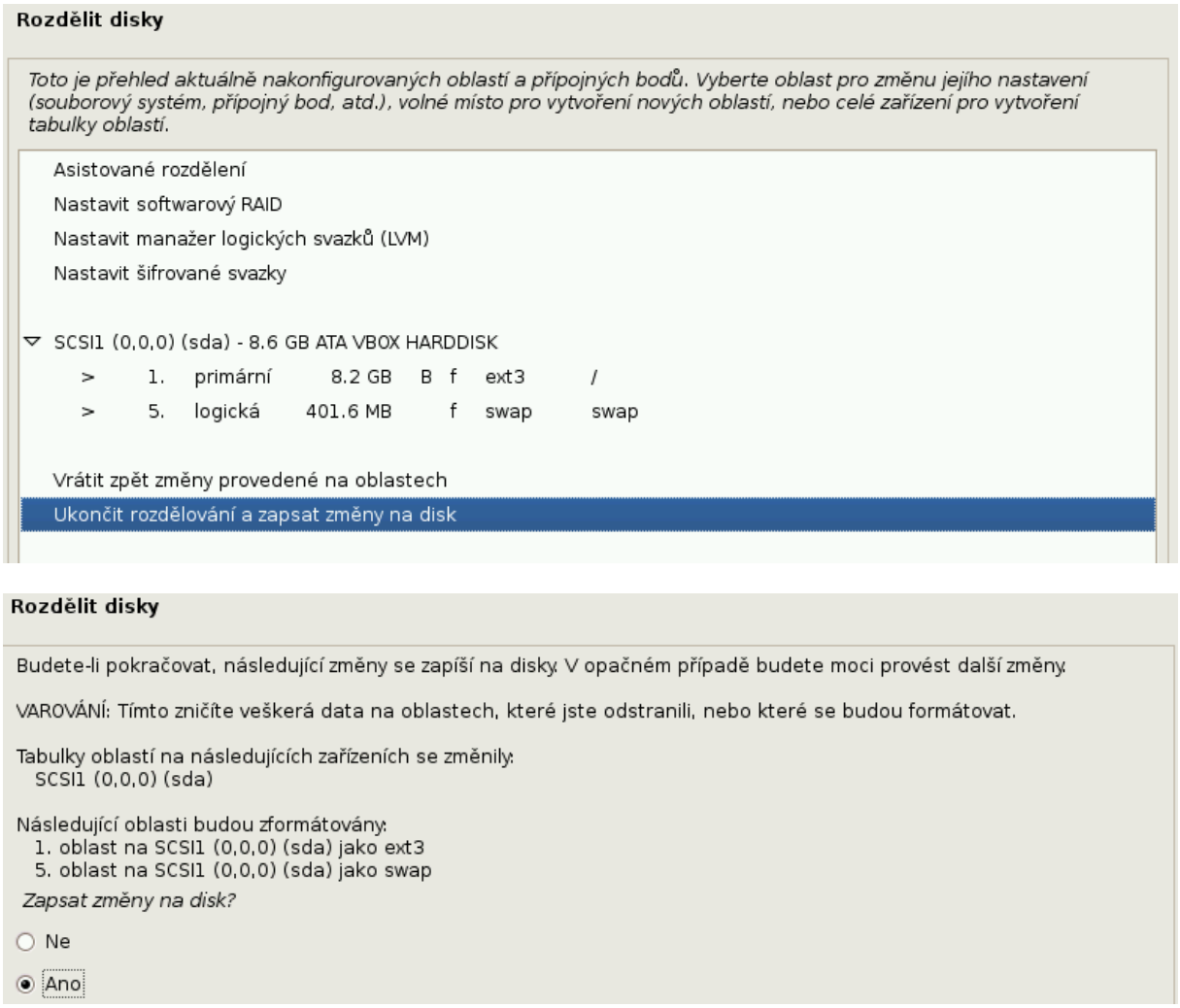


Obrázek 19 – Nastavení běžného uživatele

V následujícím kroku byl rozdělen disk. Nastavení je v tomto kroku individuální, můžeme se však držet doporučeného nastavení popsaného na následujících obrázkách. Podrobné info najdeme na stránkách projektu[6].

Obrázek 20 – Rozdělení disku 1

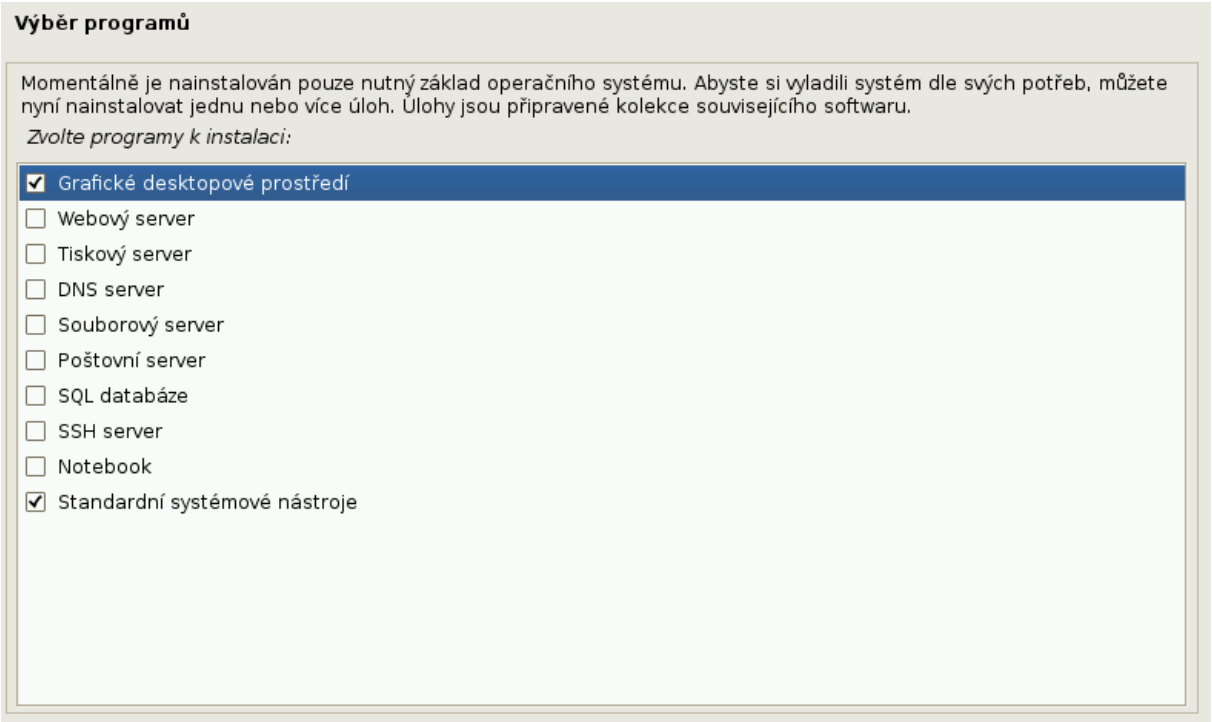
[6] http://www.debian.org/releases/stable/installmanual



Obrázek 21 – Rozdělení disku 2

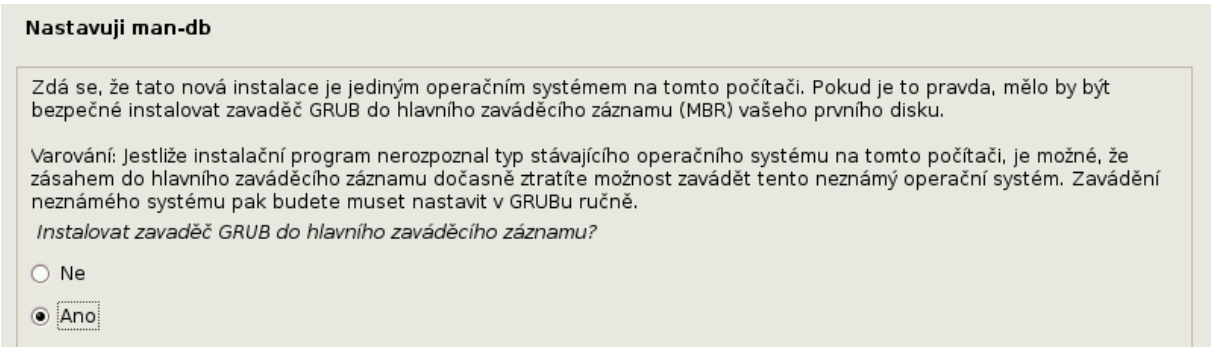
Dále byl nastaven správce balíku a vybrán repozitář pro možnost stažení dalšího softwaru.

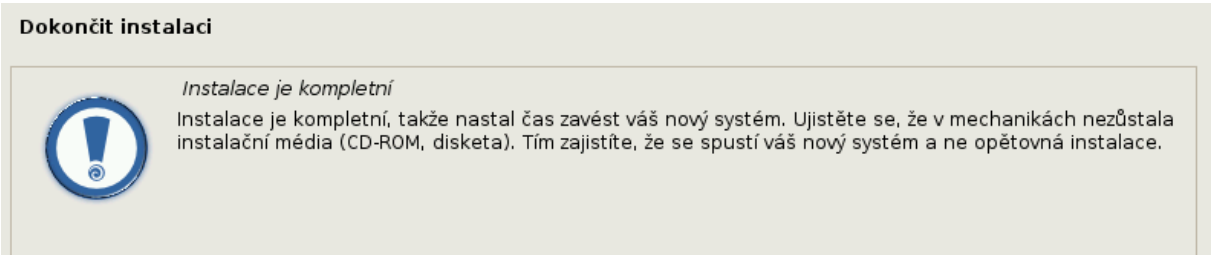
V následující části byly vybrány další části operačního systému, které budou dále využívány. Instalace webového serveru a další nebyly vybrány záměrně, jejich instalace a konfigurace je popsána samostatně.



Obrázek 22 – Výběr dalších částí operačního systému

Potvrzení nainstalování zavaděče a dokončí instalace.





Obrázek 23 – Dokončení instalace

4.2 Instalace web serveru, databáze MySQL a PHP

4.2.1 Apache

Webový server Apache byl nainstalován zadáním následujícího příkazu (instalace obsahuje potřebné balíčky):

```
# aptitude install apache2 apache2-doc
```

4.2.2 MySQL

Databáze MySQL byla nainstalována zadáním následujícího příkazu (instalace obsahuje potřebné balíčky):

```
# aptitude install mysql-server mysql-client
```

Během instalace bylo zadáno následující uživatelské jméno pro správce databáze: uživatel: **root**, heslo: **asdfg**.

4.2.3 PHP

Instalace PHP byla spuštěna zadáním následujícího příkazu (instalace obsahuje potřebné balíčky):

```
# aptitude install php5 php5-mysql libapache2-mod-php5
```

Na konec byl proveden restart webového serveru Apache:

```
# /etc/init.d/apache2 restart
```

4.3 Otestování nainstalovaných služeb

4.3.1 Apache

Apache byl otestován zadáním adresy *http://localhost/* do adresového řádku v prohlížeči, pokud se zobrazí stránka *„It's Works!“,* pak Apache funguje správně.

4.3.2 PHP

Pokud Apache funguje, můžeme otestovat PHP. To bylo uděláno následujícím způsobem. Byl vytvořen testovací soubor *testing.php*, příkazem:

```
# nano /var/www.testing.php
```

Vložen do něj příkaz pro zjištění nastavení PHP:

```
<?php phpinfo(); ?>
```

a nakonec byl soubor uložen.

