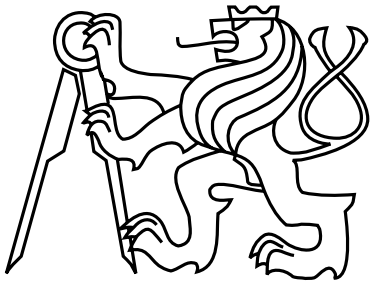
Systém pro evidenci hospitací v Ruby on Rails

Rozšiřte prototyp aplikace pro evidenci hospitací (http://kvalitavyuky.felk.cvut.cz/) tak, aby jej bylo možné nasadit do reálného provozu na ČVUT FEL. Systém implementujte na platformě Ruby on Rail. Vývoj provádějte iterativním způsobem a postupně zapracovávejte požadavky zadavatele. Celý vývoj řádně dokumentujte a výsledky své práce otestujte. Funkčnost systému demonstrujte na hospitacích, které budou probíhat v programu STM v LS 2011/2012.

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Systém pro evidenci hospitací v Ruby on Rails

Tomáš Turek

Vedoucí práce: Ing. Martin Komárek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

7. února 2012

Poděkování

Zde můžete napsat své poděkování, pokud chcete a máte komu děkovat.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V České Lípě dne 5. 2. 2012 .....

Abstract

Translation of Czech abstract into English.

Abstrakt

Tato práce se zabývá rozšířením prototypu aplikace pro spravování hospitací, tak aby jej bylo možné nasadit do reálného provozu na ČVUT FEL. Systém má za účel zjednodušit současnou administrativní náročnost hospitací.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Na FEL ČVUT byl zaveden pro studijní program STM (Softwarové technologie a management) systém pro ověřování kvality výuky. Cílem tohoto systému je zkvalitnit vyučované předměty. Jedním ze zdrojů informací jsou kontrolní návštěvy ve výukách (hospitace). Účelem těchto návštěv je získání komplexního obrazu o kvalitě výuky pro garanty (vedoucí) jednotlivých předmětů. A zároveň slouží pro pedagogy jako zpětná vazba z přednášek a cvičení.

Cílem mé práce je navrhnout a vytvořit informační systém pro správu hospitací, který zjednoduší a zrychlí administrativu, ke které se doposud používala e-mailová komunikace a www stránky Rady programu.

Kapitola 2

Analýza a návrh řešení

Tato kapitola pojednává o analýze a návrhu vhodného řešení aplikace. Výstupem této analýzy jsou funkční a nefunkční požadavky. Dále návrh a popis domén a nejdůležitější případy užití s aktéry.

2.1 Požadavky

Požadavky na systém se dělí na dvě sekce: obecné a funkční požadavky. Pro definování těchto požadavků jsem vycházel z oficiálního zadání práce, protože mi přesně definuje návrh aplikace a pro implementaci systému i potřebné technologie.

2.1.1 Obecné požadavky

Obecné požadavky se netýkají funkčnosti, ale celkového návrhu a použitých technologií.

1. Systém bude postaven na webovém frameworku Ruby on Rails.
2. Systém bude používat databázi MySQL.
3. Systém bude webovou aplikací.
4. Serverová část systému poběží na aplikačním serveru Apache HTTP server.
5. Systém bude používat webovou službu KOSapi.
6. Systém bude pro autentizaci používat FELid.

2.1.2 Funkční požadavky

Tato sekce se zabývá požadavky na funkčnost systému.

1. Systém umožní spravovat uživatele.
2. Systém umožní průběžné plánování hospitaci.

3. Systém umožní hospitujícímu i hospitovanému prohlížet hospitace.
4. Systém umožní vystavit závěrečné hodnocení na veřejné části aplikace.
5. Systém umožní hospitovanému sepsat stanoviska k názorům hospitujícího.
6. Systém umožní hospitujícímu nahrát naskenovaný dokument hodnocení výuky.
7. Systém umožní hospitujícímu napsat slovní hodnocení z výuky.
8. Systém umožní hospitujícímu napsat závěrečné shrnutí hospitace.
9. Systém bude odesílat emailem zprávy o naplánování hospitací a jednotlivé hodnocení a stanoviska příslušným osobám.
10. Systém umožní vyhledávat předměty z KOSapi.
11. Systém umožní vyhledávat osoby z KOSapi.

