Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informační systémy a technologie

Diplomant: Bc. Petr Hofmann

Vedoucí diplomové práce: Ing. Luboš Pavlíček

Oponent diplomové práce: Doc. Ing. Alena Buchalcevová, Phd.

Využití Ruby on Rails pro podporu týmové spolupráce vývojářů

školní rok 2008/2009

Vysoká škola ekonomická — Katedra informačních technologií

Fakulta informatiky a statistiky — Školní rok 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno : Bc. Petr Hofmann

Obor : informační systémy a technologie

Vedoucí katedry Vám ve smyslu nařízení vlády o státních závěrečných zkouškách a státních rigorózních zkouškách určuje tuto diplomovou práci:

Téma : Využití Ruby on Rails pro podporu týmové spolupráce vývojářů

1. Úvod
2. Teoretická část
   - 2.1. Ruby
   - 2.2. Ruby on Rails
3. Praktická část
   - 3.1. Popis projektu
   - 3.2. Analýza a návrh systému
   - 3.3. Příprava produkčního prostředí
   - 3.4. Vývoj
   - 3.5. Testování
   - 3.6. Implementace
4. Závěr

Seznam literatury:

Thomas, Dave. *Agile Web Development with Rails, 2Ed.* United States of America : The Pragmatic Programmers LLC, 2006. ISBN 0-9776166-3-0.

Řepa, Václav. *Analýza a návrh informačních systémů*. Praha: EkoPress, 1999. ISBN 80-86119-13-0

Vedoucí diplomové práce: Ing. Luboš Pavlíček

Datum zadání diplomové práce: 23/09/2008

....

Vedoucí katedry Děkan

V Praze dne 25/4/2010

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval(a) samostatně a že jsem uvedl(a) všechny použité prameny a literaturu, ze kterých jsem čerpal(a).

V Praze dne 25.4.2010

…….

podpis

ABSTRAKT – ČESKY

Téma této diplomové práce zní Využití Ruby on Rails pro podporu týmové spolupráce vývojářů.

Hlavním cílem je podpoření týmové spolupráce vývojářů (studentů kurzů programování) na Katedře informačních technologií Vysoké škole ekonomické v Praze.

Splnění tohototo cíle bylo dosaženo vývojem nové aplikace na školním serveru kitscm.vse.cz v Ruby on Rails. Tato aplikace slouží pro podporu týmové spolupráce. Automatizuje některé činnosti, které se doposud na serveru kitscm.vse.cz prováděly ručně. Spolupracuje s dalšími open-source aplikacemi (openldap, dokuwiki, subversion) běžícími na serveru kitscm.vse.cz.

Po úvodu do problematiky (kapitola 1), následuje seznámení s Ruby a Ruby on Rails (kapitola 2), dále pak analýza a návrh aplikace (kapitola 3), seznámení s provozní architekturou aplikace (kapitola 4) a ukázka vlastního vývoje části aplikace (kapitola 5).

Hlavním přínosem je zhotovená aplikace podporující týmovou spolupráci, využívaná vyučujícími a studenty.

Celá aplikace je k dispozici na http://sourceforge.net/projects/kitscm/ pod GPL licencí.

Klíčová slova:

ruby on rails
ruby
podpora spolupráce vývojářů
kitscm
kitscm.vse.cz
openldap dokuwiki subversion

ABSTRACT – ENGLISH

Thema of this master thesis is Using Ruby on Rails for support of team collaboration of developers.

The main objective is support of team collaboration of developers (students of programming courses) at the Catedry of Computer Science of the University of Economics Prague.

The objective has been acomplished by developement of a new application in Ruby on Rails on the school server kitscm.vse.cz. This application is used for support of team collaboration. Application automatizes some of procedures, which were made on the server kitscm.vse.cz manually in the past. Application collaborates with another open-source applications (openldap, dokuwiki, subversion) running on the kitscm.vse.cz server.

After the introduction (chapter 1), follows familiarization with Ruby and Ruby on Rails (chapter 2), analysis and design of the application (chapter 3), familirization with application operational architecture (chapter 4) and example of development of part of the application (chapter 5).

Main benefit is an application supporting team collaboration, which is used by students and teachers.

The whole application is available on http://sourceforge.net/projects/kitscm/ under GPL license.

Keywords:

ruby on rails
ruby
cooperation support
kitscm
kitscm.vse.cz
openldap dokuwiki subversion

Osnova

1. KAPITOLA 1 - ÚVOD

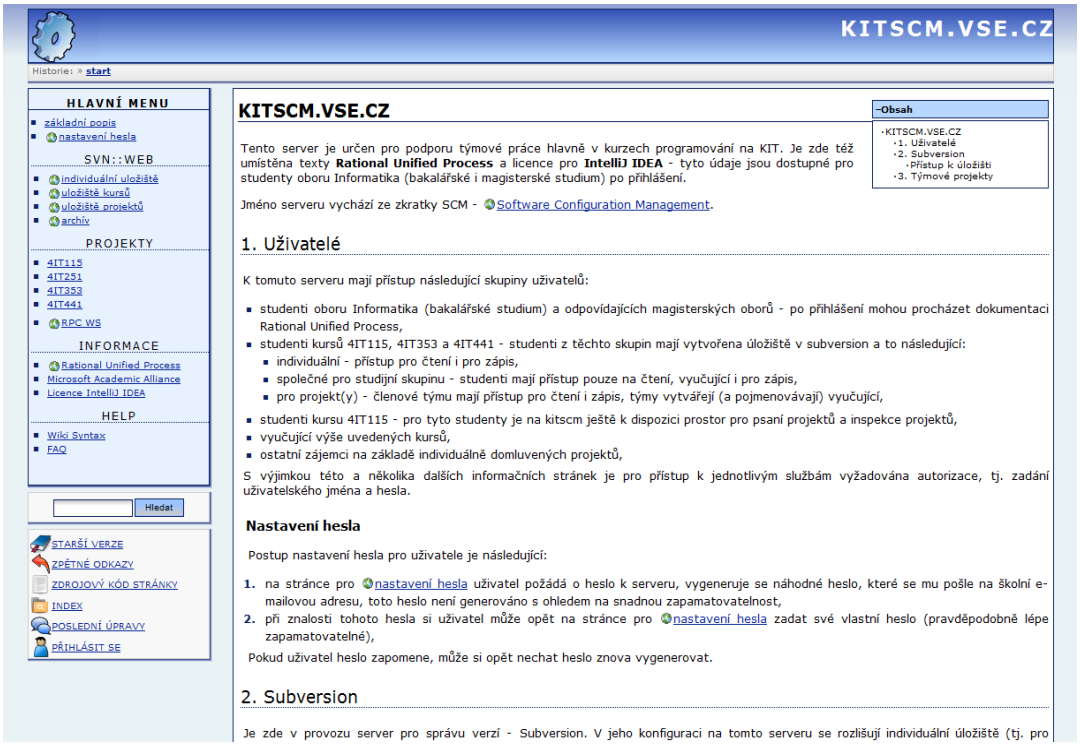
Cíle této kapitoly:	... seznámit s tématem a cílem diplomové práce ... přiblížit současný stav oblasti ... stručně popsat vyvíjenou aplikaci

Téma této diplomové práce zní **Využití Ruby on Rails pro podporu týmové spolupráce vývojářů.**

Hlavním cílem práce je podpoření týmové spolupráce vývojářů (studentů kurzů programování) na Katedře informačních technologií Vysoké školy ekonomické v Praze. Každoročně jsou na Fakultu informatiky a statistiky VŠE – obor Informatika přijaty stovky zájemců (ve školním roce 2009/2010 – 680 studentů) a jejich počet neustále vzrůstá. Součástí povinného bloku výuky bakalářského studijního programu Informatika jsou i základní kurzy programování v jazyce Java.

Právě studenti (bakalářské i magisterské studium) a vyučující těchto kurzů využívají školní server **kitcm.vse.cz,** jehož název je zkratkou **KIT** = Katedra Informačních Technologií a **SCM** = Software Configuration Management.

Server je určený primárně pro podporu týmové práce v kurzech programování.



Obrázek 1 - server KITSCM.VSE.CZ (1)

K serveru mají přístup: studenti kursů, vyučující kursů a ostatní zájemci na základě individuálně domluvených projektů.

Pro podporu týmové spolupráce jsou na serveru nainstalovány následující aplikace:

Subversion je server pro správu verzí. Na serveru se rozlišují následující druhy úložišť :

a) **Individuální úložiště** – pro studenty vybraných kurzů [KH1]

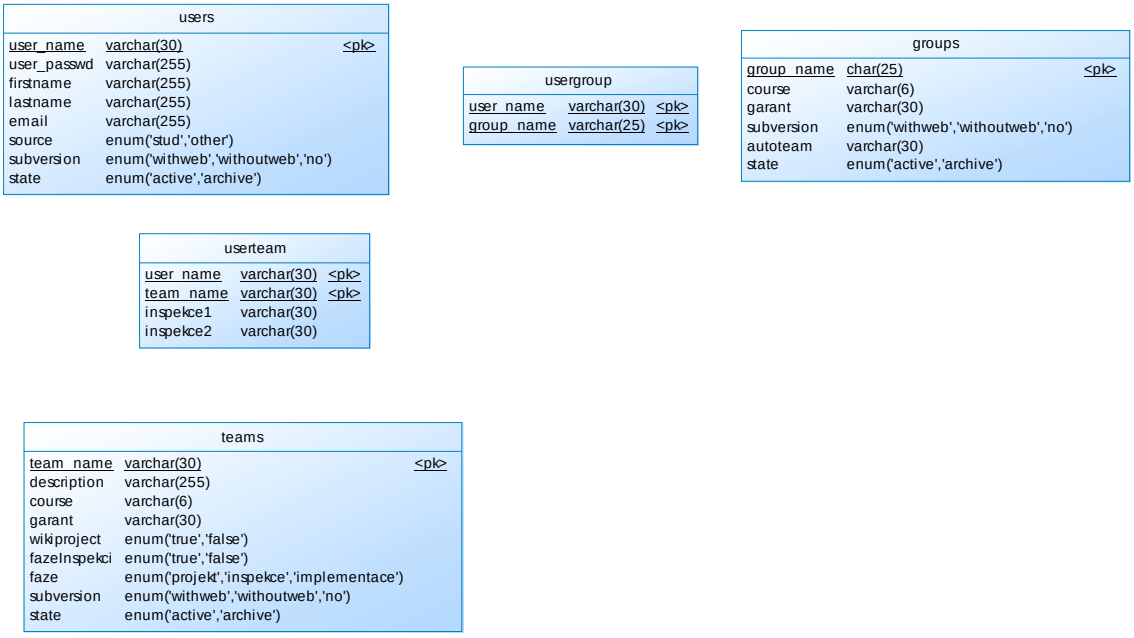
b) **Skupinová úložiště** - pro jednotlivé výukové skupiny.

c) **Úložiště pro projekty** - pro konkrétní projekty.

Prohlížení úložišť a jejich verzí umožňuje nainstalovaný program WEB:SVN.

Dokuwiki je využita jako domovská www stránka celého serveru, dále se pak v dokuwiki zakládají www stránky jednotlivých projektů, do kterých se při inspekci projektu zapisuje hodnocení.

Databáze MySQL eviduje ve třech oddělených tabulkách výukové týmy, výukové skupiny a externí uživatele probíhajících kurzů programování.



Obrázek 2 - evidence uživatelů, skupin a týmů

Veškeré úkony spojené s podporou týmové spolupráce vývojářů – zakládání uživatelů, zakládání výukových skupin, zakládání výukových týmů, rozřazení uživatelů do týmů a skupin, zakládání uživatelských, týmových a skupinových repositářů a nastavování jejich práv, zakládání týmových a skupinových wiki stránek, apod. se v současném stavu na serveru kitscm.vse.cz provádějí manuálně.

Splnění hlavního cíle této práce – podpoření týmové spolupráce vývojářů – se zajistí **vývojem nové aplikace** na serveru kitscm.vse.cz (v Ruby on Rails).

Aplikace umožní **automatizaci úloh**, které doposud museli vyučující provádět manuálně. Například aplikace umožní načíst vyexportovaný seznam všech studentů zapsaných do kurzů programování (ze systému ISIS), automaticky uživatele rozřadit do výukových skupin, automaticky založit uživatelské a skupinové repositáře a nastavit práva repositářů, automaticky založit wiki stránky skupin a nastavit jejich práva, apod.

Dále aplikace umožní **online správu** výukových skupin a projektů přes internet. Správa se rovněž centralizuje – veškeré informace budou uloženy v adresářovém serveru (OpenLDAP). Centralizací se umožní funkce jednotného přihlášení (**single sign on)** – uživatelé a učitelé budou pro různé systémy využívat pouze jedno heslo.

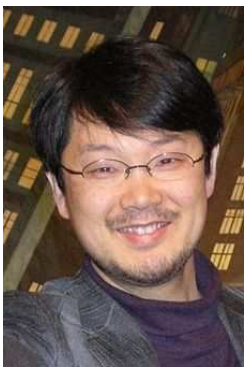
2. KAPITOLA 2 – SEZNÁMENÍ S RUBY ON RAILS

Cíle této kapitoly:	... seznámit s programovacím jazykem Ruby ... seznámit s frameworkem Ruby on Rails

Vyvíjená aplikace bude naprogramována ve webovém frameworku Ruby on Rails. Tento termín se skládá ze dvou pojmů. První z nich – **Ruby**, popisuje poměrně nový programovací jazyk. Druhý – **Rails**, označovaný většinou "Ruby on Rails" nebo někdy jen "Rails" či "RoR" představuje framework pro tvorbu webových aplikací, který je vystavený právě na programovacím jazyku Ruby.

2.1 PROGRAMOVACÍ JAZYK RUBY

V roce 1995 spatřil světlo světa další programovací jazyk. Jeho zakladatelem je jediný člověk, pan Yukihiro "Matz" Matsumoto z Japonska. Pojmenoval ho "rubín" = Ruby.



Obrázek 4 – Yukhiro Matzumoto [Zdroj: http://en.wikipedia.org/wiki/Yukihiro_Matsumoto]

Jak a proč Ruby vznikl? Tuto otázku může určitě nejlépe zodpovědět sám zakladatel, pan Matsumoto, který v listopadu 2001 poskytl pro nakladatelství OREILLY rozhovor (2) :

Stewart: *Začneme trochou historie. Kdy jste se rozhodl napsat Ruby?*

Matz: *V roce 1993, mluvil jsem s kolegou o skriptovacích jazycích. Byl jsem velmi ohromen jejich silou a možnostmi. Cítil jsem, že skriptování je ta správná cesta, kterou bych se měl vydat. Jako dlouhodobému fanouškovi objektového programování se mi zdálo, že OO programování bylo rovněž vhodné i pro skriptování. Pak jsem se podíval na internet a zjistil jsem, že Perl5, který ještě nebyl zveřejněn, bude implementovat OO rysy, ale to nebylo to, co jsem hledal. Vzdal jsem se proto jazyka Perl jako OO skriptovacího jazyka.*

Pak jsem narazil na Python. Byl to interpretovaný, OO jazyk. Ale nepůsobil na mě jako skriptovací jazyk. Byl to takový hybrid mezi procedurálním a OO programováním.

Chtěl jsem skriptovací jazyk, který by byl více výkonný než Perl a více OO než Python, proto jsem se rozhodl vytvořit svůj vlastní jazyk.

Tolik zakladatel "Matz" o důvodech, které ho vedly k vytvoření Ruby.

2.1.1 CHARAKTERISTIKA VLASTNOSTÍ RUBY

Stručně můžeme Ruby charakterizovat následujícím výčtem jeho vlastností (3,4,5):

Interpretovaný (skriptovací) jazyk

Ruby můžeme zařadit mezi skriptovací jazyky (např. dialekty UNIXových shellů, PHP, Perl, apod.), odpadá zde nutnost procesu kompilace, která je ovšem vyvážena nižší rychlostí. Kvůli nižší rychlosti byly tyto jazyky původně používány pouze pro menší projekty. Díky neustále se zvyšujícímu výkonu hardwaru se však tyto jazyky začaly používat i pro projekty většího rozsahu.

Objektově orientovaný jazyk

Oproti ostatním skriptovacím jazykům, které byly většinou navrhnuty jako procedurální a pak k nim byly přidávány objektové prvky je Ruby navržený čistě objektově a je plně objektově orientovaný. Všechny prvky jazyka Ruby jsou objekty, kořenovou třídou je třída Object. Každá třída je instancí třídy Class, která je však potomkem třídy Object.

Mezi jeho objektově orientované rysy patří hlavně:

- Datová abstrakce a modelování
- Zapouzdření
- Dědičnost

Další vlastnosti jazyka

Regulární výrazy – Ruby disponuje tímto velmi silným nástrojem, porovnávání řetězce vůči určitému vzoru.

Garbage collector - v Ruby není nutno uvolňovat alokovanou pamět. Nepoužívaná pamět se automaticky uvolňuje Garbage collectorem založeném na algoritmu Mark and Sweep. Není třeba volat destruktory.

Výjimky – Ruby obsahuje ošetření výjimek, jako např. Java.

Mix-in – v Ruby není implementována vícenásobná dědičnost. Ta je nahrazena systémem modulů, které mohou třídy importovat. Třída importující modul se nazývá Mix-in.

Multithreading – v Ruby je interně implementován vlastní systém vláken. Multithreading tedy funguje na všech OS, kde funguje Ruby (např. i v MS-DOS). Nevýhoda vlastního systému vláken je nižší rychlost a stabilita vůči systému vláken uvnitř operačního systému.

Rozšiřitelnost – existují implementace interpretu ruby v jiných jazycích. Např. Java – JVM Jruby (integrace obou jazyků). Existují však konvertory do dalších jazyků (jazyka C).

Přenositelnost – díky tomu, že je interpret jazyka Ruby napsán v jazyce C je Ruby spustitelný na mnoha platformách (Unix, Windows, Dos, MacOs, ...)

Jednoduchá syntaxe – Ruby je jazyk snadno čitelný, přehledný a lehce pochopitelný. Na druhou stranu je však gramaticky a sématicky silný.

Open source – Ruby je vyvíjený pod Open source licencí a je tedy zdarma.

Interaktivní režim – program IRB (Interactive Ruby Shell), podobný shellu umožnuje programátorovi okamžité experimentování s kódem v reálném čase.

Základní princip jazyka

Filosofii Ruby opět nejlépe vysvětlí slova zakladatele, "Matze" (2):

Stewart: *Měl jste nějakou filosofii, která Vás vedla při návrhu Ruby?*

Matz: *Ano, nazývá se "**princip nejmenšího překvapení**". Věřím, že lidé se chtějí vyjádřit, když programují. Nechtějí bojovat s programem. Programátoři musí cítit programovací jazyky, jako něco přirozeného. Zkusil jsem, aby se lidé při programování bavili a koncentrovali se na zábavu a kreativní část programování, když používají Ruby.*

Ruby je navrhnut jako lidsky-orientovaný jazyk. Odlehčuje břemeno programování. Pokouší se posunout práci zpět od lidí ke strojům. Můžete dosáhnout více úkolů s menším úsilím, v menším množství lépe čitelného kódu.

Oblast použití

Ruby se nejčastěji využívá pro zpracování textu, psaní CGI skriptů, tvorbu síťových aplikací, tvorbu GUI, práce s XML a výuku OO programování.

Ruby se rovněž začíná využívat i při rozsáhlých projektech, kde přináší především zkrácení doby vývoje, úsporu zdrojů a snadnější údržbu.

Rozšířenost jazyka

Ohledně rozšířenosti jazyka Ruby, dle společnosti TIOBE Software, která sestavuje každý měsíc index popularity programovacích jazyků, zaujímá jazyk Ruby místo v první desítce (říjen 2008).

Position Oct 2008	Position Oct 2007	Delta in Position	Programming Language	Ratings Oct 2008	Delta Oct 2007	Status
1	1	=	Java	20.949%	-0.67%	A
2	2	=	C	15.565%	+0.97%	A
3	4	↑	C++	10.954%	+1.37%	A
4	3	↓	(Visual) Basic	9.811%	-1.35%	A
5	5	=	PHP	8.612%	-0.89%	A
6	8	↑↑	Python	4.565%	+1.13%	A
7	6	↓	Perl	4.419%	-0.93%	A
8	7	↓	C#	3.767%	+0.03%	A
9	13	↑↑↑↑	Delphi	3.288%	+1.75%	A
10	10	=	Ruby	2.860%	+0.47%	A

Obrázek 5 - TIOBE index (říjen 2008) [Zdroj: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html]

Nutno podotknout, že hodnocení tohoto indexu jsou založena na celosvětovém počtu programátorů daného jazyka, kursech a školeních v daném jazyce a výrobců třetích stran, které daný jazyk využívají (6). Ke kalkulaci indexu se využívají vyhledávací enginy jako Google, MSN, Yahoo a YouTube (6).

TIOBE index tedy rozhodně nechce vyhlašovat nejlepší programovací jazyk nebo jazyk, ve kterém je napsané nejvíce řádků kódu (6). Dle společnosti TIOBE se index může spíše využít k ověření, zda jsou programovací schopnosti programátora aktuální a zda odpovídají poptávce (6). Dále se index doporučuje využít při strategických rozhodnutích, jaký to programovací jazyk má být použit při tvorbě nových informačních systémů (6).

Současný stav

Současná stabilní verze (listopad 2009) je verze Ruby 1.9.1.

2.1.2 ALTERNATIVNÍ IMPLEMENTACE RUBY

Dále existují i alternativní implementace jazyka Ruby. Každá z nich přistupuje k Ruby odlišným způsobem. Mezi nejznámější patří následující projekty:

YARV (7) - http://www.atdot.net/yarv/

YARV je akronymem pro Yet Another Ruby Virtual machine (zase další virtuální stroj pro Ruby). Jeho autorem je Koichi Sasada, proto se také někdy pojmenovává jako KRI (Koichi´s Virtual Machine). YARV je nový interpret pro Ruby, jehož hlavním cílem je zvýšení rychlosti Ruby programů. YARV byl také zvolen oficiálním interpretem pro Ruby od verze 1.9.

JRuby (8) - http://jruby.codehaus.org/

JRuby je open source Javovská implementace interpretu jazyka Ruby napsaná v Javě. Podporuje just-in-time kompilaci. Celá implementace je těsně spojena s Javou, což jí umožňuje vložit interpret Ruby do jakékoli Java aplikace. Je při tom oboustranně podporován přístup mezi Javou a Ruby kódem.

Rubinius (9)- http://rubini.us/

je další virtuální stroj a kompilátor pro Ruby, jehož cílem je poskytování bohatého a vysoce výkonného prostředí pro běh Ruby programů.

IronRuby - (10) http://rubyforge.org/projects/ironruby/

je interpret Ruby pocházející od společnosti Microsoft. Cílem je vývoj interpretu Ruby pro Microsoft .NET framework. Interpret je založený na knihovně Dynamic Language Runtime.

MacRuby (11) – www.macruby.org

je verze Ruby 1.9 upravená tak, aby mohla být přenesena přímo na Mac OS X.

2.1.3 RUBY GEMS

RubyGems jsou balíčkovacím systémem pro jazyk Ruby. Jednotlivé programy, knihovny a další utility jazyka Ruby jsou označovány jako gemy. RubyGems definují také standartní formát těchto gemů. Na straně klienta poskytují RubyGems nástroj, díky kterému lze velmi jednoduše řídit instalaci jednotlivých gemů.

Například instalace nového balíčku se provede příkazem:

```
gem install název_balíčku
```

Seznam instalovaných gemů na počítači se provede příkazem:

```
gem list --local
```

Na straně druhé RubyGems poskytují server pro distribuci těchto gemů.

Protože jsou RubyGems napsány v Ruby, jsou stejně tak jako Ruby multiplatformní. Jako ostatní vyspělé balíčkovací systémy automaticky RubyGems kontrolují závislosti.

Domovská stránka projektu RubyGems je www.rubygems.org.

2.1.4 RUBYFORGE

RubyForge je systém pro řízení vývoje software, který je určený pro projekty v jazyce Ruby.

Vývojáři na projektech spolupracují, cílem je zapojit do projektů co nejvíce vývojářů a poskytnout jim zázemí. RubyForge spolupráci ulehčuje pomocí různých služeb – např. mailing listy, wiki, fóra, správa kódu (CVS), nástroje pro sledování projektu, apod.

V březnu 2008 server RubyForge hostoval více než 5000 projektů a měl více než 26000 registrovaných uživatelů (13).

2.1.5 FRAMEWORKY VYUŽÍVAJÍCÍ JAZYK RUBY

V jazyce Ruby je také naprogramováno několik frameworků, pojďme si některé z nich stručně představit.

Ruby on Rails (14) - http://rubyonrails.org/

Ruby on Rails jsou open-source frameworkem pro vývoj webových aplikací, založeným především na návrhovém vzoru MVC.

V současnosti se jedná o nejpoužívanější framework napsaný v jazyce Ruby. Právě díky Ruby on Rails získal programovací jazyk Ruby značnou popularitu.

V tomto frameworku je napsána i aplikace pro podporu týmové spolupráce. Detailně si tento framework popíšeme v následující kapitole.

Merb (15)- http://merbivore.com/

Merb je MVC framework na vývoj webových aplikacích založený na frameworku Ruby on Rails. Původně byl Merb pouhým patchem do Ruby on Rails pro upload souborů. Postupem času se vyvinul v plnohodnotný webový framework.

Oproti Rails se filosofie celého frameworku zcela liší. Základní myšlenkou je **konfigurovatelnost** a **nezávislost**. Merb se snaží o **nezávislost** na konkrétním ORM (Objektově Relačním Mapování), knihovně JavaScriptu, šablonovacímu jazyku. Upřednostňuje se možnost přidání potřebné funkcionality pomocí pluginů - před použitím jedné masivní knihovny, která obsahuje veškerou funkcionalitu v jádru. V projektu si sám vývojář může zvolit, které nástroje ve své aplikaci bude chtít užívat, např. pro ORM je k dispozici jak ActiveRecord, tak pro Merb typický DataMapper či Sequel. Volba komponent aplikace tedy záleží na jednotlivých uživatelích frameworku, nikoli na jeho tvůrcích.

Nitro (16)- http://www.nitroproject.org/

Nitro je další framework pro tvorbu webových aplikací založený na jazyce Ruby. Je vyvíjený pod BSD licencí. Základní filosofií je skutečnost, že Nitro striktně nediktuje strukturu webové aplikace. Vývojář může používat šablony s vloženým kódem (podobně jako např. v PHP) nebo si může zvolit přístup MVC, popřípadě může zajít i dále volbou dalšího architektonického vzoru.

Camping (17) - http://camping.rubyforge.org/files/README.html

Camping je micro webový framework. Proč mikro? Celý totiž zabírá méně než 4kb kódu. Kompletní zdrojový kód může být zobrazen na jedné stránce.

Základní ideou je uložení kompletního základu webové aplikace do jednoho souboru, který je ale organizovaný do MVC částí, stejně jako Rails aplikace. Pak lze aplikaci velmi snadno migrovat do Ruby on Rails.

2.2 RUBY ON RAILS

Cíle této kapitoly:	... představit RoR … seznámit s hlavními součástmi RoR

Ruby on Rails jsou open source frameworkem pro vývoj webových aplikací. Jsou napsány v programovacím jazyce Ruby. Jejich název je někdy zkracován na "Rails" či "RoR". Vytvořil je dánský programátor David Heinemeier Hansson dá se říci "jako vedlejší produkt" při práci na svém projektu Basecamp.

Časopis Computerworld je zařadil mezi 5 nejvýznamnějších technologií pro rok 2007 (19).

V praxi Rails využívá celá řada organizací. Od organizací začínajících, přes organizace neziskové, včetně velkých komerčních organizací (www.twitter.com, www.yellowpages.com, a další). (14)

Framework (20)

Nejdříve se pozastavme nad slovem **framework**, co to vůbec framework je? Framework můžeme definovat jako určitou softwarovou strukturu, která podporuje programování, vývoj a organizaci softwarových projektů. Součástí frameworku mohou být různé podpůrné programy, doporučené postupy při vývoji, návrhové vzory, knihovny, apod. Hlavním cílem frameworku je **zjednodušení a ulehčení práce** při vývoji softwarových projektů.

Nejčastější námitkou proti použití frameworků jsou vyšší hardwarové nároky a nižší rychlost aplikací. (20)

Design Rails

Design Rails je ovlivněn několika základními koncepty (21)

Klíčovým konceptem je koncept **nejste unikátní**. Naprostá většina webových aplikací je si podobná – jako vejce vejci. Získávají data z HTML formulářů, ukládají je do databáze, z ní je pak opět načítají a zobrazují je v HTML stránkách uživatelům.