Následně bylo přepnuto zpět do webového prohlížeče a zadána adresa *http://localhost/testing.php/*. Pokud se zobrazí konfigurace PHP, pak je PHP na webovém serveru funkční.

4.3.3 MySQL

Pro databázi MySQL je třeba zajistit, že bude adresa vázána na adresu *localhostu*, ta by měla být *127.0.0.1*, to ověříme příkazem:

```
# cat /etc/hosts | grep localhost
```

Nyní je ještě třeba ověřit správnost bind adresy, ta se nastavuje v souboru *my.cnf*, byla zobrazena následujícím příkazem:

```
# cat /etc/mysql/my.cnf | grep bind-address
```

Pokud se zobrazí tato IP adresa, pak je nastavení správné:

```
bind-address = 127.0.0.1
```

Pokud je adresa jiná, je třeba ji ručně změnit.

4.4 Instalace phpMyAdmin

Instalaci balíčku byla spuštěna zadáním následujícího příkazu:

```
# aptitude install libapache2-mod-auth-mysql phpmyadmin
```

Instalace vyzve k výběru webového serveru pro automatickou konfiguraci. Byl vybrán server **apache2**.

Následující dialog se táže, zdali chceme nastavit novou databázi pomocí dbconfig-common. Protože se jedná o novou instalaci, bylo zvoleno ano.

Následně je vyzváno k zadání root hesla k MySQL. Bylo zadáno **asdfg**.

Nyní je instalace phpMyAdminu kompletní.

Instalace byla otestována zadáním adresy *http://localhost/phpmyadmin/*. Po správné instalaci se načte úvodní přihlašovací stránka phpMyAdmin. Nyní už zbývá jen otestovat přihlášení uživatele **root** s příslušným heslem zadaným při instalaci.

4.5 Instalace LDAP serveru

Instalace byla zahájena provedením příkazu *aptitude*, byl nainstalován balíček obsahující daemona OpenLDAP serveru *slapd* a balíček *ldap-utils*, který obsahuje utility pro LDAP:

```
# aptitude install slapd ldap-utils
```

Instalace proběhla s touto konfigurací.

```
Omit OpenLDAP server configuration? No
DNS domain name: ldap
Organization name? server01
Administrator password: asdfg
Confirm password: asdfg
Database backend to use: BDB
Do you want the database to be removed when slapd is purged? No
Allow LDAPv2 protocol? No
```

4.5.1 Konfigurace slapd

Konfigurace LDAP byla provedena pomocí konfiguračního souboru:

```
/etc/ldap/slapd.d/cn=config/olcDatabase={1}hdb.ldif
```

Bylo nastaveno BaseDN na *dc=ldap*. Nastavení bylo provedeno nahrazením defaultní domény touto hodnotou ve všech atributech, které ji obsahovaly.

Aby mohlo být přistupováno ke konfiguraci přes LDAP, bylo zkopírováno heslo ze souboru *olcDatabase={1}hdb.ldif* do souboru */etc/ldap/slapd.d/cn=config/olcDatabase={0}config.ldif*.

Jednalo se o řádek začínající hodnotou **olcRootPW**.

Heslo nastavené při instalaci se vztahuje k uživateli **cn=admin**, který se nachází v BaseDN (*dc=ldap)*. Toto heslo lze použít i k připojení konfigurace LDAP, stačí zadat uživatele *cn=admin* a jako BaseDN zvolit *cn=config*.

4.5.2 Konfigurace ldap-utils

Konfigurační soubor *ldap-utils* se nachází v umístění */etc/ldap/ldap.conf* a slouží k tomu, aby *ldap-utils* věděly, ke kterému serveru se mají připojovat, a jaké má tento server *BaseDN*.

Byly nastaveny následující hodnoty:

```
BASE dc=ldap
URI ldap://127.0.0.1/
```

Server byl následně spuštěn tímto příkazem:

```
# /etc/init.d/slapd start
```

4.6 Instalace phpLDAPadmin

Instalace byla spuštěna příkazem *aptitude*, kterým byl nainstalován balíček *phpldapadmin*:

```
# aptitude install phpldapadmin
```

Po instalaci byl upraven soubor s konfigurací, tu najdeme v souboru */etc/phpldapadmin/config.php*.

A nastaven následující řádek (hodnota v poli) na **dc=ldap**:

```
$servers->setValue('server','base',array('dc=ldap'));
```

Tímto byl nastaven defaultní DN LDAP serveru, lze jej však při přihlášení změnit.

Další konfiguraci si aplikace se během instalace nastavila sama.

PhpLDAPadmina spustíme z *adresy http://localhost/phpldapadmin/*

4.6.1 Otestování LDAP

K otestování LDAPu byla použita nainstalovaná aplikace phpLDAPadmin. Byly zadány následující údaje:

```
Přihlašovací DN: cn=admin, cn=config
Heslo: asdfg
```

Pokud se úspěšně přihlásíme a uvidíme strom **dn=ldap**, pak je základní konfigurace úspěšná.

4.7 Instalace Mantis

Bylo přepnuto do složky *localhostu* a přímo z repozitáře mantisu stažena poslední verze. Stažený soubor byl rozbalen a složka přejmenována na Mantis. Původní soubor byl smazán a byla nastavena práva pro čtení uživateli www-data na složku mantis:

```
# cd /var/www
# wget http://sourceforge.net/projects/mantisbt/files/mantis-
stable/1.2.11/mantisbt-1.2.11.tar.gz/download -O mantisbt-
1.2.11.tar.bz2
# tar xf mantisbt-1.2.11.tar.bz2
# mv mantisbt-1.2.11 mantis
# rm mantisbt-1.2.11.tar.bz2
# chown www-data:www-data -R /var/www/mantis
```

Následně byla v prohlížeči zadána adresa *http://localhost/mantis/*.

Byla nastavena tato konfiguraci pro databázi:

```
Type of Database: MySQL
Hostname: localhost
Username: mantis
Password: asdfg
Database name: mantis
Admin User name: root
Admin Password: asdfg
```

Nastavení bylo potvrzeno tlačítkem *Install* a aplikace mantis se nakonfiguroval. Následně stačilo přejmenovat nebo smazat složku admin přímo ve složce mantis:

```
# mv mantis _mantis
```

Aplikaci spustíme z webového prohlížeče zadáním *http://localhost/mantis/*.

Aplikace byla otestována přihlášením jako administrátor – login: **administrator**, heslo: **asdfg**.

Nastavení LDAP autorizace uživatelů Mantisu

Dle konfigurace na stránkách projektu[7] byly v konfiguračním souboru *config_default_inc.php* nastaveny následující parametry:

```
$g_ldap_server = 'ldap://localhost/';
$g_ldap_port = 389;
$g_ldap_root_dn = 'dc=ldap';
$g_ldap_organization = '';
$g_ldap_uid_field = 'uid';
$g_ldap_realname_field = 'cn';
$g_ldap_bind_dn = 'cn=admin,dc=ldap';
$g_ldap_bind_passwd = 'asdfg';
$g_use_ldap_email = ON;
$g_use_ldap_realname = ON;
$g_ldap_protocol_version = 3;
$g_ldap_follow_referrals = OFF;
$g_ldap_simulation_file_path = '';
```

Na závěr bylo třeba ještě ve stejném souboru změnit metodu autorizace na LDAP:

```
$g_login_method = LDAP;
```

Tímto byl zajištěn zabezpečený přístup a autentizace uživatelů přes LDAP. Přístup je povolen všem uživatelům nacházejícím se v kořeni LDAP. Jejich právo je defaultně *reporter*, administrátor jej může v nastavení aplikace změnit.

4.8 Instalace MediaWiki

Instalace balíčku mediawiki byla spuštěna zadáním následujícího příkazu [25]:

```
# aptitude install mediawiki
```

Následně bylo třeba odkomentovat následující řádek v souboru */etc/mediawiki/apache.conf*:

```
Alias /mediawiki /var/lib/mediawiki
```

[7] http://www.mantisbt.org/manual/manual.customizing.mantis.ldap.php

Nakonec byl restartován Apache:

```
# /etc/init.d/apache2 restart
```

Po restartu Apache byla zadána adresa *http://localhost/mediawiki/config/index*.php a byla následně nastavena tato konfigurace:

```
Site config:

Wiki name: Vývoj v PHP
Contact e-mail: admin@server01.cz
Language: cs-Česky
Copyright/license: No license metadata
Admin username: admin
Password: asdfg
Password confirm: asdfg
Object caching: No caching

Database config:

Database type: MySQL
Database host: localhost
Database name: wikimedia
DB username: wikiuser
DB password: heslo_uzivatele_wikiuser
DB password confirm: heslo_uzivatele_wikiuser

Superuser account: Use superuser account
Superuser name: root
Superuser password: asdfg

Storage Engine: InnoDB
Database character set: MySQL 4.1/5.0 UTF-8
```

Po úspěšné konfiguraci bylo, dle instrukce, nahráno nastaveni do souboru */etc/mediawiki/* a uživatele www-data nastaven jako vlastník konfiguračního souboru *LocalSetting.php*:

```
# mv /var/lib/mediawiki/config/LocalSettings.php
/etc/mediawiki/LocalSettings.php
# chgrp www-data /etc/mediawiki/LocalSettings.php
```

Wikimedii lze následně spustit na této adrese *http://localhost/mediawiki/*.

Nastavení LDAP autorizace uživatelů MediaWiki

Bylo přepnuto do složky */usr/share/mediawiki/extensions* a přímo z repozitáře mediawiki[8] byla stažena poslední verze extenze pro ověřování uživatele přes LDAP. Stažený soubor byl rozbalen a původní soubor nakonec smazán:

```
# cd /usr/share/mediawiki/extensions
# wget http://upload.wikimedia.org/ext-dist/LdapAuthentication-
MW1.19-108775.tar.gz
# tar xf LdapAuthentication-MW1.19-108775.tar.gz
# rm LdapAuthentication-MW1.19-108775.tar.gz
```

Dále byl editován soubor */mediawiki/LocalSettings.php* a na jeho konec vložen následující kód:

```
require_once("$IP/extensions/LdapAuthentication/LdapAuthentication
.php");
$wgAuth = new LdapAuthenticationPlugin();

$wgLDAPDomainNames = array('MyLDAP');
$wgLDAPServerNames = array('MyLDAP' => 'ldap');
$wgLDAPEncryptionType = array('MyLDAP' => 'clear');
$wgLDAPLowerCaseUsername = array( 'MyLDAP'=> true );
$wgLDAPUseLocal = false;
$wgLDAPSearchAttributes = array('MyLDAP' => 'uid');
$wgLDAPBaseDNs = array('MyLDAP' => 'dc=ldap');
$wgLDAPUserBaseDNs = array( 'MyLDAP' => 'ou=Vyvoj,dc=ldap');
$wgLDAPDebug = 3;
$wgDebugLogGroups['ldap'] = '/tmp/ldap.log';
```

Tímto byla zajištěna autentizace uživatelů přes LDAP. Přístup je však omezen pouze pro uživatele organizační jednotky „Vývoj“.