2.2 Uživatelské role

V systému je celkem 6 uživatelských rolí, ty jsou odvozeny z modelu aktérů na obrázku 3.1. Máme tři základní uživatelské role, které jsou základem systému - přihlášený uživatel, nepřihlášený uživatel a administrátor hospitací. Další role, jako hospitovaný a hospitující uživatel, jsou přiděleny příslušným osobám zainteresovaných v jednotlivých hospitacích.

V systému jsou následující role:

2.2.1 Nepřihlášený uživatel

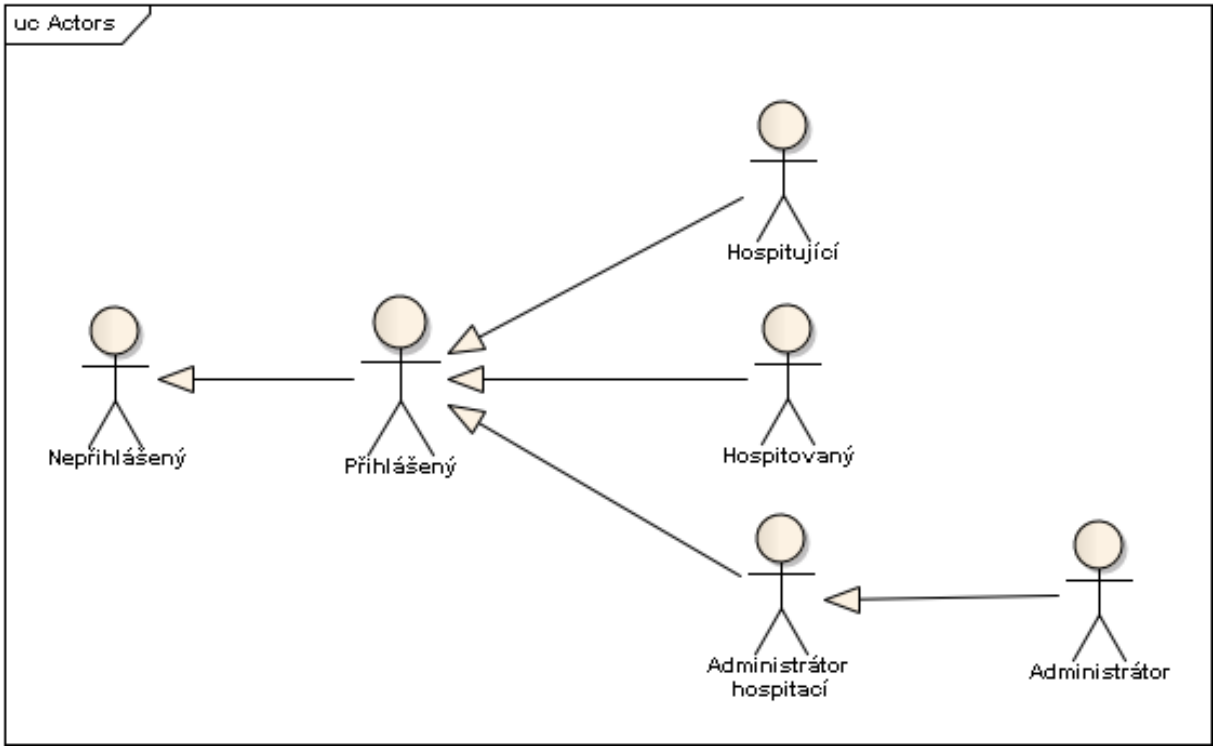
Nepřihlášený uživatel je role pro hosty naší aplikace. V systému má ze všech rolí nejmenší pravomoc. V tomto stavu je každý uživatel, který se doposud nepřihlásil do systému a je mu umožněno vykonat následující akce: přihlásit se, zobrazit seznam naplánovaných veřejných hospitací na aktuální a následující semestr, má také přístup k závěrečným shrnutím hospitací.