Sám David Heinemeier Hannson během přednášky na konferenci Future of Web Applications v únoru 2006 řekl: *„Toto je sněhová vločka. Vaše aplikace není jedna z nich. Většina věcí, které většina lidí dělá, není nijak unikátní. Vaše potřeby nejsou nijak ‚zvláštní‘.“* (22)

Dalším konceptem je **DRY** – zkratka anglického Don´t Repeat Yourself (česky neopakuj se). V Rails je tento koncept zahrnut jako **Konvence má přednost před konfigurací**. Po programátorovi je

vyžadována pouze specifikace nekonvenčního chování aplikace. Pokud se bude řídit konvencemi je standartní chování předdefinované. Rails takto automatizuje mnoho úloh a šetří čas programátora. Pokud se programátor potřebuje odchýlit od konvencí, může je jednoduše přepsat (konfigurovat).

Rails jsou také významně ovlivněny **Agilním programováním**. V roce 2001 se sešlo v lyžařském areálu Snowbird ve státě Utah 17 prominentních osobností agilního vývoje za účelem diskuse o lehčím, rychlejším a více lidsky orientovaném způsobu vytváření software (23). Vytvořili Agilní manifest (24), ve kterém jsou formulovány pilíře agilního programování.

1. osobnosti a komunikace místo procesů a nástrojů
2. funkční software místo podrobné dokumentace
3. spolupráce se zákazníkem místo vyjednáváním smlouvy
4. reakce na změnu místo dodržováním plánu

Rails jako framework jsou navrženy pro přímou podporu agilní metodiky vývoje software.

Pomocí Rails lze velmi snadno a rychle pomocí generátorů vytvořit kousky funkčního kódu (části webové aplikace). Tyto části pak lze jednoduše a snadno otestovat, předložit zákazníkovi a diskutovat s ním o změnách. Celou aplikaci Rails lze velmi snadno a rychle změnit a iterativně pokračovat v dalším vývoji.

Klíčové součásti Ruby on Rails (22 str. 28)

Mezi klíčové součásti Ruby on Rails patří v první řadě **návrhový vzor MVC**, na kterém jsou celé Rails postaveny.

Jádro frameworku je rozděleno do **pěti hlavních balíčků**:

1. active record
2. action pack (action controller + action view)
3. active resource
4. active support
5. action mailer

Dále lze funkcionalitu lze rozšířit mnoha **pluginy**.

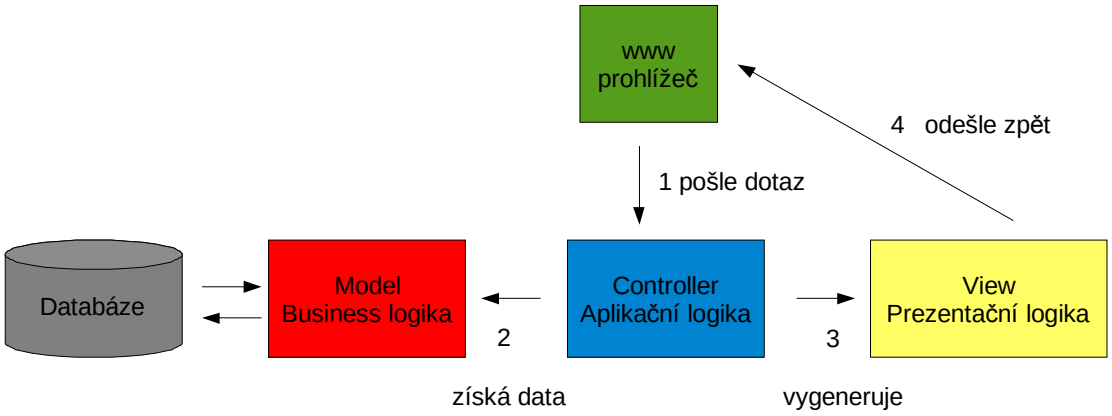
K dispozici jsou i **generátory kódu**, buildovací program **Rake**, nástroj pro **deployment Capistrano** a mnoho dalšího.

V následujících podkapitolách si postupně přiblížíme jednotlivé části Ruby on Rails.

2.2.1 MODEL – VIEW - CONTROLLER

Architektonicky jsou Ruby on Rails postaveny na návrhovém vzoru MVC – **Model, View, Controller**. Poprvé tento návrhový vzor popsal pan Trygve Reenskaug v roce 1979 (25). Pan Reenskaug přišel s novou architekturou, která rozděluje aplikace do 3 oddělených částí: model, view a controller. Odděluje se tak **datový model** aplikace (model), **uživatelské rozhraní** (view) a **řídící logika** aplikace (controller).

MVC aplikace zpracovává uživatelské požadavky následujícím způsobem:



Obrázek 7 – funkcionalita MVC

1. Uživatel v prohlížeči odešle serveru požadavek (dotaz) na internetovou stránku Controlleru.
2. Controller si vyžádá data příslušná data od modelu. Model je získá z DB.
3. Controller ze získaných dat vygeneruje stránku a pošle ji View.
4. View odešle a zobrazí stránku zpět klientovi

Rozdělení softwarové aplikace do těchto třech oddělených částí s sebou nese následující výhody :

- zlepšuje se **škálovatelnost** (možnost růstu) aplikace. Díky oddělení změna 1 části aplikace neovlivní další komponenty.
- oddělením komponent se zlepšuje **správa**, komponenty na sobě mají nízkou závislost
- **znovupoužití** – jeden model může využívat vícero views
- **distrubovatelnost** aplikace – oddělením kódu je možné každou komponentu převést v případě potřeby na samostatný server

MVC je v Rails implementován ve třech základních komponentách:

1. ActiveRecord (Model)

2. ActionController (Controller)
3. ActionView (View)

Pro pojmenování těchto prvků Rails využívají následující **konvence** (26 str. 241)**.**

Modely jsou uložené v adresáři app/models jako ruby soubory a pojmenovány dle názvu třídy modelu (user.rb), jméno třídy je v jednotném čísle, jméno tabulky v DB v čísle množném.

Views jsou v adresáři app/views/jméno_modelu. Je k nim automaticky vytvořený helper (user_helper.rb), který je v adreásři app/helpers.

Controllery jsou v adresáři app/controllers, jsou pojmenovány dle názvu třídy + Controller (UserController), taktéž jsou to soubory ruby (user_controller.rb) a automaticky je k nim vygenerován layout app/views/layout/název_modelu (user.rhtml).

Model	
tabulka	Users
soubor	app/models/user.rb
třída	User

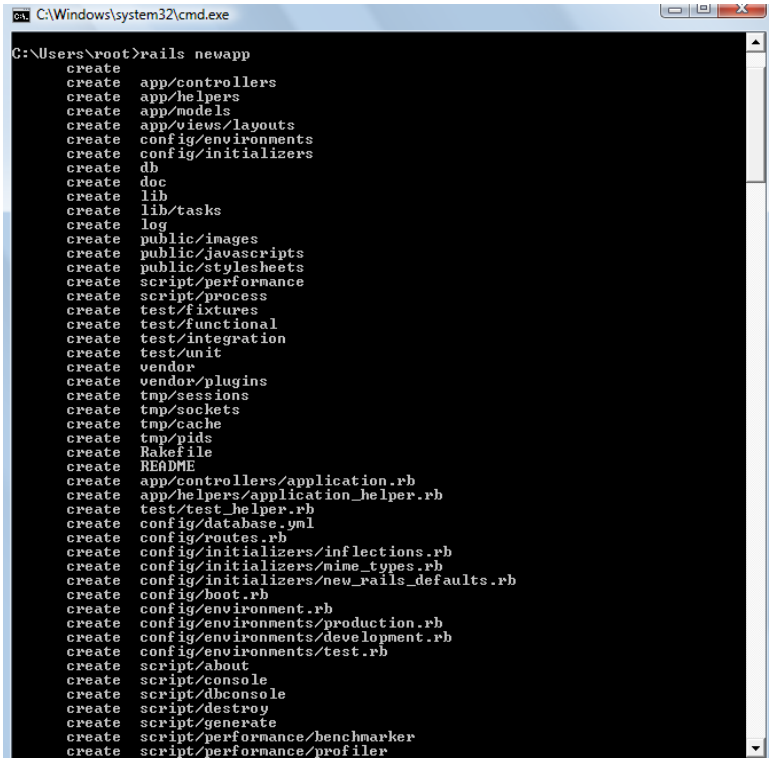
View	
URL	http://.../user/show
soubor	app/views/user/show.rhtml (.rxml, .rjs)
helper	UserHelper
soubor	app/helpers/user_helper.rb

Controller	
URL	http://.../user/show
soubor	app/controllers/user_controller.rb
třída	UserController
metoda	show
layout	app/views/layout/user.rhtml

2.2.2 GENERÁTORY KÓDU, RAKE TASKY, CAPISTRANO, KONZOLE, …

V Rails se používají **pasivní generátory kódu**, které programátorům šetří mnoho práce. V podstatě to jsou parametrizované šablony, které při zavolání generují výstup (v případě Rails – adresáře a soubory Rails aplikace). Jakmile je výsledek vygenerován, stane se plnohodnotnou součástí Rails projektu. Může být např. editován, spouštěn, verzován jako jakákoli jiná část Rails projektu. Zapomene se na jeho původ.

Již samotné založení Rails aplikace (spuštění příkazu *rails + jméno_aplikace*) je spuštění takovéhoto generátoru, který automaticky založí všechny potřebné části Rails aplikace. Vše je vidět na následujícím obrázku, kde se postupně zakládají jednotlivé adresáře (controllery, views, modely, apod.) a soubory (*.rb) aplikace *newapp*.



Obrázek 8 - založení nové Rails aplikace (Windows Vista)

Generátory se využívají i během tvorby celé Rails aplikace. Zakládání modelů, views, controllerů se dělá pomocí skriptu (soubor script/generate.rb). Například nový model uvnitř Rails aplikace vygenerujeme příkazem:

```
ruby script/generate model User
```

Generátor opět v souladu s konvencemi Rails (a konceptem **Konvence má přednost před konfigurací**) automaticky založí všechny potřebné části Rails aplikace (migrační schéma pro model, testy pro model, helper pro model, apod.)

```
C:\Users\root\newapp>ruby script/generate model User
      exists  app/models/
      exists  test/unit/
      exists  test/fixtures/
      create  app/models/user.rb
      create  test/unit/user_test.rb
      create  test/fixtures/users.yml
      create  db/migrate
      create  db/migrate/20081107141215_create_users.rb

C:\Users\root\newapp>
```

Obrázek 9 - založení nového modelu

Adresářová struktura Rails aplikace (26 str. 228)

Každá Rails aplikace má následující strukturu:

moje_aplikace/	
README	Soubor obsahující základní informace o instalaci a použití.
Rakefile	Buildovací skript.
app/	Adresář s Modely, Views a Controllery. Hlavní kód aplikace.
config/	Adresář pro konfigurační parametry aplikace a databáze.
db/	Databázová schémata a migrace.
doc/	Automaticky vygenerovaná dokumentace (po spuštění rake doc:app).
lib/	Sdílený kód. Dodatečné knihovny.
log/	Logovací soubory aplikace.
public/	Veřejně přístupný adresář. WWW server ho používá jako root aplikace.
script/	Skripty (generátor kódu, konsole, instalátor pluginů, apod.).
test/	Testovací adresář (unit, funkční, integrační, fixtury a mocky).
tmp/	Dočasně soubory potřebné pro běh aplikace (sessions, cache, sockety).
vendor/	Kód importovaný od 3 stran (pluginy, apod.).

Adresář app obsahuje podadresáře, ve kterých je dle souladu s MVC architekturou oddělena struktura aplikace:

app/	**Hlavní kód aplikace**
controller/	Controller
helpers/	
models/	Model
views/	View

Rake (27)

Rake je buildovací systém pro Rails. Je ekvivalentem programu make. Rake je napsán kompletně v jazyce Ruby a stejně tak jeho soubory Rakefiles (ekvivalent Makefiles) využívají syntaxi Ruby.

Rake ulehčuje práci při vývoji Rails aplikací, např. při tvorbě gem balíčků, čištění cache paměti, migraci databáze, apod.

Capistrano (28)

Pro deployment (nahrání a aktualizace kódu na server) Rails aplikací se využívá open source nástroj Capistrano, který je napsaný v Ruby. Základem je verzovaný kód Rails aplikace (v Subversion nebo Git) a jeho automatizované nahrání na server, aktualizace, případně rollback (návrat) k předchozí (funkční) verzi Rails aplikace.

Konzole

Součástí Rails je i konzole, kterou lze spustit ruby script/console (v domovském adresáři Rails aplikace). Konzole umožní práci s běžící aplikací v terminálu. Je možné ji spustit v každém prostředí – třeba i nad produkčním a "ostrou" databází (*ruby script/console production*). Pro programátora představuje konzole neocenitelný nástroj. S její pomocí lze provádět administrátorské zásahy do běžící aplikace (např. změnu a nastavení hesel), přistupovat k datům modelů, helperům a omezeně i ke controllerům. Velmi efektivně lze s ní Rails aplikace ladit.

Rails prostředí (26 str. 237)

V Rails je zaveden koncept třech oddělených prostředí. Development, test a production. Každé z těchto prostředí má své vlastní konfigurační parametry. Aplikaci je možné provozovat v každém z těchto prostředí a mezi prostředími se může přepínat:

```
ruby script/server –e development
ruby script/server –e test
ruby script/server –e production
```

Velmi výhodné je, že při přechodu mezi prostředími nedochází k žádným změnám v kódu Rails aplikace.

Testování (26 str. 184)

Rails v sobě mají zabudovanou podporu pro automatizované testování aplikací.

Testy nalezneme v adresáři test/.

Rozlišují se:

1. Unit tests – testují modely a jejich business logiku
2. Funkcional tests – testují 1 metodu v controllerech
3. Integrational tests – testují tok skrze controllery a modely

Dále Rails podorují i testovací data – fictures.

2.2.3 RUBY, RUBY GEMS, RAILS, ZÁSUVNÉ MODULY

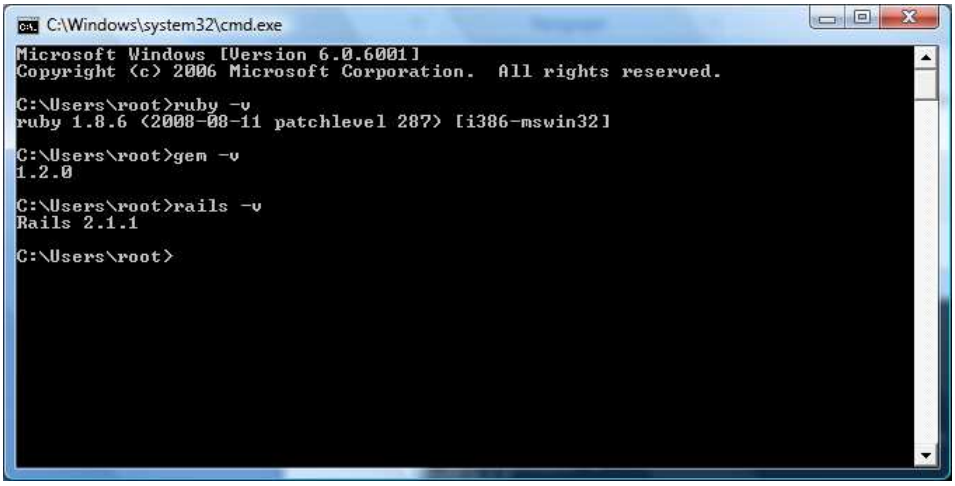
Pokud chceme Rails nainstalovat, první věcí, kterou musíme mít, je programovací jazyk **Ruby**. Na www.ruby-lang.org lze stáhnout nejnovější verzi. Podporovaná je většina operačních systémů (Windows, Linux, OS X, …). Pro Windows existuje instalační soubor .exe, pro Linux/Unix jsou k dispozici zdrojové kódy a balíčky, lze využít i balíčkovací systémy (Ubuntu – *apt-get install ruby*).

Jazyk Ruby využívá balíčkovací systém **Ruby-Gems**. Pomocí něho lze instalovat jak knihovny pro Ruby, tak i s Ruby spojené ulitily (Rake, Capistrano). Domovská stránka je Ruby Gems je www.rubygems.org, zde lze Ruby Gems stáhnout. Instalují se spuštěním Ruby programu setup.rb (*ruby setup.rb*).

Samotný framework **Ruby on Rails** je balíček RubyGems. Instaluje se pomocí balíčkovacího systému Ruby Gems – příkazy:

```
gem update

gem  install rails --include-dependencies
```

Funkčnost Ruby, Rubygems a Rails lze ověřit příkazy:

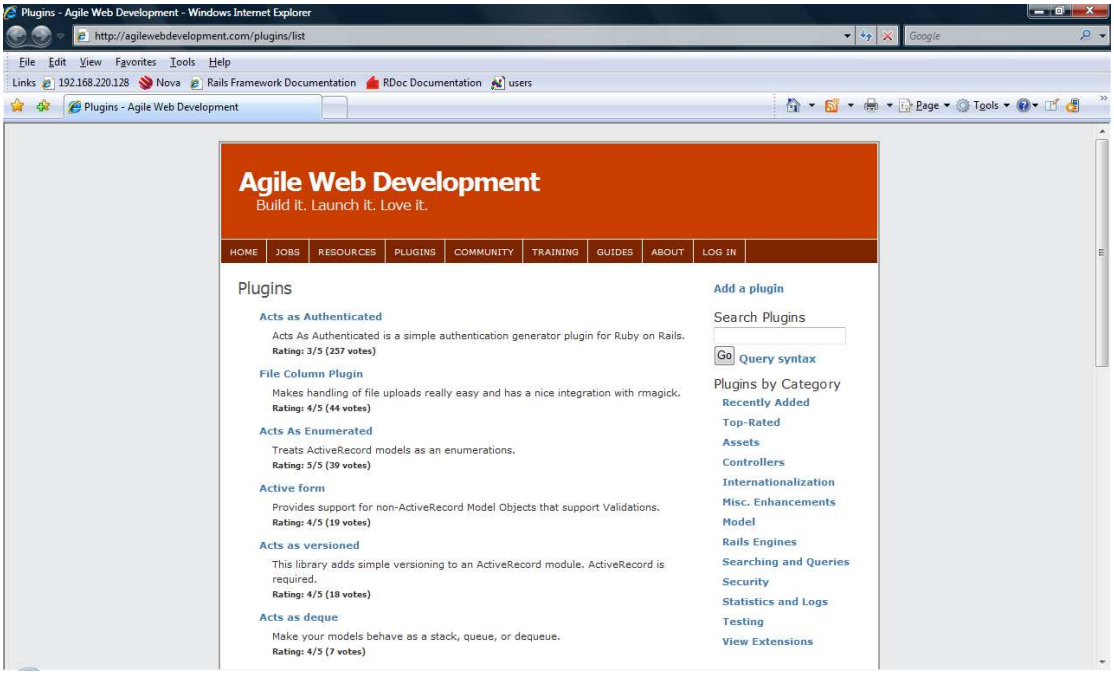


Obrázek 10 - ověření funkčnosti Ruby, Ruby Gems a Rails

Pro Rails existuje značné množství zásuvných modulů - plugins. Pomocí zásuvných modulů lze velmi rychle (a pohodlně) do aplikace implementovat funkcionalitu, kterou již před námi někdo napsal. Můžeme říci, že to jsou vlastně již hotové komponenty, které lze používat v Rails aplikacích. Nainstalujeme je pomocí skriptu uloženého v adresáři script/ :

ruby plugin/install název_pluginu (nebo URL repositáře)

Seznam zásuvných modulů je na http://agilewebdevelopment.com/plugins/list v současné době (říjen 2008) je jich na této straně k dispozici 1231.

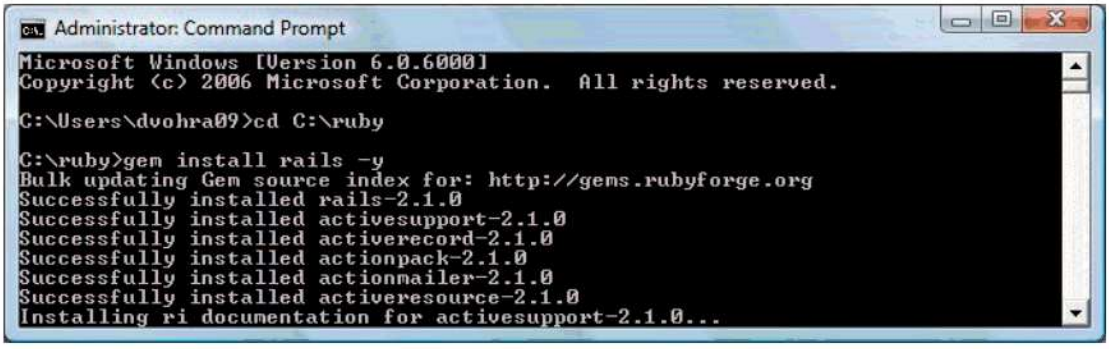


Obrázek 11 - pluginy pro Rails [Zdroj: http://agilewebdevelopment.com/plugins/list]

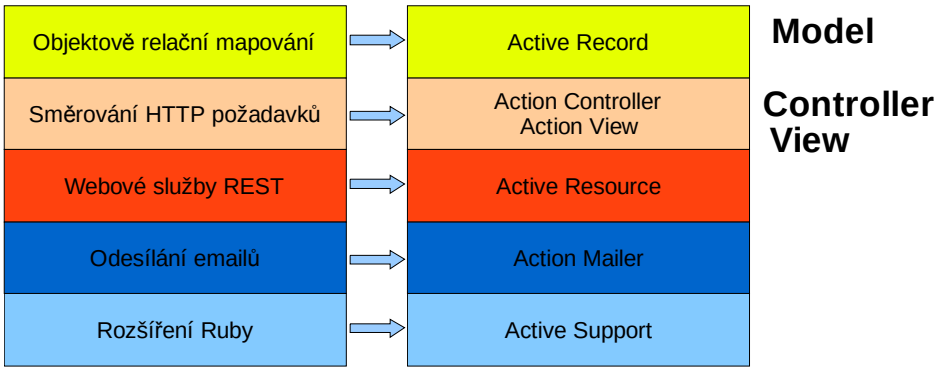
2.2.4 JÁDRO FRAMEWORKU

Samotné jádro frameworku RoR tvoří 5 základních komponent.

Již samotná instalace RoR proběhne tak, že balíčkovací systém Ruby Gems nainstaluje celkem 6 balíčků, 5 základních komponent, které dohromady tvoří 6 balíček – celý framework RoR.



Následující obrázek zachycuje vztah mezi funkcionalitou RoR, jednotlivými balíčky RoR a návrhovým vzorem M-V-C.



V následující části stručně popíšeme každou z těchto komponent tvořících jádro RoR.

2.2.5 ACTIVE RECORD (MODEL)

Active Record je návrhový vzor pro přístup k datům, původně pojmenovaný Martinem Fowlerem v jeho knize *Patterns of Enterprise Application Architecture* (29).

Rails využívají Active Record jako integrovanou ORM (Object-Relational Mapping) vrstvu. **ORM** je anglická zkratka **objektově-relační mapování**. Jedná se o programovací techniku, která má za úkol konverzi dat mezi relačními databázemi a OO programovacími jazyky. ORM knihovny mapují tabulky z databáze na třídy objektů. Tabulky jsou namapovány na třídy, řady na objekty a sloupce na atributy objektů (30).

Jako příklad si vezmeme relační tabulku **people**.

ID	NAME	SURNAME	E-MAIL
1	John	Smith	smith@gmail.com
2	Jan	Novak	novak@seznam.cz

Tradičně lze záznamům tabulky people přistupovat pomocí **SQL**:

```
INSERT INTO people (name, surname, email) VALUES (’John’, ’Smith’,
’smith@gmail.com’);
SELECT * FROM people;
SELECT * FROM people WHERE id = 1;
```

Ekvivalentní **Rails syntaxe** pomocí Active Record vypadá:

```
# ActiveRecord::Base.establish_connection …
Person.create(:name => ‘John‘, :surname => ‘Smith‘, :email =>
‘smith@gmail.com‘)
Person.all
Person.find(1)
Person.find_by_surname(‘Smith‘)
```

ActiveRecord mapuje relační tabulku People na třídu Person. Jednotlivé záznamy (řádky) tabulky budou odpovídat objektům Person (instancím třídy). Sloupec tabulky ID odpovídá atributu třídy Person.ID.

tabuka people

ID	Name	Surname	E-mail
1	John	Smith	smith@hotmail.com
2	Susanna	Reddick	susan@hotmail.com

třída Person

Person
ID
Name
Surname
E-mail

Obrázek 12 - tabulka x třída

ORM mapování v Rails proběhne při vytvoření podtřídy **ActiveRecord::Base**. ActiveRecord zabalí databázové tabulky do objektových tříd.

Dle Rails konvencí jsou databázové tabulky v databázi pojmenovány jako množné číslo modelu třídy (anglicky). Model třídy User tedy standartně nalezneme v DB jako tabulku users.

Jméno třídy	Jméno tabulky
User	users
TaxAgency	tax_agencies
Batch	batches
Person	people
Quantity	quantities
Datum	data

Tabulka 1 - konvence ORM

S mapováním řádků relačních tabulek na atributy objektů souvisí i datové typy. ActiveRecord musí zajistit mapování datových typů SQL na datové typy jazyka Ruby.

SQL typ	Ruby třída
int, integer	Fixnum
decimal, numeric	BigDecimal*
interval, date	Date
clob, blob, text	String
float, double	Float
char, varchar, string	String
datetime, time	Time
boolean	Boolean

Obrázek 13 – datové typy SQL a Ruby

ActiveRecord v Rails podporuje tři základní typy vztahů mezi tabulkami:

one-to-one - asociace zahrnuje vztahy 1:0 a 1:1. Cizí klíč jedné řady jedné tabulky odkazuje na nejvýše jednu řadu tabulky druhé.

one-to-many - lze vytvářet kolekce objektů 1:M. Rodičovský objekt může obsahovat kolekci potomků. Tabulka potomka používá belongs_to k identifikaci svého rodiče.

many-to-many - lze vytvářet vztahy M:N. Uvnitř databáze se pro tento vztah vytvoří asociační tabulka.

V modelech se **asociační vztahy** deklarují direktivami:

belongs-to - specifikuje, že třída **má rodičovský vztah** ke třídě, která tuto direktivu obsahuje. Konvence ActiveRecord říká, že tabulka obsahující cizí klíč patří k tabulce na níž se odkazuje.

has-one - deklaruje, že specifikovaná třída **je potomkem** dané třídy. Tabulka potomka bude obsahovat cizí klíč referující zpětně k tabulce rodičovské. Jelikož se jedná o direktivu, která upravuje asociaci na pozici rodičovského prvku, ActiveRecord přidává i způsob na ošetření referenční integrity dat.

has many - definuje atribut rodičovské třídy, který se chová jako kolekce objektů potomka. Odpovídá vztahu 1:M. K potomkům lze v rodičovské třídě přistupovat jako k poli, lze pole prohledávat, přidávat nové potomky, odebírat, procházet pole, apod.

has and belongs to many - v mnohém se podobá has_many. Stejně jako tato direktiva vytváří atribut, ve kterém je uchovávána kolekce objektů, akorát se zde oproti jednostrannému vztahu 1:M definuje vztah oboustranný M:N.

act_as_list - přidáváme funkcionalitu **seznamu**. Rodič může své potomky v seznamu přemisťovat, přidávat je, odebírat. Seznam je implementován přiřazením čísla pozice každému potomkovi. V tabulce potomka musí existovat sloupec, který toto pořadí zaznamenává (defaultně :position).

act_as_tree - funkcionalita pro organizaci řádků tabulek do hierarchické struktury, tzv. stromu.

ActiveRecord je nejobsáhlejší komponenta RoR, zahrnuje i mnoho dalších funkcí, např. : validace, callback, transakce, zamykání tabulek, migrace, apod. V této práci však není prostor pro popis těchto funkcí.

Pro vývoj aplikace je nejdůležitější především ORM – způsob mapování relačních tabulek na objekty. V dalších kapitolách dojde k praktickému využití podobného principu – pluginu active-ldap, který pobobným způsobem mapuje LDAP záznamy na objekty.

2.2.6 ACTION PACK, ACTION CONTROLLER, ACTION VIEW

Webové aplikace zjednodušeně fungují tak, že z www prohlížeče přijmou požadavek, zpracují ho a pak na něj určitým způsobem odpoví. V Rails se o zpracování příchozích požadavků a generování odchozích odpovědí starají dva balíčky – Action Controller a Action View. Dohromady tyto dva moduly tvoří balíček **Action Pack.**

ACTION CONTROLLER (Controller)

Controller je logické centrum celé aplikace. Koordinuje navzájem interakci mezi uživatelem, view a modelem. Většinu těchto interakcí Rails zvládají automaticky - na pozadí aplikace.

Hlavními akcemi, které controller pro aplikaci zajišťuje, jsou:

Routing

Routing je subsystém, který určuje co udělat s příchozím dotazem. Příchozí dotaz je v Rails aplikaci nejdříve zpracován **routerem (směrovačem),** který je součástí controlleru.

Původně se v Rails (do verze 1.1) standartně používalo mapování URL, které zpracovávalo dotaz následujícím způsobem:

```
http:// www.example.com/people/display/1
map.connect '/:controller/:action/:id'.
@params = {:controller => ‘people‘,:action => ‘display‘,:id => ‘1‘}
```

Z URL (www.example.com/people/display/1) tak Rails aplikace získala informace o Controlleru (people), jeho metodě (display) a parametrech (1).