4.9 Instalace Subversion

Instalace byla spuštěna nainstalováním aktuální verze SVN a modulu SVN do Apache:

```
# aptitude update && aptitude install subversion libapache2-svn
subversion-tools
```

Dále určíme umístění SVN repositářů, např. */home/svn* a vytvoříme repozitář pro konkrétní projekt, např. *projek*t:

```
# mkdir /home/svn
# svnadmin create /home/svn/projekt
```

Byl nastaven přístup k repositáři pro Apache, práva uživateli www-data:

```
# chown www-data:www-data -R /home/svn/projekt
```

[8] http://www.mediawiki.org/wiki/Extension:LDAP_Authentication

Nastavení LDAP autorizace uživatelů SVN

Nastavení autorizace uživatelů SVN přes LDAP byla provedena konfigurací Apache. Byl editován soubor */etc/apache2/sites-enable/default* a přidány do něj následující řádky:

```
<Location /svn>
    DAV svn
    SVNParentPath /home/svn
    AuthBasicProvider ldap
    AuthName "LDAP autentizace"
    AuthLDAPURL
"ldap://ldap:389/ou=Programatori,ou=Vyvoj,dc=ldap?uid?sub?(objectC
lass=*)"
    AuthLDAPBindDN cn=admin,dc=ldap
    AuthLDAPBindPassword asdfg
    AuthType Basic
    Require valid-user
</Location>
```

Tímto byl zajištěn zabezpečený přístup a autentizace uživatelů přes LDAP. Přístup je však omezen pouze pro uživatele organizační jednotky „Programátoři“.

Nakonec byl proveden restart Apache:

```
# /etc/init.d/apache2 restart
```

Správnou autorizaci uživatelů ověříme přes url: *http://localhost/svn/projekt.*

4.10 Instalace OwnCloud

Instalace byla zahájena nainstalováním potřebných balíčků:

```
# aptitude install libapache2-mod-php5 php5-json php5-gd php5-
sqlite php5-curl curl libcurl3 bzip2
```

Dále bylo přepnuto do složky */var/www* a stažena z repozitáře owncloud poslední verze. Soubor byl následně rozbalen a původní smazán. Nakonec byla nastavena práva pro čtení uživateli www-data na složku owncloud:

```
# cd /var/www
# wget http://download.owncloud.org/releases/owncloud-
4.0.5.tar.bz2
# tar xf owncloud-4.0.5.tar.bz2
# rm owncloud-4.0.5.tar.bz2
# chown www-data:www-data -R /var/www/owncloud
```

Do prohlížeče byla zadána následující adresa *http://localhost/owncloud/* a spuštěna konfigurace aplikace. Byly nastaveny následující hodnoty:

```
Username: root
Password: asdfg
Data folder: /var/www/data
Configure database: MySQL
Database root: root
Database  password: asdfg
Database name: owncloud
Server: localhost
```

Nastavení LDAP autorizace uživatelů ownCloud

V prohlížeči byla zadána adresa *http://localhost/owncloud/* a přihlášeno jako administrátor – uživatel root. V levém menu vybráno *Nastavení->Aplikace*. V seznamu aplikací dále zvoleno *„LDAP user and group backend“* a vybráno **enable**. Tímto byl povolen modul pro autentizaci uživatelů přes LDAP server.

Dále bylo třeba v menu *Admin* nastavit na záložce *LDAP* basic následující konfiguraci:

```
Host: ldap
Base: dc=ldap
Name: cd=admin,dc=ldap
Password: asdfg
User Login Filter: uid=%uid
```

A následně na záložce *Advanced*:

```
Port: 389
Base User Tree: ou=Vyvoj,dc=ldap
```

Tímto byla zajištěna autentizace uživatelů přes LDAP serveru. Přístup je však omezen pouze pro uživatele organizační jednotky „Vývoj“.

5 Konfigurace test serveru

V ukázce platformy se však bude jednat pouze o další webový server, který naslouchá na portu 8080.

Nejdříve byla vytvořena speciální složka pro nový webový server, *testwww*, která bude sloužit jako webový prostor testovacího serveru a byla nastavena práva na čtení uživateli www-data:

```
# mkdir /var/testwww
# chown www-data:www-data -R /var/testwww
```

Dále byl editován *soubor /etc/apache2/ports.conf* a doplněn následující kód:

```
NameVirtualHost *:8080
Listen 8080
```

A do dalšího souboru */etc/apache2/sites-enable/default* na konec vložen následující kód:

```
<VirtualHost *:8080>
     DocumentRoot /var/testwww
</VirtualHost>
```

Na závěr byl restartován Apache:

```
# /etc/init.d/apache2 restart
```

5.1 Konfigurace cronu

Konfigurace cronu byla provedena za účelem automatického updatu projektů z SVN na testovací server.

Z konfigurace SVN je založen první repozitář, tj. **projekt**. Pro tento repozitář je následně popsán automatický update na testovacím serveru s využitím cronu.

Následně byl proveden checkout repozitáře **projekt** do složky testovacího serveru. Toto bylo provedeno následujícím příkazem:

```
# svn checkout file:///home/svn/projekt /var/testwww/projekt
```

Následně bylo možně na adrese *http://localhost:8080/projekt* otestovat, že projekt běží na testovacím serveru.

Nyní bylo možně nakonfigurovat samotný cron. To bylo provedeno přes jeho editor, který byl spuštěn následujícím příkazem:

```
# crontab -e
```

Spustil se editor crontablu a do něj byl vepsán následující řádek:

```
0**** svn update /var/testwww/projekt
```

Tímto bylo nastaveno, že se update projektu spustí každou 0. minutu – tedy každou hodinu.

6 Instalace klientských softwarů

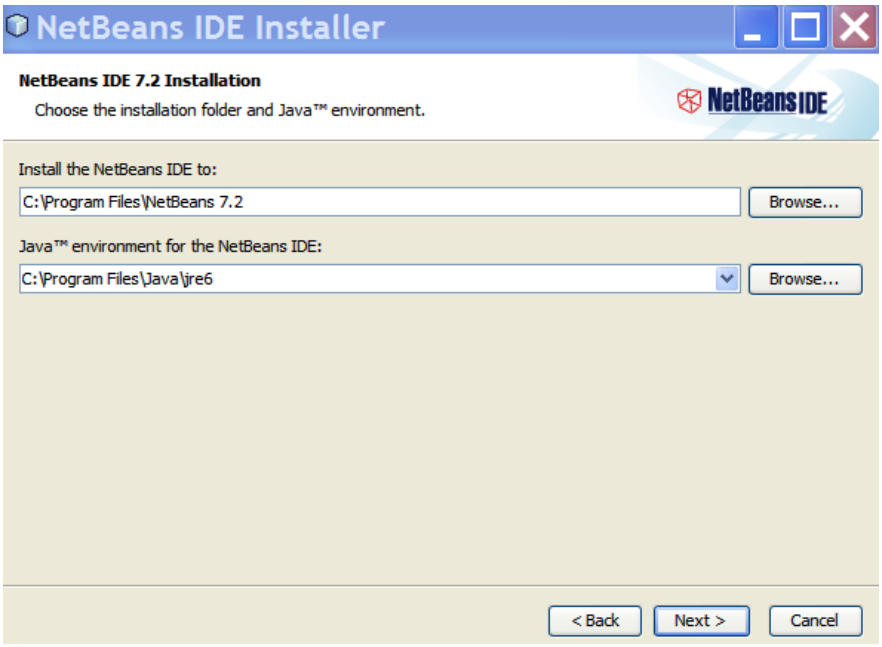
Tato kapitola popisuje instalaci a konfiguraci softwarů na pracovní stanici.

6.1 NetBeans IDE

Následný popis instalace produktu NetBeans IDE je popsán pro v prostředí OS MS Windows 7.

Ze stránky projektu[9] byla vybrána a stažena aktuální verze instalace (*netbeans-7.2-ml-php-windows*), která podporuje technologii PHP.

Během instalace bylo vybráno umístění aplikace a prostředí Java, viz Obrázek 24.



Obrázek 24 – Instalace NetBeans IDE

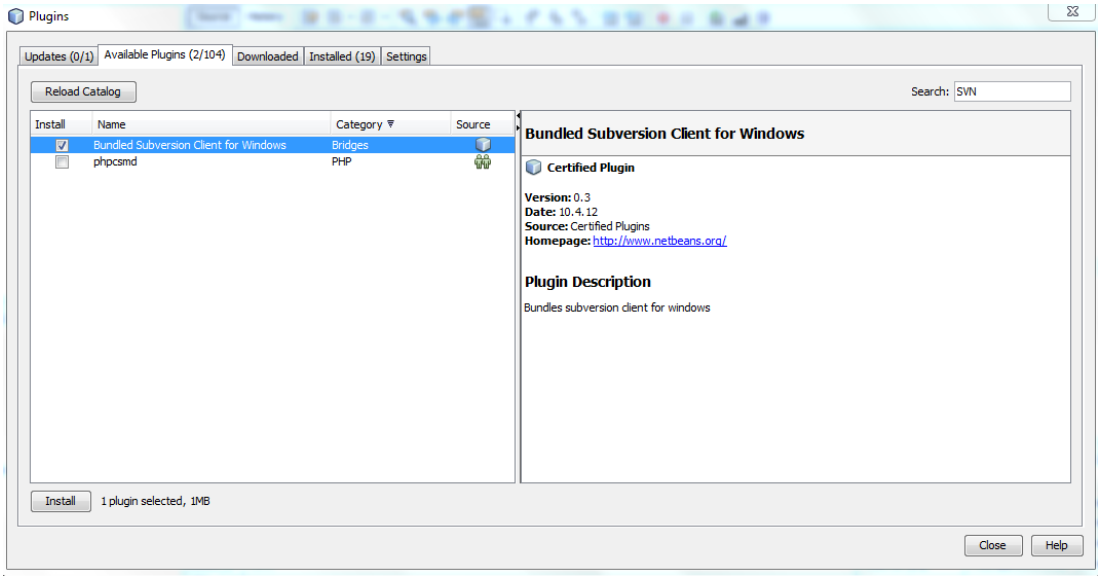
Po potvrzení se dokončila instalace aplikace.

Instalace pluginů

Po instalaci NetBeans IDE je třeba doinstalovat další, potřebné pluginy pro vývoj. Byla nainstalována integrace klienta verzovacího systému Subversion. Na následujících obrázcích je popsán postup instalace pluginu pro využití SVN.

V menu *Tools->Plugins* byla vybrána druhá záložka – *Available Plugins*. Do pole Search zbyl zadán hledaný plugin, v našem případe SVN, viz Obrázek 25.

[9] http://netbeans.org/downloads/



Obrázek 25 – Vyhledání požadovaného pluginu

Byl nalezen odpovídající plugin. Ten byl následně zaškrtnut a instalován. Po ukončení instalace byl proveden restart aplikace, po kterém začal plugin standardně fungovat.

Podobně lze doinstalovat např. plugin pro podporu frameworku.

6.2 Další klientské aplikace

Dále byl nainstalován na klientskou stanici webový server – balíček Wamp, jehož instalace a konfigurace je dobře popsána na stránkách produktu[10].

Posledním větším produktem byl TortoiseSVN, jehož instalace a konfigurace je rovně velmi dobře popsána na stránkách produktu[11]. Samotná práce s ním je dále popsána v metodických pokynech pro programátory.

[10] http://www.wampserver.com
[11] http://tortoisesvn.net

7 Metodické pokyny

Metodické pokyny obsahují základní pokyny prováděné na serverové a klientské části.

