České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Portál pro doučování

Pavel Pokorný

Vedoucí práce: Ing. Božena Mannová, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

23. května 2014

Poděkování

Děkuji vedoucí mé bakalářské práce Ing. Boženě Mannové, Ph.D. za cenné rady a připomínky, jež mi poskytla. Děkuji také svým blízkým za podporu a porozumění, díky kterým jsem se mohl na vypracování této práce plně soustředit.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 23.5.2014 ….....

Abstrakt

Cílem této bakalářské práce je analýza, návrh a implementace webové portálu pro zprostředkování doučování s využitím moderních opensource technologií.

Abstract

The aim of this bachelor thesis is to analyze, design and partially implement web application for home tutoring support with the use of modern open source technologies.

Obsah

Kapitola 1 – Úvod

Úkolem této práce je vytvořit webový portál pro zprostředkování doučování. Tento portál by měl sloužit oběma stranám, to znamená lektorům i zájemcům o doučování. Lektorům, kteří doučování nabízejí, k zadávání nabídek doučování a také k údržbě vlastního profilu, který bude předmětem hodnocení jimi doučovaných studentů. Pomocí tohoto profilu bude lektor moci zvýšit svoji důvěryhodnost pro potenciální nové zájemce, kterým profil bude zároveň poskytovat bližší podrobnosti o osobě lektora.

Zájemcům o doučování pak tento portál bude poskytovat přehled o existujících nabídkách a lektorech, kteří je vystavují. Pokud by jim žádná z nabídek nevyhovovala, anebo by se s žádným z vystavujících lektorů nedohodli na přesných podmínkách, budou mít možnost vystavit inzerát se svou poptávkou po doučování a svými kontaktními údaji.

Není pochyb o tom, že na českém internetu portály s podobným zaměřením už existují. Tomu, jaká je jejich současná nabídka, co nabízejí, a jak důvěryhodně a používaně působí na případné zájemce, je věnována úvodní rešerše. Další kapitoly této práce jsou postupně věnovány analýze a návrhu takového portálu, následně je probírán popis implementace a testování jeho funkčního prototypu.

1.1 Motivace

Doučování jsem se věnoval po větší část svého studentského života. S kamarádem jsme si kdysi založili jednoduché statické webové stránky, které i přes svoji jednoduchost a absenci optimalizace pro webové vyhledávače po léta sbíraly nezanedbatelné množství přístupů. Vezmeme-li v úvahu, že lidé se obecně snaží využít prvních vyhledaných odkazů, a že naše stránky se málokdy umístily na lepší než sedmé stránce výsledků, a to i při specifických dotazech typu „doučování matematiky Praha“, lze říci, že takový stav byl velice překvapivý. Rozhodl jsem se proto prozkoumat současnou situaci a zkusit zpracovat vlastní řešení.

Kapitola 2 – Rešerše existujících produktů

První fáze analýzy byla zaměřena na průzkum doučovacích inzertních portálů, které lze v českém prostředí nalézt. Pro jejich výběr byly použity webové vyhledávače Google.com a Seznam.cz. Při vyhledávání jsem se soustředil na stránky nabízející inzerci. Osobní stránky lektorů či specializovaných agentur, stejně jako jednostranně zaměřené stránky jsem nehodnotil. Z použitelných nalezených odkazů byly dále odstraněny ty, které podle svého obsahu a vzhledu nevypadají jako aktivně udržované.

Ze zbylých portálů jsem vybral 6 reprezentativních případů, které jsem dále zkoumal, a to jak z pohledu zájemce o doučování, tak z pohledu lektora nabízejícího doučování.

2.1 Sledované vlastnosti

Zkoumané portály jsem zkoumal z následujících hledisek:

- nabízené služby
- možnost filtrování dle předmětu, úrovně či geografické lokality
- nutnost registrace zadavatele inzerátu
- umožňuje či vyžaduje registraci lektorů?
- vyžaduje platby za inzeráty, anebo za registraci lektorského profilu?
- umožňuje hodnocení jednotlivých lektorů?
- práce s kontakty na uživatele – jsou údaje o inzerentech veřejně dostupné?
- celkový vzhled
- intuitivita uživatelského rozhraní
- používanost – frekvence, s jakou jsou přidávány nové inzeráty
- umožňuje zprostředkování platby za doučování?

Pro zkoumání uživatelského rozhraní, průzkum práce s ovládáním inzerátů od neregistrovaných uživatelů a pro ověření funkčnosti aplikace bylo vytvořeno i několik testovacích inzerátů.

2.2 Popis vybraných portálů

Pro podrobnější průzkum a popis byly vybrány následující portály:

- Doučování-inzerce
- Individuální doučování
- Kvalitní doučování
- Hledáme doučování
- Naučím
- Síť Doučka.cz

Dále jsou uvedeny popisy a postřehy k jednotlivým portálům:

Doučování-inzerce

http://doucovani-inzerce.cz/

Jednoduchý, ale funkční a stále aktuální web s přibližně pěti publikovanými inzeráty na den. Nabízí filtrování podle předmětu a kraje i komentáře k jednotlivým inzerátům. Kontakt pomocí zveřejněného emailu. K žádnému z úkonů nevyžaduje registraci, administrace vlastních příspěvků řešena pomocí IP adresy, anebo vyhledání dle emailu, poté zadání hesla nastaveného při tvorbě inzerátu. Vše zdarma. Na účet na seznamu přicházejí emaily ve špatném kódování.

Individuální doučování

http://www.individualnidoucovani.cz/

Jednoduchá stránka, kontakt probíhá přes vyplnění formuláře, lektoři by se měli ozvat zpět. Objeví se zde okolo 20 inzerátů za týden. Stránky obsahují českou i slovenskou lokalizaci. Jsou poměrně mladé, byly založeny v roce 2012. Uživatelské rozhraní je místy mírně nevyhovující, například nelze vybrat oblast „celá Praha“, což překáží hlavně při vyhledávání doučování. Při pokusu o vložení inzerátu se mi vícekrát objevila nepěkné php chybové hlášení, přičemž všechna data vyplněná ve formuláři zmizela.

Kvalitní doučování

http://www.kvalitni-doucovani.cz/

Přehledná stránka, vypadá velmi živě – kolem 20 inzerátů za den. Při vkládání inzerátu lze vybrat úroveň v doučovaném předmětu, cena může být nespecifikována (možnosti jako „dohodou“, „nerozhoduje“). Bez registrace povoleno vložení jednoho inzerátu za 4 dny. Mazání inzerátů řešeno pomocí odkazu zaslaného na uvedený email. Pro vytvoření lektorského profilu či pro zvýraznění inzerátu je nutná placená registrace.

Co se týče optimalizace pro vyhledávače, google.com ji zařazuje na první místa při vyhledávání řetězců obsahujících slovo „doučování“, přestože je uživateli internetových prohlížečů v databázi WOT (Web Of Trust) označena jako „nepříliš důvěryhodná“.

Hledáme doučování

http://www.hledamedoucovani.cz/

Jedná se o malý a nepříliš využívaný portál s přibližně třemi inzeráty za týden. Registrace zde vůbec není, inzeráty jsou upravovány pomocí hesla nastaveného při jejich zadání. Dále je možné vložit k inzerátu fotografii.

Naučím

http://www.naucim.cz/

V podstatě se jedná o online databázi lektorů, neexistuje zde sekce pro vložení poptávky doučování. Lektoři se registrují zdarma, později platí provize za doučování 15Kč/hodinu (platí se zpětně). K dispozici jsou bližší údaje o lektorovi, tedy jeho věk, fotografie a hodnocení ve formě jedné až pěti hvězdiček. Jako dobrý nápad vypadá zveřejňovaný orientační rozvrh lektora a to, zda a kam je ochoten dojíždět, případně zda je možné doučování v jeho prostorách. Stránky jsou vcelku jednoduché a přehledné, i když některé komponenty jsou přehuštěné textem (popis rozvrhu, mapa Prahy).