2.2.2 Přihlášený uživatel

Přihlášený uživatel vychází z role nepřihlášeného uživatele. Je to uživatel, který má vytvořený účet v systému a již se přihlásil. Tato role přidává možnost odhlásit se a procházet veškeré ohlášené hospitace. Jedná se o roli základní pro všechny další role, které z ní vychází.

2.2.3 Hospitovaný

Hospitovaný je role pro přihlášeného uživatele v systému. Je přidělena pro každého vyučujícího, který vyučuje předmět, na němž byla naplánovaná hospitace. Tato role přidává možnost přístupu k naplánovaným hospitacím, kde figuruje jako hospitovaný. U těchto hospitací má právo na zobrazení všech hodnotících dokumentů a umožní mu sepsat dokument se stanovisky k názorům hospitujícího.



Obrázek 2.1: Aktéři

2.2.4 Hospitující

Hospitující je role pro přihlášeného uživatele v systému. Přiděluje ji administrátor hospitací osobám, které mají za úkol vykonat hospitaci. U přidělených hospitací má uživatel právo na zobrazení informací o jeho hospitaci a povinnost sepsat hodnotící dokumenty z vykonané hospitace.

2.2.5 Administrátor hospitací

Tato role patří mezi nejdůležitější role v systému, protože spravuje hospitace a přiděluje uživatelům nové role. Hlavním úkolem této role je plánovat hospitace na předměty a posléze je spravovat.

2.2.6 Administrátor

Administrátor je super uživatel, který má nejvyšší pravomoc v systému. Má přístup ke všem zdrojům aplikace a může nastavovat aplikaci.

2.3 Doménový model

Doménový model na obrázku 3.2 reprezentuje klíčové domény systému a jejich vzájemné vztahy.

Popis domény jsem pro přehlednost rozdělil podle zdroje na dvě základní skupiny. V první skupině jsou domény, které používám z KOSapi a druhou skupinou jsou domény specifické pro moji aplikaci.