7.1 Administrátoři

V metodických pokynech pro administrátory je vycházeno z nainstalované a nakonfigurované serverové části, dle kapitoly Instalace a konfigurace jednotlivých produktů serveru. Pokud je třeba upravit konfiguraci platformy, je dobré vycházet z této kapitoly. V instalaci a konfiguraci je rovněž popsáno vytváření nových repozitářů a jejich automatické kopírování na testovací server, což se rovněž v metodických pokynech již nevyskytuje.

7.1.1 Správa LDAP uživatelů

Správa uživatelů je prováděna přes webové rozhraní aplikace phpLDAPadmin, která je spustitelná na adrese *http://localhost/phpldapadmin/*.

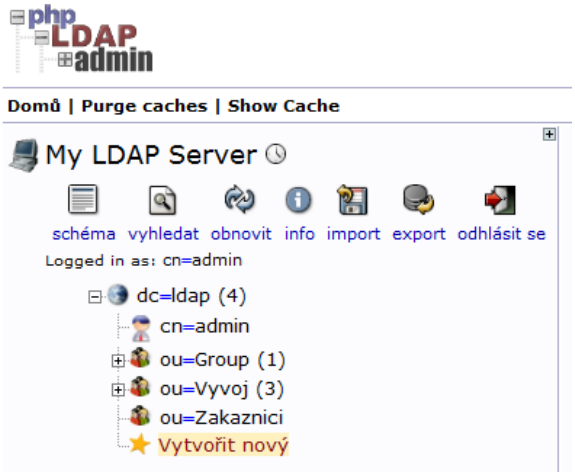
Jako administrátor se přihlásíme s DN: **cn=admin, dc=ldap** a heslem: **asdfg**, viz Obrázek 26.

Obrázek 26 – Přihlášení do aplikace phpLDAPadmin

V levé části se nám zobrazí strom objektů na LDAP serveru. Ten zde tvoří organizační jednotky *„Vývoj“* s podložkami *„Analytici“*, *„Programátoři“* a *„Testeři“*. Každá složka obsahuje jiná práva na přístup do konkrétních aplikací platformy. Na úrovni organizační jednotky *„Vývoj“* jsou *„Zákazníci“*, kteří mají přístup pouze do aplikace Mantis, pro hlášení případných připomínek při testování nových úprav. Práva pro jednotlivé aplikace jsou popsána u každé aplikace v kapitole Instalace a konfigurace jednotlivých produktů serveru. Struktura LDAP stromu je znázorňuje Obrázek 27.

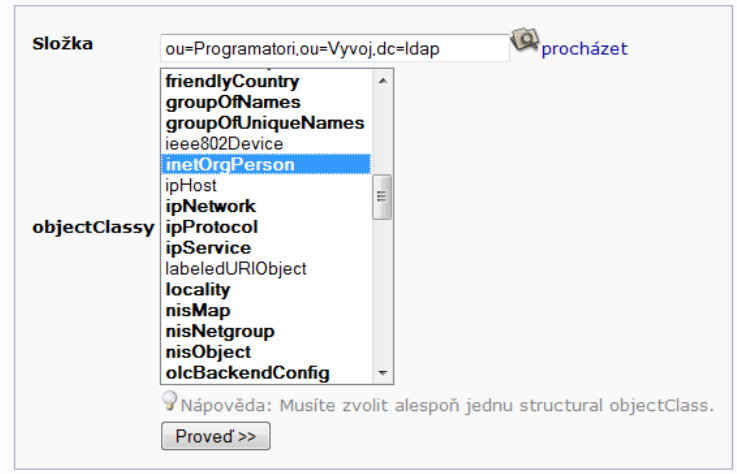
Založení nového uživatele

Nového uživatele založíme přes menu *Vytvořit nový*, viz Obrázek 27.



Obrázek 27 – Založení nového uživatele

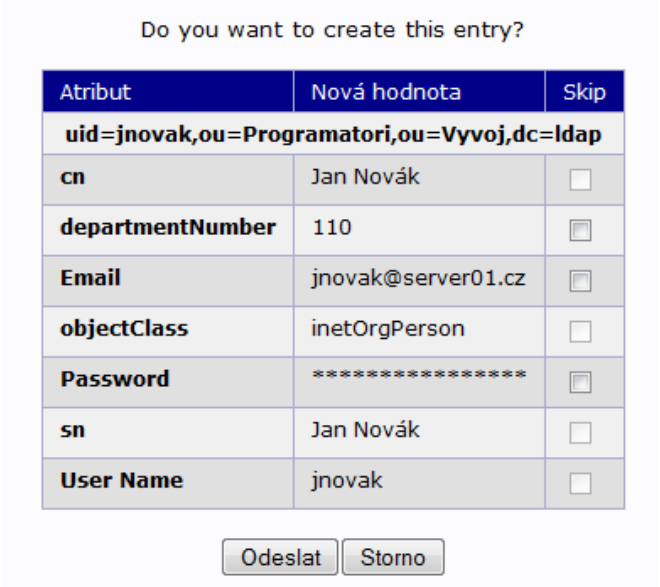
Následně vybereme výchozí šablonu pro založení uživatele. V následující nabídce zvolíme příslušnou *skupinu*, do které chceme nového uživatele zařadit a vybereme třídu objektu *inetOrgPerson*, viz Obrázek 28.



Obrázek 28 – Třída a složka uživatele

V následujícím kroku vyplníme všechny povinné i další potřebné atributy definované třídou inetOrgPerson, která je pro základní použití nejvhodnější.

Důležitý je atribut RDN, který definuje jednoznačné rozlišení uživatelů, unikátní název. Zvolíme UID, což představuje uživatelské jméno (login) do systémů. Nakonec potvrdíme a zobrazí se nám výpis všech nastavených atributů včetně přidělené složky (organizační jednotky), viz Obrázek 29.

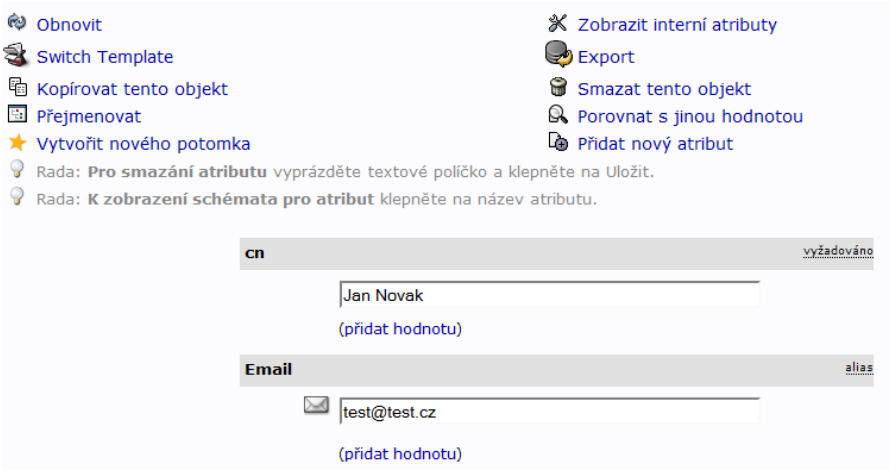


Obrázek 29 – Vyplněné atributy nového uživatele

Následně dáme *odeslat* a dojde k založení nového uživatele, kterého můžeme zobrazit v levém stromu.

Editace a odstraněni uživatele, přiřazení do jiné složky

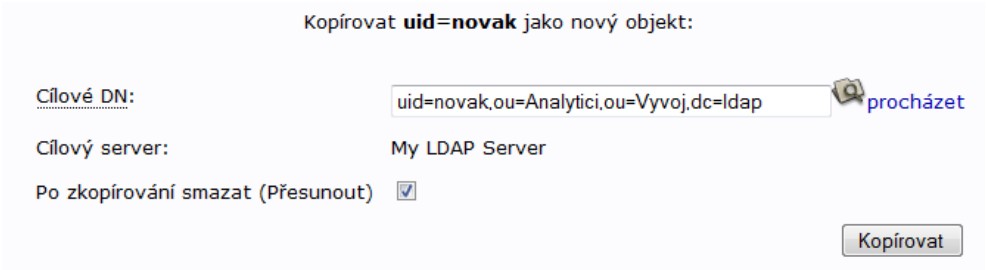
V LDAP stromu zvolíme kliknutím konkrétního uživatele a výchozí šablonu pro jeho úpravu. Na následně zobrazené stránce můžeme upravit atributy uživatele, včetně změny hesla, viz Obrázek 30. Změny uložíme tlačítkem *Update Object*.



Obrázek 30 – Editace uživatele

Uživatele odstraníme přes položku v menu *„Smazat tento objekt“*.

Přesunutí do jiné složky provedeme přes položku „Kopírovat tento objekt“, kde následně zvolíme novou složku (organizační jednotku) a potvrdíme, viz Obrázek 31.



Obrázek 31 – Přesun uživatele do jiné složky

7.1.2 Správa jednotlivých produktů

Tato kapitola obsahuje správu uživatelů a skupin v jednotlivých aplikacích platformy.

Mantis

Do aplikace Mantis se jako administrátor přihlásíme přes adresu *http://localhost/mantis/*, uživatelské jméno: **administrator**, heslo: **asdfg**.

Aplikace má poměrně mnoho nastavení a možností, budou tedy zde popsány pouze základní funkce.

Správa uživatelů

Mantis vyhledává uživatele na LDAP serveru a následně si dělá jejich kopii do své databáze, kde si k nim přiřazuje různá práva. Správu uživatelů najdeme pod menu *Správa->Správa uživatelů*, viz Obrázek 32.

Obrázek 32 – Správa uživatelů v aplikaci Mantis

Na detailu uživatele potom můžeme přiřadit různé role, jsou to: *správce, vedoucí, vývojář, aktualizátor, reportér a recenzent.* Role může přiřazovat pouze uživatel *správce*. Uživatelé mají defaultní roli *reportér*.

Správa projektů

Spravovat projekty mohou uživatelé v roli správce a vedoucí. Tuto nabídku najdeme pod menu *Správa->Správa projektů*, viz Obrázek 33.

Obrázek 33 – Správa projektů v aplikaci Mantis

Uživatel s právy na tuto položku zde může zakládat nové složky s jednotlivými projekty a přiřadit k nim uživatele.

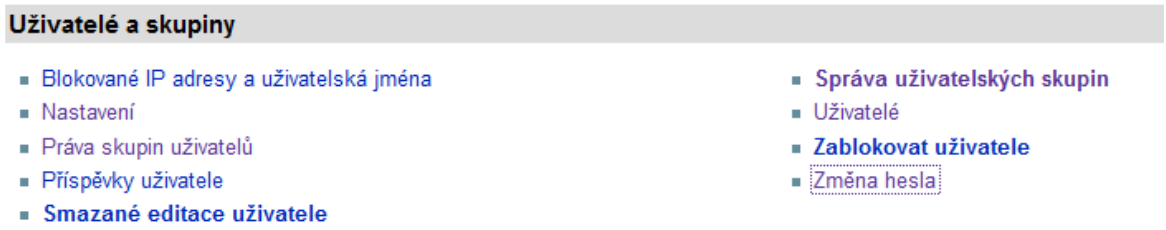
MediaWiki

Do aplikace Mantis se jako administrátor přihlásíme přes adresu *http://localhost/mediawiki/*, uživatelské jméno: **administrator**, heslo: **asdfg**.

Aplikace má rovněž mnoho nastavení a možností, budou tedy zde popsány pouze základní funkce.