Síť Doučka.cz

http://www.doucka.cz/, *http://www.doucovanipraha.cz/*

Tento webový portál jako jediný z nalezených nabízí moderní vzhled. První stránka (www.doucka.cz) funguje vlastně jen jako rozcestník na další regionální stránky s nabídkami doučování, které jsou součástí tohoto projektu. Uživatelské rozhraní je příjemné a intuitivní.

Na druhou stranu je třeba říci, že se nejedná o klasický inzertní portál, což s sebou nese i určité nevýhody jak pro lektory, tak pro zájemce o doučování. Jak to tedy funguje? Lektor se zaregistruje, vyplní profil a čeká na nabídky. Když přijde zájemce o doučování, má možnost si podle svých požadavků a preferencí vybrat buď konkrétního lektora, anebo odeslat požadavek na doučování. Tým z Doučka.cz pak postupně kontaktuje vhodné lektory, kteří mohou nabídku přijmout či odmítnout. V případě, že s nabídkou souhlasí, musí výměnou za kontaktní údaje zaplatit agentuře jednorázovou provizi (jejíž výše je stanovena podle klientovy představy o tom, kolik doučování potřebuje uskutečnit). Pokud nebyl lektor zákazníkem přímo vybrán, musí doučovat za fixní cenu (pro Prahu 200 Kč/hod.).

Jak je vidět, tento systém je nevýhodný především pro ty zájemce, kteří nejsou ochotni platit 200 Kč/hod. a raději by vystavili inzerát s nižší cenou a počkali, zda se někdo ozve.

Mezi nesporné výhody pak patří možnost hodnocení lektorů a přehled o tom, s kolika klienty byl ten který lektor spojen.

2.3 Vyhodnocení

Celkově lze říci, že portál s přesně takovou funkcionalitou, jakou požadujeme od toho našeho, nebyl na českém internetu nalezen. Velká část zkoumaných stránek navíc nabízí zastaralý vzhled, což zvláště pro mladého uživatele může být lehce odrazujícím faktorem.

Je však pravdou, že dva z hodnocených portálů jsou velmi propracované a i jejich vzhled působí na potenciálního uživatele profesionálně. Především portály ze sítě Doučka.cz se svým moderním vzhledem působí uživatelsky přitažlivě a příjemně. V tomto případě ovšem vstupuje do hry také faktor nastavení podmínek spolupráce. Portály ze sítě Doučka.cz jsou spíše zprostředkovací agenturou, a podle toho vypadají i jejich podmínky spolupráce, což je podle mě nejmarkantnější u určené ceny. Dále také nemusí každému lektorovi vyhovovat, že za kontakt na

zájemce o doučování musí zaplatit předem, a to někdy i tolik, kolik si vydělá za první hodinu. Přímé zprostředkování plateb ale nenabízí, v čemž je na tom stejně jako všechny ostatní analyzované portály.

Dalším zkoumaným kritériem byla možnost hodnocení lektorů. To umožňují toliko dva ze zkoumaných portálů (Doucka.cz a Naucim.cz), přičemž jen portál Doucka.cz zveřejňuje i jiné podrobnosti a komentáře, než jen pouhý aritmetický průměr, jak je tomu v případě portálu Naucim.cz.

Kapitola 3 – Analýza

V této sekci jsou nejprve analyzovány požadavky na vyvíjený portál, ty jsou dále rozvinuty do případů užití a problémová doména je následně zachycena v analytickém modelu tříd.

3.1 Specifikace funkčních a ostatních požadavků

Formalizované požadavky na softwarový systém jsou základním kamenem jeho vývoje. Pokud je systém vyvíjen na objednávku pro nějakého zákazníka, měly by být určeny ve spolupráci s ním a měly by být pevně zakotveny v případné smlouvě. Funkční požadavky odrážejí zákazníkovo očekávání toho, co by měl systém umožňovat, zatímco ostatní[1] požadavky jsou spíše zachycením různých omezení a podmínek na systém kladený, které přímo nesouvisí s jeho funkcionalitou. Podle [1] by oba tyto druhy požadavků měly být zachyceny v přesně definované a měřitelné formě.

V této práci vystupuje zadání bakalářské práce v roli zákazníka, vyplývají z něj totiž základní funkční požadavky na navrhovaný portál. Tyto funkční požadavky lze rozdělit do několika podkategorií: práce s inzeráty, hodnocení lektorů a správa uživatelů.

3.1.1 Katalog funkčních požadavků

Balíček Správa uživatelů:

Číslo požadavku	**Název**	**Popis**
REQ001	Přidání uživatele	Systém bude umožňovat přidávat do něj uživatele.
REQ002	Aktivace prostřednicvím emailu	Systém bude umožňovat aktivaci uživatelského účtu až po zadání kódu, který bude zaslán na uvedenou emailovou adresu.
REQ003	Blokace uživatelského účtu	Systém bude umožňovat dočasnou i trvalou blokaci uživatelského účtu na základě akce moderátora.
REQ004	Nahlášení nevhodného inzerátu či chování	Systém bude umožňovat nahlásit uživatele z důvodu nevhodného textu inzerátu či jiného nevhodného chování.
REQ005	Změna uživatelských údajů	Systém bude umožňovat změnit uživatelské údaje uvedené při registraci.
REQ006	Přihlašování do systému	Systém bude umožňovat uživatelům se do něj přihlásit.
REQ008	Odblokování uživatelského účtu	Systém bude moderátorům umožňovat odblokovat dočasně zablokované uživatelské účty.

Tab. 1: Specifikace požadavků - balíček Správa systému

1 V angličtině *non-functional*

Balíček Hodnocení lektorů:

Číslo požadavku	**Název**	**Popis**
REQ007	Schválení lektora	Systém bude moderátorům umožňovat schvalovat jednotlivé lektory.
REQ009	Ohodnocení lektora uživatelem	Systém bude uživatelům umožňovat ohodnotit lektora po proběhlé hodině.
REQ010	Ohodnocení lektora sebou samým	Systém bude lektorům umožňovat ohodnotit vlastní znalosti a dovednosti.
REQ011	Zobrazení hodnocení	Systém bude uživatelům umožňovat zobrazit si hodnocení jednotlivých lektorů.

Tab. 2: Specifikace požadavků - balíček Hodnocení lektorů

Balíček Práce s inzeráty:

Číslo požadavku	**Název**	**Popis**
REQ012	Vložení nabídky doučování	Systém bude lektorům umožňovat vložit nabídkový inzerát na doučování.
REQ013	Vložení poptávky	Systém bude uživatelům umožňovat vložit poptávkový inzerát.
REQ014	Prohlížení nabídek	Systém bude uživatelům (i nepřihlášeným) umožňovat prohlížet si nabídky doučování.
REQ015	Prohlížení poptávek	Systém bude lektorům umožňovat prohlížet si poptávkové inzeráty v jejich oborech.
REQ016	Reakce na inzerát	Systém bude umožňovat reagovat na všechny druhy prohlížených inzerátů a bude umožňovat zprostředkování nabízeného/poptávaného doučování.
REQ017	Smazání inzerátu	Systém bude umožňovat smazat vystavený inzerát.
REQ018	Prohlížení vlastních vystavených inzerátů	Systém bude umožňovat prohlížet si inzeráty, které uživatel sám podal.
REQ019	Editace vlastního inzerátu	Systém bude umožňovat editaci vystaveného inzerátu.

Tab. 3: Specifikace požadavků - balíček Hodnocení lektorů

3.1.2 Ostatní požadavky

Číslo požadavku	Název	Popis
REQ020	Open source	Celá aplikace bude vybudována na opensourcových technologiích.
REQ021	Perzistence	Aplikace bude data ukládat do některé opensourcové databáze.
REQ022	Kompatibilita	Webové rozhraní aplikace bude kompatibilní minimálně s prohlížeči Mozilla Firefox, Internet Explorer a Google Chrome, v nejnovějších verzích.
REQ023	Obslužná kapacita	Aplikace bude schopna obsloužit alespoň jeden uživatelský požadavek za vteřinu. Celkový počet uživatelů a inzerátů, které bude schopna pojmout, bude řádově 500 uživatelů a 2000 inzerátů.
REQ024	Testování	Aplikace bude otestována jednotkovými, integračními i uživatelskými testy. U uživatelských testů budou testovány minimálně všechny pozitivní průchody daného případu užití.