Od Rails 1.2 je již možno využít mapování požadavků typu RESTful. V této práci a vyvíjené aplikaci však bude REST mapování nebude využito. Zájemce odkazuji na (35).

Odpovědi (Responses)

Jedním z úkolů controlleru je odpověď uživateli. V rails existují 4 základní druhy odpovědí:

1. generování šablon (template) – nejpoužívanější způsob (dle MVC template = view), view převezme data od controlleru a použije je k vygenerování www odpovědi (stránky) do prohlížeče

2. vrácení řetězce přímo do okna prohlížeče – bez zavolání view (používá se např. při chybových hlášeních)
3. žádná odpověď – využití při AJAX požadavcích
4. data – např. při stahování souboru

Controller vždy vygeneruje 1 odpověd na 1 požadavek. Taktéž může existovat více šablon s rozdílnými příponami (.rhtml, .rxml, .rjs).

Cookies (26 str. 434)

Cookies umožňují webovým aplikacím uchovávat data mezi jednotlivými požadavky. Rails schovávají cookies za jednoduché rozhraní. Každý Controller má k dispozici atribut cookies, který je objektem typu hash. Při obdržení požadavku je automaticky objekt cookies inicializován a hodnoty jsou odeslány z prohlížeče do aplikace. Aplikace kdykoli může přidávat nové páry klíč/hodnota do objektu cookies. Ty se pak odešlou prostřednictvím view a prohlížeče uživateli.

```
cookies[:the_time] = Time.now.to_s
```

Sessions (26 str. 437)

Session je podobně jako cookie hash objekt, jehož stav se mezi jednotlivými požadavky nemění. Narozdíl od cookies sessions mohou obsahovat různé objekty. Používají se tedy především pro udržení stavu (např. obsah nákupního košíku) mezi jednotlivými požadavky (procházením stránek internetového obchodu).

Rails sessions mají 2 části:

1. na straně uživatele a jeho prohlížeče – do uživatelova počítače se standartně uloží ve formě cookie 32 hex znakový klíč (s názvem session_id)
2. na straně serveru Rails uchovává data v datovém skladu, který je indexován dle session_id

K dispozici je několik typů datových skladů pro sessions, např.:

- CGI::Session::PStore v souboru formátu PStore
- :active_record_store v databázi
- :memory_store v lokální paměti aplikace

Filtry (26 str. 447)

Filtry umožňují napsat kód v controlleru a nechat ho provést před/po akcích daného controlleru. Takto Rails implementují autentifikaci, logování, apod. Rails podporují 3 typy filtrů: before, after a around. **Before** se provede před metodou Controlleru, **after** po a **around** umožní ohraničení metody z obou stran, lze tak např. změřit, jak dlouho trvá její provedení.

Verifikace (26 str. 453)

Mechanismus verifikace je abstrakce, díky které lze konfigurovat konkrétní omezení detailněji než pomocí filtrů.

Lze tak vyjádřit např. podmínku, aby session obsahovala platného uživatele

```
verify :only => :post_comment,
:session => :user_id,
:add_flash => { :note => "Musíte se p•ihlásit!" },
:redirect_to => :index
```

Metoda post_comment tak musí obsahovat v :session klíč :user_id, pokud ne je přesměrována na index.

Caching (26 str. 455)

Caching umožní Rails aplikaci význačně snížit vytížení serveru a snížit dobu odezvy aplikace. Místo toho, aby se starý obsah generoval pokaždé znovu, principem cachování je **zapamatování vygenerovaného obsahu**. Při dalším požadavku na stejnou stránku, ji můžeme načíst z Cache paměti (což je mnohem rychlejší) než ji znova celou generovat.

Rails využívají 3 přístupy ke Cachingu - page caching, action caching a fragment caching.

ACTION VIEW (View)

Balíček Action view zapouzdřuje veškerou funkcionalitu nutnou pro generování šablon, nejčastěji v html, xml či rjs podobě, zpět uživateli.

Šablony (templates) (26 str. 465)

V Rails je dynamický obsah generován šablonami (templates = views). View je zodpovědné za tvorbu buď části, nebo celé www stránky zobrazené v prohlížeči. Je to vlastně kus HTML kódu, který zobrazuje jak statický text, tak se k němu může připojit i obsah dynamický (vytvořený metodou v controlleru).

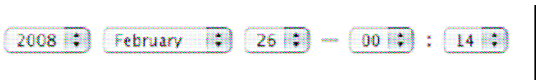
Existují tři druhy šablon:

1. rhtml – vkládá kousky Ruby kódu do HTML kódu view, pomocí nástroje **ERB** (embedded ruby).
2. rxml – umí pomocí Ruby zakládat XML dokumenty
3. rjs – vytváří JavaScript, který se vykoná v prohlížeči, typicky pro www s technologií AJAX

Helpery (26 str. 471)

Helper je modul, který obsahuje metody, které pomáhají view. Metody helperu jsou zaměřené především na výstup. Jejich účelem je generování HTML (nebo XML nebo JavaScript) kódu. Helper rozšiřuje chování šablony. Ke každému controlleru je vygenerován při založení automaticky helper. (UsersController => UsersHelper). Rails obsahují celou řadu Helper-Metod, které jsou dostupné pro všechna views.

Pro formuláře je k dispozici např. helper **datetime_select**, který pomáhá při výběru data.



Obrázek 14 - Helper datetime_select Zdroj: (22 str. 61)]

2.2.7 ACTIVE RESOURCE

Pomocí Rails je velice snadné vybudovat webové služby typu REST, které spolupracují jak s webovými prohlížeči, tak i programovatelným webem. Ve skutečnosti pouhým dodržováním REST principů se zjednodušuje mnoho práce – nemusíme naším klientům sdělovat, jakým způsobem vytvářet, číst, aktualizovat či mazat zdroje. Stačí pouze, když klienti budou správně využívat metody HTTP protokolu (GET, PUT, POST a DELETE).

Jediná věc, kterou tak musíme udělat, je klienty správně **nasměrovat**. Je nutné specifikovat kombinaci uživatelského jména/hesla k přístupu ke službě - k této specifikaci využijeme URL.

```
class Person < ActiveResource::Base
      self.site = „http://username:password@www.example.com:3000/“
end
```

Pomocí ActiveResource pak přistupujeme ke vzdáleným zdrojům stejně, jako pomocí ActiveRecord při přístupu k DB – způsob práce je stejný.

ActiveResource je vlastně **abstraktní vrstva k přístupu ke vzdáleným zdrojům**, podobně jako je např. ActiveRecord abstraktní vrstvou k přístupu k databázi. Komunikace probíhá pomocí XML dokumentů.

2.2.8 ACTION MAILER

ActionMailer je jednoduchá komponenta Rails pro odesílání a příjem elektronické pošty.

Odesílat poštu lze buď před program **Sendmail** nebo pomocí **SMTP** serveru. Třetí možností je **testovací prostředí** pro elektronickou poštu, ve kterém nedochází k odesílání emailů, ale pouze k jejich ukládání do pole.

2.2.9 ACTIVE SUPPORT

ActiveSupport rozšiřuje některé třídy jazyka Ruby o **extra metody.** Je to sada knihoven, které sdílí všechny Rails komponenty. Většinu tohoto modulu využívají Rails interně.

Uveďme si dva příklady.

Pole - o metodu **group_by,** která pole setřídí.

```
users = users.group_by { |user| user.jmeno }
```

Řetězce – o metody **pluralize** a **singularize**, které konvertují jméno do jednotného (množného čísla)

```
puts "user".pluralize                => users
puts "users".singularize        => user
```

Rails framework je velmi obsáhlý a obsahuje značné množství různých komponent. V této kapitole byly popsány klíčové součásti – 5 balíčků, ze kterých se RoR skládá – activerecord, actionpack, activeresource, actionmailer a activesupport. Největší prostor byl věnován popisu komponent nejvíce využitých při vývoji aplikace pro podporu týmové spolupráce – balíčkům activerecord a actionpack, ve kterých je zahrnuta funkcionalita návrhového vzoru M-V-C.

3. KAPITOLA 3 – ANALÝZA A NÁVRH APLIKACE

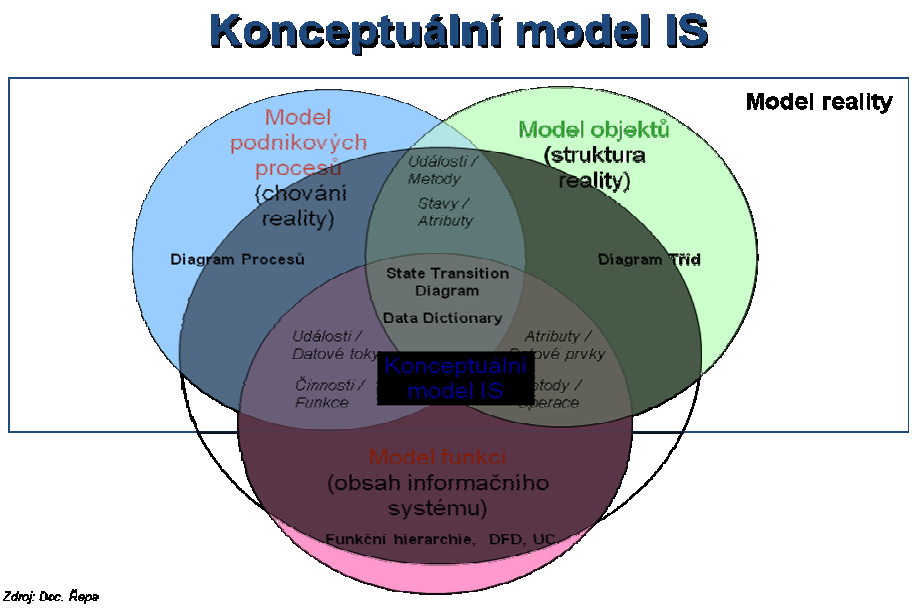
Cíle této kapitoly:	…	analýza aplikace
	…	sestavení analytických modelů

Pro analýzu a návrh systému jsem využil znalosti získané v kurzech analýzy na KIT VŠE Praha. Zájemce o bližší informace odkazuji na publikaci (32).

Výchozí model pro tvorbu IS a formulaci požadavků na ně je **Konceptuální model**. Tento model se skládá ze tří vzájemně propojených modelů (32):

1. modelu podnikových procesů
2. modelu objektů
3. modelu funkcí

Každý z těchto modelů představuje jednu dimenzi návrhu IS. Popisují IS z hlediska jeho chování (procesy), struktury (objekty) a obsahu (funkce).



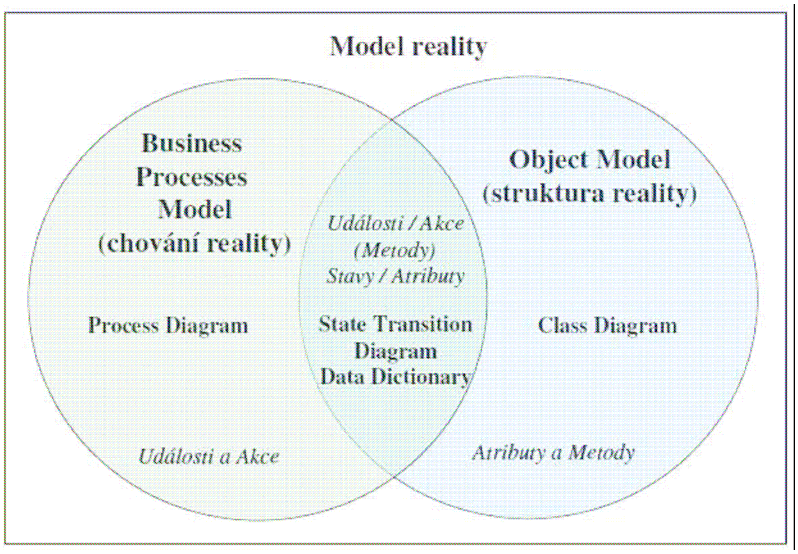
Obrázek 15 - Konceptuální model IS [32]

3.1 PROCESNÍ MODEL

Procesní model je pro IS/ICT zdrojem poznání reality, požadavků na informace a informační podporu

Spolu s objektovým modelem tento model tvoří **model reality** (32).

1. **Model podnikových procesů** modeluje dynamické chování modelu reality (32). Popisuje akce a jejich vzájemné časové návaznosti na událostech a stavech procesů (32).
2. **Model objektů** modeluje statickou strukturu modelu reality, její podstatu nezávislou na konkrétní technologii a implementačním prostředí (32).



Obrázek 16 - Model reality [32]

Proces podpora studentů

Aplikace pro podporu týmové spolupráce je navržena pro podporu jediného procesu - podpora studentů. Každý semestr vypíše KIT VŠE v Praze kurzy programování. Pro každý z těchto kurzů je nutné kromě výuky zajistit jak softwarovou podporu (subversion repositáře, dokuwiki, nastavení přístupových práv, …), tak i další administrativu (rozdělení do skupin a týmů, přiřazení garantů, apod.).

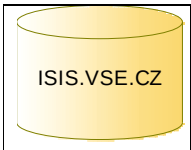
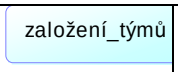
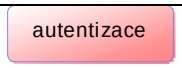
V současnosti se většina administrativních úkonů provádí ručně, vyvitnutá aplikace by měla řadu těchto úkonů automatizovat. Proces lze popsat následující tabulkou:

Tabulka 2 - Proces podpora studentů

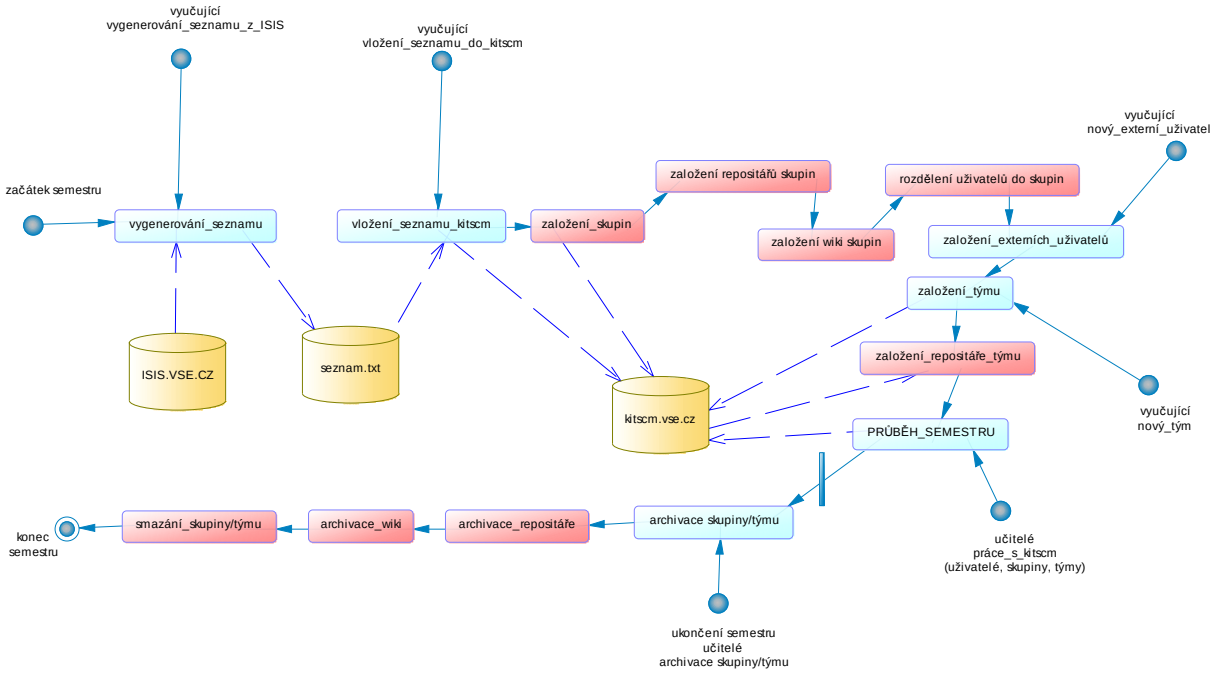
Id procesu	P1
Název procesu	Podpora studentů
Strategický cíl	Informační podpora kurzů programování
Událost	Nový semestr – začátek kurzů
Produkt / služba	Aplikace pro podporu kurzů programování
Účel	Podpora kurzů, automatizace administrace,
Vlastník procesu	KIT VŠE Praha
Zákazník procesu	Vyučující, studenti kurzů programování
Oblasti pro zlepšení / problémy	Automatizace procesu, usnadnění, zrychlení a zefektivnění práce
Metriky	Uspořený čas, chyby systému
Podmínky	
Informační systémy / aplikace	ISIS VŠE Praha, Subversion, Dokuwiki, OpenLdap
Dokumenty	

Pro modelování jsem zvolil metodiku BPMN (33) a jako CASE nástroj Power Designer 12.1 (34).

Tabulka 3 - použitá notace

					
Úložiště dat	Činnost procesu	Automatizovaná činnost	Počáteční událost	Koncový stav	čekání

Model procesu



Obrázek 17 - model procesu Podpora výuky

Popis procesu

Celý proces se opakuje každý semestr. Předchází mu vypsání kurzů programování a registrace studentů na tyto kurzy (v systému ISIS). **Počáteční událostí** je zahájení výuky (první výukový týden semestru).

Vyučující (garant kurzu) **vygeneruje do textového soboru** ze systému ISIS seznam studentů zapsaných na příslušný kurz.

Tento seznam garant kurzu importuje do aplikace pro podporu týmové spolupráce na serveru kitscm.vse.cz. Na základě tohoto seznamu **aplikace automaticky na serveru kitscm.vse.cz:**

1. založí výukové skupiny (v LDAP serveru VŠE)
2. založí skupinové repositáře v subversion a nastaví příslušná práva
3. založí skupinové wiki a nastaví příslušná práva
4. rozřadí uživatele do výukových skupin
5. založí studentské repositáře v subversion (pro každého studenta jeden) a nastaví příslušná práva.

Vygenerovaný soubor ze systému ISIS bude možné importovat do aplikace ne pouze jednou – na začátku semestru, ale i vícekrát – např. 2 týden výuky, kdy dojde k zapsání dalších studentů na kurzy.

Aplikace tedy musí obsahovat vestavěnou **ochranu proti přepsání** existujících uživatelských účtů a repositářů.

V prvních týdnech výuky kurzů programování seznámí vyučující studenty se serverem kitscm.vse.cz. Studenti se rozdělí do týmů. Po rozřazení studentů do týmů založí vyučující v aplikaci týmy a přidá do nich studenty. Po založení týmu aplikace automaticky založí subversion repositář týmu a nastaví příslušná práva.

V průběhu semestru zajistí aplikace vyučujícím přístup k LDAP záznamům:

1. externích uživatelů
2. výukových skupin
3. uživatelských týmů

Učitelé tak budou moci dodatečně přidávat, editovat, mazat externí uživatele, týmy a skupiny.

Při **ukončení semestru** garanti kurzů provedou archivaci každé skupiny/týmu. Při archivaci skupiny/týmu aplikace přesune adresář s repositářem do adresáře pro archivace, dále se stejným způsobem archivuje wiki skupiny/týmu, pak aplikace skupinu/tým smaže. Externí uživatelé v aplikaci zůstanou a mazat se nebudou.

3.2 OBJEKTOVÝ MODEL

Druhým analytickým modelem je objektový model. Tímto modelem se modeluje **statická struktura reality** navrhovaného IS. Jsou zde zachyceny typy objektů a jejich základní vlastnosti a vztahy. Hlavním cílem tohoto modelu je pochopit realitu, její strukturu, pojmy a složitost.

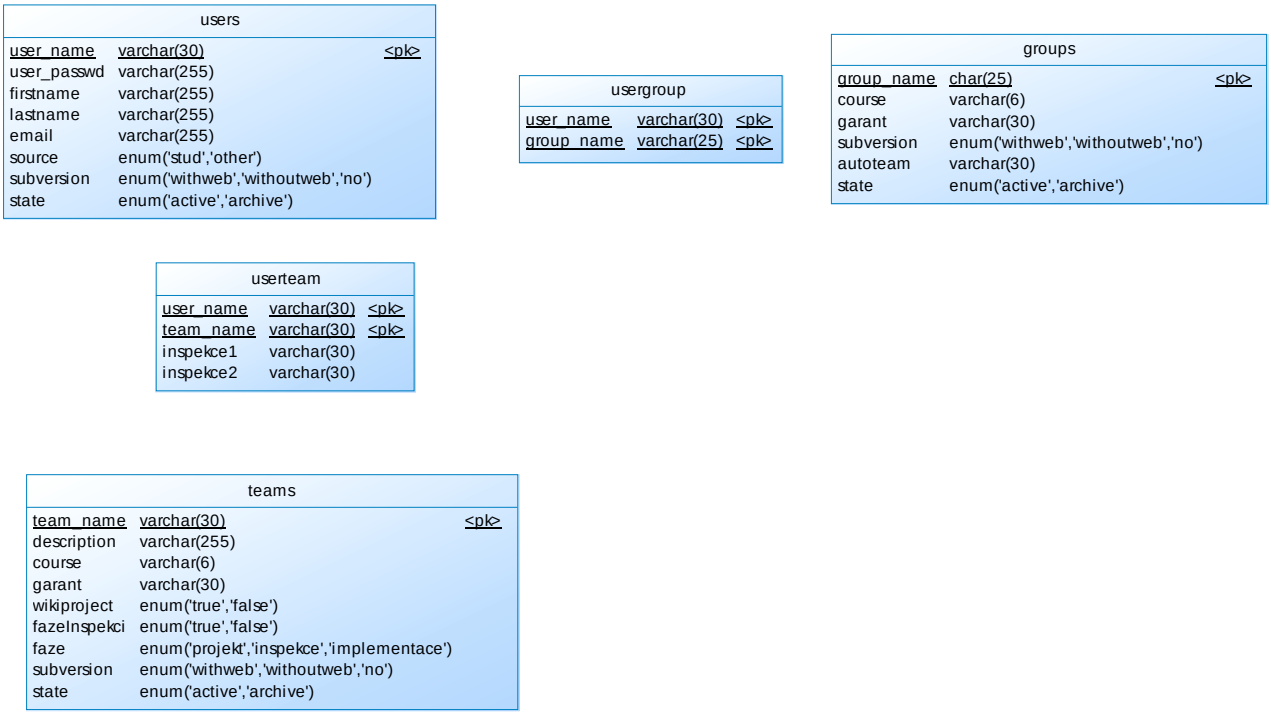
Rozhodl jsem se objektový model **nesestavovat** . Sestavování objektového v tomto případě – aplikace pro podporu týmové spolupráce nemá smysl**.**

Původně vyučující kurzů programování využívali relační databázi MySQL. Struktura reality dat uložených na serveru kitscm.vse.cz, její objekty a vztahy – vše je známo (obrázek 51) – datový model.

Jedním z hlavních požadavků na vývoj aplikace (Tabulka 4 - 6.5) je přímé propojení aplikace s adresářovým OpenLDAP serverem VŠE, **bez použití databáze**. Je tedy nutné relační databázi transformovat do struktury adresářového serveru. To je hlavní účel této části analýzy.

Původní stav systému

Ve třech hlavních tabulkách (users, groups, teams) databáze MySQL byly uchovávány informace o jednotlivých uživatelích, výukových skupinách a projektových týmech. Dále v databázi byly dvě asociační tabulky (userteam, usergroup), které sloužily pro zařazení uživatelů do pracovních skupin a projektových týmů.



Obrázek 18 - původní datový model

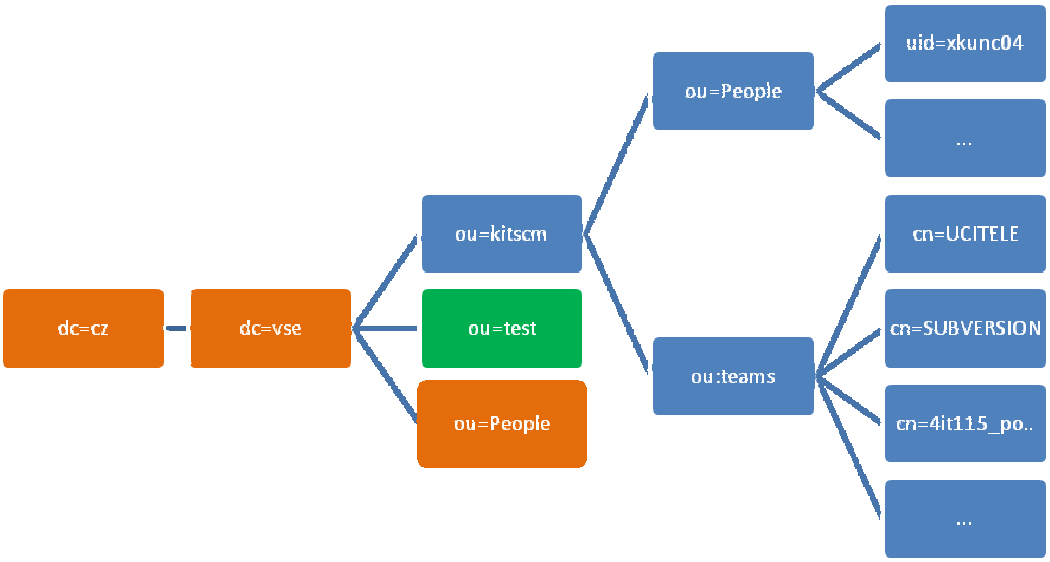
LDAP adresář

Původní LDAP adresář VŠE tvořila doména dc=vse, dc=cz s ou=People obsahující uživatelské účty všech uživatelů VŠE a ou=test, určená pro testování.

Obrázek 19 - Původní LDAP struktura

Nová struktura LDAP vznikla přidáním ou=kitscm, obsahující:

1. ou=People – zde budou uloženi externí uživatelé (nestudenti VŠE) využívající tento systém
2. ou=teams – zde budou uloženy jednotlivé výukové skupiny a týmy, dále 2 speciální skupiny: cn=SUBVERSION (uživatelé této skupiny budou mít repositář v SUBVERSION) a cn=UCITELE (seznam učitelů)



Obrázek 20 – Nová struktura LDAP

Uživatelé

Pro evidenci uživatelů se místo původní tabulky:

users		
user_name	varchar(30)	<pk>
user_passwd	varchar(255)	
firstname	varchar(255)	
lastname	varchar(255)	
email	varchar(255)	
source	enum('stud','other')	
subversion	enum('withweb','withoutweb','no')	
state	enum('active','archive')	

Obrázek 21 - Původní tabulka users

využijí 2 adresářové stromy:

1. adresářový strom **ou=People,dc=vse,dc=cz** , ve kterém jsou uloženi všichni uživatelé VŠE. Nad tímto stromem nebude aplikace provádět žádné změny, pouze ověřovat existenci uživatele a jeho autentizaci heslem.
2. adresářový strom **ou=People,ou=kitscm,dc=vse,dc=cz ,** ve kterém budou evidováni **externí uživatelé systému** (nečlenové VŠE). Učitelé budou moci pomocí aplikace tyto uživatele manuálně přidávat, odebírat a aktualizovat jejich údaje.