Správa uživatelů

MediWiki uživatelé vyhledává na LDAP serveru a následně si dělá jejich kopii do své databáze, kde si k nim přiřazuje různá práva. Správu uživatelů najdeme pod levým menu *Speciální stránky->Uživatelé a skupiny*, viz Obrázek 34.



Obrázek 34 – Správa uživatelů a skupin v aplikaci MediaWiki

Administrátor zde může spravovat uživatelské skupiny, uživatele a přiřazovat je do skupin.

Nastavení prostředí aplikace

Základní nastavení aplikace je přístupné přes pravé horní menu, položku *Nastavení*, viz následující Obrázek 35.

Obrázek 35 – Nastavení prostředí aplikace MediaWiki

Je zde možnost upravit údaje o přihlášeném uživateli, vzhled aplikace, nastavení povolených parametrů importovaných souborů, nastavení formátu používaného data, vizuální zobrazení oken v aplikaci, editor stránek, upozornění posledních sledovaných stránek, možnosti vyhledávání v daných jmenných prostorech, formátování stránek a další.

Podrobnější informace ke správě aplikace je umístěno přímo v nápovědě MediaWiki.

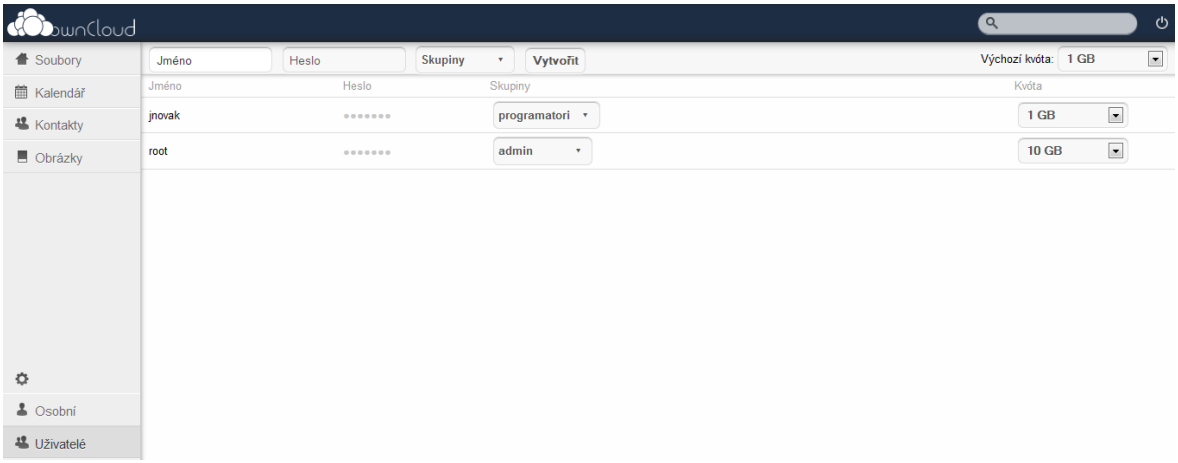
ownCloud

Do aplikace ownCloud se jako administrátor přihlásíme přes adresu *http://localhost/owncloud/*, uživatelské jméno: **root**, heslo: **asdfg**.

Stejně jako předchozí aplikace, má i ownCloud mnoho nastavení a možností, budou tedy popsány pouze ty základní.

Správa uživatelů

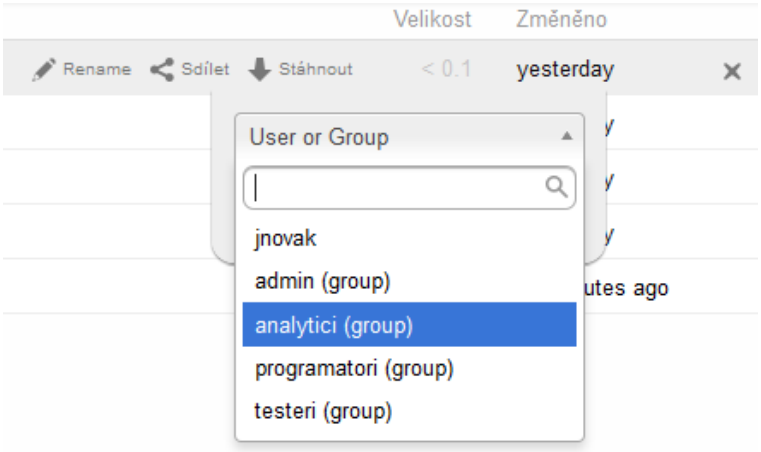
OwnCloud vyhledává uživatele na LDAP serveru a následně si dělá jejich kopii do své databáze, kde je může administrátor přiřadit do různých skupin, které sám nadefinuje. Správu uživatelů najdeme pod menu *Nastavení->Uživatelé*, viz Obrázek 36.



Obrázek 36 – Správa uživatelů v aplikaci ownCloud

Administrátor aplikace má možnost každému uživateli nastavit povolenou kvótu vložených dat a přidělit uživatele do různých skupin.

Současně může zakládat nové skupiny, skupina se založí přes menu *Skupiny->Add group*. Jednotlivé skupiny mají potom význam pro sdílení složek a dokumentů, kdy se přes sdílení přiřadí dokument konkrétní skupině nebo uživateli. Toto najdeme v menu *Soubory*, u každého souboru/složky potom pod menu *Sdílet*, viz Obrázek 37.



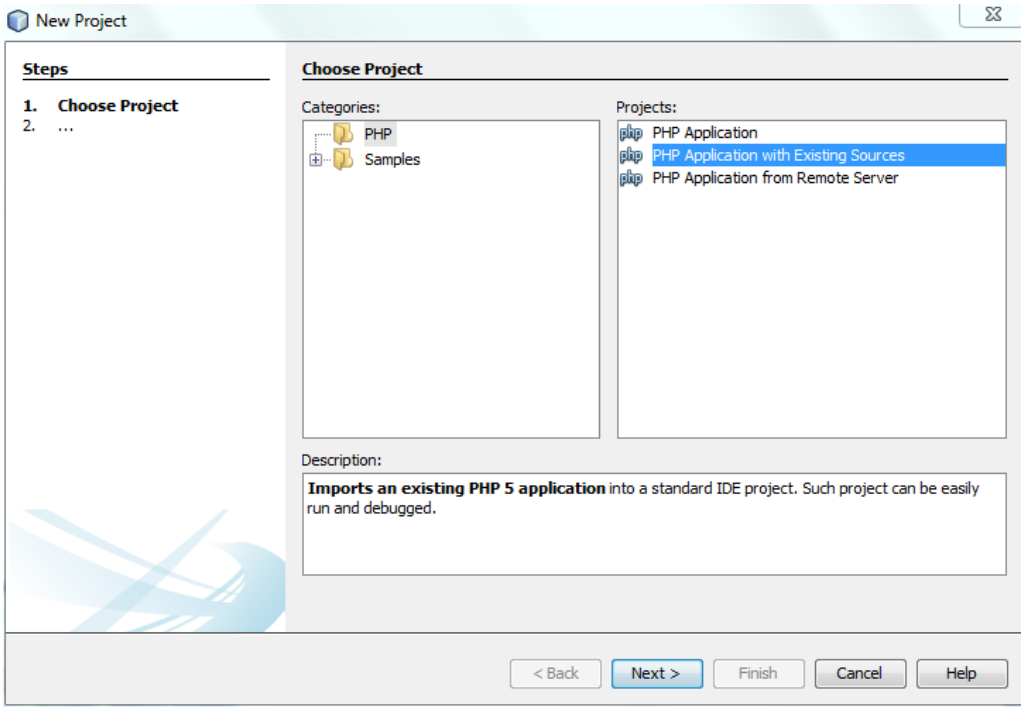
Obrázek 37 – Sdílení souborů/složky

7.2 Programátoři

Kapitola obsahuje základní metodické pokyny pro programátora – založení projektu v NetBeans IDE, popis základních pravidel pro zápis PHP kódu, práce s repozitářem a dalšími produkty platformy.

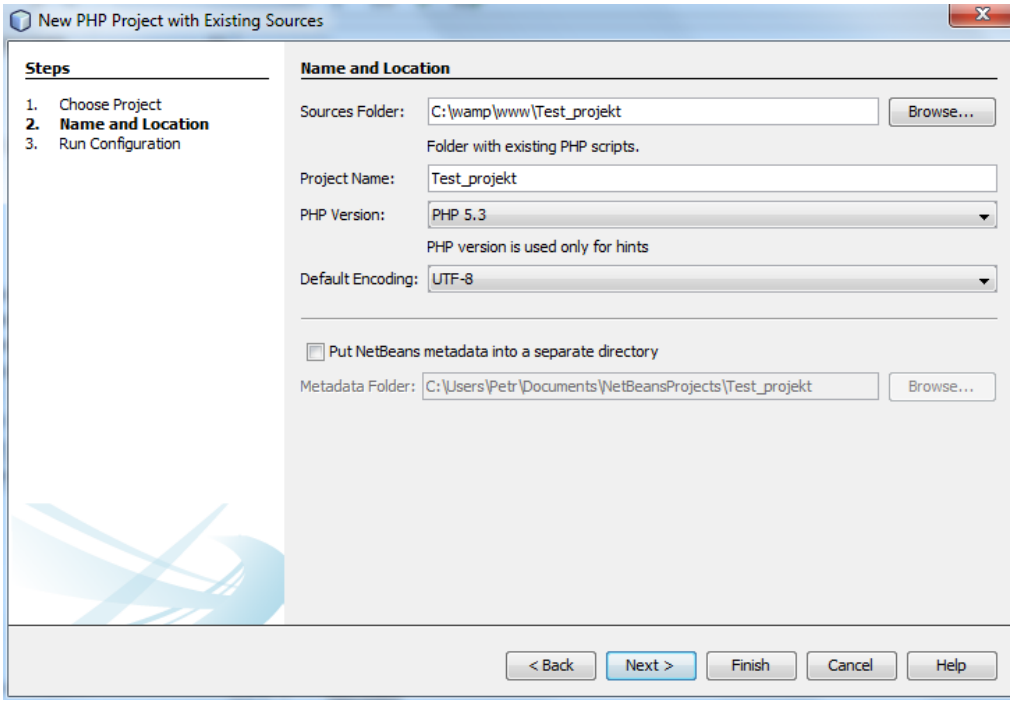
7.2.1 Založení projektu v NetBeans IDE

Nový projekt založíme z menu *File->New Project*. Protože budeme jistě vycházet z nějakého základu, např. přeinstalovaného frameworku, vybereme v menu *Project* možnost PHP aplikace s existujícím kódem. V menu *Categories* vybereme PHP. Potvrdíme tlačítkem *Next*. Vše znázorňuje následující Obrázek 38.