Tab. 4: Specifikace požadavků - ostatní požadavky

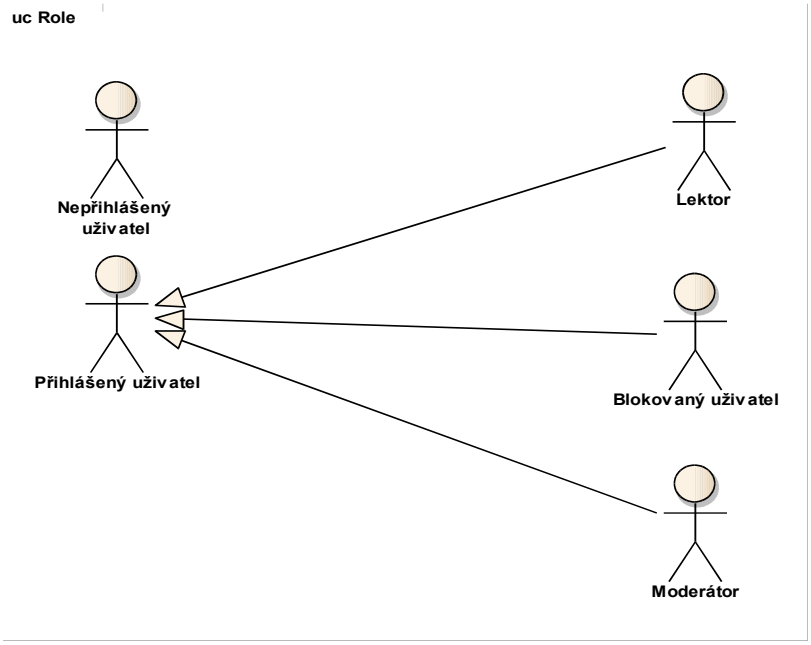
3.2 Uživatelské role

V systému bude definováno několik uživatelských rolí:

- Nepřihlášený uživatel – každý uživatel, který zavítá na stránky; mezi jeho možnosti patří vlastně jen zaregistrovat se anebo přihlásit se, i když je možné, že v budoucnu by bylo zpřístupněno například prohlížení inzerátů, případně jiné funkce „pouze pro čtení“
- Přihlášený uživatel – přihlášený uživatel má přístup do svého profilu, může zadávat vlastní inzeráty a reagovat na cizí, může hodnotit lektory a prohlížet si jejich profily, může také nahlašovat nevhodné inzeráty či chování jiných uživatelů
- Blokovaný uživatel – uživatel, který je dočasně blokován moderátorem, především po přezkoumání ohlášení o nevhodném chování tohoto uživatele
- Lektor – hlavní rozdíl mezi lektorem a přihlášeným uživatelem je ten, že lektor má aktivován vlastní profil, který si mohou ostatní uživatelé prohlížet; dále může zadávat nabídkové inzeráty
- Moderátor – moderátor řeší hlášení od uživatelů, může je blokovat a odblokovávat

Původně byl definován i uživatel „Zájemce o doučování“, ale nakonec od něj bylo odstoupeno, jelikož všechny jeho funkcionality by měly mít i další role, které jsou všechny speciálnějšími případy role „Přihlášený uživatel“.

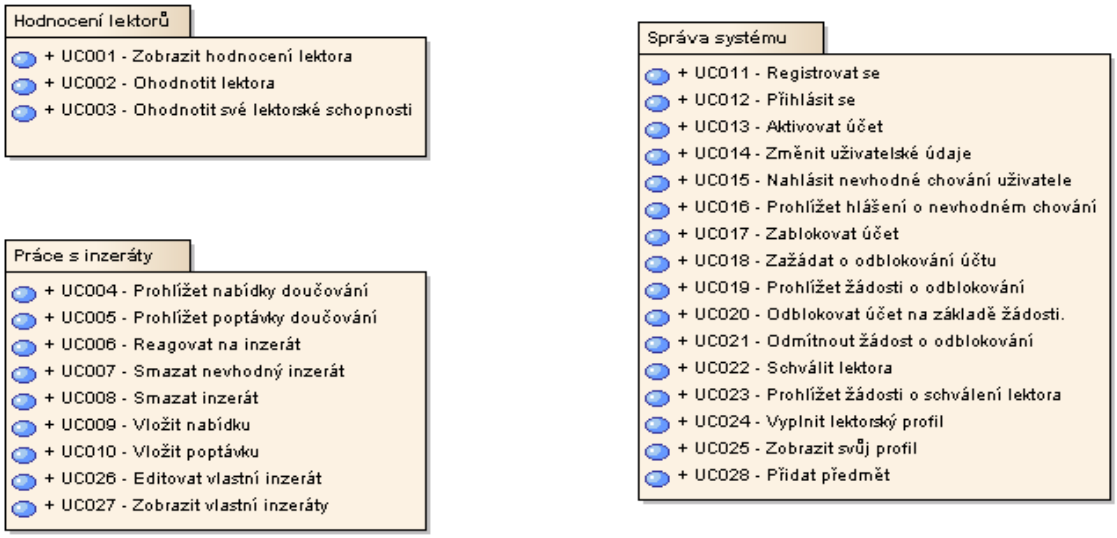
Níže je přehledový diagram:



Obr. 1: Uživatelské role

3.3 Případy užití

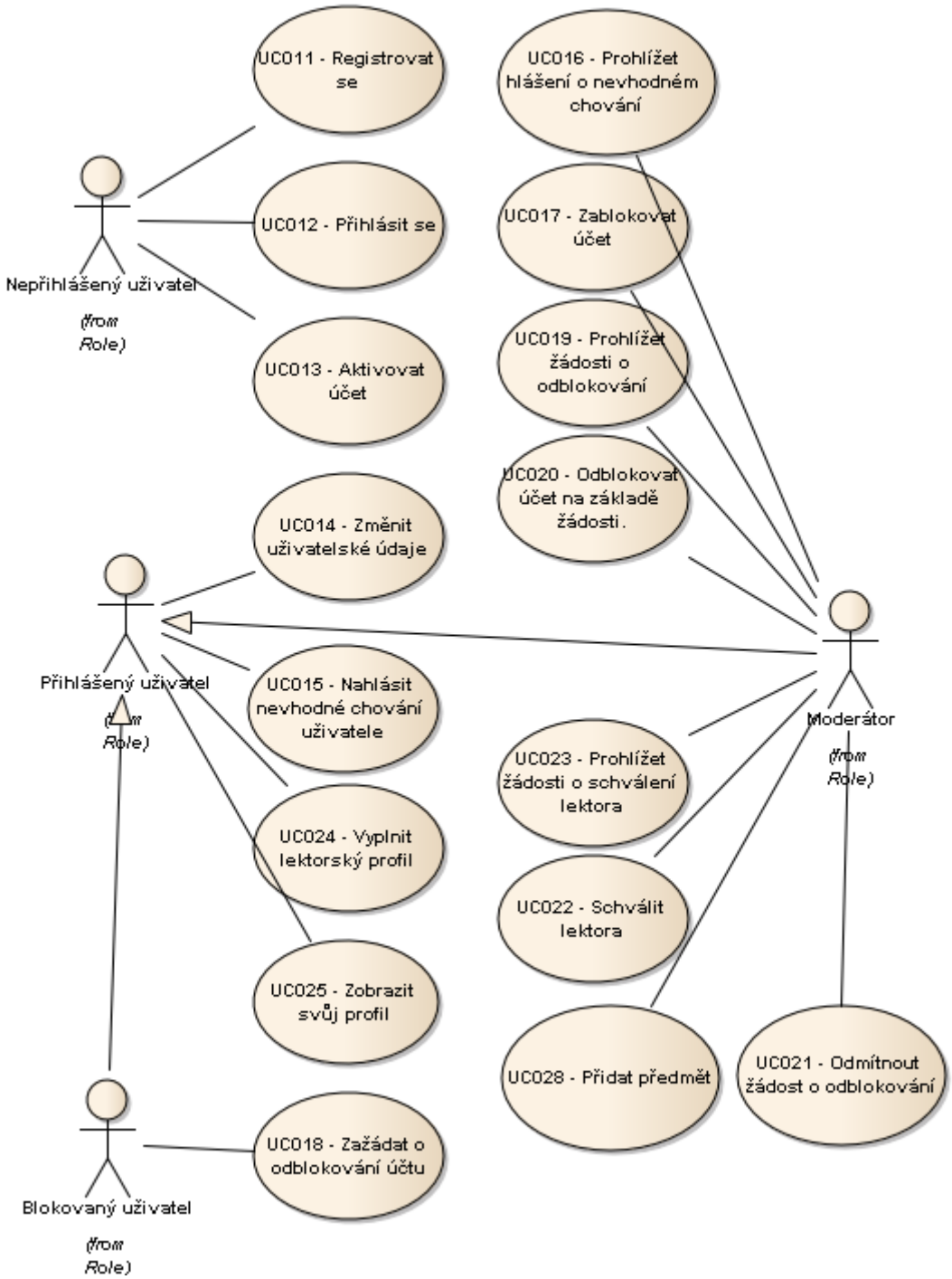
Jednotlivé funkční požadavky byly rozpracovány do případů užití, při jejichž tvorbě bylo postupováno částečně podle [2], ovšem s některými zjednodušeními navrhovanými ve [3]. Jelikož případy užití přímo vyplývají z funkčních požadavků, jsou seskupeny do stejné struktury balíčků.