2.3.1 Domény z KOSapi

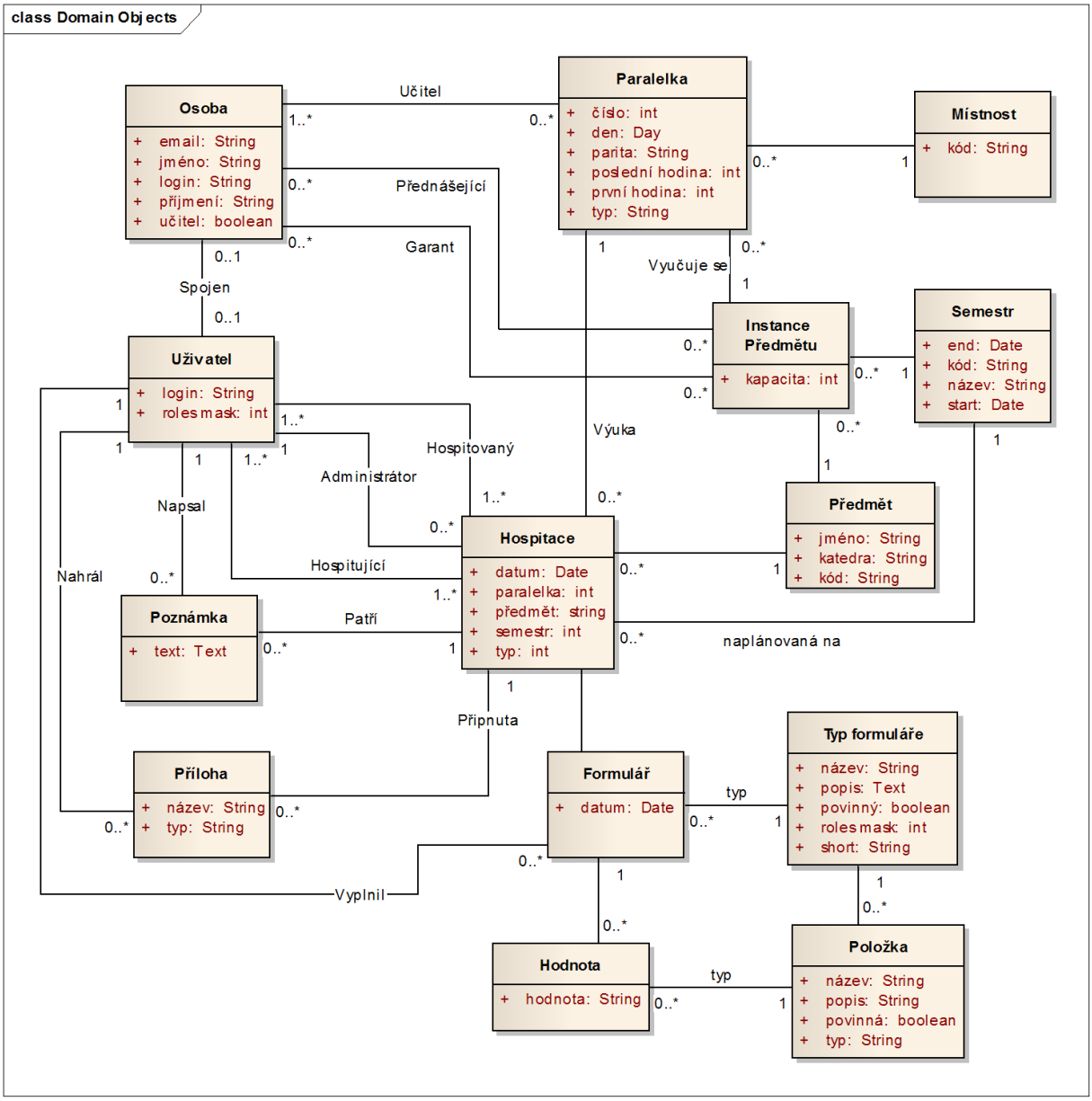
- Osoba - informace o osobě nacházející se na FEL. Každá osoba může být učitelem a studentem.
- Semestr - informace o jednotlivých semestrech.
- Předmět - popis jednotlivých předmětů vyučovaných na FEL.
- Instance předmětu - jsou instance předmětu vypsané v konkrétním semestru.
- Paralelka - je vypsaná rozvrhová paralelka pro instanci předmětu.
- Místnost - informace o místech, kde probíhá výuka předmětů

2.3.2 Domény aplikace

- Uživatel - obsahuje dodatečné informace a role pro osobu v aplikaci.
- Hospitace - obsahuje informace o naplánování hospitace.
- Poznámka - je textová poznámka pro hospitaci napsaná uživatelem.
- Příloha - je připojený datový soubor k hodnocení hospitace.
- Formulář - je vyplněný formulář pro hospitaci.
- Hodnota - je vyplněná hodnota jedné formulářové položky.
- Typ formuláře - udává formát dokumentu, který se používá pro hospitace.
- Položka - je šablona jedné položky ve formuláři.

2.4 Životní cyklus hospitace

Cílem této části analýzy je popsat životní cyklus, kterým hospitace prochází.



Obrázek 2.2: Doménový model

2.4.1 Vytvoření

Životní cyklus hospitace začíná jejím vytvořením. Toto zajišťuje administrátor hospitací, který založí hospitaci a definuje semestr, kdy se má hospitace uskutečnit, a předmět vyučovaný na fakultě. Při vytváření hospitace se určí typ hospitace a tím i její způsob zviditelnění, pro ostatní aktéry v aplikaci.