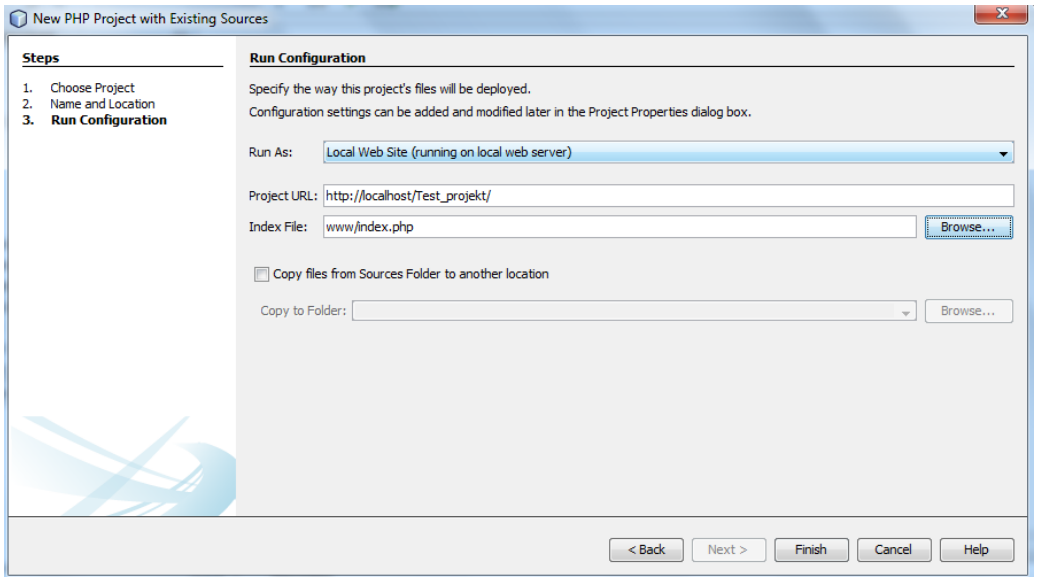
Obrázek 38 – Založení projektu v NetBeans IDE

V následujícím kroku zvolíme cestu ke složce zdrojového kódu (ten si předem nahrajeme do složky *localhostu*), zvolíme název projektu a vybereme nejvyšší verzi PHP, viz Obrázek 39. Potvrdíme tlačítkem *Next*.



Obrázek 39 – Název a složka projektu

V posledním kroku zadáme umístění souboru *index* a můžeme změnit adresu projektu, viz Obrázek 40. Vše potvrdíme tlačítkem *Finish*.



Obrázek 40 – Konfigurace projektu

7.2.2 Základní pravidla pro zápis PHP kódu

Kapitola obsahuje základní pravidla pro zápis kódu v jazyce PHP. Tyto pravidla se také často označují jako tzv. *štábní kultura* nebo *programátorská kultura*. Pravidla jsou převzata z doporučených pravidel frameworku Zend[12].

Pravidla zápisu kódu

Soubory, které obsahují pouze PHP kód, by nikdy neměly končit řetězcem *"?>"*. Pravidlo pomůže vyhnutí chybám typu *"Cannot modify header information - headers already sent by"*.

Pro odsazování používat 4 mezery (nepoužívat tabulátor).

Skripty by neměli být širší víc než 80 znaků (maximální povolená hranice je 120 znaků).

Název třídy vždy začíná velkým písmenem a obsahuje znaky abecedy a podtržítko. Pokud třídu pojmenujeme *Php_Paging*, pak by měla být umístěna v adresáři *Php/Paging.php*. Podtržítko představuje nový adresář. Číslice jsou povoleny, ale nedoporučují se. První písmeno slova je vždy velké, ostatní jsou malá. Například *Php_XML* je chybný název, *Php_Xml* je správně.

Rozhraní vždy končí slovem *Interface*, například *Php_Validator_Interface*.

Metody i atributy dodržují takzvanou velbloudí notaci, kdy první písmeno je malé a další slova jsou oddělena velkým písmenem, například getAllResult().

Používat standardní předpony pro metody se stejným účelem. Předponu *get* mají metody, které vracejí nějakou hodnotu, *set* metody, které nějakou hodnotu nastavují, *is* je předpona pro metody, které vrací boolean hodnotu a podobně.

Snažit se používat dostatečně dlouhé názvy atributů a metod. Pouhé result, get nebo resource nic neříkají.

U privátních a chráněných atributů a metod používat na začátku názvu podtržítko (např. *_setup*).

Pomocné proměnné mohou být pojmenovány podle vzoru $n, ne však v cyklech, které jsou delší než 20 řádků.

Konstanty mají v názvu všechna písmena velká a slova jsou oddělena podtržítkem (např. *INTERVAL_CONSTANTA*).

Používat vždy plný zápis PHP značek, tedy „*<?php“*, nikoli pouze „*<?“*.

[12] http://framework.zend.com/manual/en/coding-standard.html

Pro uzavření řetězce používat *jednoduché uvozovky*. Takzvané palcové, nebo též dvojité uvozovky používat jen tehdy, pokud jsou uvnitř nich uzavřeny řetězce uvozené jednoduchými uvozovkami.

Řetězce se mohou spojovat pomocí tečky (.), musí však být od tečky odděleny jednou mezerou (*'text'* . *'text'*).

Spojování řetězců lze rozdělit na více řádků (tečka vždy musí být pod rovnítkem):

```
1. $sql = "SELECT `id`, `name` from `people` "
2.       . "WHERE `name`='Fred' OR `name`='Susan'";
```

Pole nezačínat zápornými indexy.

Při deklaraci pole a volání funkce vždy oddělit hodnoty mezerou:

```
1. $sampleArray = array(1, 2, 3, 'Zend', 'Studio');
```

Při deklaraci asociativního pole psát každou hodnotu na nový řádek:

```
1. $sampleArray = array(
2.     'firstKey'  => 'firstValue',
3.     'secondKey' => 'secondValue',
4. );
```

Složená závorka obsahující tělo třídy či metody je vždy na novém řádku.

```
1. /**
2. * Documentation Block Here
3. */
4. class SampleClass
5. {
6.     // all contents of class
7.     // must be indented four spaces
8. }
```

V každém souboru může být pouze jedna třída.

Atributy třídy nebo instance jsou uvedeny na začátku třídy.

Hodnota příkazu return nesmí být uzavřena v závorkách.

Podmíněné konstrukce oddělte z obou stran mezerou, metody naopak mezeru neobsahují. Operátory oddělovat mezerou.

```
1. if ($a != 2) {
2.     $a = 2;
3. }
```

V podmíněné konstrukci switch používat odsunutí:

```
1. switch ($numPeople) {
2.     case 1:
3.         break;
4.
5.     case 2:
6.         break;
7.
8.     default:
9.         break;
10. }
```

V konstrukci switch nikdy nezapomenout na část default.

Psát co nejvíce komentářů.

Pro komentáře používat syntaxi phpDoc[13].

[13] http://phpdoc.org/

Každý soubor by měl začínat zápisem *phpDoc*:

```
/**
 * Short description for file
 *
 * Long description for file (if any)...
 *
 * LICENSE: Some license information
 *
 * @category   Zend
 * @package    Zend_Magic
 * @subpackage Wand
 * @copyright  Copyright (c) 2005-2011 Zend Technologies USA Inc.
(http://www.zend.com)
 * @license    http://framework.zend.com/license   BSD License
 * @version    $Id:$
 * @link       http://framework.zend.com/package/PackageName
 * @since      File available since Release 1.5.0
 */
```

Každá třída by měla obsahovat na začátku komentář:

```
/**
 * Short description for class
 *
 * Long description for class (if any)...
 *
 * @category   Zend
 * @package    Zend_Magic
 * @subpackage Wand
 * @copyright  Copyright (c) 2005-2011 Zend Technologies USA Inc.
(http://www.zend.com)
 * @license    http://framework.zend.com/license   BSD License
 * @version    Release: @package_version@
 * @link       http://framework.zend.com/package/PackageName
 * @since      Class available since Release 1.5.0
 * @deprecated Class deprecated in Release 2.0.0
 */
```

Rovněž každá metoda musí obsahovat svůj popis (nezapomenout na *@throws* pro vyhozené výjimky).

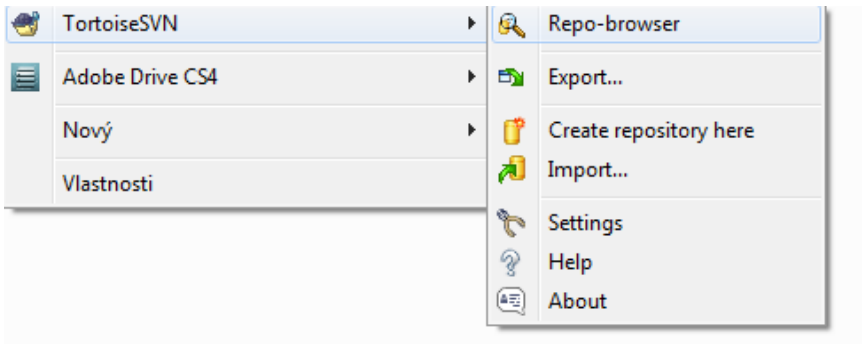
7.2.3 Práce s repozitářem

Kapitole popisuje základní operace s repozitářem Subversion. Toto je demonstrováno na klientu Tortoise SVN.

Vytvoření repozitáře, základní struktura složek

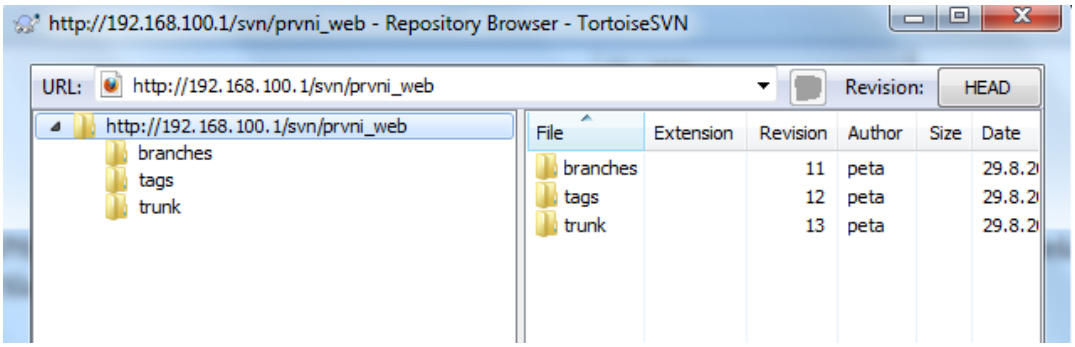
Samotný repozitář na serveru zajistí administrátor, který má k tomuto vlastní metodické pokyny.

Před prvním importem dat do repozitáře je na něm dobré vytvořit základní strukturu složek. To uděláme za pomocí *Repo-browseru*, který vyvoláme přes prvé tlačítko myší kdekoliv v průzkumníku, viz Obrázek 41.



Obrázek 41 – Vyvolání Repo-browseru

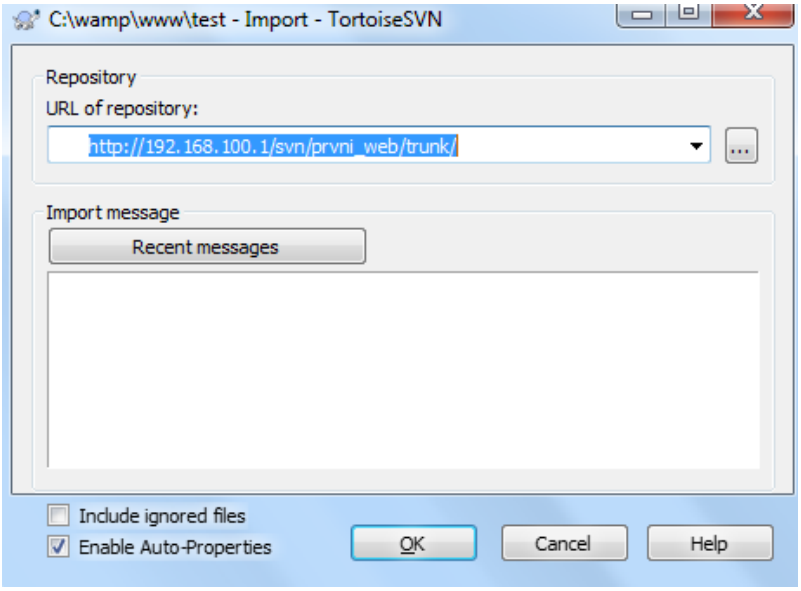
Připojíme se do nového repozitáře a vytvoříme strukturu složek. Složky zakládáme přes pravé tlačítko myši a výběru *Create folder*. Strukturu popisuje Obrázek 42.