Obr. 2: Rozdělení případů užití do balíčků

3.3.1 Balíček Správa systému

V tomto balíčku jsou obsaženy případy užití, které souvisí se správou uživatelů a inzerátů.



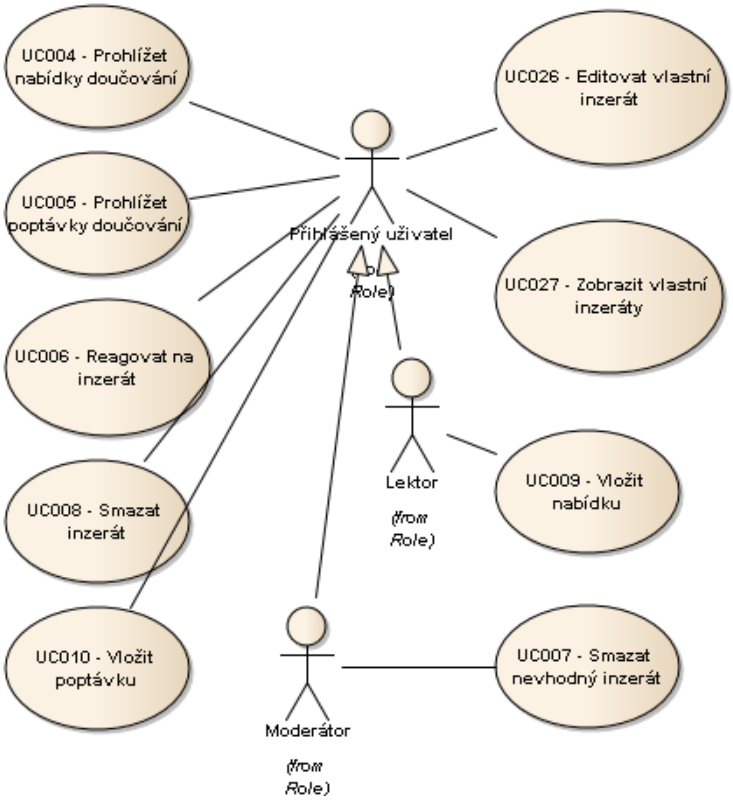
Obr. 3: Případy užití - balíček Správa systému

3.3.2 Balíček Práce s inzeráty

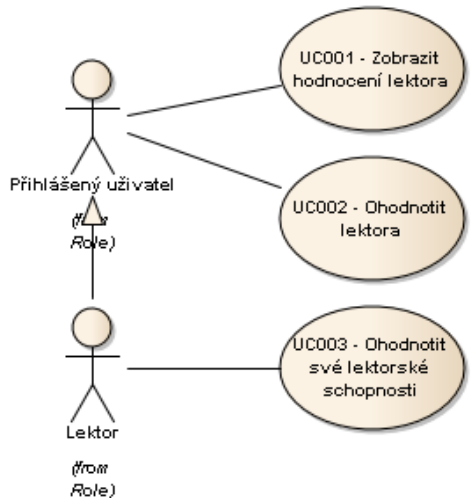
V tomto balíčku jsou obsaženy případy užití související s prací s inzeráty.

3.3.3 Balíček Hodnocení lektorů

V tomto balíčku jsou případy užití zabývající se požadavky na systém z hlediska hodnocení lektorů. Z balíků požadavků je sice nejmenší, do budoucna by ale tato oblast byla jednou z prvních, která by byla rozšiřována.