Organizační jednotka **ou=People,ou=kitscm,dc=vse,dc=cz** bude mít následující atributy:

Attribute Description	Value
objectClass	*domainRelatedObject (auxiliary)*
objectClass	*organizationalUnit (structural)*
objectClass	*top (abstract)*
associatedDomain	kitscm
ou	People

Obrázek 22 – Atributy organizační jednotky dn:ou=People,ou=kitscm,dc=vse,dc=cz

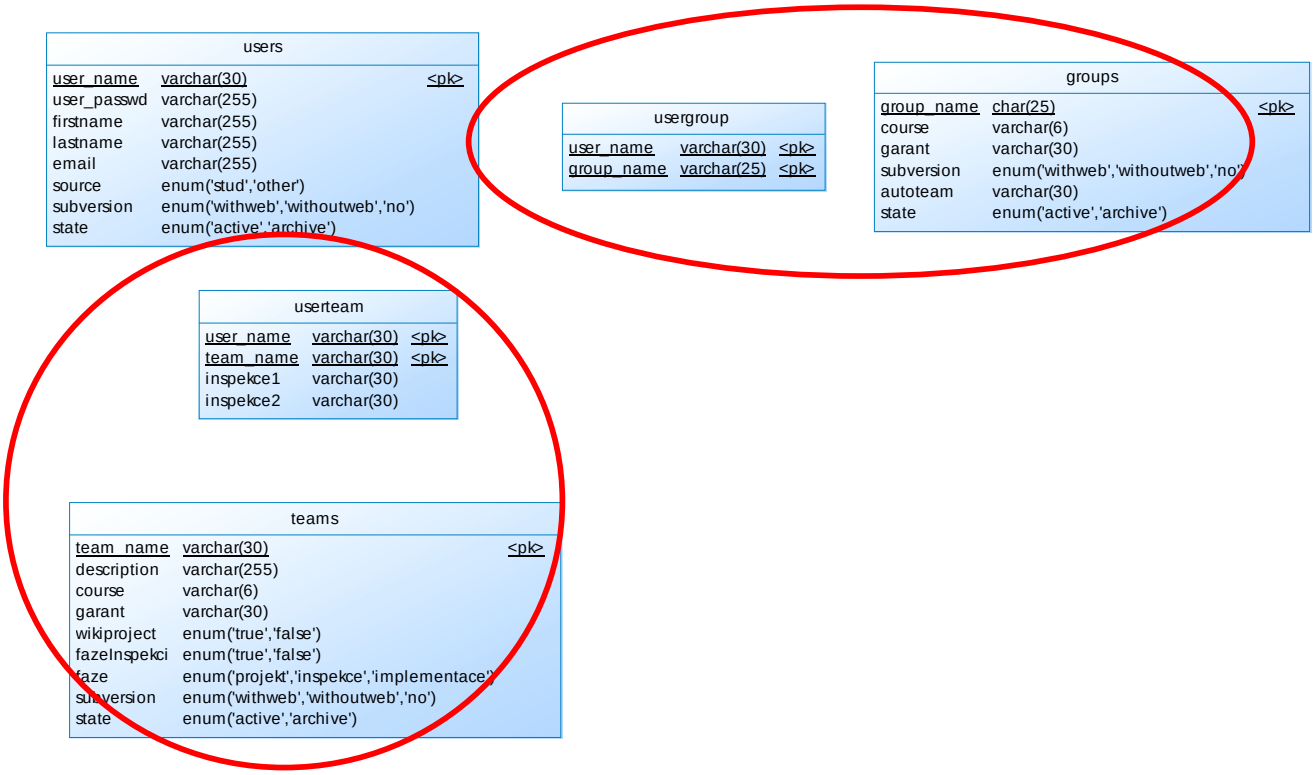
Externí uživatelé uložení v této organizační jednotce budou mít následující atributy:

Attribute Description	Value
objectClass	*inetOrgPerson (structural)*
objectClass	*organizationalPerson (structural)*
objectClass	*person (structural)*
objectClass	*posixAccount (auxiliary)*
objectClass	*top (abstract)*
cn	Petr
gidNumber	0
homeDirectory	/home/xhofp10/
sn	Hofmann
uid	xhofp10
uidNumber	0
userPassword	SHA hashed password

Obrázek 23 – Atributy záznamu externího uživatele dn:uid=xhofp10,ou=People,ou=kitscm,dc=vse,dc=cz

Výukové skupiny a týmové projekty

V původní databázi byly výukové skupiny a týmové projekty odděleny ve dvou tabulkách - teams a groups. Příslušnost uživatele k dané skupině či týmu se evidovala ve dvou asociačních tabulkách userteam a usergroup.



Obrázek 24 - Původní stav týmů a skupin

V novém designu LDAP serveru se tyto tabulky sloučí do jednoho LDAP stromu **ou=teams,ou=kitscm,dc=vse,dc=cz**.

Attribute Description	Value
objectClass	***groupOfNames (structural)***
objectClass	***top (abstract)***
objectClass	***x-cz-vse-kitscm-team2 (structural)***
cn	**4it251_po_12:45_p107**
member	**uid=xbaum09,ou=People,dc=vse,dc=cz**
member	**uid=xbobp00,ou=People,dc=vse,dc=cz**
description	Skupina 4it251_po_12:45_p107 založena: Mon Oct 26 21:08:41 +0100 2009
vseCourse	4IT251
vseGarant	uid=pavlicek,ou=People,ou=test,dc=vse,dc=cz
vseSubversion	TRUE
vseSubversionRO	FALSE
vseWikiproject	FALSE

Obrázek 25 - Atributy skupin a týmů dn:ou=teams,ou=kitscm,dc=vse,dc=cz

Odlišovat skupiny od týmů bude jejich název – atribut cn.

Výukové skupiny budou vždy označeny ve formátu **předmět_den_hodina_učebna** (např. 4it115_po_09:15_p109).

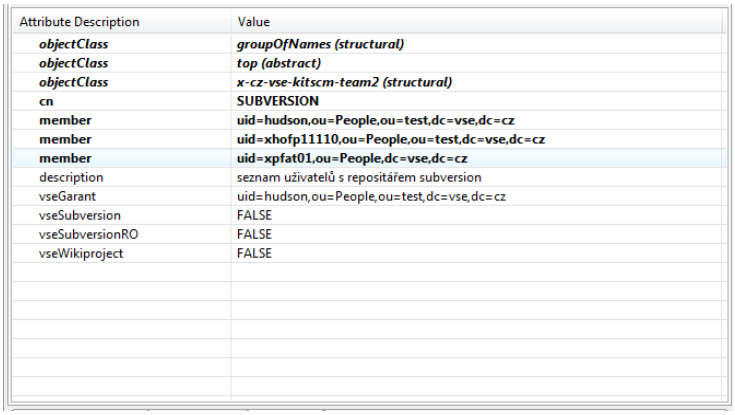
Týmy budou vždy označeny **názvem týmu**, který si první hodinu výuku zvolí sami jeho členové (např. tym_prazaci, …).

Jednotliví členové týmu/skupiny budou uloženi jako atribut member příslušného týmu/skupiny.

Speciální skupiny – UCITELE a SUBVERSION

Součástí ou=teams,ou=kitscm,dc=vse,dc=cz jsou i dvě speciální skupiny:

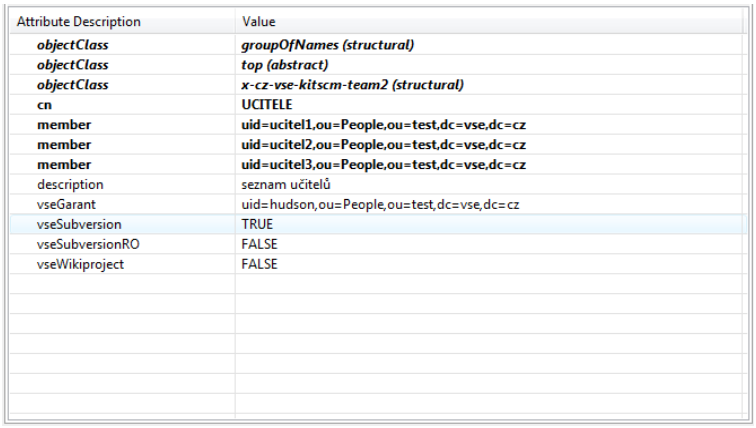
Skupina **SUBVERSION** eviduje, kdo z uživatelů má uživatelský repositář v Subversion. Je tedy seznamem uživatelů s vlastním uživatelský repositářem Subversion. Pokud do ní přidáme uživatele (member), automaticky se mu založí vlastní repositář Subversion a nastaví práva.



Attribute Description	Value
objectClass	***groupOfNames (structural)***
objectClass	***top (abstract)***
objectClass	***x-cz-vse-kitscm-team2 (structural)***
cn	**SUBVERSION**
member	**uid=hudson,ou=People,ou=test,dc=vse,dc=cz**
member	**uid=xhofp11110,ou=People,ou=test,dc=vse,dc=cz**
member	**uid=xpfat01,ou=People,dc=vse,dc=cz**
description	seznam uživatelů s repositářem subversion
vseGarant	uid=hudson,ou=People,ou=test,dc=vse,dc=cz
vseSubversion	FALSE
vseSubversionRO	FALSE
vseWikiproject	FALSE

Obrázek 26 - Atributy dn:cn=SUBVERSION,ou=teams,ou=kitscm,dc=vse,dc=cz

Ve skupině **UCITELE** se evidují učitelé kurzů (member). Při zakládání týmu/skupiny je nutné určit garanta týmu/skupiny. Garantem může být pouze učitel evidovaný v této skupině.



Attribute Description	Value
objectClass	***groupOfNames (structural)***
objectClass	***top (abstract)***
objectClass	***x-cz-vse-kitscm-team2 (structural)***
cn	**UCITELE**
member	**uid=ucitel1,ou=People,ou=test,dc=vse,dc=cz**
member	**uid=ucitel2,ou=People,ou=test,dc=vse,dc=cz**
member	**uid=ucitel3,ou=People,ou=test,dc=vse,dc=cz**
description	seznam učitelů
vseGarant	uid=hudson,ou=People,ou=test,dc=vse,dc=cz
vseSubversion	TRUE
vseSubversionRO	FALSE
vseWikiproject	FALSE

Obrázek 27 - Atributy dn:cn=UCITELE,ou=teams,ou=kitscm,dc=vse,dc=cz

Pro týmy/skupiny bylo vytvořeno speciální OPENLDAP schema - kitscm.schema (příloha 1).

3.3 FUNKČNÍ MODELOVÁNÍ

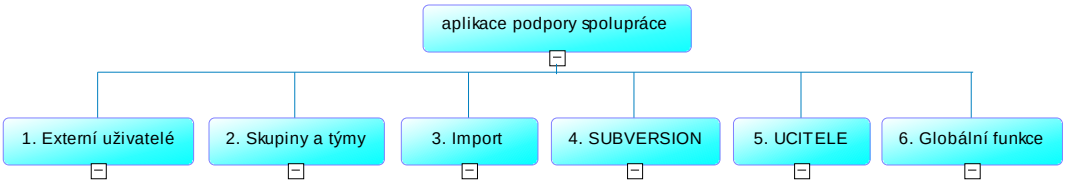
Třetí částí analýzy je funkční modelování. **Funkce** je prvkem chování systému a vyjadřuje **CO** má systém dělat, bez důrazu na to JAK (32). Je charakterizována svými vstupy a výstupy, nikoliv postupem zpracování, kterým je naopak charakteristický proces (32).

Souhrn funkcí IS určuje jeho funkcionalitu (32). Popis funkcionality je důležitý zejména při analýze požadavků daného IS, zákazník resp. uživatel při funkčním modelování upřesňuje představy o systému - co má systém dělat, co má poskytovat (32).

Hierarchie funkcí IS

Na **celkovou funkcionalitu** systému dává pohled **model hierarchie funkcí** (32). Jedná se o statický pohled na IS (32). Popisuje hierarchický rozpad funkcí IS (32). Funkce vyšší úrovně jsou pojmenováním určité množiny logicky souvisejících funkcí podřízené úrovně (32). Na následujícím

modelu je zachycená pouze první úroveň funkcionality, podúrovně jsou uvedeny detailně v popisu funkcí.



Obrázek 28 - Hierarchie funkcí – první úroveň

Popis funkcí

V popisu funkcí se jednotlivě popíší elementární funkce spadající do funkčních oblasti zachycených v modelu hierarchie funkcí, které mají uživatelé aplikace k dispozici. U každé funkce je stručný popis funkcionality (požadavku na aplikaci).

Tabulka 4 - Popis funkcí aplikace

1	**Externí uživatelé**	
1.1	Správa externích uživatelů	Aplikace bude umět spravovat externí uživatele, tzn. zakládat, zobrazovat, editovat a mazat.
1.2	Generování a šifrování hesla	Při založení nového uživatele se heslo uživatele vygeneruje automaticky a zašifruje se pomocí SHA1.
1.3	Změna hesla	Při změně hesla (provádí uživatel) se vygeneruje heslo nové a zašifruje se pomocí SHA1.
2	**Výukové skupiny a týmy**	
2.1	Správa skupin a týmů	Aplikace bude umět spravovat výukové skupiny a týmy, tzn. zakládat, zobrazovat, editovat a mazat. Dále pak aplikace bude umět do skupin/týmů přidávat a odebírat uživatele a to jak uživatele interní – studenty VŠE evidované v systému ISIS, tak i uživatele externí – evidované v aplikaci.
2.2	Založení repositáře skupiny / týmu v subversion	Při založení/editaci skupiny/týmu v aplikaci bude možné zapnout/vypnout automatické založení repositáře skupiny/týmu + automatické nastavení příslušných práv repositáře.
2.3	Přístupová práva skupin/týmů	V aplikaci bude možné při založení/editaci skupiny/týmu všem členům nastavovat dva druhy přístupových práv (read only, read write).
2.4	Wiki stránky skupiny/ týmu	Aplikace bude umět založit wiki stránky skupiny/týmu a nastavit příslušná práva.
2.5	Nový člen - subversion	Při přidání nového člena do skupiny/týmu bude možné zapnout/vypnout i automatické přiřazení tohoto člena do skupiny SUBVERSION.
2.6	Archivace repositářů a wiki stránek	Aplikace bude umět archivovat skupinu/tým. Při archivaci se přesune repositář a wiki skupiny/týmu do archivu, smaže se skupina/tým, externí uživatelé v aplikaci zůstávají a nemažou se.
3	**Import**	

3.1	Import vygenerovaného souboru ze systému ISIS	Aplikace bude umět načíst vygenerovaný soubor .txt ze systému ISIS, automaticky dle něho založit výukové skupiny, skupinové repositáře a nastavit příslušná práva repositářů. Dále pak rozřadit uživatele do výukových skupin.
3.2	Ochrana proti přepsání	Při opakovaném importu dat – souboru ze systému ISIS a automatickém zakládání skupin, skupinových repositářů a uživatelských repositářů je nutné zajistit ochranu proti přepsání existujících uživatelských skupin, repositářů a wiki stránek.
4	**SUBVERSION**	
4.1	Skupina SUBVERSION	Aplikace bude ve speciální skupině SUBVERSION, která nelze smazat, evidovat subversion repositáře uživatelů – interních i externích. Všichni členové této skupiny budou mít založený repositář. Při přidání člena se automaticky založí členovi repositář a nastaví se mu příslušná práva. Při odebrání člena se členovi pouze zruší práva na repositář, repositář se nesmaže.
5	**UCITELE**	
5.1	Skupina UCITELE	Aplikace bude ve speciální skupině UCITELE evidovat učitele kurzů. Tato skupina nelze smazat. Garant skupiny/týmu musí být člen této skupiny.
6	**Globální funkce**	
6.1	Čeština	Aplikace bude pouze v češtině
6.2	Bezpečnost	Přenos dat bude šifrován pomocí SSL, hesla uložená v LDAP budou šifrována pomocí SHA1.
6.3	Nápověda	V aplikaci bude nápověda, pro každou stránku zvlášť.
6.4	Autentizace uživatelů	Všichni uživatelé (externí, interní - ISIS) budou autentizováni v LDAP serveru VŠE.
6.5	LDAP	Celá aplikace NEbude pracovat s databází, ale přímo s LDAP, všechna data budou uložena v LDAP.

4. KAPITOLA 4 – PROVOZNÍ ARCHITEKTURA

Cíle této kapitoly:

... seznámit s provozní architekturou
… architektura operačního systému
… architektura ruby, gems, rails
… architektura dalších aplikací

V této kapitole si popíšeme provozní architekturu, na které aplikace pro spolupráci pracuje. Porozumění této architektuře je prvním krokem jak pro vývoj, tak později i pro nasazení aplikace.

Aplikace pro podporu týmové spolupráce spolupracuje s celou řadou dalších aplikací. Na straně jedné stojí jádro aplikace vytvořené v Ruby on Rails. S aplikací nejtěsněji spojenou další aplikací je OpenLDAP server, který se využívá místo databáze pro uchování dat uživatelů, skupin a týmů. Pro propojení se využívají speciální knihovny a pluginy jak pro programovací jazyk Ruby, tak i framework RoR (Ruby/LDAP, ruby-net-ldap, ruby-activeldap).

Dále aplikace spolupracuje především s Subversion a Dokuwiki, opět jsou zde nutné speciální knihovny a pluginy pro propojení s Ruby a RoR.

4.1 OPERAČNÍ SYSTÉM

Základem provozní architektury je **OS Linux**. Veškeré součásti, se kterými aplikace spolupracuje (Subversion, Apache, Dokuwiki, apod.) jsou open source.

Zvolenou distribucí je **Ubuntu – 8.10 Intrepid server 32bit,** především pro svoji jednoduchost a přívětivost. Samozřejmě celou architekturu je možné implementovat i na ostatních linuxových distribucích. Například server kitscm.vse.cz využívá linuxovou distribuci Gentoo.

V této práci není prostor pro detailní popis instalace OS Linux, čtenáře odkazuji na dokumentaci příslušné linuxové distribuce – v mém případě (www.ubuntu.cz), kde by měli naleznout návody.

V Ubuntu je nejjednodušší způsob již při instalaci OS označit kolekce: LAMP server a OpenSSH.

Obrázek 29 - instalace Ubuntu (výběr programů)

Aktualizace systému

Po instalaci OS je nutné provést jeho aktualizaci.

```
sudo apt-get update
sudo apt-get upgrade
```

4.2 RUBY

Framework RoR i balíčkovací systém Rubygems jsou naprogramovány v programovacím jazyce Ruby (kapitola 2.1). Prvním krokem je tedy instalace programovacího jazyka Ruby a potřebných knihoven.

```
sudo apt-get install ruby irb rdoc ri

sudo apt-get install libyaml-ruby libzlib-ruby libopenssl-ruby
ruby1.8-dev build-essential libmysql-ruby
```

4.3 RUBY GEMS

Ruby gems jsou balíčkovací systém pro programovací jazyk Ruby. Způsobů instalace existuje několik (pomocí apt-get, download). Oficiální doporučený způsob je stáhnutí aktuální verze balíčku z RubyForge (http://rubyforge.org/frs/?group_id=126) a jeho instalace v domácím adresáři uživatele.

```
wget http://rubyforge.org/frs/download.php/60718/rubygems-1.3.5.tgz

tar xvzf rubygems-1.3.5.tgz

cd rubygems-1.3.5

sudo ruby setup.rb
```

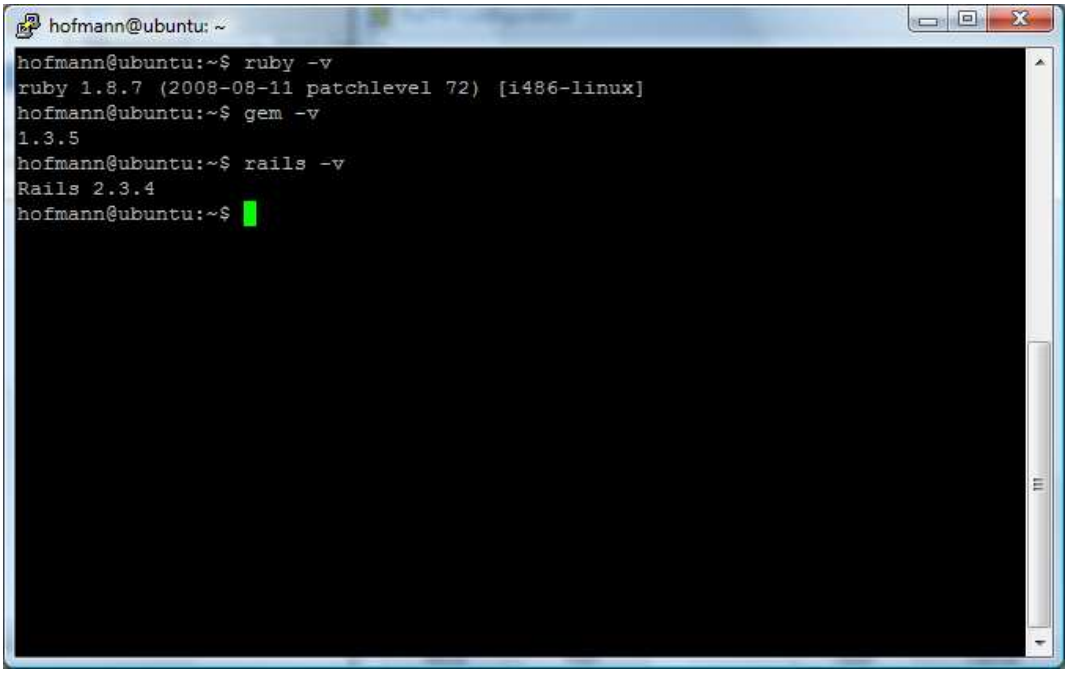
V Ubuntu je nutné ještě vytvořit symbolický link.

```
ln –s /home/hofmann/rubygems-1.3.5/bin/gem /usr/bin/gem
```

4.4 RUBY ON RAILS

Framework Ruby on Rails je vlastně balíčkem Rubygems (kapitola 2.2.3), RoR se instalují jako gem.

```
sudo gem install rails
```



Obrázek 30 - kontrola ruby,rubygems,rails

4.5 DODATEČNÉ BALÍČKY RUBYGEMS

Dále je nutné instalovat další potřebné balíčky.

Mongrel

Mongrel je webový server (podobně jako Apache) pro běh Ruby on Rails aplikací, standartně je součástí Rails webový server Webrick. Při vývoji aplikace bude Mongrel použitý pouze pro vývoj a testování.

```
sudo gem install mongrel
```

MySQL

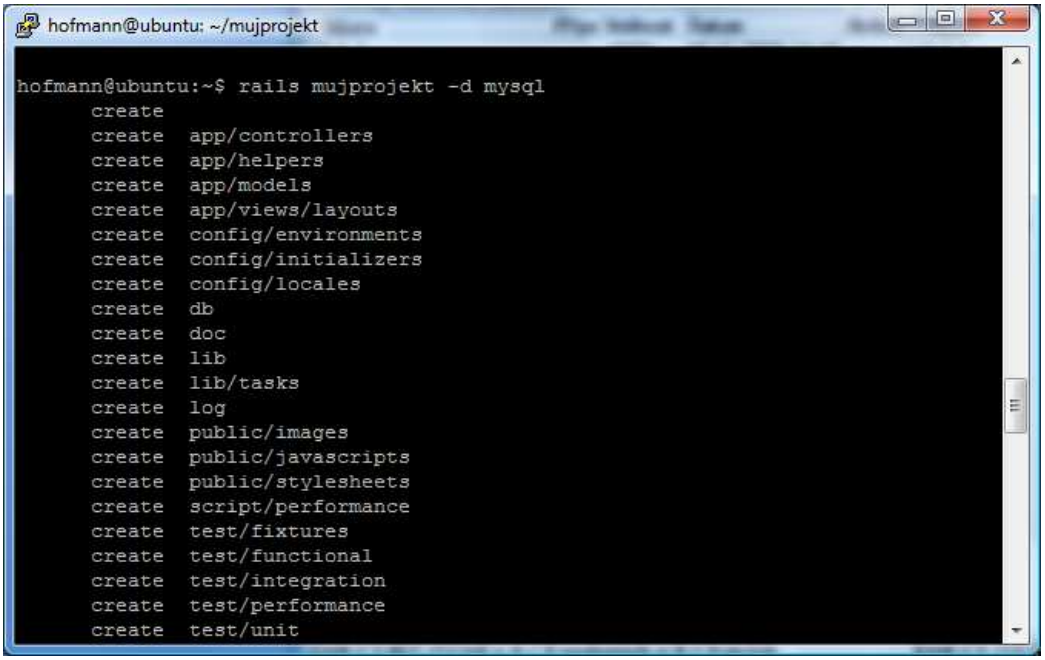
MySQL je open-source relační databáze, pro propojení s MySQL je nutné doinstalovat.

```
sudo apt-get install libmysqlclient-dev
sudo gem install  mysql
```

4.6 TEST ARCHITEKTURY RAILS

Celou Rails architekturu je vhodné předem otestovat založením testovací aplikace. Parametr –d (database) specifikuje využití databáze MySQL (standartně je přednastavena databáze SQLite)

```
rails mujprojekt –d mysql
```



Obrázek 31 - založení Rails projektu s MySQL

Generátor rails vygeneruje celou kostru aplikace (kapitola 2.2.2). Pro Rails aplikaci se musí založit testovací databáze mujprojekt_test.

```
mysqladmin create mujprojekt_development –u root -p
```

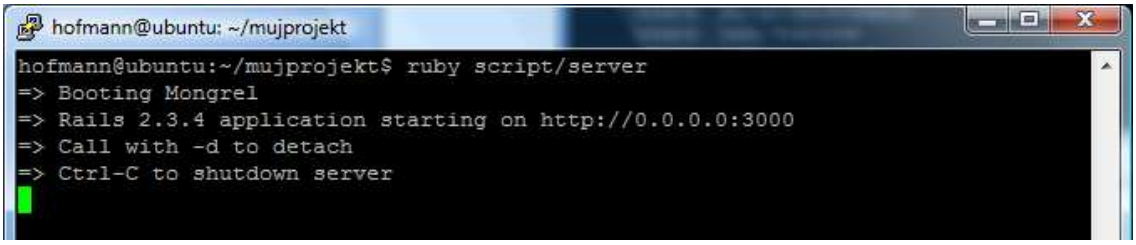
V souboru **mujprojekt/config/database.yml** se nastavují přihlašovací údaje do databáze.

```
development:
```

```
  adapter: mysql
  encoding: utf8
  reconnect: false
  database: mujprojekt_development
  pool: 5
  username: root
  password: heslo
  socket: /var/run/mysqld/mysqld.sock
```

Pak stačí spustit v adresáři aplikace webový server Mongrel, standartně běžící na portu 3000.

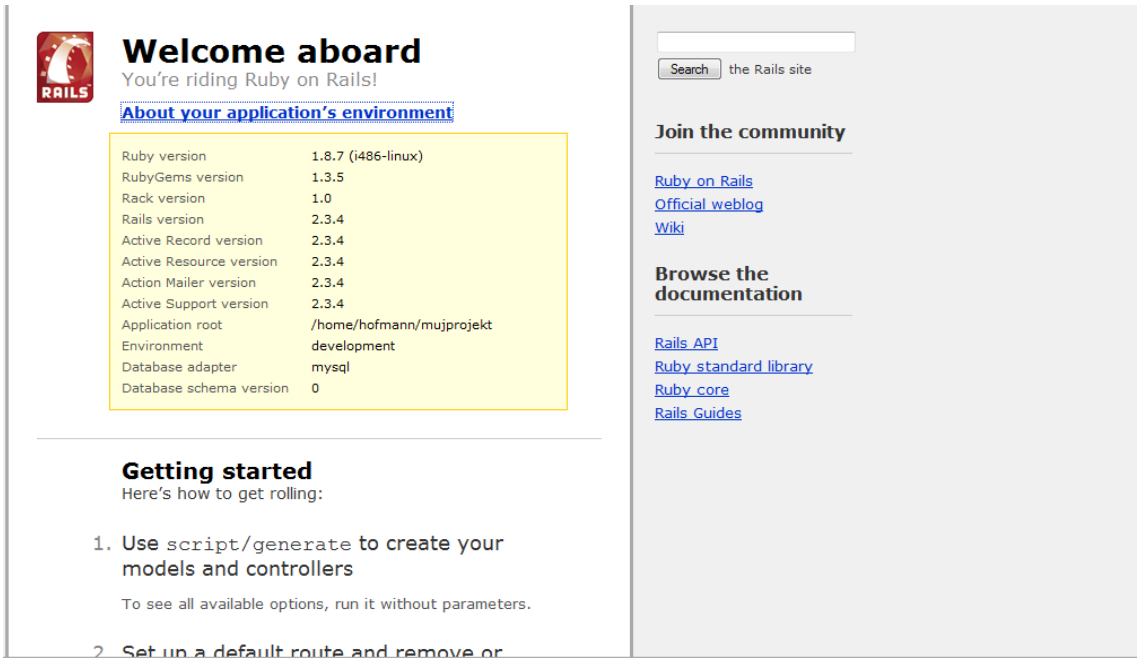
```
ruby script/server
```



Obrázek 32 - spuštění webového serveru Mongrel

A v internetovém prohlížeči otestovat stav celého Rails prostředí.

```
http://localhost:3000
```



Obrázek 33 - test Rails prostředí

Po úspěšném testu Rails prostředí je možné nainstalovat další aplikace (openldap, dokuwiki, subversion, apod.), se kterými aplikace spolupracuje, plus zásuvné moduly – pro propojení a možností ovládání z RoR

4.7 OPENLDAP

Aplikace místo databáze veškerá data ukládá do adresářového serveru OpenLDAP. Vysoká škola ekonomická v Praze používá pro autentizaci uživatelů svůj vlastní OpenLDAP server. Provádět ale na tomto běžícím serveru jakékoli dotazy spojené s vývojem a testováním nové aplikace by bylo příliš riskantní. Z tohoto důvodu je součástí provozní architektury aplikace i OpenLDAP server, jehož havárie nenese žádná rizika. Aplikace se vyvine a otestuje nad daty uloženými v tomto OpenLDAP serveru. OpenLDAP se instaluje příkazem.

```
sudo apt-get install slapd ldap-utils
```

Během instalace je nutné nastavit administrátorské heslo a LDAP server nakonfigurovat. Nejdříve se LDAP server musí zastavit.

```
sudo /etc/init.d/slapd stop
```

Musí se smazat všechny soubory v adresáři **/var/lib/ldap/.** Pak se spustí příkaz pro rekonfiguraci LDAP serveru.

```
sudo dpkg-reconfigure slapd
```

Pro aplikaci se nastaví následující údaje.

```
Doména:                     vse.cz
Název Organizace:           vse
Databázový backend:         HDB
Smazat  starou databázi:    ano
P•esunout databázi:         ne
Heslo:                      heslo
LDAPv2:                     ne
```

A otestuje se funkcionalita LDAP serveru.

```
ldapsearch -xLLL -b "dc=vse,dc=cz"
```

PHPLDAPadmin

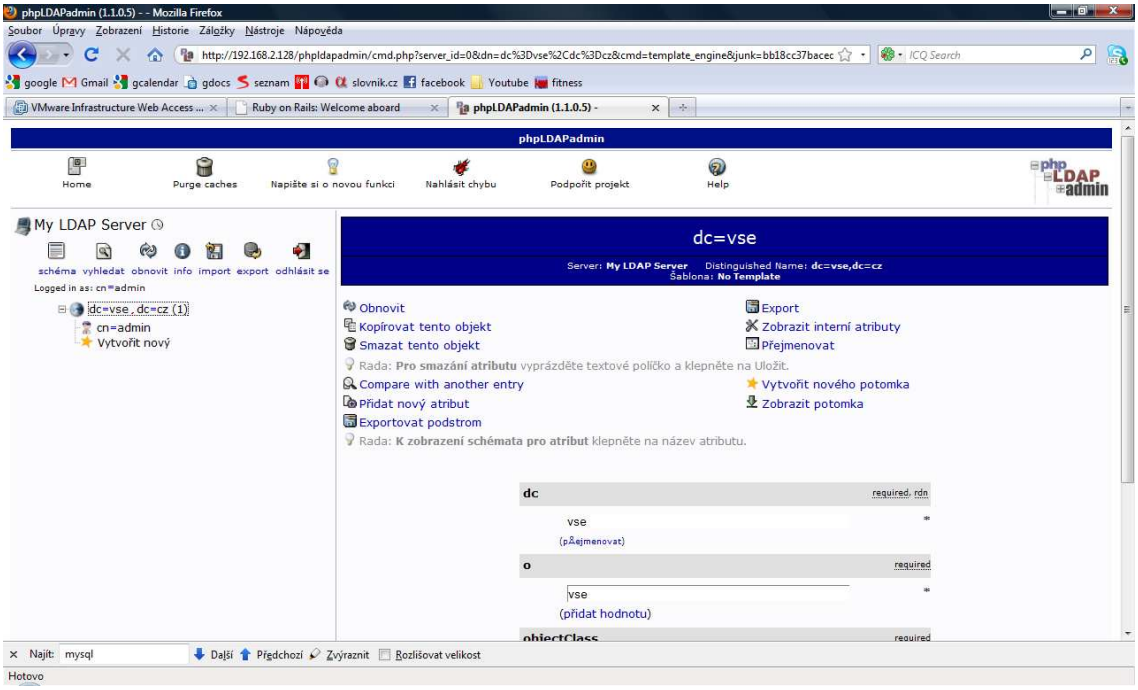
Pro jednoduchý přístup a správu LDAP se nainstaluje program phpldapadmin.

```
sudo apt-get install phpldapadmin
```

V souboru /etc/phpldapadmin/config.php se nakonfiguruje base DN a DN uživatele LDAP.

```
$ldapservers->SetValue($i,'server','base',array('dc=vse,dc=cz'));

$ldapservers->SetValue($i,'login','dn','cn=admin,dc=vse,dc=cz');
```

PHPLdapAdmin se otestuje v internetovém prohlížeči http://server/phpldapadmin . Přihlašovací DN je cn=admin,dc=vse,dc=cz a heslo kitscm (nastavilo se při rekonfiguraci LDAP). Programem phpldapadmin lze velmi rychle a snadno administrovat LDAP strom.