Obrázek 42 – Základní struktura složek repozitáře

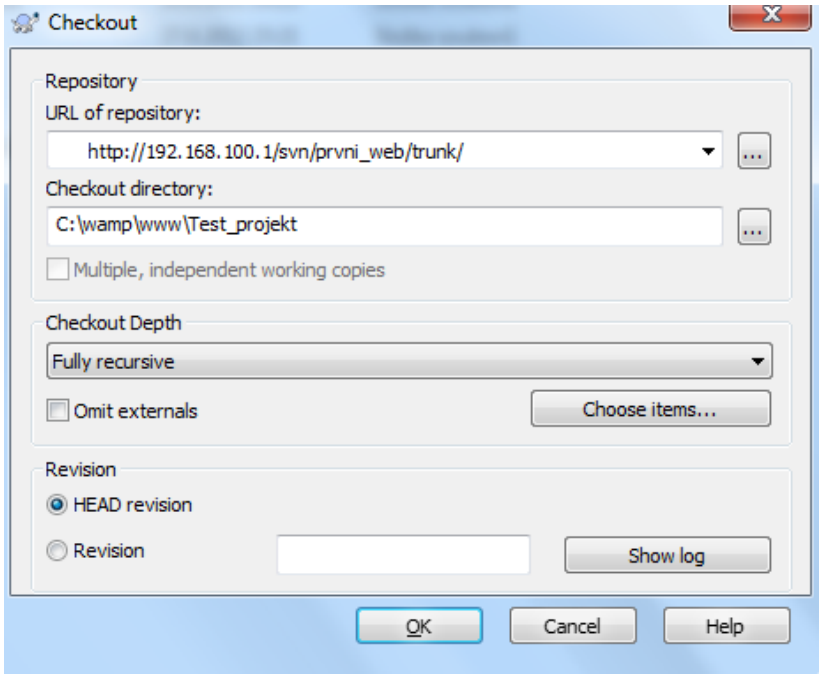
Základní struktura slouží k tomu, abychom vedli hlavní kmen projektu – **trunk** a mohli také vést větve – **branches** a dělat nálepky – **tags** pro konkrétní vydané verze.

Protože jsme již založili testovací projekt při založení projektu v NetBeans IDE, můžeme tento projekt naimportovat na SVN. To provedeme přes pravé tlačítko myši na složce projektu a v menu TortoiseSVN vybereme *Import*. Zadáme cestu do *trunku* nového repozitáře první web a potvrdíme OK, viz následující Obrázek 43.



Obrázek 43 – Import projektu do repozitáře

Následně je třeba provést *checkout* a přepsat složku nového projektu na lokálním disku, dojde k „propojení“ s SVN a doplnění o speciální, skryté soubory. *Checkout* opět vyvoláme přes pravé tlačítko myši. Po nastavení potvrdíme OK a dojde k updatu dat ve složce projektu, viz Obrázek 44.



Obrázek 44 – Checkout projektu z repozitáře

Po úspěšném *checkoutu* můžeme začít repozitář běžně používat k práci.

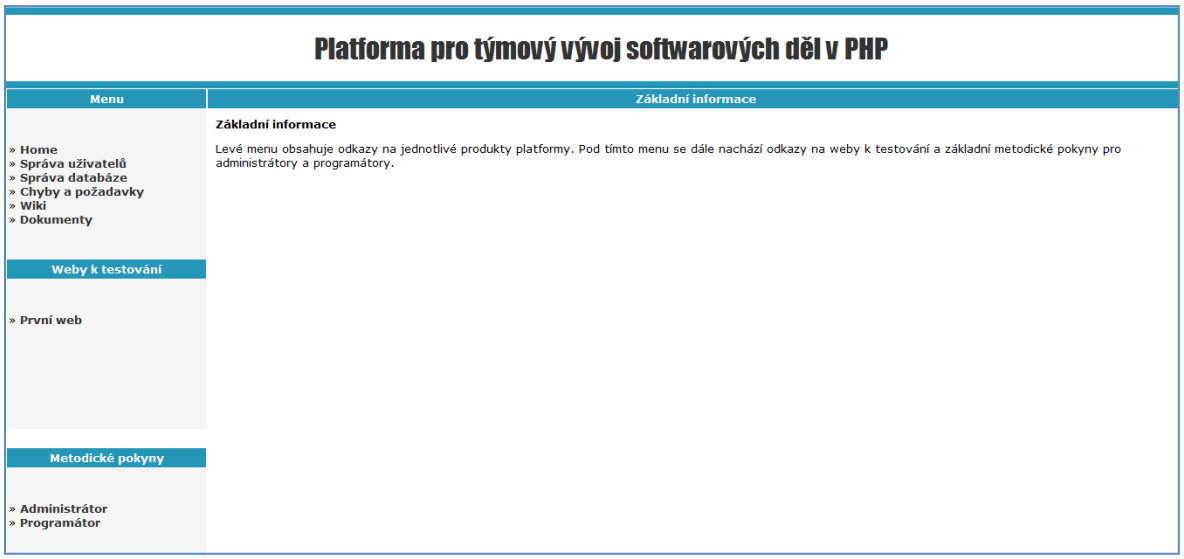
Základní metodika pro využívání šťítků (tags) a větví (branchces)

Tagy slouží pro zachování určité verze repozitáře v čase, říká se jim také *tzv. snapshot*. Tag vytvoříme pomocí kopie verze z trunku, kterou chcete označit štítkem. Štítky se tvoří např. při nasazení, předání zákazníkovi k otestování atd.

Branches představují v rámci repozitáře to samé co tags, jen s nimi jinak pracujeme. Tagy slouží k zaznamenání určité konkrétní verze. Oproti tomu větev branches slouží k tomu, aby byla dále upravována apod. Na větvi můžeme pracovat paralelně jako na hlavní větvi trunk a až budeme považovat za nutné, opět ji můžeme spojit s hlavní větví trunk.

7.2.4 Úvodní rozcestník platformy

Úvodní stránku zobrazíme z prohlížeče pod adresou *http://192.168.100.1/*, na serveru potom pod *localhostem*. Náhled úvodního rozcestníku znázorňuje Obrázek 45.



Obrázek 45 – Náhled rozcestníku platformy

V levém menu má každý uživatel k dispozici rozcestník k jednotlivým aplikacím platformy. Přístup má ovšem dle svého zařazení a tím i práv. Pod menu se dále nachází seznam webů připravených k testování. Součástí úvodního rozcestníku jsou i samostatné dokumentace s metodickými pokyny pro Administrátory a Programátory.

Závěr

V práci byla obecně představena problematika samotného vývoje a následně zaměřeno na týmový vývoj softwarových děl v PHP. Zde byla analýza zaměřena na jednotlivé požadavky při vývoji, jenž tento vývoj obnáší. Následně byly požadavky přeneseny na softwarové produkty, které je dokáží řešit. Ty byly omezeny dle vydefinovaných kritérií, přičemž nejdůležitějším bylo, že se musí jednat o open source produkty.

Vybrané produkty byly samostatně rozebrány, pospána jejich funkčnost a možnosti vzájemné integrace. Na závěr každého produktu bylo popsáno, proč byl produkt zvolen a vytyčeny jeho výhody, případně i nevýhody.

Produkty byly implementovány a popsána jejich konfigurace. Dohromady vytvořili jednoduchou platformu, která splňuje základní požadavky pro týmový vývoj softwarových děl v PHP.

Platforma obsahuje obecné metodické pokyny pro administrátory a programátory. Pokyny pro administrátory byly zaměřeny na centrální správu identit uživatelů ve všech požadovaných produktech platformy a základní správu těchto aplikací. Pokyny pro programátory byly zaměřeny na základní použití integrovaného vývojového prostředí, práci se systémem správy verzí a pravidel zápisu PHP kódu dle předepsané štábní kultury jazyka.

Platforma byla doplněna o úvodní rozcestník v podobě HTML stránky, který nabízí odkazy na jednotlivé produkty.

Práci určitě nelze hodnotit jako převratné řešení pro kladené nároky týmového vývoje. Poskytuje však všechny základní potřeby jednotlivých rolí v týmu. Největším přínosem práce hodnotím popis konfigurace jednotlivých produktů platformy, protože je někdy obtížné tuto konfiguraci někde dohledat. Zde je pohromadě popsána a může být nápomocí pro řešení jiných úkolu. Jedná se převážně o propojení aplikací s centrálním úložištěm identit.

Možnost následného vývoje této práce vidím v další aplikaci, která by umožňovala centrálně spravovat další sdílená data ve všech produktech. Může se i například jednat o definici přístupu uživatelů ke konkrétnímu repozitáři v systému správy verzí, to má opodstatnění při využití externího programátora, kterému chceme dát přístup pouze k softwaru, na kterém pracuje. K těmto účelům může rovněž posloužit implementovaná centrální správa identit.

Literatura a zdroje

[1] VONDRÁK, Ivo. *Úvod do softwarového inženýrství* [online]. Ostrava, 2002 [cit. 2012-02-10]. Dostupné z www: http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf. VŠB – Technická univerzita Ostrava.

[2] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací*: *objektově orientovaná analýza a návrh prakticky*. Brno: Computer Press, 2007, 53,54,59,63-68. ISBN 978-80-251-1503-9.

[3] HRABÍ, STANISLAV. *APLIKOVANÁ METODIKA VÝVOJE SOFTWARU V MALÝCH A STŘEDNÍCH (SME) FIRMÁCH* [online]. Brno, 2011 [cit. 2012-02-10]. Dostupné z: http://is.muni.cz/th/98664/fi_m/. Diplomová práce. Masarykova Univerzita.