Obr. 4: Případy užití - balíček Práce s inzeráty

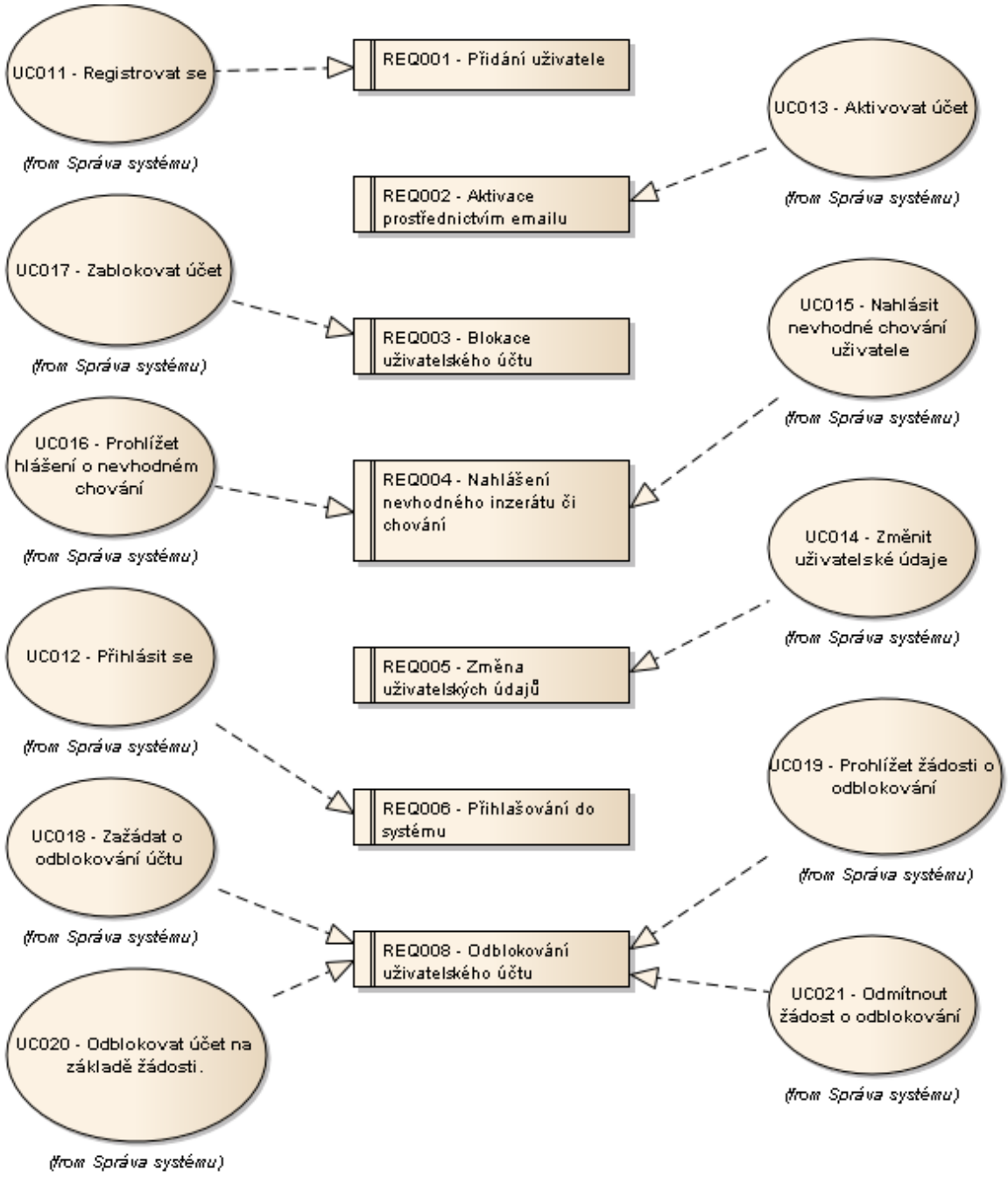


Obr. 5: Případy užití - balíček Hodnocení lektorů

3.4 Mapování případů užití na funkční požadavky

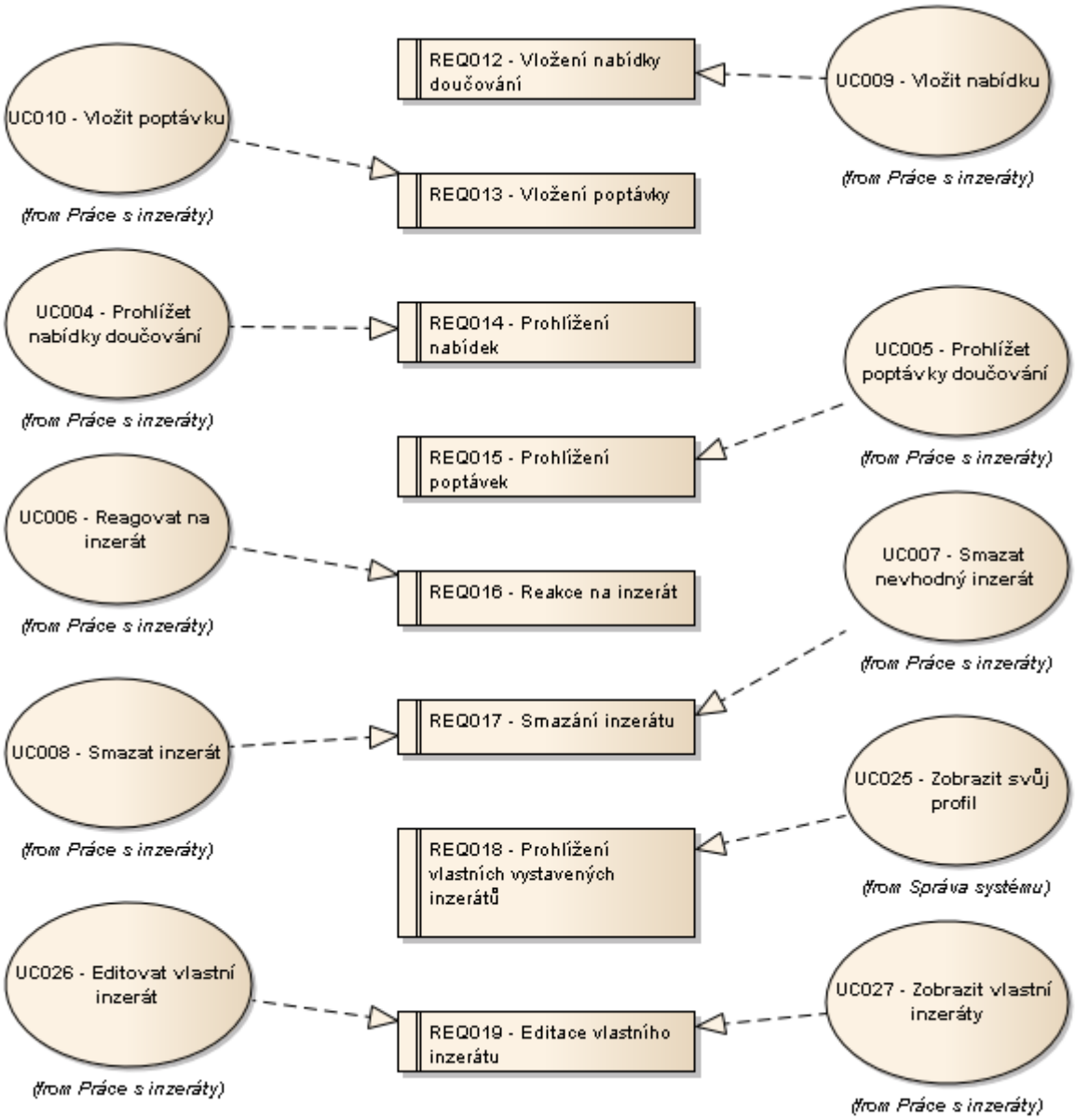
Každý z případů užití vychází z některého z funkčních požadavků. V nástroji Enterprise Architect, ve kterém byly případy užití i funkční požadavky zakreslovány, byly pro lepší přehlednost vytvořeny i diagramy mapování jednotlivých případů užití na tyto požadavky.

3.3.1 Balíček Správa systému



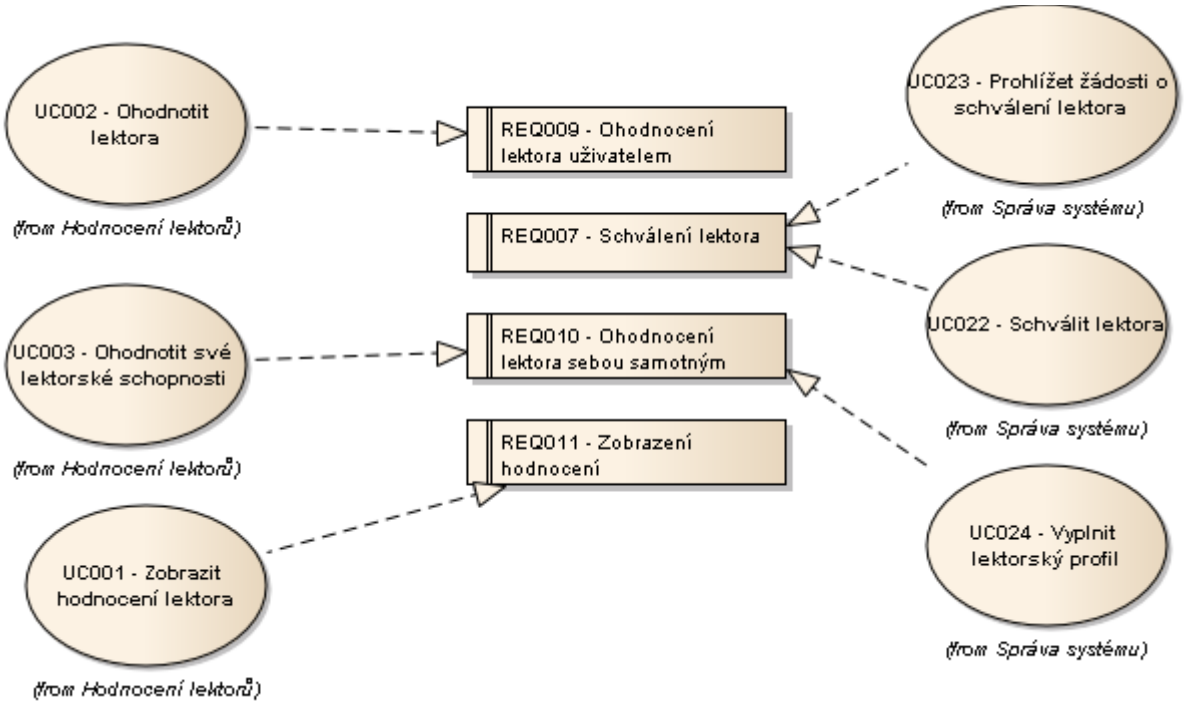
Obr. 6: Mapování požadavků - balíček Správa systému