Obrázek 34- phpldapadmin

Pokud se při spuštění zobrazí chybová hláška memory limit – musí se v souboru `/etc/php5/apache2/php.ini` změnit parametr memory_limit z původních 16M například na 64M a restartovat webový server `sudo /etc/init.d/apache2 restart`.

Pro správu LDAP stromu lze použít phpldapadmin, nebo lze manuálně importovat do LDAP stromu soubory .ldif.

```
ldapadd -x -D cn=admin,dc=vse,dc=cz -W -f soubor.ldif
```

4.8 LDAP PODPORA V RUBY A RUBY ON RAILS

Pro propojení jazyka Ruby a Rails aplikací s LDAP serverem se využije několik různých zásuvných modulů, toto je již ale poměrně komplikovaná instalace a nastavení.

Ruby/LDAP

Nejdříve se instaluje knihovna Ruby/LDAP s potřebnými knihovnami.

```
sudo apt-get install libldap2-dev
sudo apt-get install libssl-dev

wget http://downloads.sourceforge.net/ruby-ldap/ruby-ldap-
0.9.7.tar.gz?modtime=1155098708&big_mirror=0
tar xvzf ruby-ldap-0.9.7.tar.gz
cd cd ruby-ldap-0.9.7/

ruby extconf.rb --with-openldap2 –without-libssl
make
sudo make install
```

Ruby ActiveLDAP

Ruby ActiveLDAP je objektově orientované rozhraní mezi Ruby/Ruby on Rails a LDAP. Instaluje se jako gem.

```
sudo gem install ruby-activeldap
```

Ruby Net LDAP

Pro autentizaci je nutný ruby-net-ldap

```
sudo gem instal ruby-net-ldap –y
```

ActiveLdap

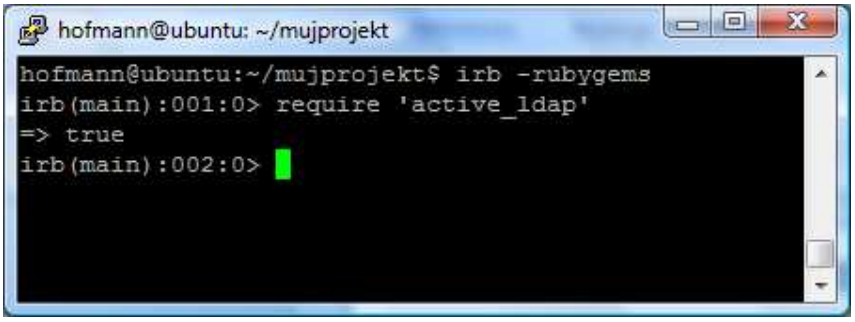
Poslední součástí je gem ActiveLdap.

```
sudo gem install activeldap
```

Propojení lze velmi snadno otestovat v irb.

```
cd /home/hofmann/mujprojekt
irb –rubygems
require 'active_ldap'
```

Pokud je vše dobře nastaveno, irb vrací hodnotu => true



Obrázek 35 - funkční ActiveLdap

4.9 SUBVERSION

Další komponenta, kterou je nutno nainstalovat je aplikace pro správu verzí – Subversion a potřebná knihovna pro volání příkazů Subversion z RoR.

```
sudo apt-get install subversion

sudo apt-get install libsvn-ruby
```

Subversion lze testovat příkazem.

```
svn help
```

4.10 DOKUWIKI

Další potřebnou součástí provozní architektury je wiki aplikace Dokuwiki.

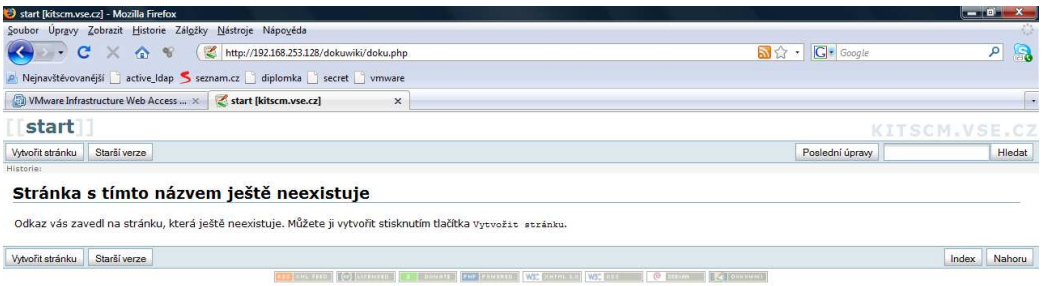
```
sudo apt-get install dokuwiki
```

V Dokuwiki je standartně přednastavený přístup pouze z lokálního počítače (localhost) – 127.0.0.1 . V souboru **/etc/apache2/conf.d/@dokuwiki.conf** je nutné přístup rozšířit pro všechny uživatele.

```
#allow from 127.0.0.1
allow from all
```

Konfigurační soubor Dokuwiki je ***/etc/dokuwiki/dokuwiki.php***. Jelikož jsou v ale tomto souboru implicitní hodnoty, nedoporučuje se s tímto souborem jakákoli manipulace. Doporučený postup je vytvoření souboru local.php, ve kterém budou změněné parametry (parametry ze souboru local.php mají přednost před parametry v souboru dokuwiki.php).

```
touch local.php
<? php
$conf['lang']        = 'cs';              //your language
$conf['title']       = 'kitscm.vse.cz'; //what to show in the title
$conf['superuser']   = 'hofmann';    //The admin can be user or @group
$conf['manager']     = 'hofmann';    //The manager can be user or @group
```



Obrázek 36 - dokuwiki

Po instalaci všech výše uvedených částí je provozní architektura potřebná pro vývoj aplikace připravena.

5. KAPITOLA 5 - VÝVOJ

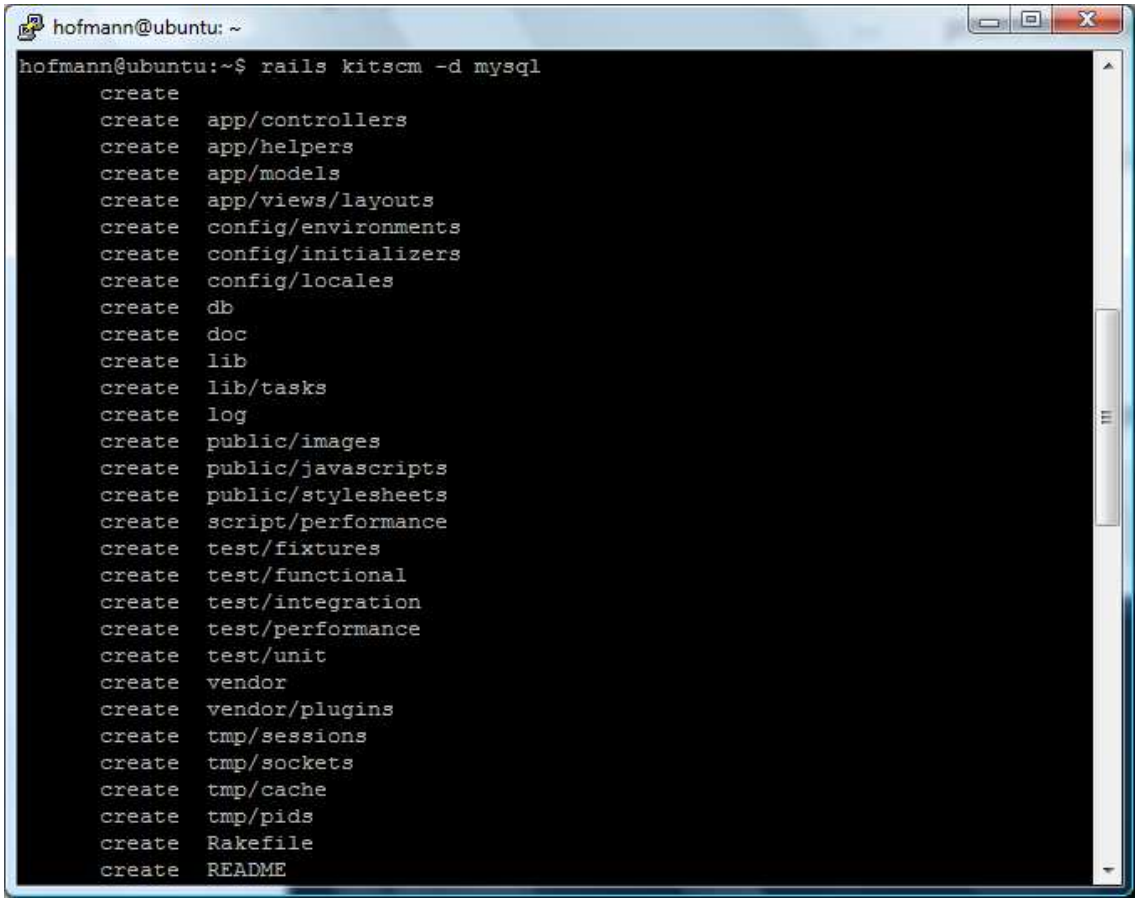
Cíle této kapitoly:	...	prakticky seznámit s vývojem v Rails
	…	vyvinout část aplikace – uživatelé

V této kapitole se krok po kroku popíše vývoj části aplikace (uživatelé). Na konkrétním příkladě se je možné co nejlépe seznámit s frameworkem RoR, jeho výhodami i nevýhodami. Výchozím předpokladem pro tvorbu je připravená a fungující provozní architektura (předchozí kapitola).

5.1 ZALOŽENÍ RAILS PROJEKTU

Prvním krokem je spuštění generátoru (kapitola 2.2.2), který založí celý RoR projekt (s MySQL databází).

```
rails kitscm –d mysql
```



Obrázek 37 – Založení Rails projektu

5.2 ZALOŽENÍ A NASTAVENÍ DATABÁZE MYSQL

Standartně většina Rails aplikací data ukládá do databáze. V případě aplikace pro podporu týmové spolupráce je situace jiná – veškerá data se budou ukládat do OpenLDAP. I když v MySQL databázi nebudou uložena žádná data, především pro pro účely testování se databáze založí.

```
sudo mysql –u root –p
```

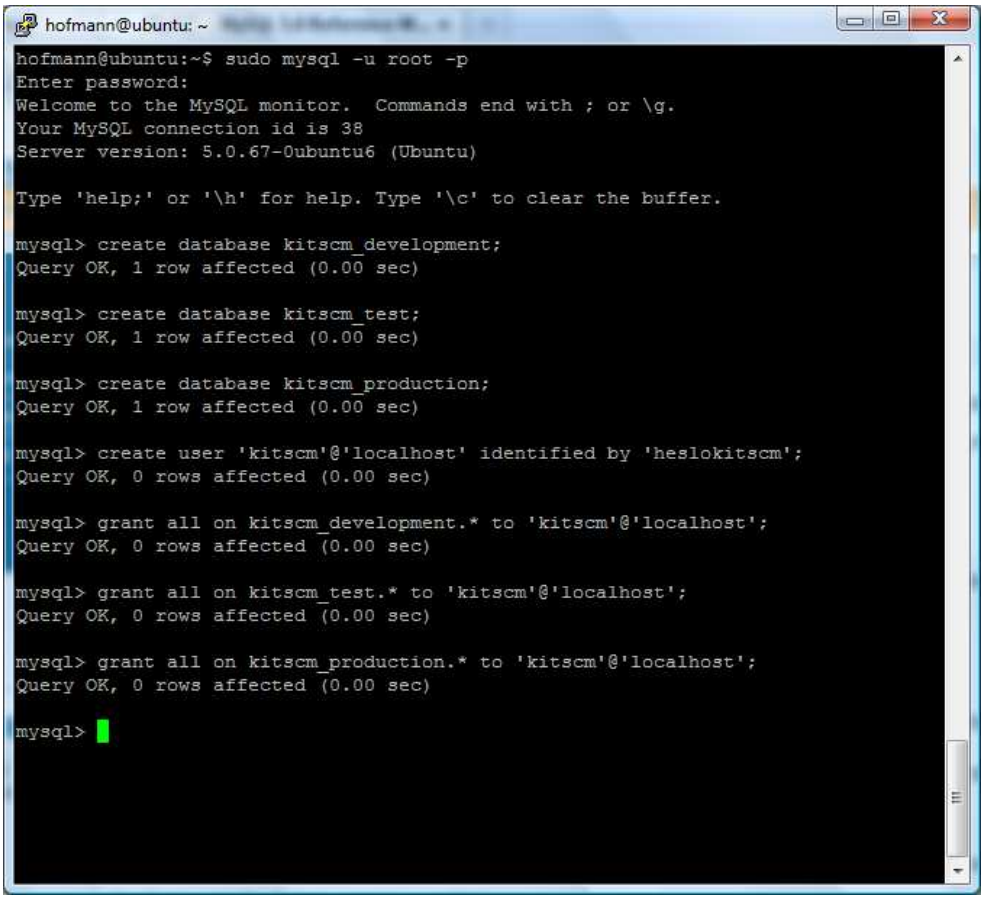
V kapitole 2.2.2 byly popsány Rails prostředí, pro každé prostředí se vytvoří oddělená databáze.

```
create database kitscm_development;
create database kitscm_test;
```

```
create database kitscm_production;
```

Pak je nutné v databázi založit nového uživatele, nastavit mu heslo práva na dané 3 databáze

```
create user 'kitscm'@'localhost' identified by 'heslokitscm';
grant all on kitscm_development.* to 'kitscm'@'localhost';
grant all on kitscm_test.* to 'kitscm'@'localhost';
grant all on kitscm_production.* to 'kitscm'@'localhost';
quit
```



Obrázek 38 - MySQL konfigurace

Propojení Rails aplikace s DB se definuje v konfiguračním souboru **config/database.yml.** Každé prostředí Rails aplikace je propojené s jinou databází.

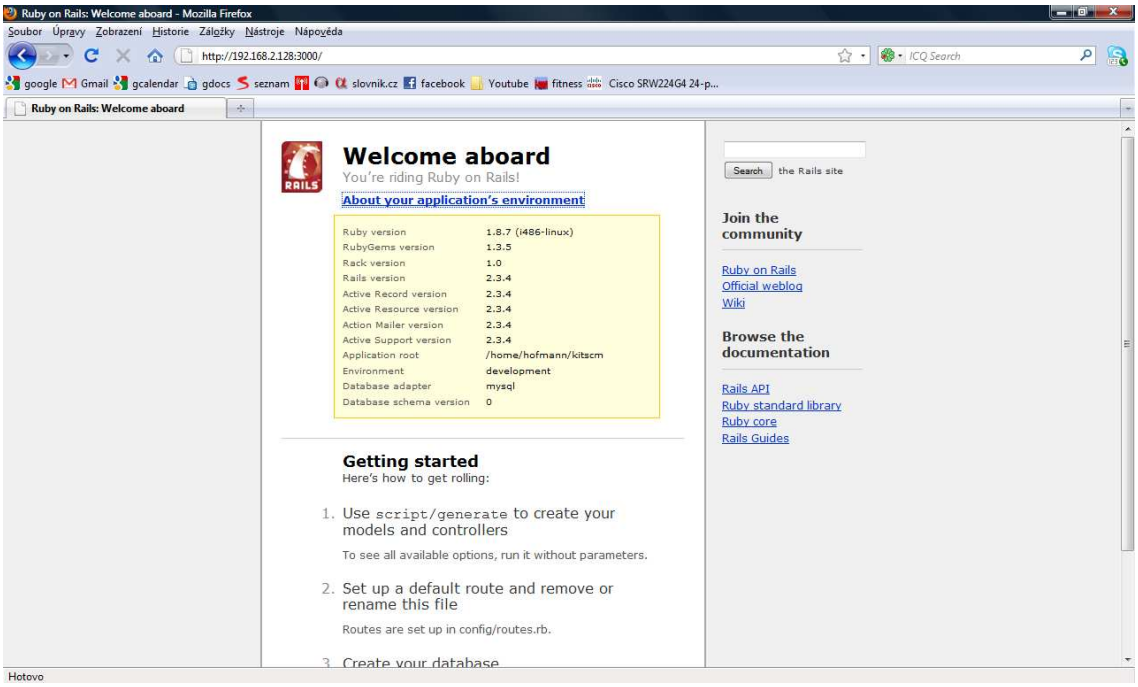
```
development:
  adapter: mysql
  encoding: utf8
  database: kitscm_development
  username: kitscm
  password: heslokitscm
  socket: /var/run/mysqld/mysqld.sock

test:
  adapter: mysql
  encoding: utf8
  database: kitscm_test
  username: kitscm
```

```
  password: heslokitscm
  socket: /var/run/mysqld/mysqld.sock

production:
  adapter: mysql
  encoding: utf8
  database: kitscm_production
  username: kitscm
  password: heslokitscm
  socket: /var/run/mysqld/mysqld.sock
```

5.3 PRVOTNÍ TESTOVÁNÍ

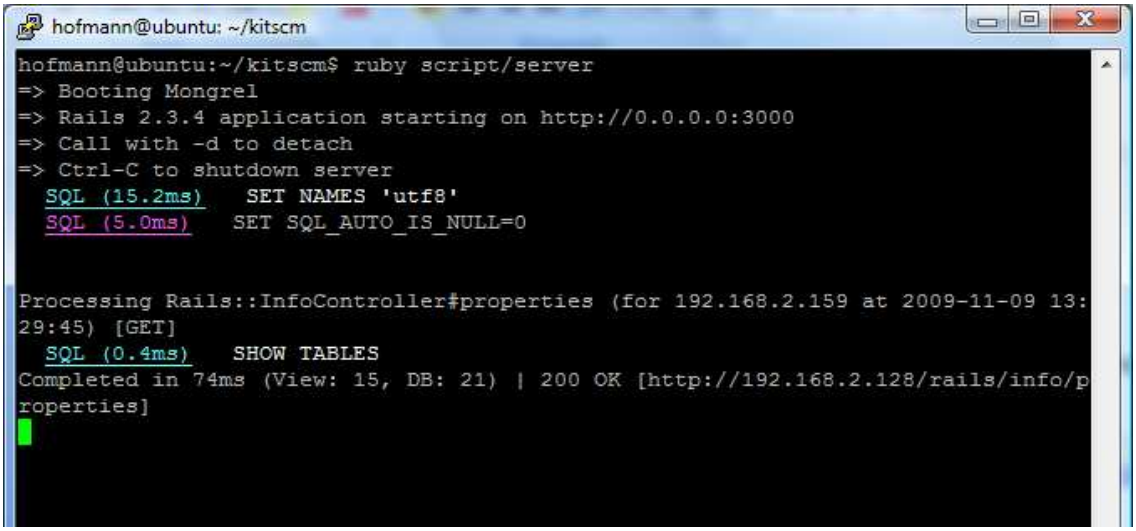
Pro testování aplikace je nejjednošší spustit testovací server Mongrel, standartně běžící na portu 3000, a aplikaci otestovat v prohlížeči.

```
cd kitscm
ruby script/server
```



Obrázek 39 – Testování aplikace

Na konzoli (kapitola 4.2.2) spuštěný webový server Mongrel začne vypisovat log.



Obrázek 40 - Mongrel log

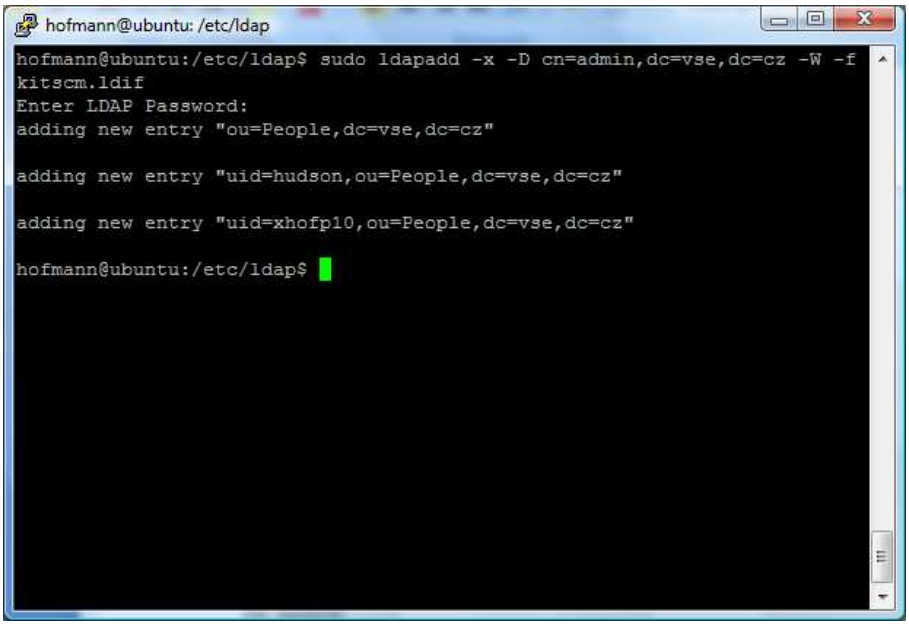
5.4 NAPLNĚNÍ LDAP DATY

Pro import dat do OpenLDAP se nejdříve vytvoří .ldif soubor.

```
sudo touch /etc/ldap/kitscm.ldif
```

Do tohoto souboru se vloží kód (příloha 2). Založí se organizační jednotka ou=users a do ní se vloží 2 uživatelé. Soubor se importuje do LDAP stromu.

```
sudo ldapadd -x -D cn=admin,dc=vse,dc=cz -W -f kitscm.ldif
```



Obrázek 41 - vložení dat do LDAP

Funkčnost lze otestovat následujícími příkazy.

```
ldapsearch -xLLL -b "dc=vse,dc=cz"
```

```
ldapsearch -xLLL -b "ou=People,dc=vse,dc=cz" uid=xhofp10
```

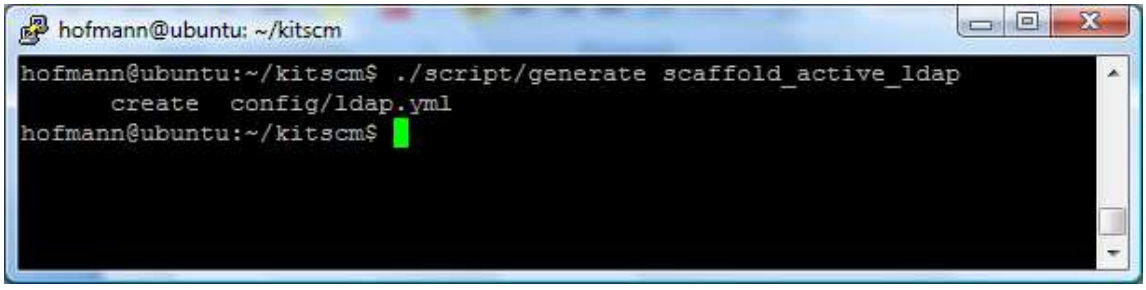
5.5 PROPOJENÍ RAILS APLIKACE S LDAP

Jakmile je OpenLDAP funkční a naplněný potřebnými daty, může se Rails aplikace propojit s OpenLDAP. Přímo do aplikace se nainstaluje zásuvný modul (kapitola 2.2.3) activeldap.

```
cd kitscm

svn co http://ruby-activeldap.googlecode.com/svn/trunk
vendor/plugins/activeldap-trunk
```

Pomocí activeldap se vygeneruje soubor **config/ldap.yml**

```
./script/generate scaffold_active_ldap
```



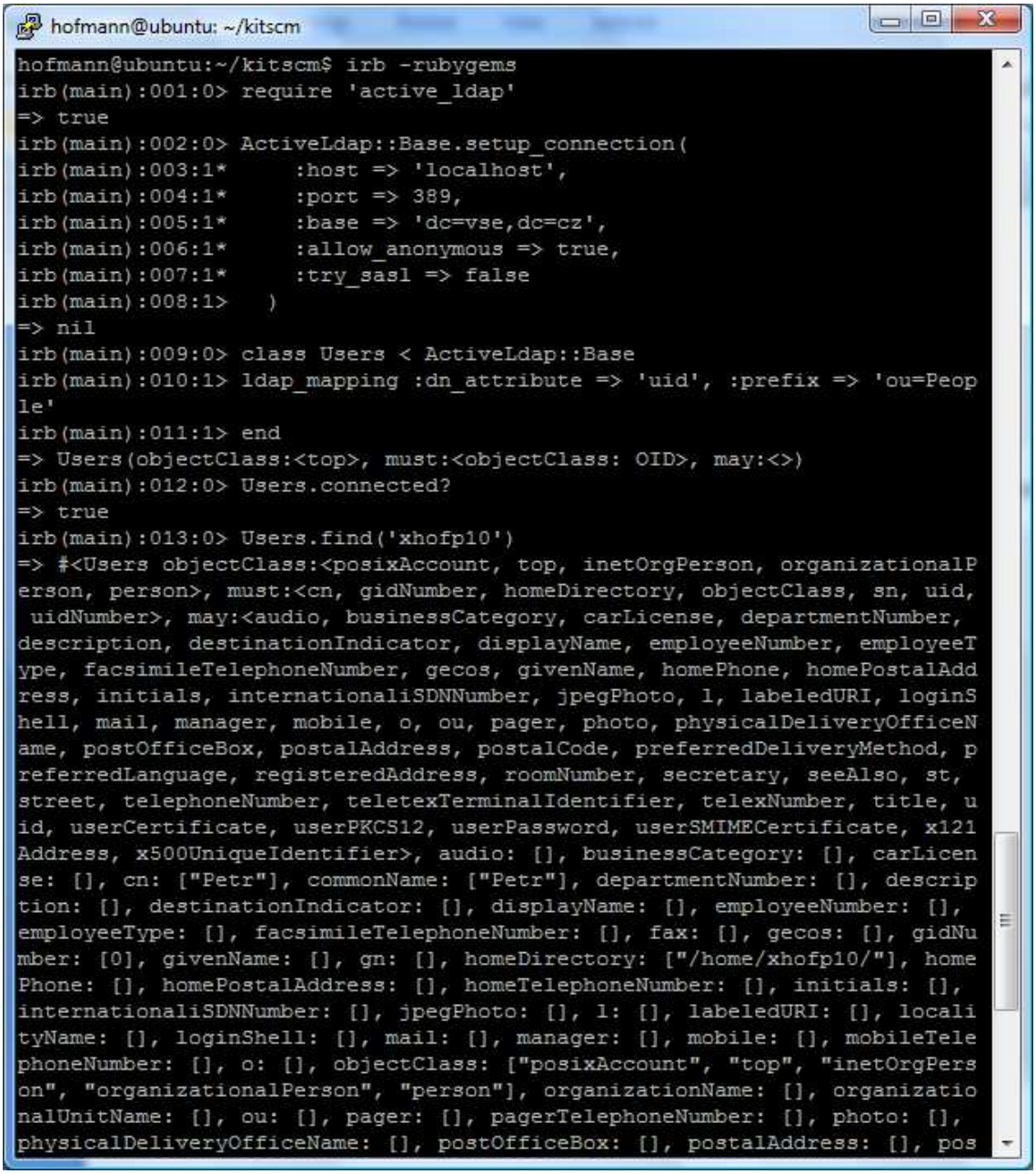
Obrázek 42 - Generování active ldap

V tomto souboru se nastavuje propojení Rails aplikace s OpenLDAP serverem. Opět zvlášť za jednotlivá prostředí, stejně jako v případě databáze (kapitola 4.6).

```
development:
host: 127.0.0.1
port: 389
base: dc=vse,dc=cz
bind_dn: cn=admin,dc=vse,dc=cz
password: heslo
```