2.4.2 Naplánování

Při plánování je také hlavním aktérem administrátor hospitace. V této části životního cyklu administrátor určí hospitovanou paralelku předmětu a datum, kdy se hospitace uskuteční.

Administrátor také v této fázi přidělí hospitující z řad pedagogů určených k vykonání hospitace.

2.4.3 Hodnocení

Poté, co proběhla kontrola hospitace, začíná nová fáze, ve které se hodnotí vyučování. Do této fáze už nezasahuje administrátor hospitace, ale přicházejí na scénu dva jiní aktéři: hospitovaný a hospitující.

V první fázi musí hospitující vyplnit, nebo nahrát naskenovaný formulář pro Hodnocení výuky při hospitaci. Tento formulář slouží k dokumentaci průběhu hospitace.

V druhé fázi jeden z hospitujících sepíše slovní hodnocení hospitační návštěvy.

Ve třetí fázi může hospitovaný do dvou dnů vyplnit stanovisko hodnoceného k názorům hospitujícího.

V poslední fázi jeden z hospitujících sepíše poslední formulář Závěrečné shrnutí. Po vyplnění tohoto formuláře se hospitace stává ukončenou a tím končí její životní cyklu.

2.5 Technologie

Tato část popisuje jednotlivé technologie potřebné pro implementaci aplikace.

2.5.1 Ruby on Rails

Ruby on Rails, zkráceně Rails, je jedno z implementačních omezení, které se nachází přímo v zadání práce. Je to framework postavený na jazyce Ruby a je primárně určen pro tvorbu webových aplikací napojených na databázi. Rails je postaven na návrhovém vzoru model-view-controller. Tento framework používá dva hlavní principy. Prvním principem je Convention over Configuration a druhým je Don't Repeat Yourself.

2.5.2 KOSapi

KOSapi je webová služba poskytující aplikační rozhraní v podobě RESTful webové služby. Je určená pro vznik školních aplikací, které potřebují mít přístup k datům souvisejících s výukou. Pro instance FEL a FIT čerpá služba data z KOSu.

Z této služby čerpám hlavně data předmětů a osob v KOSu. Pro připojení ke KOSapi jsem rozšířil již existující knihovnu napsanou v Ruby pány Tomášem Linhartem a Tomášem Jukínem ve školním projektu VyVy.

2.5.3 FELid

FELid je globální autentizační a autorizační systém pro webovské aplikace na síti FEL. Poskytuje jednotný a bezpečný způsob přihlášení uživatelů a přenos jejich údajů do různých aplikací na webu. Zároveň podporuje jednorázové přihlášení (tzv. single sign-on). Znamená to, že se uživatel přihlašuje pouze do první použité aplikaci a u dalších aplikací už nemusí zadávat svoje přihlašovací údaje.

Tuto službu používám pro autentizaci uživatelů do systému. Abych mohl používat v aplikaci FELid je nutné splnit technické požadavky, které jsou napsány na stránkách FELid.

2.5.4 Aplikační server

Pro zprovoznění aplikace do reálného provozu jsem použil webový server Apache HTTP server ve verzi 2. Tato verze webového serveru totiž umožňuje instalaci zásuvných modulů Passenger a Shibboleth. Passenger umožňuje nasazení rails aplikací na serveru. Druhý modul Shibboleth zprostředkovává single sign-on autentizaci mezi aplikací a službou FELid.

2.5.5 Databáze

Ruby on Rails poskytuje možnost připojení k různým databázovým systémům prostřednictvím adaptérů. Díky tomu není aplikace závislá na použitém databázovém systému a díky tomu mohu používat pro vývoj a testování aplikace jednoduchý databázový systém SQLite, který pro tyto účely bohatě postačuje a není potřeba jej složitě konfigurovat. Pro samotné nasazení aplikace do provozu už používám databázový systém MySQL.