3.3.2 Balíček Práce s inzeráty



Obr. 7: Mapování požadavků - balíček Práce s inzeráty

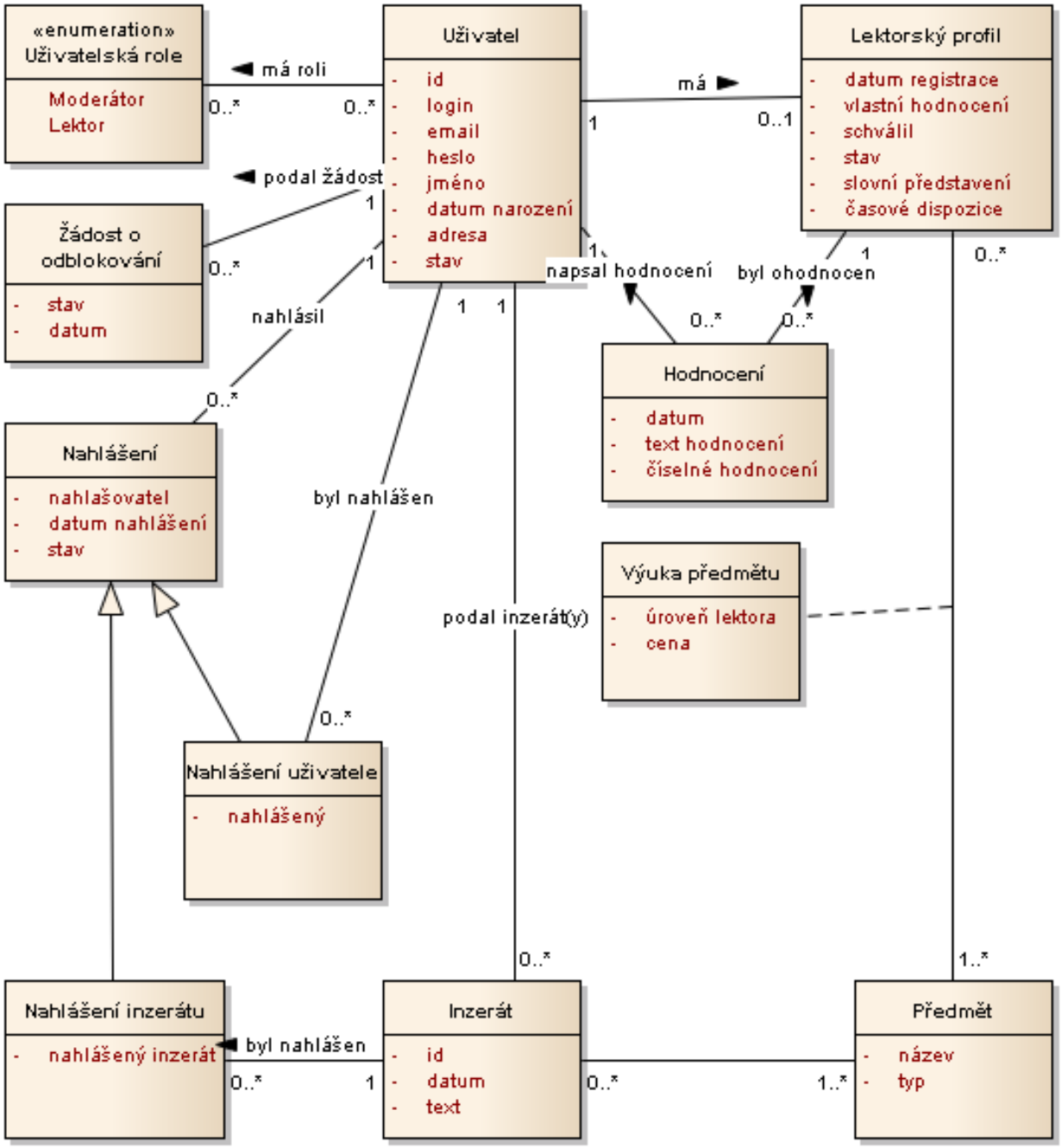
3.3.3 Balíček Hodnocení lektorů



Obr. 8: Mapování požadavků - balíček Hodnocení lektorů

3.5 Analytický model tříd

Analytický model tříd zachycuje strukturu tříd, které jsou pro business logiku systému nejdůležitější. Zároveň také ukazuje jejich nejpodstatnější atributy. Návrhový model tříd je popsán dále v textu (5.2.1) a je složitější v tom smyslu, že zobrazuje o poznání více informací o třídách.



Obr. 9: Analytický model tříd

Kapitola 4 – Návrh řešení

Tato kapitola přibližuje postup návrhu řešení. Začíná od výběru vhodných technologií a frameworků, jejich popisu a zdůvodnění jejich nasazení. Dále obsahuje již implementačně závislý popis struktury a tříd systému, stejně jako diagramy komponent a nasazení.

4.1 Použité technologie

4.1.1 Java Enterprise Edition

Celý projekt stojí především na využití Java technologie, což je kombinace programovacího jazyka a platformy.

Jazyk Java byl vybrán proto, že je široce používaným objektově orientovaným programovacím jazykem vyšší úrovně abstrakce. Tento jazyk je architektonicky nezávislý, robustní a existuje pro něj řada rozšiřujících a doplňujících knihoven.

Edice Enterprise Edition je pak součástí Java platformy, která je určená pro vývoj informačních systémů a webových aplikací. Od Standard Edition se liší mnoha přidanými knihovnami, které vývojáři umožňují využít velkého množství funkcionalit. Podle [5] jsou součástí platformy Java EE především specifikace pro:

- vývoj webových aplikací
- vývoj sdílené business logiky
- přístup ke zprávovému middleware
- přístup k legacy systémům
- komponenty zajišťující integraci webových aplikací a portálů
- podpora technologií webových služeb

4.1.2 Spring Framework

Jelikož vývoj aplikací v čisté Java Enterprise Edition může být složitý a nepřehledný, vznikaly v průběhu času různé frameworky, které byly motivovány snahou tento vývoj zjednodušit a zpřehlednit. Jedním z nejúspěšnějších a nejživotaschopnějších se ukázal být framework Spring. Tento framework vznikl především jako nenáročná (rozuměj výpočetně i programátorsky) alternativa k dříve těžkopádným Enterprise Java Beans.

Podle [6] patří mezi jeho hlavní výhody patří především:

- je snadno integrovatelný s mnoha frameworky a nadstavbovými knihovnami
- využívá návrhové vzory Inversion of control (řídí životní cyklus instancí v něm obsažených

tříd) a Dependency injection (zajišťuje, že daná instance bude mít nastavenou správnou implementaci)

- je neinvazivní, nezávislý na koknrétním programovacím paradigmatu
- obsahuje podporu transakcí
- jedná se o open source
- neklade zvláštní nároky na obsažené Javovské třídy (často stačí, že mají settery a gettery)
- upřednostňuje konvenci před konfigurací (často vystačíme s defaultní konfigurací)
- lze konfigurovat jak pomocí konfiguračních xml souborů, tak pomocí anotací umístěných přímo ve třídách
- základní modul Spring Core lze libovolně kombinovat s dalšími, rozšiřujícími moduly (v této práci využity především Spring Security, Spring Test a Spring ORM)

Právě z těchto důvodů byl Spring upřednostněn před Enterprise Java Beans, ačkoliv jejich nová verze se svými vlastnostmi Springu značně přibližuje a v některých ohledech jej i předčí.

4.1.3 Apache Maven

Enterprise Java aplikace mívá zpravidla rozsáhlý soubor závislostí na podpůrných knihovnách. Pro řízení buildu aplikace můžeme využít některý ze základních nástrojů – v prostředí programovacího jazyka Java to bývají nejčastěji Ant či Maven. Právě druhý jmenovaný je o mnoho lépe uzpůsoben pro řízení buildů rozsáhlejších enterprise aplikací. To platí obzvlášť, pokud předpokládáme, že bude potřeba aplikaci sestavovat ze zdrojových souborů na více než jednom počítači.

Apache Maven pro vyhledávání knihoven, na nichž je program závislý, používá lokální i online repozitáře, do kterých jsou nahrávány jejich rozsáhlé soubory. Konfigurace se provádí v souboru pom.xml. Podle svých tvůrců patří mezi jeho hlavní oblasti působnosti [7]:

- zjednodušení buildovacího procesu
- vytvoření jednotného způsobu buildování
- poskytování kvalitních informací o buildovaném projektu
- podpora doporučených vývojářských postupů
- transparentní prostředí pro přechod k novým verzím používaných knihoven

4.1.4 JPA – Hibernate

Obecně vzato, mapování tříd objektově orientovaného programu na tabulky v relační databázi (objektově-relační mapování neboli ORM) není právě triviálním úkolem, a to i pokud nebereme v úvahu rozdíly mezi jednotlivými relačními databázemi.