[4] Metodologie vývoje softwaru. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-02-10]. Dostupné z: http://cs.wikipedia.org/wiki/Metodologie_vyvoje_softwaru

[5] VŠETIČKA, Martin. *Mantis* [online]. [cit. 2012-03-08]. Dostupné z: http://www.martinvseticka.eu/index.php?sekce=browse&page=147

[6] ALDORF, F. *ZÁKLADNÍ CHARAKTERISTIKY RUP* [online]. 2008 [cit. 2012-03-08]. Dostupné z: http://objekty.vse.cz/Objekty/Rup2

[7] LDAP. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-03-08]. Dostupné z: http://cs.wikipedia.org/wiki/LDAP

[8] Active Directory. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-03-20]. Dostupné z: http://cs.wikipedia.org/wiki/Active_directory

[9] Mantis. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-03-20]. Dostupné z: http://cs.wikipedia.org/wiki/Mantis

[10] *MediaWiki* [online]. 2008 [cit. 2012-04-07]. Dostupné z: http://www.mediawiki.cz/cz/46*o-wiki

[11] *NetBeans IDE* [online]. [cit. 2012-04-07]. Dostupné z: http://netbeans.org/

[12] Vývojové prostředí. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-07]. Dostupné z: http://cs.wikipedia.org/wiki/Vyvojove_prostredi

[13] Správa dokumentů. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-12]. Dostupné z: http://cs.wikipedia.org/wiki/Sprava_dokumentu

[14] *Cloudové úložiště* [online]. [cit. 2012-04-13]. Dostupné z: http://www.root.cz/clanky/owncloud-vlastni-cloudove-uloziste/

[15] *Management znalostí* [online]. [cit. 2012-04-20]. Dostupné z: http://www.systemonline.cz/clanky/management-znalosti.htm

[16] Wiki. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-20]. Dostupné z: http://cs.wikipedia.org/wiki/Wiki

[17] Verzování. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-24]. Dostupné z: http://cs.wikipedia.org/wiki/Verzovani

[18] Apache Subversion. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-24]. Dostupné z: http://cs.wikipedia.org/wiki/Apache_Subversion

[19] *Síťové operační systémy* [online]. [cit. 2012-04-25]. Dostupné z: http://www.uniscomp.cz/t/sitove-operacni-systemy-linux-solaris-netware-windows

[20] *Debian* [online]. [cit. 2012-05-02]. Dostupné z: http://www.debian.cz

[21] Debian. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-03]. Dostupné z: http://cs.wikipedia.org/wiki/Debian

[22] *TortoiseSVN* [online]. [cit. 2012-05-03]. Dostupné z: http://tortoisesvn.net/docs/nightly/TortoiseSVN_cs/tsvn-preface.html

[23] INTERVAL.CZ. *Databáze a jazyk SQL* [online]. [cit. 2012-05-03]. Dostupné z: http://interval.cz/clanky/databaze-a-jazyk-sql/

[24] LINUXSOFT. *Netbeans IDE* [online]. [cit. 2012-05-07]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1761

[25] *Vodopádový vývoj* [online]. [cit. 2012-04-08]. Dostupné z: http://vyvojari.oxyonline.cz/vodopadovy-vyvoj-obhajoba

[26] *Prototypování* [online]. [cit. 2012-04-08]. Dostupné z: http://pss.tym.cz/prednasky/P02/foil21.html

[27] *Spirálový model* [online]. [cit. 2012-04-08]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/spiralovy-model/

[28] *Fáze metodiky UP* [online]. [cit. 2012-04-08]. Dostupné z: http://pss.tym.cz/prednasky/P04/foil25.html

[29] *Charakteristika RUP* [online]. [cit. 2012-04-08]. Dostupné z: http://objekty.vse.cz/Objekty/Rup2

[30] *LDAP adresářový strom* [online]. [cit. 2012-04-08]. Dostupné z: http://homel.vsb.cz/~las03/ldap/index.html

Příloha A – Konfigurace VirtualBoxu

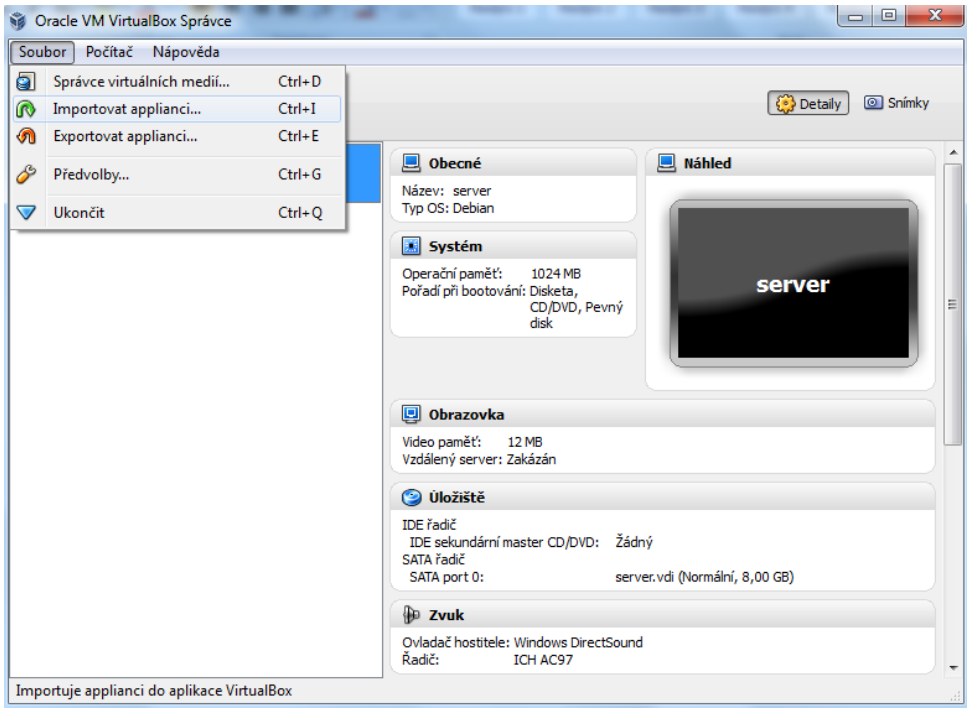
Tato příloha obsahuje popis základních kroků pro spuštění platformy.

Instalace, konfigurace a metodiky vychází z využití virtuálního počítače, na kterém je nainstalován server. Klient je suplován lokální stanicí. Server má přidělenu speciální virtuální síťovou kartu, která má nastavenu v operačním systému statickou IP adresu a lokální stanice s ním přes tuto adresu komunikuje. Adresa je nastavena na *192.168.100.1*.

Veškeré nastavení včetně samotné image, na kterém se nachází serverová část platformy, byla vyexportována do speciálního archivu VirtualBoxu. Výhodou je komprimace dat (snížení velikosti na třetinu), snadné přenesení a spuštění na jiném stroji.

Na následujících obrázcích je popsán import nastavení a virtuálního serveru ve VirtualBoxu. Virtual Box lze stáhnout ze stránek projektu[14], kde se rovněž nachází popis a manuály. Instalace s exportním souborem je rovněž přiložena na DVD, jenž je součástí diplomové práce.

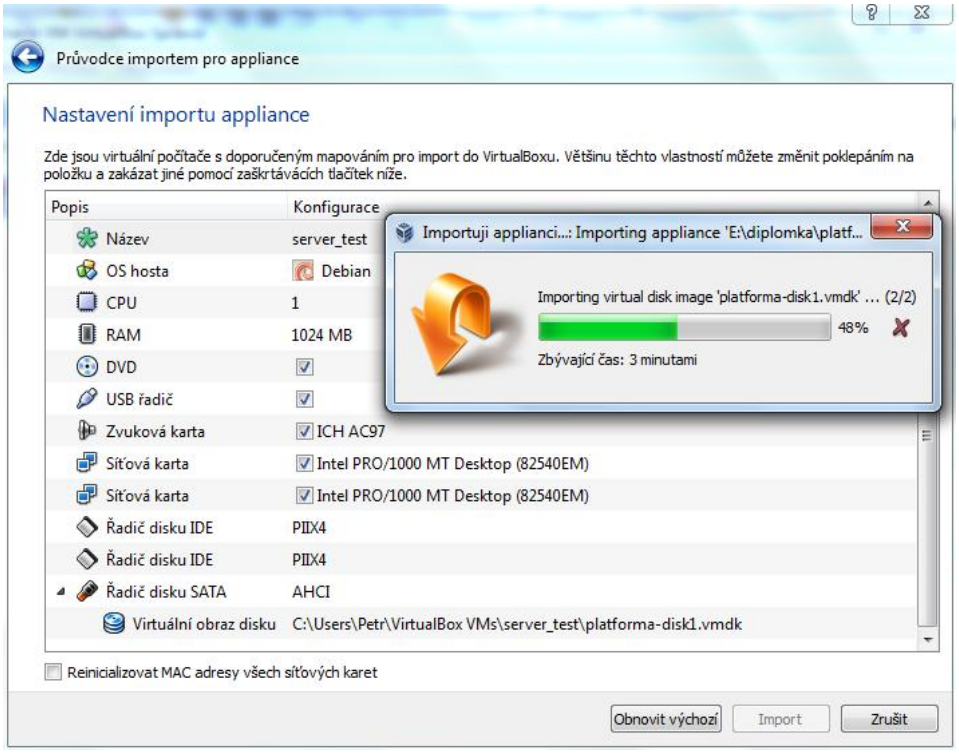
Spustíme správce VirtualBoxu a vybereme v menu *Soubor->Importovat appliance*, viz Obrázek 46.



Obrázek 46 – Import virtuálního stroje

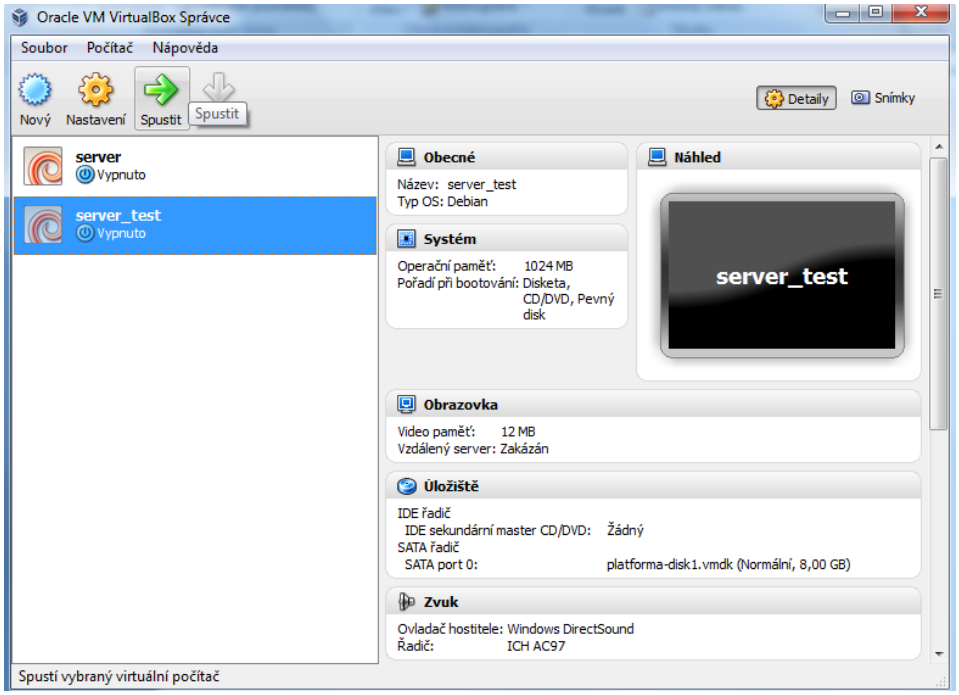
[14] https://www.virtualbox.org

Následně potvrdíme konfiguraci virtuálního stroje a stiskneme tlačítko *Import*, viz Obrázek 47.



Obrázek 47 – Potvrzení konfigurace a import

Po úspěšném importu vybereme naimportovaný virtuální stroj a dáme spustit, viz Obrázek 48.



Obrázek 48 – Spuštění virtuálního stroje

Po spuštění operačního systému se přihlásíme pod testovacím loginem: **petr** a heslem: **petr**.

Příloha B – Obsah přiloženého DVD

Přiložené DVD obsahuje:

- dokument *VacekP_PlatformaTymovy_LC_2012.pdf*, který obsahuje text diplomové práce v elektronické podobě,
- archiv virtuálního stroje (serverová část) ve VirtualBoxu, soubor *server.ova*,
- instalaci Virtual Boxu, instalační soubor *VirtualBox-4.1.18-78361-Win.exe*.