Důležité je otestovat funkčnost v irb, nejprve anonymní propojení.

```
irb –rubygems

require 'active_ldap'
ActiveLdap::Base.setup_connection(
    :host => 'localhost',
    :port => 389,
    :base => 'dc=vse,dc=cz',
    :allow_anonymous => true,
    :try_sasl => false
  )
class Users < ActiveLdap::Base
ldap_mapping :dn_attribute => 'uid',  :prefix => 'ou=People'
end
Users.connected?
Users.find('xhofp10')
quit
```

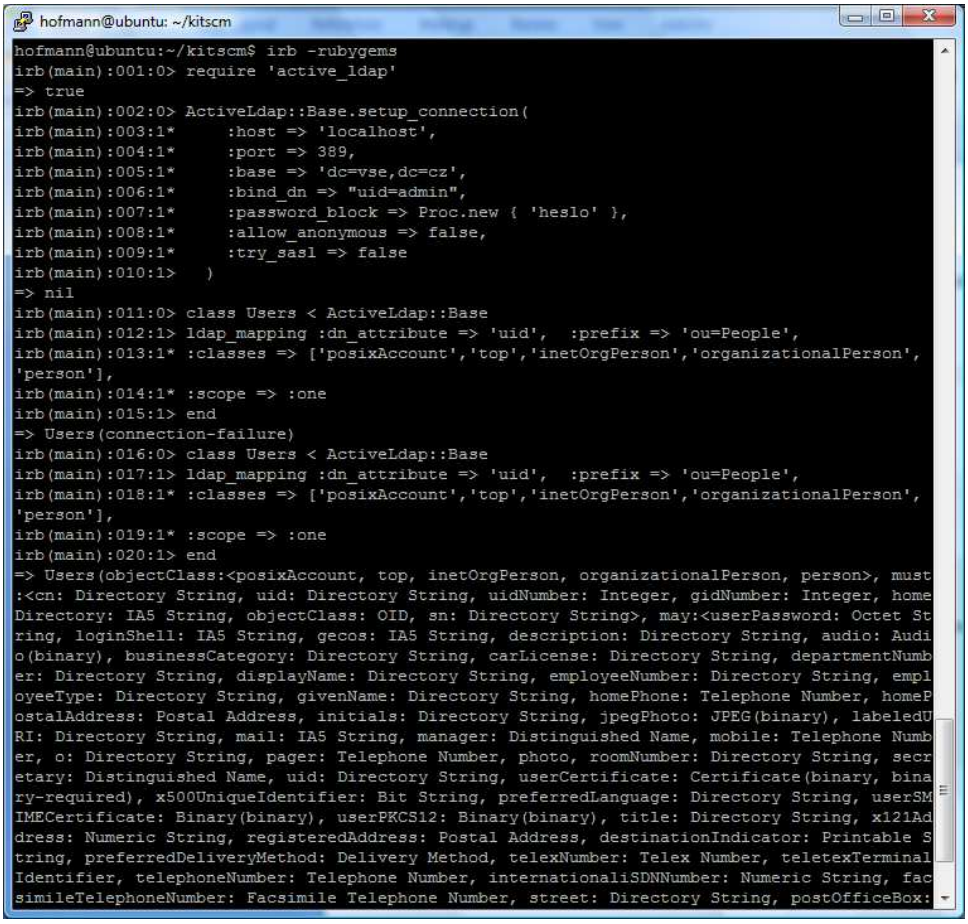


Obrázek 43 - testování propojení Rails & LDAP (anonymní)

Dále zabezpečené propojení , které je ověřeno LDAP heslem

```
irb –rubygems

require 'active_ldap'
ActiveLdap::Base.setup_connection(
    :host => 'localhost',
    :port => 389,
    :base => 'dc=vse,dc=cz',
    :bind_dn => "uid=admin",
    :password_block => Proc.new { 'heslo' },
    :allow_anonymous => false,
    :try_sasl => false
```

```
  )
class Users < ActiveLdap::Base
ldap_mapping :dn_attribute => 'uid',  :prefix => 'ou=People',
:classes =>
['posixAccount','top','inetOrgPerson','organizationalPerson',
'person'],
:scope => :one
end
Users.connected?
Users.find('xhofp10')
quit
```



Obrázek 44 - zabezpečené propojení Rails & active_ldap

Jakmile je Rails aplikace propojená s OpenLDAP může se začít s vývojem aplikace.

5.6 VÝVOJ - UŽIVATELÉ

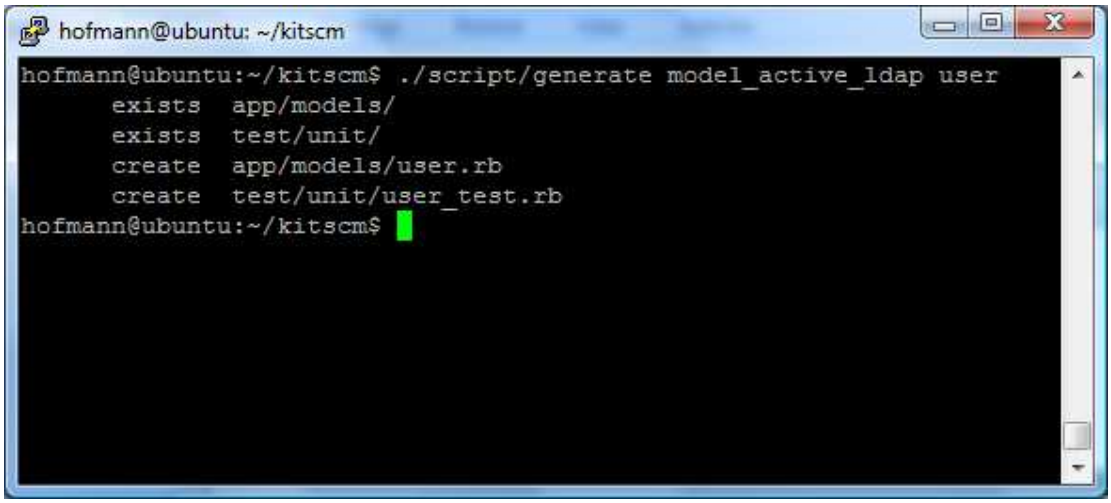
V této práci a kapitole není prostor pro ukázku vývoje celé aplikace. Pro praktické seznámení s vývojem v Rails zde bude popsán vývoj požadavku číslo 1.1 správa externích uživatelů (tabulka 4). Rails je MVC framework (kapitola 4.1.4) - první věcí, kterou se vývoj zahajuje, je generování modelu.

5.6.1 MODEL

Pro vytvoření modelu je v Rails, stejně jako pro většinu potřebných komponent, dostupný generátor (kapitola 4.1.5). Ve standartní Rails aplikaci s databází by se pro vytvoření modelu user spouštěl standartní generátor *./script/generate model user.*

V modelu se definuje datový model Rails aplikace. V případě aplikace pro podporu spolupráce data nejsou ukládána standartně do databáze, ale do OpenLDAP. Proto se pro pro generování modelu využívá generátor zásuvného modulu activeldap.

```
./script/generate model_active_ldap user
```



Obrázek 45 - Generování modelu

Ve vygenerovaném souboru modelu `app/models/user.rb`, se nastavuje propojení aplikace a OpenLDAP. V kapitole 2.2.5 je popsáno ORM – mapování objektových tříd na relační tabulky. Zásuvný modul activeldap podobným způsobem mapuje Rails objektové třídy na záznamy OpenLDAP. Třída User je namapována na LDAP záznam User.

Attribute Description	Value
objectClass	*inetOrgPerson (structural)*
objectClass	*organizationalPerson (structural)*
objectClass	*person (structural)*
objectClass	*posixAccount (auxiliary)*
objectClass	*top (abstract)*
cn	Petr
gidNumber	0
homeDirectory	/home/xhofp10/
sn	Hofmann
uid	xhofp10
uidNumber	0
userPassword	SHA hashed password

Obrázek 46- LDAP záznam User

V souboru **`app/models/user.rb`** se specifikují parametry propojení.

```
class User < ActiveLdap::Base
  ldap_mapping :dn_attribute => "uid",
           :prefix => "ou=People",
           :classes =>
['inetOrgPerson','organizationalPerson','person','posixAccount','top
']
end
```

Parametr :prefix je pouze `"ou=People", "dc=vse,dc=cz"` se nastavuje v souboru `config/ldap.yml`.

5.6.2 CONTROLLER

Po vytvoření modelu a definici datového modelu aplikace, následuje vytvoření controlleru, v němž je je obsažena řídící logika aplikace. Pro vytvoření controlleru se využije standartní generátor.

```
./script/generate controller users
```

Obrázek 47 - Generování controlleru

V souboru **`app/controllers/users_controller.rb`** je kód řídící logiky aplikace. Po vytvoření je standartně prázdný.

```
class UserController < ApplicationController
end
```

Při postupném rozšiřování logiky aplikace se do controlleru přidávají jednotlivé metody. Pro aplikaci týmové spolupráce se přidáme první metodu - index, která načte všechny existující uživatele a uloží je jako proměnnou @users. Controller dle obrázku 7 získává data prostřednictvím modelu User.

```
class UserController < ApplicationController
def index
  @users = User.find(:all)
end
end
```

5.6.3 VIEW

Třetí součástí pro vytvoření funkční části aplikace, po modelu a controlleru, je view. Ve view se definuje uživatelské rozhraní aplikace. Pro vytvoření se již nevyužívá žádný generátor, pouze se v adresáři **app/views/user** zakládají příslušné soubory .html.erb. Pro metodu index definovanou v controlleru manuálně založíme soubor **index.html.erb**

```
touch index.html.erb
```

Do tohoto souboru view vložíme následující kód.

```
<h1>Seznam uživatel• </h1>
<% if @users.empty? -%>
<p><%= "Žádný uživatel." %></p>
<% else -%>
<table>
<% poradi = 1 %>
<% for user in @users %>
<tr>
    <td><%= poradi %>. </td>
    <td><b><%= h(user.uid) %></b></td>
</tr>
<% poradi += 1 %>
<% end %>
</table>
<br/>
<% end -%>
```

Ve view se kombinuje Ruby kód a standartní HTML kód. Ruby kód uzavřený v blocích **<% %>** se vykoná, ale nezobrazí se. Oproti tomu Ruby kód uzavřený v blocích **<%= %>** vloží na své místo hodnotu posledního uvnitř provedeného výrazu (využívá se jako echo). Ostatní kód je standartní HTML kód.

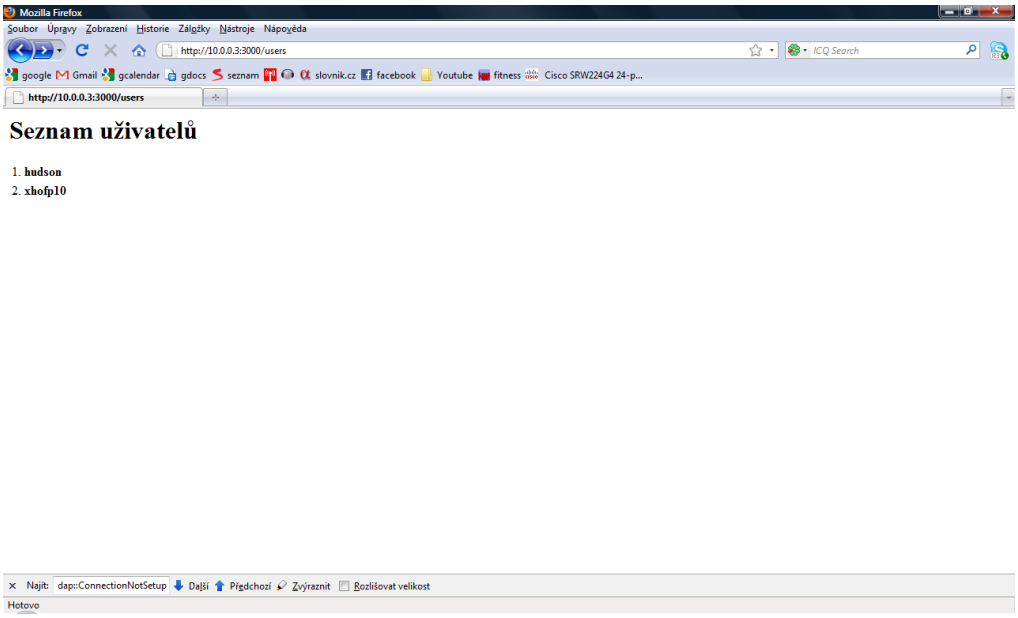
Výše uvedené view zobrazí nadpis "Seznam uživatelů", formátovaný dle HTML jako nadpis 1. Pokud je pole @users prázdné, zobrazí se odstavec "Žádný uživatel". V opačném případě se definuje promenna poradi a do tabulky se pro každého uživatele (hodnotu v @users) do řádku vypíše pořadové číslo (poradi) a jeho uživatelské jméno (uid).

Pokud by se z view odstranily bloky <% %> a <%= %> zůstal by souvislý Ruby kód.

```
if @users.empty?
"Žádný uživatel."
else
     poradi = 1
      for user in @users
        poradi
        h(user.uid)
        poradi += 1
     end
end
```

Po vytvoření všech částí MVC – modelu, controlleru a view se vše může otestovat. Spustí se webový server Mongrel (kapitola 4.6 str. 48) a v prohlížeči se vše otestuje zadáním adresy.

http://server:3000/users nebo http://server:3000/users/index



Obrázek 48 - Testování prvního view

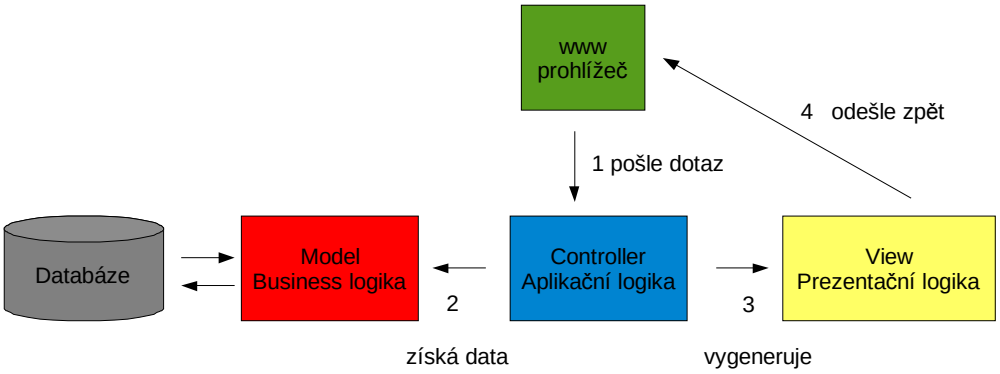
V kapitole 2.2.1 str. 20 byla na obrázku č. 7 popsána funkcionalita MVC. Nyní si vše ukážeme konkrétně na příkladu vyvíjené aplikace:

1 – poslání dotazu proběhně v prohlížeči zadáním **http://server/users**

2 - **/users** je controller, volaná metoda je **index**, v ní se získají data z modelu User, model zprostředkuje získání dat z LDAP serveru (místo databáze) pomocí activeldap

3 – ve view se data **@users** zpracují, vygeneruje se HTML kód

4 – HTML kód se odešle zpět do www prohlížeče uživatele



Obrázek 49 - MVC funkcionalita

5.7 DALŠÍ METODY – CRUD

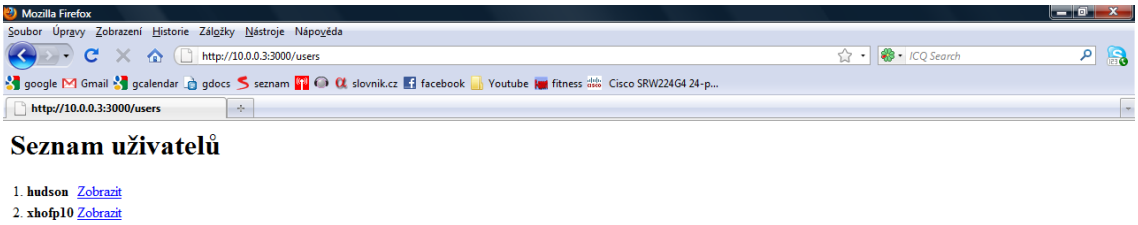
V této podkapitole se do aplikace doplní další operace CRUD – Create (založení nového uživatele), Read (načtení a zobrazení uživatele), Update (aktualizace dat uživatele) a Delete (smazání uživatele). Jelikož se rozšiřuje řídící logika aplikace a uživatelské rozhraní, na modelu se nic nemění. Při rozšiřování se přidají příslušné metody do controlleru users_controller.rb a založí nová views.

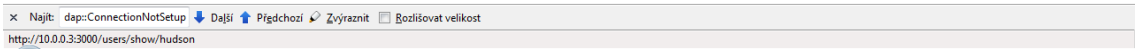
5.7.1 READ

Operace read načte data příslušného uživatele a vypíše je. Nejprve se do seznamu uživatelů - view **index.html.erb** doplní odkaz, kterým se příslušný uživatel zobrazí.

```
<td><%= link_to 'Zobrazit', :action => 'show', :id => user %></td>
```

```
<% for user in @users %>
<tr>
    <td><%= poradi %>. </td>
    <td><b><%= h(user.uid) %></b></td>
    <td><%= link_to 'Zobrazit', :action => 'show', :id => user
%></td>

</tr>
<% poradi += 1 %>
<% end %>
```





Obrázek 50 – Seznam uživatelů

Kliknutím na odkaz "Zobrazit" se zavolá controller **http://server:3000/users/show/uid_uživatele**. Do controlleru se přidá metoda show, která nalezne příslušného uživatele.

```
def show
    @user = User.find(params[:id])
```

```
end
```

Velmi důležité je v Rails porozumět předávání parametrů z view do controlleru.

1. Ve view se definuje, že se kliknutím na tlačítko **Zobrazit**, má zavolat funkce **show**, jako parametr :**id** se má předat daný uživatel **user** (dn_attribute je uid)

```
<td><%= link_to 'Zobrazit', :action => 'show', :id => user %></td>
```

2. V controlleru se pak vyhledá daný uživatel.

```
@user = User.find(params[:id])
```

Jako poslední je nutné založit nové view **show.html.erb**, ve kterém se zobrazí data o uživateli.

```
<h1>Zobrazení uživatele: <%= @user.uid %></h1>
<table>
<tr><td id="levy">Uid: </td><td><%= h(@user.uid)%></td></tr>
<tr><td id="levy">Heslo: </td><td><%=
h(@user.userPassword)%></td></tr>
<tr><td id="levy">Jméno: </td><td><%= h(@user.cn)%></td></tr>
<tr><td id="levy">Příjmení: </td><td><%= h(@user.sn)%></td></tr>
<tr><td id="levy">Email</td><td><%= h(@user.mail)%></td></tr>
<tr><td id="levy">Domovský adresář</td><td><%=
h(@user.homeDirectory)%></td></tr>
</table>
</br>
<%= link_to 'Zpět', :action =>'index' %> </p>
```

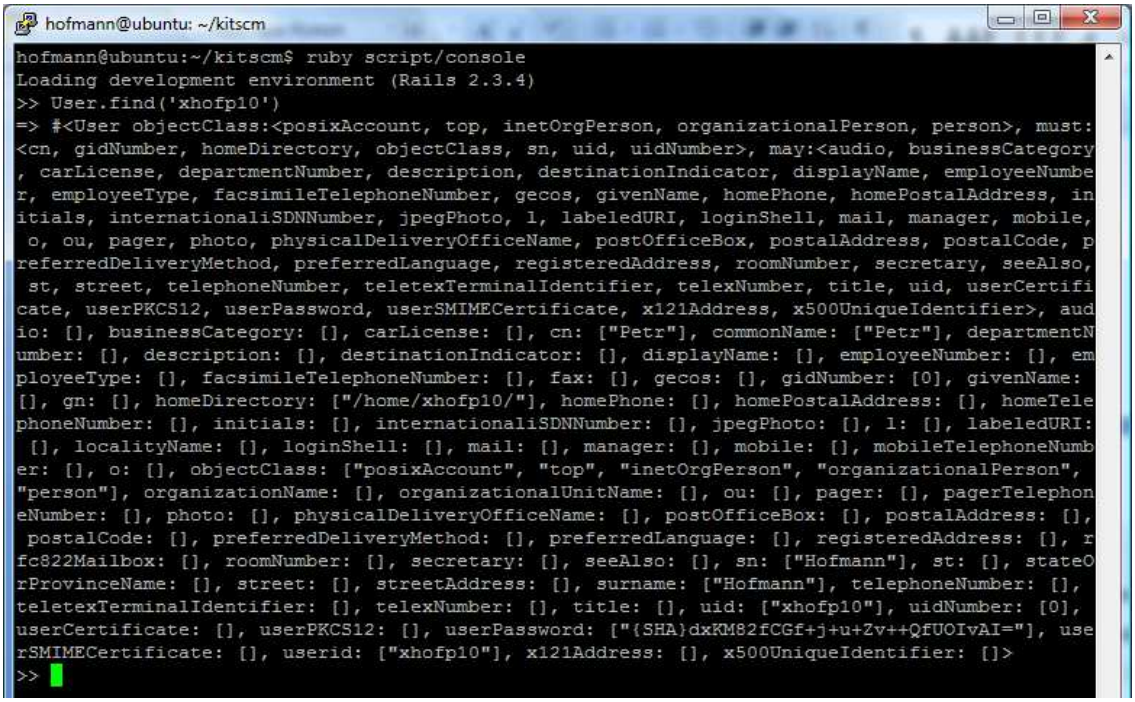
Obrázek 51 - Zobrazení uživatele

5.7.2 CREATE

Další operací je operace Create, která využívá 2 metody - new a create. V této operaci mohou nastat různé chyby, proto je výhodné nejdříve zkusit zakládání záznamů v irb či v konzoli. Pro tento příklad je vhodnější konzole (kapitola 2.2.2). Konzole se spustí v domovském adresáři aplikace příkazem.

```
ruby script/console
```

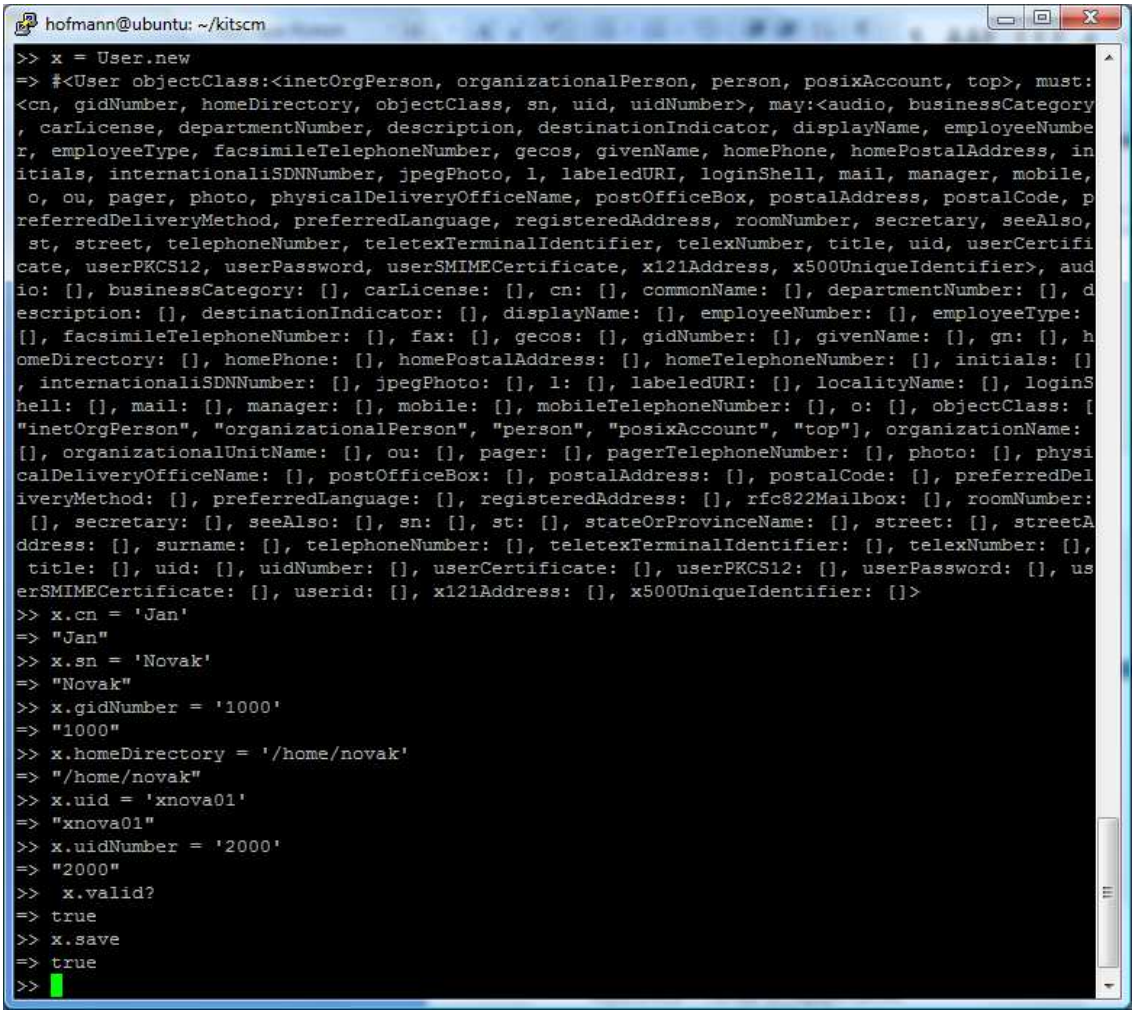
Do konzole se ihned nahraje příslušné prostředí, v našem případě vývojové. Na rozdíl od irb se nemusí pro test nejprve propojovat RoR aplikace s OpenLDAP (kapitola 5.5). V konzoli se může rovnou začít pracovat s daty v OpenLDAP, např. v hledání uživatelů.