2.6 Architektura

V této části popisuji architektonické vzory a konvence používané pro vývoj aplikace.

2.6.1 MVC

MVC (Model-view-controller) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.

Ruby on Rails obsahuje ve svém jádru tuto architekturu, proto je její implementace automatická.

2.6.2 DRY

DRY (Don't repeat yourself) je princip vývoje softwaru zaměřený na snížení opakování psaní stejného kódu a tím zvyšuje čitelnost a znovupoužitelnost kódu. To znamená, že informace se nacházejí na jednoznačném místě. Pro příklad Ruby on Rails získává definici sloupců pro třídu modelu přímo z databáze.

2.6.3 CoC

CoC (Convention over Configuration) je další princip používaný v Rails pro zlepšení čitelnosti a znovupoužitelnosti kódu. Tento princip znamená, že konvence má přednost před konfigurací a to tak, že Rails předpokládá to, co chcete udělat, místo toho, aby vás nutil specifikovat každou drobnost v konfiguraci.

2.6.4 REST

REST (Representational State Transfer) je architektonický vzor pro webové aplikace. Je založen na HTTP protokolu a hlavní myšlenkou je poskytovat přístup ke zdrojům dat. Všechny zdroje jsou identifikovány přes URI. REST definuje čtyři základní metody pro přístup ke zdrojům. Jde o metody POST, GET, PUT a DELETE, které zprostředkovávají základní operace create, read, update a delete.

Kapitola 3

Realizace

Za účelem postupného nasazování aplikace pro letní semestr 2011/2012 byl zvolený iterační vývoj aplikace. V této kapitole popisuji cíl, postup a výstup jednotlivých iterací.

3.1 První iterace

3.1.1 Cíl

První iterace měla za cíl vytvořit základní architekturu aplikace s připojením na KOSapi a k tomu vytvořit funkční aplikaci pro naplánování hospitací.

3.1.2 Postup

3.1.2.1 Datová vrstva

Protože aplikace využívá dva datové zdroje KOSapi a databázi aplikace, bylo potřeba nejdřív vyřešit jak se připojit ke KOSapi. V této části vycházím z prototypu aplikace pro správu hospitací. Poté jsem vytvořil modely tak, aby umožnily komunikaci mezi KOSapi a databází.

3.1.2.2 Autentizace a autorizace

Pro autentizaci, v této fázi vývoje, jsem zatím neimplementoval autentizaci prostřednictvím FElid. Dočasně jsem pro tento účel použil modul authlogic.

Pro autorizaci používám modul CanCan. CanCan je modul pro Ruby on Rails, který určuje, k jakým zdrojům má daný uživatel povolený přístup. Všechna oprávnění jsou definována na jednom místě (třída Ability). Umí filtrovat controllery, views i databázové dotazy.

3.1.2.3 Uživatelské prostředí

Pro vytvoření uživatelského prostředí jsem použil již existující knihovnu Bootstrap od Twitteru. Použil jsem tuto knihovnu, protože obsahuje jak kompletní CSS tak i Javascript a je open-source. Další výhodou je popularita uživatelského prostředí.

Pro usnadnění implementace Bootstrapu do aplikace jsem zprovoznil dva moduly. První modul SimpleForm zjednoduší vytváření formulářů tak, že si předem nadefinujeme jejich styl pro Bootstrap. A druhý modul WillPaginate slouží k implementaci stránkování dat.

3.1.3 Výstup

Po ukončení této iterace vznikla nasaditelná část aplikace pro plánování hospitací.

Kapitola 4

Testování

- Způsob, průběh a výsledky testování.
- Srovnání s existujícími řešeními, pokud jsou známy.

Kapitola 5

Závěr

- Zhodnocení splnění cílů DP/BP a vlastního přínosu práce (při formulaci je třeba vzít v potaz zadání práce).
- Diskuse dalšího možného pokračování práce.