V prostředí javovských programů naštěstí existuje způsob, jak se tohoto úkolu zhostit o mnoho jednodušeji. Řešením je specifikace JPA – Java Persistence API [8]. Toto API definuje rozhraní mezi aplikací a poskytovatelem připojení k databázi. Konfiguraci z hlediska aplikace je možné provádět anotacemi (@Entity, @Column a další), které vyjadřují vlastnosti a vztahy entit, které mají být perzistovány (uloženy) v databázi. Poskytovatelem připojení je pak některý z produktů, který na jedné straně splňuje specifikaci JPA a na straně druhé je schopen komunikovat s použitou databází na základě použitých anotací i specifických dotazů vyjádřených v JPA dotazovacím jazyce JPQL.

Jednou z implementací tohoto API je tak například knihovna Hibernate (od verze 3 plně implementuje JPA), která je pro svou vyspělost a univerzálnost použita i v tomto projektu.

Jako samotná databáze je zde použita Apache Derby, především pro svou integraci do prostředí NetBeans, ve kterém je projekt vyvíjen. Vzhledem k využití knihovny Hibernate by však přeorientování se na jakoukoliv jinou databázi nebylo nic obtížného.

4.1.5 Java Server Faces

Pro prezentační vrstvu aplikace je použita implementace JSF standardu (Java Server Faces) od firmy Oracle. JSF představují komponentově orientovaný framework, který se využívá pro tvorbu webových aplikací. Výhodou JSF je v podstatě propojení html stránek a Java tříd, a to pomocí instance třídy FacesServlet, který celou tuto komunikaci zprostředkovává a je ze strany jednotlivých JSF komponent volán vyvolanými eventy.

Mezi další výhody patří uchovávání stavu v javovských třídách (tzv. Managed Beans, také označovány jako backing beans), a to po různě definovanou dobu. Dále je možné využívat tzv. Expression Language, kdy je možné používat atributy a volat metody backing bean přímo z kódu stránky (např. po stisknutí tlačítka).

Z dalších funkcí JSF jmenujme ještě alespoň definování pravidel navigace po webových stránkách, a to jak pravidel statických v souboru faces-config.xml, tak dynamických, kdy backing beana může vrátit JSF komponentě řetězec, podle kterého se navigace uskuteční.

V této práci byly JSF použity především proto, že (podle [9]) programátora efektivně odstiňují od detailního ovládání http protokolu, od servletů i starostí o ukládání formulářů, jejich repopulaci a validaci. To sice může být někdy na škodu (viz [11]), v našem případě jsou ale jimi poskytované funkce plně vyhovující, a to i s ohledem na návaznost další technologie použité pro implementaci prezentační vrstvy: knihovny PrimeFaces.

4.1.6 PrimeFaces

PrimeFaces jsou opensourcovou knihovnou komponent, které různými způsoby rozšiřují základní JSF komponenty. PrimeFaces[2] jsou zaměřeny jednak na pěkný vzhled, jednak na zachování uspokojivého výkonu.

Vybrány však byly především pro jednoduchost implementace, lehce měnitelný vzhled a bohatou nabídku komponent.

2 Dostupné z www.primefaces.org

4.1.7 Zabezpečení

Jak již bylo naznačeno v oddíle popisujícím framework Spring, vyvíjená webová aplikace je zabezpečena pomocí modulu Spring Security. Tento modul má vlastní konfigurační soubory a umožňuje zabezpečení na všech vrstvách aplikace – na prezentační (dle konfigurace v souboru application-security.xml), na servisní i na datové (zde pomocí anotací). Úroveň zabezpečení i mechanismus autentikace jsou přitom v mnoha ohledech konfigurovatelné, nabízí například i podporu více rolí pro jednoho přihlášeného uživatele, definování dodatečných kontrol po přihlášení, vynucení šifrovaného protokolu https a podobně.

4.1.8 Apache Tomcat

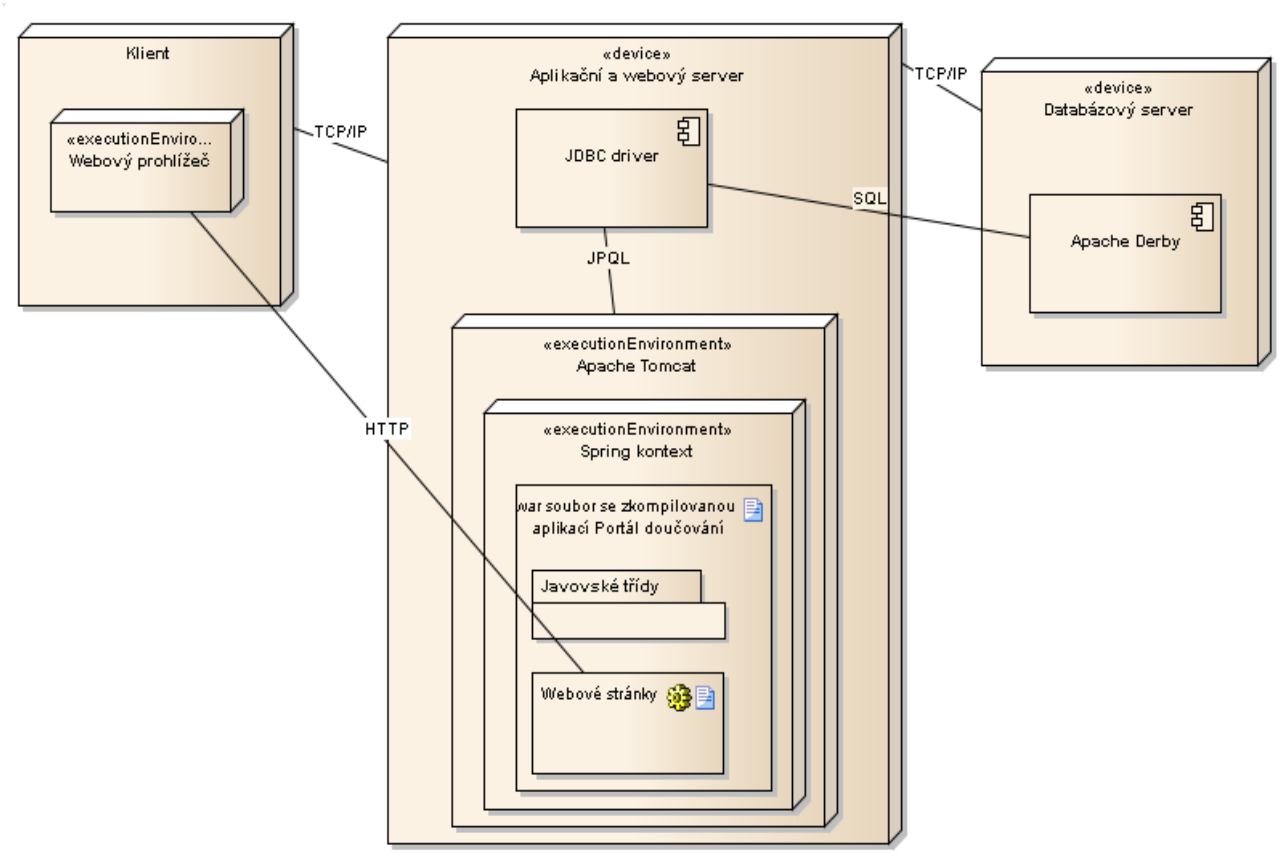
Celá webová aplikace poběží na opensourcovém serveru a servletovém kontejneru Apache Tomcat. Jedná se o nejpoužívanější webový server. Je platformně nezávislý (běží na Java Virtual Machine), jeho základní konfigurace je jednoduchá a není příliš náročný na systémové prostředky. To všechno jsou důvody, proč dostal přednost před některým z jiných serverů, jako je např. Glassfish.