Obrázek 52 - ruby script/console

V konzoli se otestuje založení nového uživatele. Uživatel se založí metodou new, nastaví se mu příslušné atributy (některé jsou povinné), otestuje se jeho validita – metoda valid? a uživatel se uloží do OpenLDAP - metoda save.

```
x = User.new
x.cn = 'Jan'
x.sn = 'Novak'
x.gidNumber = '1000'
x.homeDirectory = '/home/novak'
x.uid = 'xnova01'
x.uidNumber = '2000'
 x.valid?
x.save
```



Obrázek 53 - Založení uživatele ruby/console

Pokud je manuální vkládání dat pomocí konzole funkční, lze tuto funkcionalitu velmi jednoduše doplnit do Rails aplikace. Do controlleru se přidají 2 nové metody. Metoda **new –** založí nového uživatele:

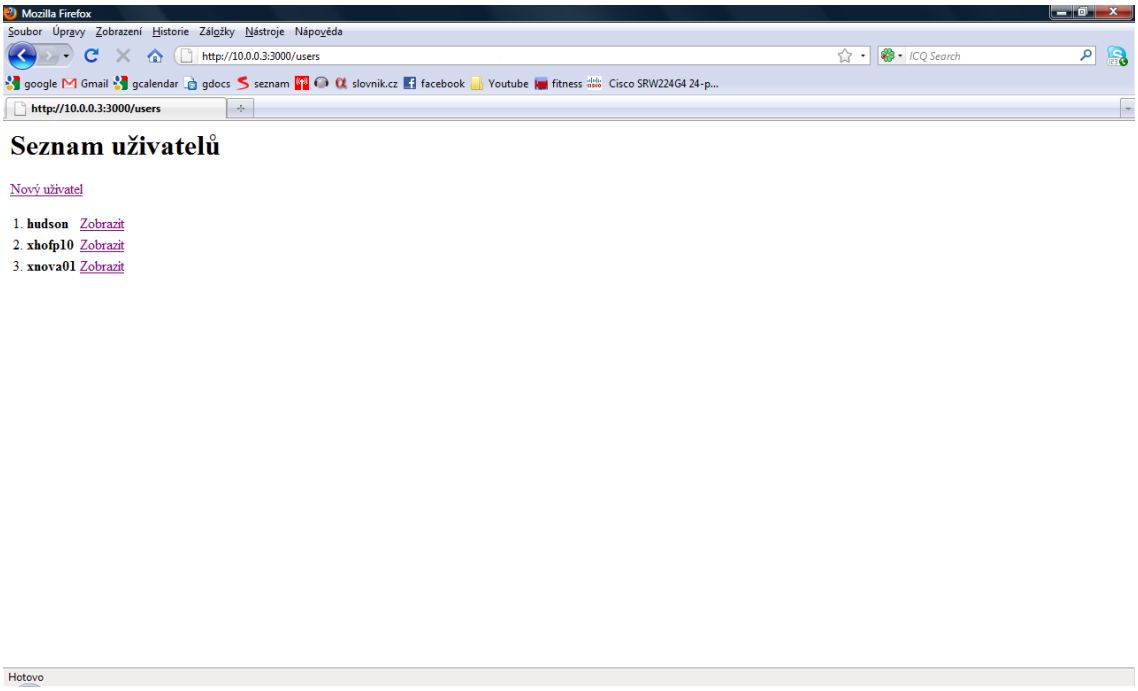
```
def new
    @user = User.new
end
```

a metoda **create** – novému uživateli předá data, jako atributy doplní parametry (které dostane odesláním formuláře – viz. dále).

```
def create
    @user = User.new(params[:user])
   if @user.save
        flash[:notice] = 'Uživatel byl úsp•šn• založen.'
        redirect_to(:action => 'index')
    else
      render (:action => 'new')
  end
end
```

Do view seznamu uživatelů (index.html.erb) se přidá odkaz, který přesměruje na stránku založení nového uživatele

```
<p id='tlacitko'><%= link_to 'Nový uživatel', :action => 'new'
%></p>
```

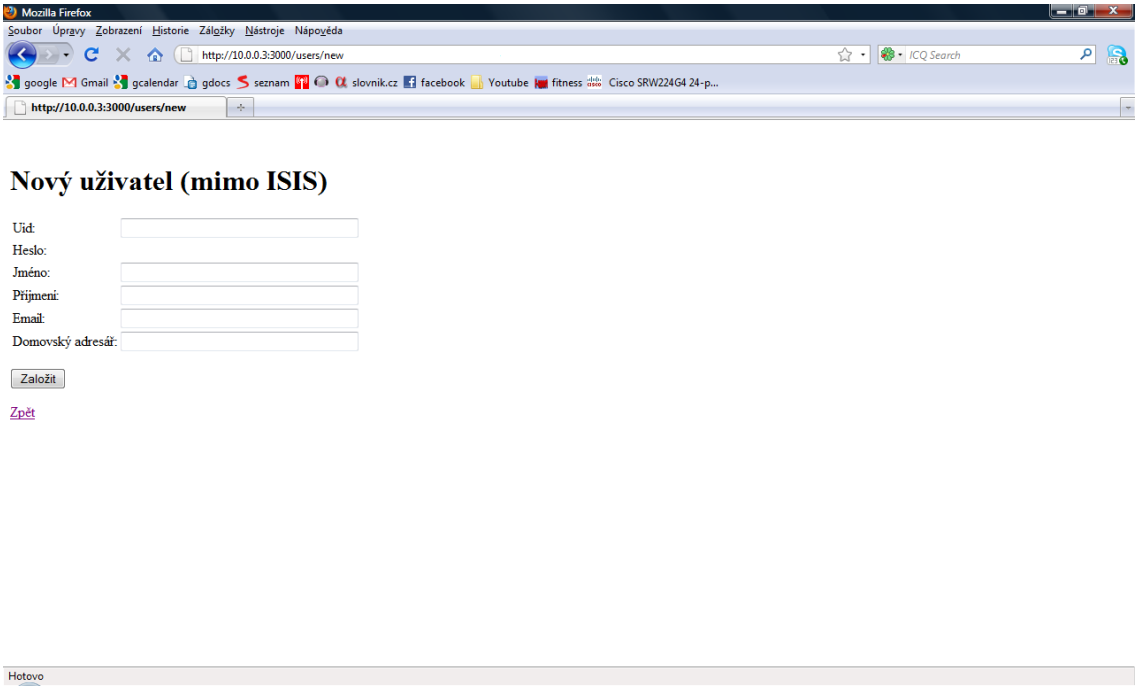


Obrázek 54 - Nový uživatel

Pro založení uživatele vytvoříme nové view **new.html.erb.**

```
<br/><h1>Nový uživatel (mimo ISIS)</h1>
<% form_for :user, :url => 'create' do |f| %>
<%= f.error_messages %>
<table>
<tr><td id="levy"><label for="uid" >Uid: </label></td>
<td><%= f.text_field :uid, :size => 40 %></td></tr>
<tr><td id="levy"><label for="userPassword" >Heslo: </label></td>
<td><%= f.hidden_field :userPassword, :size => 40, :value =>
Array.new(8){(rand(122-97)+97).chr}.join %></td></tr>
<tr><td id="levy"><label for="cn" >Jméno: </label></td>
<td><%= f.text_field :cn, :size => 40 %></td></tr>
<tr><td id="levy"><label for="sn" >P•íjmení: </label></td>
<td><%= f.text_field :sn, :size => 40 %></td></tr>
<tr><td id="levy"><label for="mail" >Email: </label></td>
<td><%= f.text_field :mail, :size => 40 %></td></tr>
<tr><td id="levy"><label for="homeDirectory" >Domovský adresá•:
</label></td>
<td><%= f.text_field :homeDirectory, :size => 40 %></td></tr>
<%= f.hidden_field :gidNumber, :value => "9" %>
<%= f.hidden_field :uidNumber, :value => "9" %>
</table>
  <p>
    <%= f.submit "Založit" %>
  </p>
<% end %>
<p><%= link_to 'Zpět', :action => 'index' %></p>
```

Toto view je již složitější. Založení uživatele proběhne pomocí Rails form helperu (form_for), pomocí kterého se vytvoří nová instance modelu (:user).



Obrázek 55 - Založení nového uživatele

Helper form_for vygeneruje HTML kód.

Funkcionalitu helperů lze nejsnadněji ukázat na podobném příkladě. Například následující formulář

```
<% form_for :person, @person, :url => { :action => "create" } do |f| %>
      <%= f.text_field :first_name %>
      <%= f.text_field :last_name %>
      <%= submit_tag 'Create' %>
<% end %>
```

Vygeneruje následující HTML kód:

```
<form action="/persons/create" method="post">
<input id="person_first_name" name="person[first_name]" size="30"
type="text" />
<input id="person_last_name" name="person[last_name]" size="30" type="text"
/>
<input name="commit" type="submit" value="Create" />
</form>
```

V Rails formulářích se využívají standartní formulářové prvky – textová pole (f.text_field), skrytá pole (f.hidden_field), tlačítko k odeslání (f.submit), apod. Heslo (atribut :userPasssword) se vygeneruje příkazem `Array.new(8){(rand(122-97)+97).chr`

Pokud se do formuláře ve view **new.html.erb** zadají data, a odešlou se (kliknutím na tlačítko Založit), active_ldap nejprve pokusí zjistit, zda záznam (UID užívatele) již v LDAP stromu existuje:

```
LDAP: search: FAILED (0.0ms):
{:filter=>"(&(uid=xkurz11)(&(objectClass=inetOrgPerson)(objectClass=organiz
ationalPerson)(objectClass=person)(objectClass=posixAccount)(objectClass=to
```

```
p)))", :error_message=>"No such object",
:base=>"uid=xkurz11,ou=People,dc=vse,dc=cz", :scope=>:base,
:error=>"LDAP::ResultError", :attributes=>[]}
Ignore error ActiveLdap::LdapError::NoSuchObject(No such object): filter
(&(uid=xkurz11)(&(objectClass=inetOrgPerson)(objectClass=organizationalPers
on)(objectClass=person)(objectClass=posixAccount)(objectClass=top))):
attributes: []
```

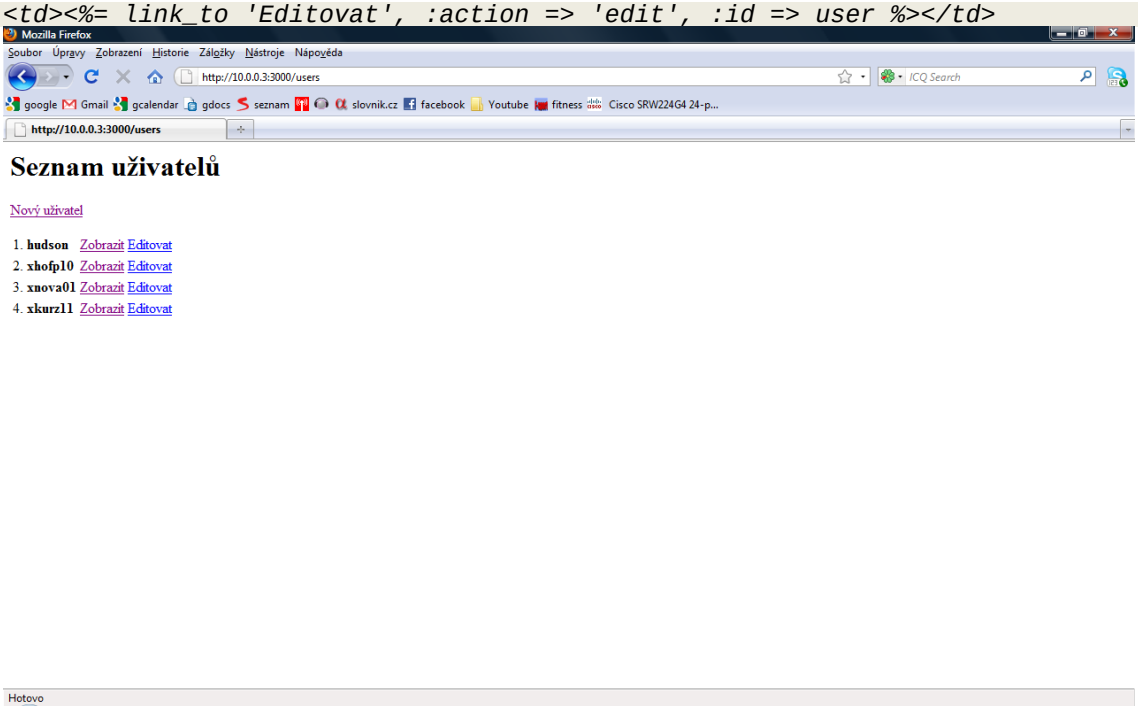
Pokud záznam neexistuje, přidá se do OpenLDAP.

```
LDAP: add (13.1ms): {:dn=>"uid=xkurz11,ou=People,dc=vse,dc=cz",
:attributes=>[[:add, "uid", {"uid"=>["xkurz11"]}], [:add, "objectClass",
{"objectClass"=>["inetOrgPerson", "organizationalPerson", "person",
"posixAccount", "top"]}], [:add, "cn", {"cn"=>["Jakub"]}], [:add,
"gidNumber", {"gidNumber"=>["9"]}], [:add, "homeDirectory",
{"homeDirectory"=>["/home/xkurzj11"]}], [:add, "mail",
{"mail"=>["kurz@seznam.cz"]}], [:add, "sn", {"sn"=>["Kurz"]}], [:add,
"uidNumber", {"uidNumber"=>["9"]}], [:add, "userPassword",
{"userPassword"=>["kohcvuaa"]}]]}
Redirected to http://10.0.0.3:3000/users
Completed in 86ms (DB: 2) | 302 Found [http://10.0.0.3/users/create]
```

5.7.3 UPDATE

Další operací je operace Update umožnující aktualizaci/editaci dat uživatele. Postup je podobný jako u operace new. Nejprve se do seznamu uživatelů – view **index.html.erb** přidá odkaz pro editaci.

```
<td><%= link_to 'Editovat', :action => 'edit', :id => user %></td>
```



Obrázek 56 - Seznam uživatelů - editace

Do controlleru se přidají metody **edit** a **update**, které najdou příslušného uživatele a aktualizují jeho atributy.

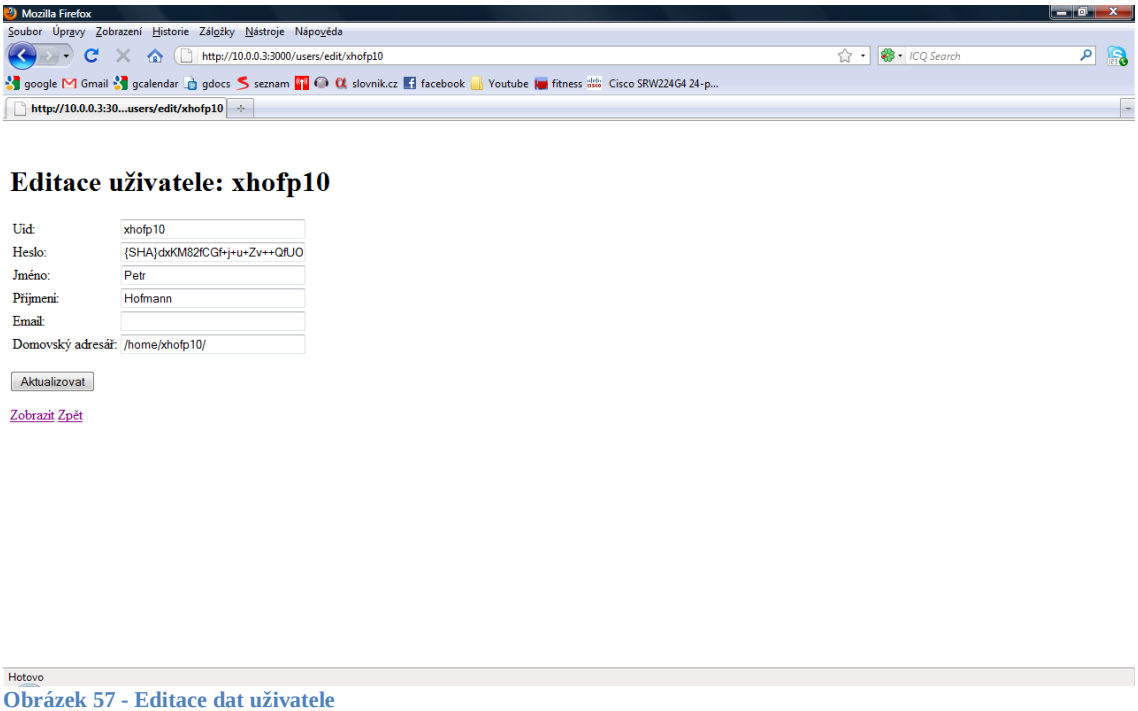
```
def edit
    @user = User.find(params[:id])
end

def update
    @user = User.find(params[:id])
```

```
    if @user.update_attributes(params[:user])
        flash[:notice] = 'Uživatel byl úsp•šn• aktualizován.'
        redirect_to(:action => 'index')
    else
        render (:action => "edit")
    end
end
```

Pro editaci se založí další view – **edit.html.erb**

```
<h1>Editace uživatele: <%= @user.uid %></h1>
<table>
<% form_for :user, :url => {:action => 'update', :id => @user} do |f| %>
<tr><td id="levy"> <label for="uid" >Uid: </label></td><td> <%=
f.text_field (:uid) %> </td></tr>
<tr><td id="levy"> <label for="userPassword" >Heslo: </label></td><td> <%=
f.text_field (:userPassword) %></td> </tr>
<tr><td id="levy"> <label for="cn" >Jméno: </label> </td><td><%=
f.text_field (:cn) %> </td></tr>
<tr><td id="levy"> <label for="sn" >P•íjmení: </label> </td><td><%=
f.text_field (:sn) %> </td></tr>
<tr><td id="levy"><label for="mail" >Email:</label></td><td><%=
f.text_field (:mail) %></td></tr>
<tr><td id="levy"><label for="homeDirectory" >Domovský
adresá•:</label></td><td><%= f.text_field (:homeDirectory) %></td></tr>
</table>
  <%= f.error_messages %>
  <p><%= f.submit "Aktualizovat"%></p>
<% end %>
<p><%= link_to 'Zobrazit', :action => 'show', :id => @user %>
<%= link_to 'Zp•t', :action => 'index' %> </p>
```

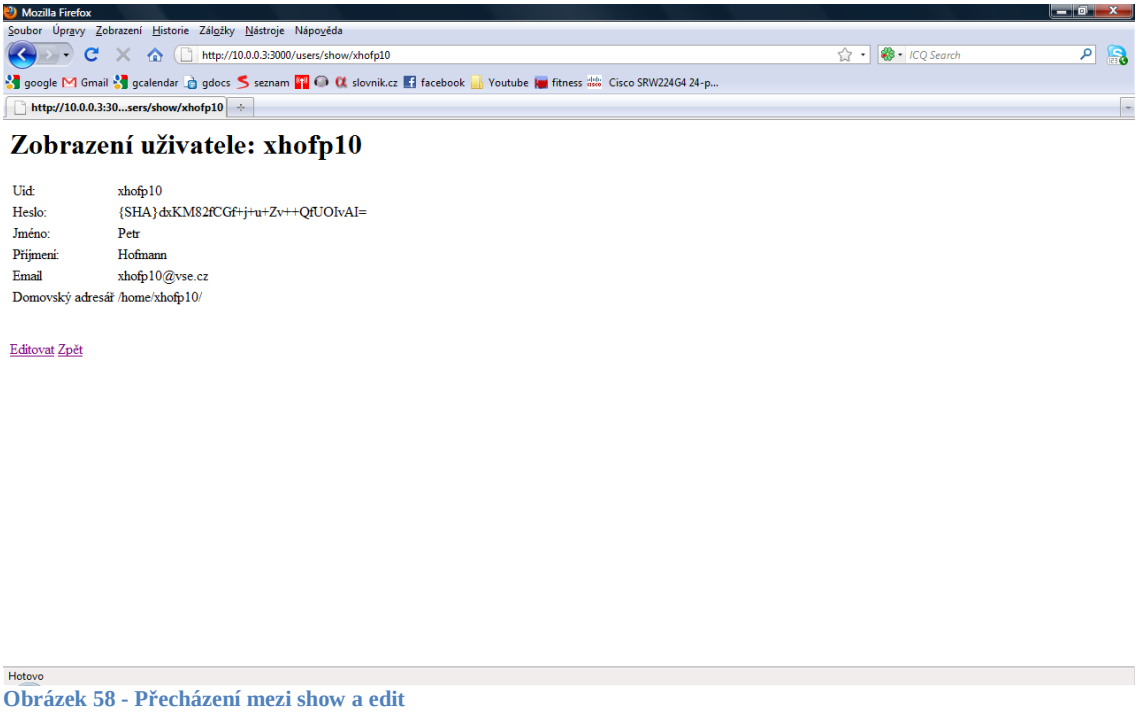
Editace uživatele probíhá podobně jako při zakládání nového uživatele pomocí Rails formulářů – form_for helperu.



Obrázek 57 - Editace dat uživatele

Do view zobrazení uživatele se přidá odkaz edit – pro možnost přecházení ze zobrazení rovnou k editaci.

```
<p><%= link_to 'Editovat', :action =>'edit', :id => @user %>
```

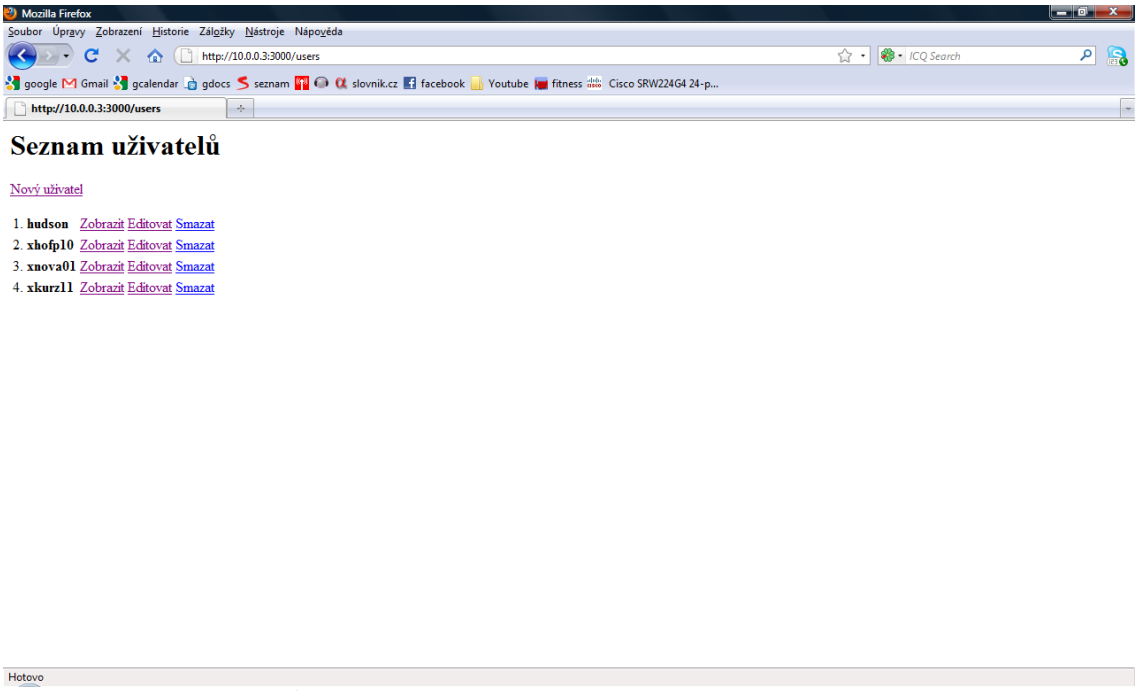


Obrázek 58 - Přecházení mezi show a edit

5.7.4 DELETE

Poslední základní operací je operace delete, která smaže daného uživatele z OpenLDAP. Do seznamu uživatelů – index.html.erb se přidá další odkaz – Smazat.

```
<td><%= link_to 'Smazat', {:action => 'destroy', :id => user}, :confirm =>
'Smazat uživatele?', :method => :post %></td>
```



Obrázek 59 - Seznam uživatelů - smazání

Při kliknutí na tento odkaz je vhodné ověřit, zda se uživatel má skutečně (parametr :confirm).

Obrázek 60 - Potvrzení smazání uživatele

Do controlleru se přidá metoda destroy, která nalezne daného uživatele a smaže ho z OpenLDAP.

```
def destroy
    @user = User.find(params[:id])
    @user.destroy
    redirect_to(:action => 'index')
end
```

5.8 DESIGN - LAYOUT A CSS

V kapitolách 5.6.1 až 5.7.4 byla do aplikace pro podporu týmové spolupráce zakomponována funkcionalita týkající se požadavku 1.1 z tabulky 4 – správa externích uživatelů. Konkrétně jde následující funkce:

1. zobrazení seznamu externích uživatelů
2. založení nového uživatele
3. zobrazení vybraného uživatele
4. aktualizace dat vybraného uživatele
5. smazání vybraného uživatele

Krok za krokem byl v těchto kapitolách popsán způsob vývoje Rails aplikace. V kapitole 5.8 se popíše základní způsob formátování Rails aplikace – pomocí layoutu a css.

Pro grafické formátování Rails aplikací se využívají layouty a kaskádové styly css.

Layout můžeme definovat jako jednotnou grafickou úpravu (šablonu). Layout pro aplikaci se založí založením souboru v adresáři layouts.

```
touch  app/views/layouts/users.rhtml
```

Tento layout bude "šablonou" pro controller users_controller.rb. Veškerá views v app/views/users budou formátovány tímto layoutem. Do souboru **users.rhtml** se vkládá kód layoutu.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd" >
<html>
<head>
  <title>KITSCM.VSE.CZ</title>
  <%= stylesheet_link_tag "users" , :media => "all" %>
</head>

<body id="page" >
  <div id="banner" >
    <img src="/images/banner.png" />
    <%= @page_title || "kitscm.vse.cz" %>
  </div>
  <div id="columns" >
    <div id="side" >
           <h3>Aplikace</h3>
           <ul><%= link_to "Uživatelé" , :controller => 'users', :action =>
'index' %></ul>
    </div>
    <div id="main" >
       <%= yield :layout %>
    </div>
  </div>
  <div id="end" >
<br>
<a href="http://kit.vse.cz"> Copyright (C) 2009 Katedra informa•ních
technologií </a>

        </div>
</body>
```

```
</html>
```

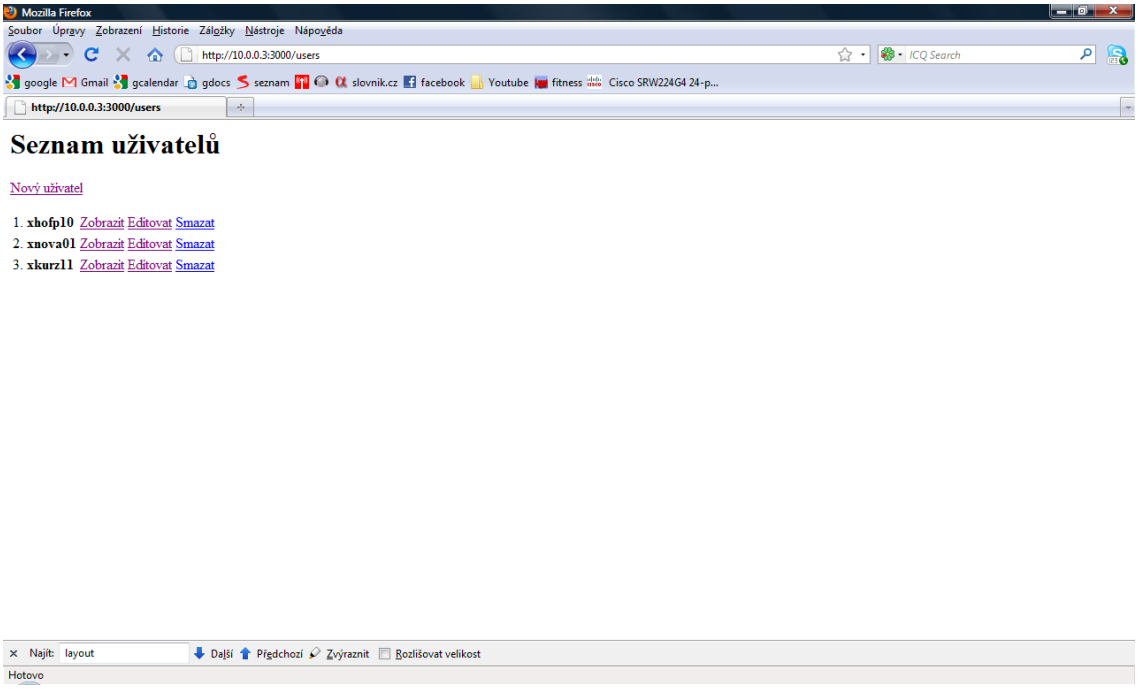
Jedná se o klasický HTML kód, formátovaný pomocí css (i layout). Pomocí ***stylesheet_link_tag "users"*** se specifikuje propojení s css stylem (users.css). Pomocí **<%= yield :layout %>** se definuje, kam se mají příslušná views (index,show,edit,…) vložit – v tomto případě do části **div id="main".** Tento layout, se bude formátovat kaskádovým stylem.

Pro styl se založí se soubor stylu, co něhož se vloží css kód (příloha 3).

```
touch app/public/stylesheets/users.css
```

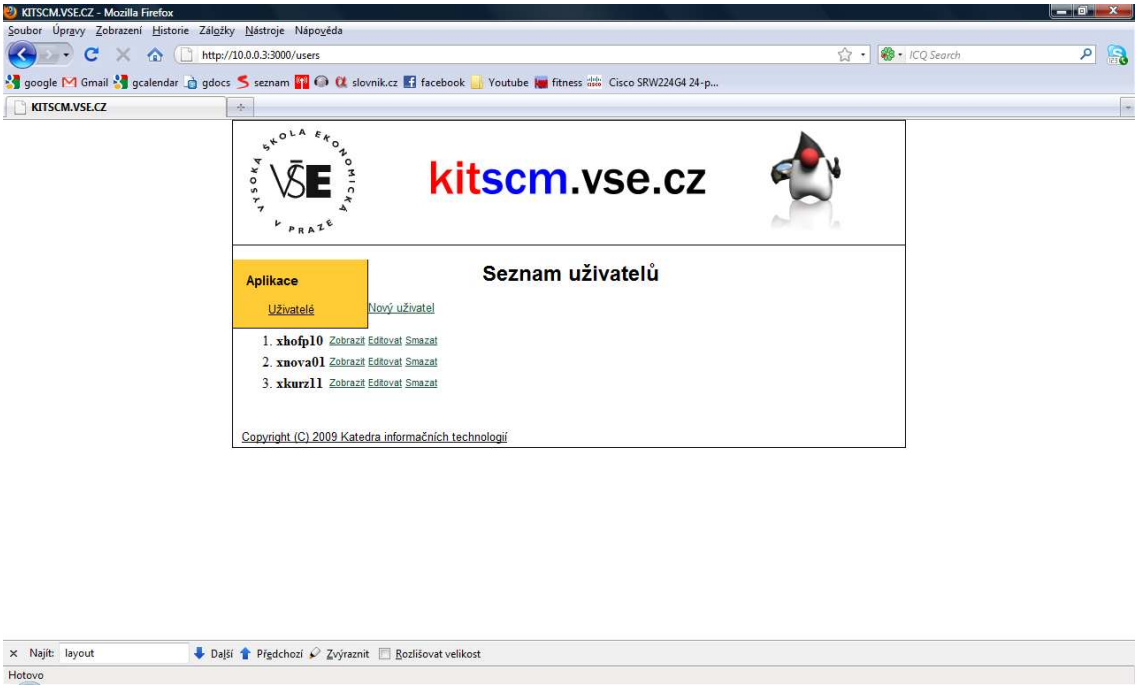
Jedná se o standartní css, kterým se formátuje jak formát celé stránky, tak i jednotlivé části.

Původní design – bez layoutu:



Obrázek 61 - Původní design

Lze porovnat s novým designem včetně layoutu:



Obrázek 62 - Design včetně layoutu[1]

Touto kapitolou končí i celá kapitola 4 - vývoj, jejímž cílem byla ukázka způsobu vývoje Rails aplikace. Jakékoli rozšíření funkcionality aplikace pro týmovou spolupráci a její další vývoj, bude probíhat podobným způsobem.

6. KAPITOLA 5 - ZÁVĚR

Cíle této kapitoly:	... popsat vývoj a předání aplikace ... ukázat koncový stav aplikace ... zhodnotit celkový projekt a dipl. práci

6.1 POPIS VÝVOJE A PŘEDÁNÍ APLIKACE

Aplikaci jsem začal vyvíjet v září roku 2008. Na svém laptopu jsem pomocí VMWARE zřídil virtuální ubuntu server, na kterém jsem vytvořil testovací prostředí. První verze aplikace byla hotova koncem prosince 2008.

V lednu 2009 proběhlo nasazení na školní server kitscm.vse.cz. Tento server běží na linuxové distribuci Gentoo. Správce serveru (Ing. Luboš Pavlíček) provedl instalaci Ruby, Rubygems, Rails a všech potřebných knihoven. Zdrojové kódy byly přeneseny z flash paměti (archiv tar.gz). Dále mně byl založen vzdálený přístup (ssh) na tento server.

[1] včetně souboru public/images/banner.png

Během roku 2009 se změnilo zadání celého projektu – požadavky na aplikaci. Úpravy, customizace a další vývoj tedy probíhaly již přímo na školním serveru a na produkčním LDAP serveru VŠE (v testovacím prostředí ou=test,dc=vse,dc=cz).

Milníkem byl začátek listopadu 2009, kdy jsme se s vedoucím diplomové práce Ing. Lubošem Pavlíčkem domluvili na finální podobě aplikace – na seznamu požadavků (tabulka 4 kapitola 3.3), po jejichž splnění aplikace mohla být předána.

6.2 KONCOVÝ STAV APLIKACE

Koncový stav aplikace splňuje všechny požadavky definované v tabulce 4 (kapitola 3.3).

Zdrojové kódy jsou k dispozici na serveru sourceforge.net - https://sourceforge.net/projects/kitscm/

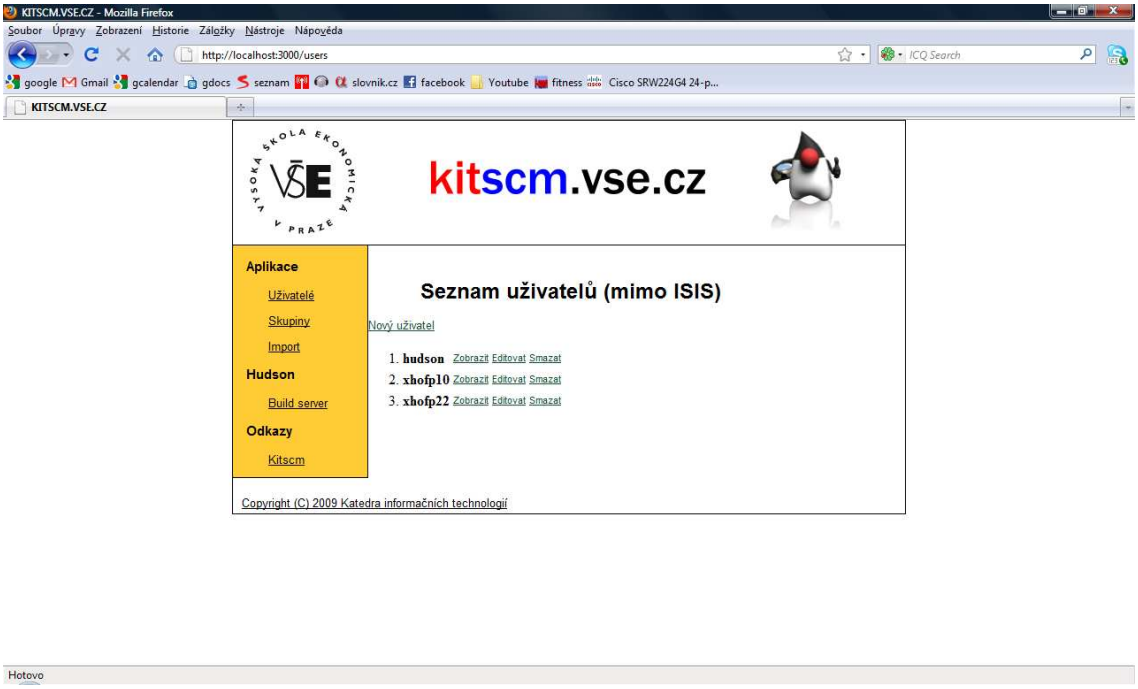
, kde si je případní zájemci mohou stáhnout a ve vývoji pokračovat.

Vyvinutou aplikaci nejvýstižněji popíší obrazovky.

Úvodní stránka - přihlášení do aplikace:

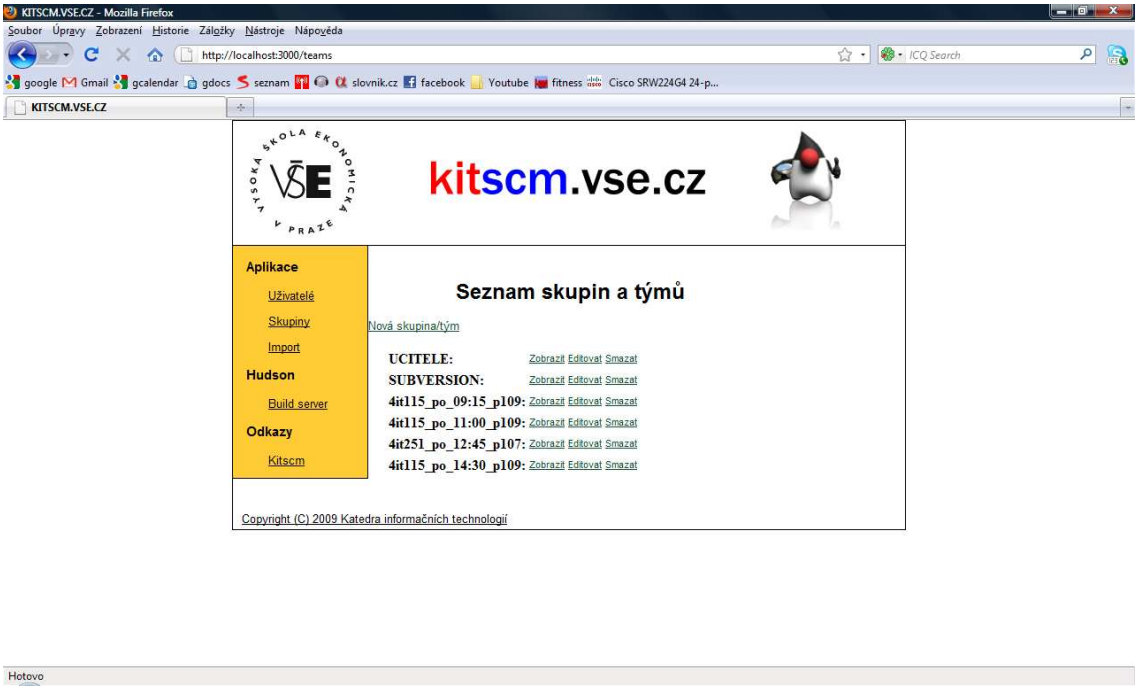
Obrázek 63 - Přihlašovací obrazovka

Sekce uživatelé slouží pro evidenci externích uživatelů (nejsou evidováni v systému ISIS):



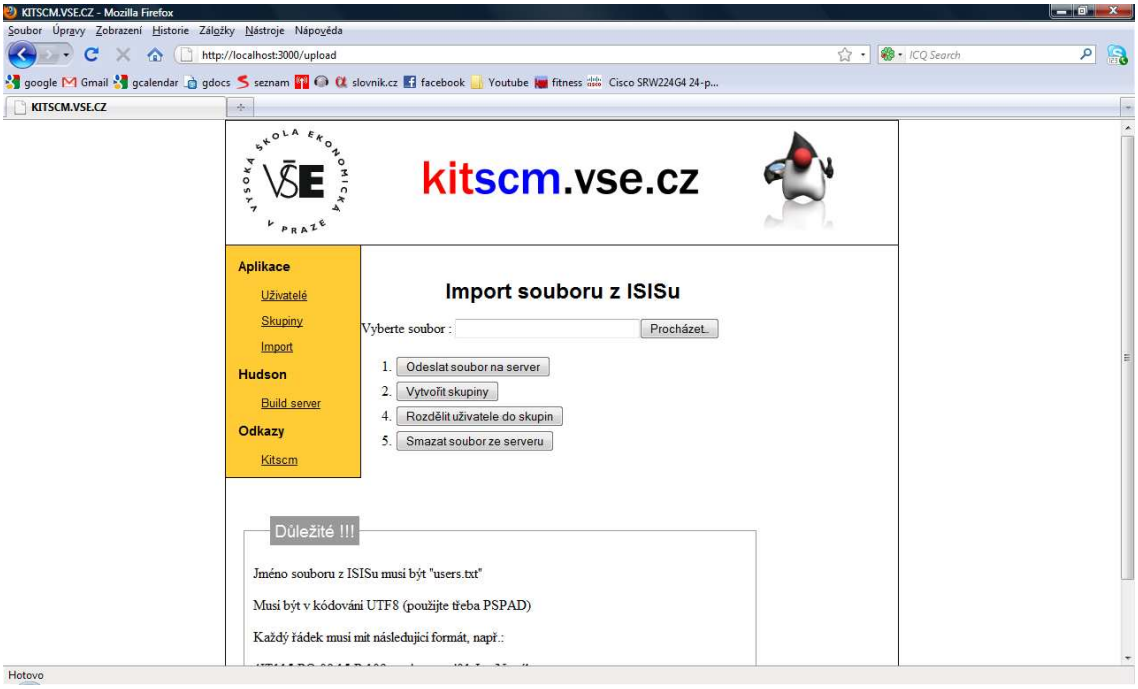
Obrázek 64 - Uživatelé

V sekci skupiny se evidují vyučující skupiny, uživatelské týmy a dvě speciální skupiny, skupina UČITELÉ a skupina SUBVERSION.



Obrázek 65 - Skupiny a týmy

V sekci import je možné do aplikace importovat seznam uživatelů ze systému ISIS:



Obrázek 66 - Import souboru z ISIS

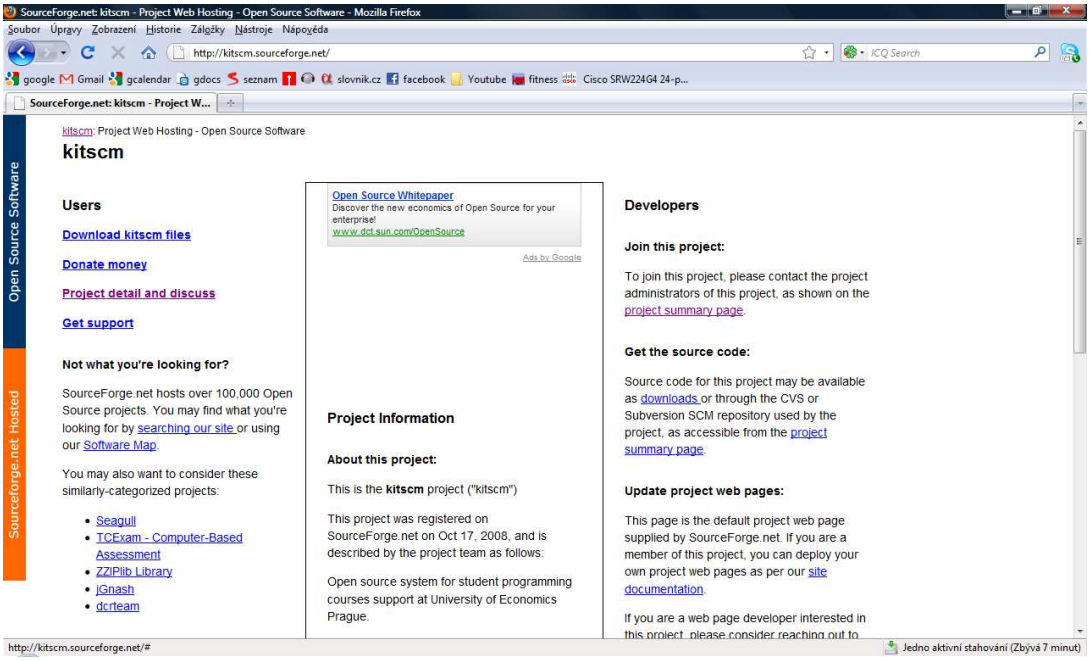
6.3 ZHODNOCENÍ PROJEKTU A DIPLOMOVÉ PRÁCE

Hlavním cílem této diplomové práce a celého projektu byla podpora týmové spolupráce na KIT VŠE v Praze. Tohoto cíle mělo být dosaženo vývojem aplikace pro podporu týmové spolupráce a implementací této aplikace na školní server kitscm.vse.cz.

Aplikace ulehčuje práci spojenou s organizací kurzů programování KIT, především automatizací řady úloh, které se doposud vykonávaly manuálně.

Hlavním přínosem je dokončená první verze aplikace. Zahájení celého projektu, analýza a návrh aplikace, vytvoření a popis provozní architektury a vývoj první verze aplikace, byly nejtěžším a časově nejnáročnějším úkolem na vývoji projektu. Případní nástupci (další vývojáři), kteří budou aplikaci rozšiřovat, již budou mít k dispozici fungující verzi aplikace, hotovou analýzu, kompletní dokumentaci provozní architektury a popsané know-how z vývoje aplikace.

Veškeré zdrojové kódy aplikace jsou k dispozici také na http://sourceforge.net/projects/kitscm/ pod GPL licencí.



Obrázek 67- http://sourceforge.net/projects/kitscm/

7. ZDROJE

1. **Pavlíček, Luboš.** kitscm.vse.cz. *kitscm.vse.cz.* [Online] VŠE v Praze. http://kitscm.vse.cz.

2. **Stewart, Brude.** An Interview with the Creator of Ruby. *O´Reilly Media.* [Online] 29. 11 2001. http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html.

3. **Slagell, Mark.** What is ruby? *Ruby User's Guide.* [Online] 2005-2008. http://www.rubyist.net/~slagell/ruby/.

4. Ruby (programovací jazyk). *Wikipedïe Otevřená encyklopedie.* [Online] http://cs.wikipedia.org/wiki/Ruby_(programovac%C3%AD_jazyk).

5. Ruby (programming language). *Wikipedia The Free Encyclopedia.* [Online] http://en.wikipedia.org/wiki/Ruby_(programming_language).

6. TIOBE Programming Community Index. [Online] 10 2008. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

7. **Sasada, Koichi.** YARV: Yet Another Ruby VM. [Online] http://www.atdot.net/yarv/.

8. JRuby Java powered Ruby implementation. [Online] 2006. http://jruby.codehaus.org/.

9. **Phoenix, Evan.** Rubinius . *Ruby, the Way It Was Meant To Be.* [Online] 2007-2008. http://rubini.us/.

10. Ironruby. *www.rubyforge.org.* [Online] http://rubyforge.org/projects/ironruby/.

11. Mac Ruby. [Online] 2008. http://www.macruby.org/trac/wiki/MacRuby.

12. RubyGems User Guide. *www.rubygems.org.* [Online] http://www.rubygems.org/read/chapter/1.

13. RubyForge. *Wikipedia The Free Encyclopedia.* [Online] http://en.wikipedia.org/wiki/RubyForge.

14. *Ruby on Rails.* [Online] http://rubyonrails.org/.

15. *Merb.* [Online] 2008. http://merbivore.com/.

16. Nitro (web framework). *Wikipedia The Free Encyclopedia.* [Online] 2008. http://en.wikipedia.org/wiki/Nitro_(web_framework).

17. Camping, the Documentation. *www.rubyforge.org.* [Online] http://camping.rubyforge.org/files/README.html.

18. Ruby on Rails. *Wikipedie Otevřená Encyklopedie.* [Online] http://cs.wikipedia.org/wiki/Ruby_on_Rails.

19. The Top Five Technologies You Need to Know About in '07. *www.computerworld.com.* [Online] March. 1 2007. http://computerworld.com/action/article.do?command=viewArticleBasic&articleId=9011969#rails.

20. Framework. *Wikipedie Otevřená Encyklopedie.* [Online] http://cs.wikipedia.org/wiki/Framework.

21. Ruby on Rails. *Wikipedia The Free Encyclopedia.* [Online] http://en.wikipedia.org/wiki/Ruby_on_rails.

22. **Minařík, Karel.** Efektivni vyvoj webovych aplikaci v Ruby on Rails (Prezentace z přednášky na konferenci Webexpo). *www.slideshare.net.* [Online] http://www.slideshare.net/karmi/efektivni-vyvoj-webovych-aplikaci-v-ruby-on-rails-webexpo-presentation?nocache=4364.

23. Agile Manifesto. *Wikipedia The Free Encyclopedia.* [Online] http://en.wikipedia.org/wiki/Agile_Manifesto.

24. Manifesto for Agile Software Development. [Online] 2001. http://agilemanifesto.org/.

25. Model-view-controller. *Wikipedie Otevřená encyklopedie.* [Online] http://cs.wikipedia.org/wiki/MVC.

26. **Thomas, Dave.** *Agile Web Development with Rails, 2Ed.* United States of America : The Pragmatic Programmers LLC, 2006. 0-9776166-3-0.

27. Rake -- Ruby Make. *www.rubyforge.org.* [Online] http://rake.rubyforge.org/.

28. *Capistrano: Home.* [Online] http://www.capify.org/.

29. Active record pattern. *Wikipedia The Free Encyclopedia.* [Online] Wikimedia Foundation. http://en.wikipedia.org/wiki/Active_record_pattern.

30. Object relational mapping. *Wikipedia The Free Encyclopedia.* [Online] Wikimedia Foundation. http://en.wikipedia.org/wiki/Object-relational_mapping.

31. **Fielding, Roy Thomas.** Architectural Styles and the Design of Network-based Software Architectures. [Online] 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

32. **Řepa, Václav.** *Analýza a návrh informačních systémů.* Praha : EkoPress, 1999. ISBN 80-86119-13-0 .

33. BPMN. [Online] Object Management Group, Inc. http://www.bpmn.org/.

34. PowerDesigner Data Modelling Software Tool . [Online] Sybase Inc. http://www.sybase.com/products/modelingdevelopment/powerdesigner.

35. RESTful Rails [Online] SitePoint Solutions

http://www.sitepoint.com/blogs/2008/02/04/restful-rails-part-i/

8. PŘEHLED OBRÁZKŮ

9. TERMINOLOGICKÝ SLOVNÍK

Termín	Zkratka	Význam [zdroj]
Ruby on Rails	RoR	Framework pro vývoj webových aplikací založený na jazyku Ruby [http://cs.wikipedia.org/wiki/Ruby_on_Rails]
Objektově orientované programování	OOP	Metodika vývoje softwaru. [http://cs.wikipedia.org/wiki/Objektov%C4%9B_orientovan%C3%A9_programov%C3%A1n%C3%AD]
Skript		Zdrojový kód programu, který je čten a vykonáván za běhu [http://cs.wikipedia.org/wiki/Skript_%28program%29]
Interpret		Speciální počítačový program, který umožňuje přímo vykonávat (interpretovat) zápis z jiného programu v jeho zdrojovém kódu ve zvoleném programovacím jazyce. Program není nutné přenášet do strojového kódu cílového procesoru (na rozdíl od překladače). [http://cs.wikipedia.org/wiki/Interpret_%28software%29]
Informační a studijní systém VŠE Praha	ISIS	Informační a studijní systém Vysoké školy ekonomické v Praze [http://isis.vse.cz]
CreateReadUpdateDelete	CRUD	Vytvoření, načtení, aktualizace a smazání záznamu. Čtyři základní funkce persistentního uložení dat. [http://en.wikipedia.org/wiki/Create,_read,_update_and_delete]
Linux, Apache, MySql, PHP	LAMP	Zkratka, která v informatice označuje sadu svobodného softwaru používaného jako platforma pro implementaci dynamických webových stránek . [http://cs.wikipedia.org/wiki/LAMP]

10.PŘÍLOHY

10.1 PŘÍLOHA 1 – OPEN LDAP SCHEMA KITSCM.SCHEMA.

```
#########################################################
# TEAMS - ATTRIBUTY
#########################################################

#course char(6)
attributetype ( 1.3.6.1.4.1.4203.666.383.101
        NAME 'vseCourse'
        DESC 'Vyukovy kurs'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{6} SINGLE-VALUE )

#garant char(30)
attributetype ( 1.3.6.1.4.1.4203.666.383.102
        NAME 'vseGarant'
        DESC 'Jmeno garanta skupiny'
        SUP member
        )

#subversion (withweb-withoutweb-no) - nahrazeno char(10)
attributetype ( 1.3.6.1.4.1.4203.666.383.103
        NAME 'vseSubversion'
        EQUALITY booleanMatch
        DESC 'Zda vytvorit subversion repositare'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE  )

attributetype ( 1.3.6.1.4.1.4203.666.383.104
        NAME 'vseSubversionRO'
        EQUALITY booleanMatch
        DESC 'Zda je pro cleny tymu pristup pouze RO'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE  )


#teamName varchar(30)
attributetype ( 1.3.6.1.4.1.4203.666.383.200
        NAME 'vseTeamName'
        DESC 'Nazev tymu'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{30} SINGLE-VALUE )

#description varchar(255)
attributetype ( 1.3.6.1.4.1.4203.666.383.201
        NAME 'vseTeamDescription'
        DESC 'Popis tymu'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255} SINGLE-VALUE )

#wikiproject (true-false)- boolean
attributetype ( 1.3.6.1.4.1.4203.666.383.204
        NAME 'vseWikiproject'
        EQUALITY booleanMatch
        DESC 'Zda zalozit wiki projekt'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE )

#faze (projekt-inspekce-implementace) - nahrazeno char(15)
attributetype ( 1.3.6.1.4.1.4203.666.383.206
        NAME 'vseFaze'
        DESC 'Faze projektu'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{15} SINGLE-VALUE )

#clenoveTymu / 1:M
attributetype ( 1.3.6.1.4.1.4203.666.383.209
        NAME 'vseClenTymu'
        DESC 'Vsichni clenove patrici do tymu'
```

```
        SUP member
        )

#inspektori projektu / 1:M
attributetype ( 1.3.6.1.4.1.4203.666.383.210
        NAME 'vseInspektoriTymu1'
        DESC 'inspektori tymu 1'
        SUP member
        )

#inspektori projektu / 1:M
attributetype ( 1.3.6.1.4.1.4203.666.383.211
        NAME 'vseInspektoriTymu2'
        DESC 'inspektori tymu 2'
        SUP member
        )

attributetype ( 1.3.6.1.4.1.4203.666.383.212
        NAME 'vseSubversionURL'
        DESC 'URL uloziste subversion'
        SUP labeledURI
        )

attributetype ( 1.3.6.1.4.1.4203.666.383.213
        NAME 'vseGroup'
        DESC 'idenfikace skupiny, do ktere patri projekt'
        SUP member
        )

########################################################
#TEAMS - OBJECT_CLASS
#OID 1.3.6.1.4.1.4203.666.383.xxx
########################################################

objectClass ( 1.3.6.1.4.1.4203.666.383.600
    NAME 'x-cz-vse-kitscm-team'
    DESC 'Tym uzivatelu systemu kitscm VSE'
    SUP top STRUCTURAL
        MUST ( vseTeamName )
        MAY  ( vseClenTymu $
                vseTeamDescription $ vseCourse $ vseGarant $ vseSubversion
$
                vseSubversionRO $ vseWikiproject $ vseFaze $
                vseInspektoriTymu1 $ vseInspektoriTymu2 )
        )

objectClass ( 1.3.6.1.4.1.4203.666.383.601
    NAME 'x-cz-vse-kitscm-team2'
    DESC 'Tym uzivatelu systemu kitscm VSE'
    SUP groupOfNames STRUCTURAL
        MAY  ( vseCourse $ vseGarant $ vseSubversion $ vseGroup $
               vseSubversionRO $ vseWikiproject $ vseFaze $
               vseInspektoriTymu1 $ vseInspektoriTymu2 $ vseSubversionURL )
        )
```

10.2 PŘÍLOHA 2 - .LDIF KÓD S DATY OPENLDAP

```
dn: ou=People,dc=vse,dc=cz
objectClass: top
objectClass: organizationalUnit
objectClass: domainRelatedObject
associatedDomain: kitscm
ou: People

dn: uid=hudson,ou=People,dc=vse,dc=cz
objectClass: posixAccount
objectClass: top
objectClass: inetOrgPerson
```

```
objectClass: organizationalPerson
objectClass: person
cn: hudson
gidNumber: 0
homeDirectory: /home/hudson
mail: hudson@vse.cz
sn: hudson
uid: hudson
uidNumber: 0
userPassword:: aGVzbG8y

dn: uid=xhofp10,ou=People,dc=vse,dc=cz
objectClass: posixAccount
objectClass: top
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
cn: Petr
gidNumber: 0
homeDirectory: /home/xhofp10/
sn: Hofmann
uid: xhofp10
uidNumber: 0
userPassword:: e1NIQX1keEtNODJmQ0dmK2o
```

10.3 PŘÍLOHA 3 – CSS STYL

```
/* Styly pro users */
#page {
        width: 760px;
        margin: 0 auto 0 auto;
        border: 1px solid black;
        background: white;
}
#banner {
  height: 140px;
  background: white;
  text-align: center;
  color: white;
  font: 200% sans-serif;
  font-weight: bold;
  border-bottom: 1px solid black;
  width: 760px;
}
#banner img {
 float: left;
 max-width: 760px;
 max-height: 140px;
 }
#columns {
  background: white;
}
#main {
  background: white;
  width: 79%;
  margin-left: 10px;
}
#side {
  width: 20%;
  float: left;
  background: #ffcc35;
  border-right: 1px solid black;
  border-bottom: 1px solid black;
  font: 80% sans-serif;
}
```

```
#side a:link {
color: black;
}
#side a:visited {
color: black;
}
#side a:hover {
text-decoration: none;
color: black;
}
#side h3 {
  color: black;
  font-weight: bolder;
  font: sans-serif;
  padding-left: 1em;
}
#main h1 {
     font: 150% sans-serif;
     text-align: center;
     font-weight: bolder;
}
#main table {
padding-left: 20px;
}
#main table a {
color: #165035;
font: 70% sans-serif;
}
#main table a:hover {
text-decoration: none;
}
#levy {
font-weight: bold;
padding-right: 10px;
}
#tlacitko {
}
#tlacitko a {
color: #EB0438;
font-weight: bold;
font: 120% sans-serif;
}
#tlacitko a:hover {
text-decoration: none;
}
#end {
        height:40px;
        clear: both;
        background: white;
}
#end a {
color: black;
font: 80% sans-serif;
margin-left: 10px;
}
#end a:hover {
text-decoration: none;
}
#main fieldset {
margin: 10px;
border: 1px solid #999999;
}
#main fieldset legend {
min-width: 120px;
max-width: 300px;
background: #999999;
padding: 5px;
margin-left: 1em;
color: white;
font: 120% sans-serif;
```

```
}
#main p a {
color: #165035;
font: 80% sans-serif;
}
#main p a:hover {
color: green;
text-decoration: none;
}
#main ol {
margin: 10px;
}
#main vystraha {
background: red;
}
```