4.2 Model nasazení

Apache Tomcat hraje roli aplikačního i webového serveru, na kterém je umístěna vlastní část aplikace a kde se generují webové stránky ve formátu xhtml, které jsou s využitím protokolu http odesílány klientskému webovému prohlížeči.

Databázový server může a nemusí běžet na stejném hardwarovém zařízení jako aplikační server. K rozdělení na více zařízení by mohlo dojít v případě předpokladu výrazného zvýšení zatížení. V tom případě by se také vyplatilo znovu uvážit, zda by jiný databázový server nevyhověl zvýšeným požadavkům lépe.

Na straně klienta by se webové stránky měly správně zobrazovat ve všech moderních prohlížečích. Jestli tomu tak skutečně je, bude jedním z předmětů testování.



Obr. 10: Model nasazení

Kapitola 5 – Implementace

Celá aplikace byla implementována ve vývojovém prostředí NetBeans IDE 7.4, které poskytovalo podporu pro veškeré použité technologie, včetně ovládání databáze a manipulace s webovým a aplikačním serverem Apache Tomcat. Projekt je konfigurován jako Maven Web Application s využitím kontejneru Spring.

Při implementaci jsem se snažil o vytvoření vrstevnaté architektury, tedy o rozdělení aplikace na následující vrstvy:

- servisní (či aplikační) vrstva – soustřeďuje vlastní logiku aplikace a propojuje ostatní vrstvy
- datová vrstva – ta je specializována na ukládání objektů problémové domény a na jejich údržbu a získávání z databáze
- prezentační vrstva – zaměřena na prezentaci dat směrem k uživateli,

Tyto vrstvy a jejich případná konfigurace jsou popsány v sekcích 5.2 – 5.4. V následující sekci 5.1 je pak popsána konfigurace dvou základních použitých technologií – buildovacího nástroje Apache Maven a frameworku Spring. Obě tyto konfigurace se týkají aplikace jako celku, proto je nelze snadno zařadit do některé ze jmenovaných vrstev.

5.1 Konfigurace Spring a Apache Maven

5.1.1 Konfigurace Mavenu – pom.xml

Kromě samotných zdrojových java souborů lze do této vrstvy zařadit i konfiguraci buildovacího nástroje Maven, která je umístěna v souboru **pom.xml**. V něm jsou definovány především všechny knihovny, které aplikace využívá, dále tzv. repozitáře, ze kterých se tyto knihovny (a knihovny, na kterých tyto závisí) stahují v případě, že nejsou k dispozici lokálně. Kromě toho jsou zde definovány podrobnosti buildovacího procesu a testování. Níže několik příkladů těchto xml elementů:

Definice repozitáře:

```
<repository>
	<id>com.springsource.repository.bundles.release</id>
	<name>EBR Spring Release Repository</name>
	<url>http://repository.springsource.com/maven/bundles/release< /url>
</repository>
```

Závislost:

```
<dependency>
	<groupId>commons-dbcp</groupId>
	<artifactId>commons-dbcp</artifactId>
```

```
      <version>1.4</version>
</dependency>
```

Nastavení buildovacího procesu:

```
 <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>2.1.1</version>
      <configuration>
            <failOnMissingWebXml>false</failOnMissingWebXml>
            <packagingExcludes>WEB-INF/web.xml</packagingExcludes>
      </configuration>
</plugin>
```

5.1.2 Konfigurace Springu

Framework Spring má konfiguraci umístěnu na několika místech. Její hlavní část se nalézá v souboru **applicationContext.xml**. Start Springu zajišťuje a konfiguruje nastavení v souboru web.xml, který ovšem slouží i jiným složkám webové aplikace. A konečně některé funkce (zpřístupnění některých souborů pro testování, použití anotace @Configurable) jsou umožněny nastavením v souboru **pom.xml**.

Všechna ostatní konfigurace se provádí pomocí anotací u tříd, jejich atributů a metod. Tímto způsobem je konfigurace o poznání zpřehledněna, neboť je přenesena přímo do tříd, které jsou konfigurovány.

5.1.2.1 Nejdůležitější části souboru applicationContext.xml:

Použití konfiguračních anotací:

```
<context:annotation-config />
```

Kde hledat Springové komponenty (beany):

```
<context:component-scan base-package="com.pavel.portaldoucovani"/>
```

Podpora anotace @Configurable:

```
<context:spring-configured/>
```

Kde hledat soubory s konfigurací připojení do databáze:

```
<bean id="propertyConfigurer" class="org.springframework.beans.factory.
                                        config.PropertyPlaceholderConfigurer">
      <property name="locations">
            <list>
```

```
                <value>/WEB-INF/properties/jdbc.properties</value>
                <value>/WEB-INF/properties/jpa.properties</value>
            </list>
        </property>
</bean>
```

Připojení do databáze:

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
                                                    destroy-method="close">
        <property name="driverClassName" value="${jdbc.driverClassName}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
        <property name="initialSize" value="2" />
        <property name="minIdle" value="2" />
</bean>
```

Nastavení tovární třídy pro Entity Manager:

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.
                        LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="jpaVendorAdapter">
        <bean class="org.springframework.orm.jpa.
                                        vendor.HibernateJpaVendorAdapter">
                <property name="databasePlatform" value="${jpa.platform}"/>
                <property name="generateDdl" value="true"/>
                <property name="showSql" value="true"/>
      </bean>
        </property>
        <property name="packagesToScan" value="com.pavel.portaldoucovani" />
</bean>
```

Nastavení transakcí:

```
<bean id="txManager" class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
<bean id="transactionTemplate"
        class="org.springframework.transaction.support.TransactionTemplate">
        <property name="transactionManager">
            <ref bean="txManager"/>
```

```
      </property>
</bean>
```

Podpora deklarativního označení transakcí pomocí anotace @Transactional:

```
<tx:annotation-driven transaction-manager="txManager" />
```

5.1.2.2 Konfigurace Springu v souboru web.xml

Kde najít konfigurační soubory Springu:

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/context/*.xml</param-value>
</context-param>
```

Start Springu při startu serveru:

```
<listener>
      <listener-class>
      org.springframework.web.context.ContextLoaderListener
      </listener-class>
</listener>
```

Asociace požadavků s příslušným vláknem dotazu:

```
<listener>
      <listener-class>
            org.springframework.web.context.request.RequestContextListener
      </listener-class>
</listener>
```

5.1.3 Konfigurace Spring Security

Obdobně jako Spring, je i framework Spring Security konfigurován na více místech, konkrétně opět v souboru **web.xml** a v našem případě ještě v souboru **applicationContext-security.xml**.

5.1.3.1 Konfigurace Spring Security v souboru applicationContext-security.xml

Zabezpečení metod:

```
<global-method-security secured-annotations="enabled" pre-post-annotations=
                                                              "enabled"/>
```

Hlavní část konfigurace, kde je specifikována přihlašovací a odhlašovací stránka a dále pravidla pro přístup na jednotlivé stránky či skupiny stránek:

```
<http>
      <form-login login-processing-url="/static/j_spring_security_check"
            login-page="/login.xhtml"
            authentication-failure-url="/login.xhtml?login_error=t" />
      <intercept-url pattern="/user/**" access="ROLE_USER" />
      <logout logout-url="/static/j_spring_security_logout" />
</http>
```

A specifikace služby, která autentikaci uděluje:

```
<beans:bean id="authenticationProvider"
      class="com.pavel.portaldoucovani.service.AuthenticationService">
      <beans:property name="genericDAO" ref="genericDAO" />
      <beans:property name="transactionTemplate" ref="transactionTemplate" />
</beans:bean>
<security:authentication-manager alias="authenticationManager">
      <security:authentication-provider ref="authenticationProvider" />
</security:authentication-manager>
```

5.1.3.2 Konfigurace Spring Security v souboru web.xml

V tomto souboru jsou nakonfigurovány základní vlastnosti Spring Security, jako je uvítací stránka po přihlášení do aplikace, dobu platnosti přihlášení a hlavně pak vlastnosti filtrovacího řetězu (SecurityFilterChain), tzn. třída, která plní jeho funkci, a pak také na jaké stránky má být aplikován.

Nastavení SecurityFilterChain:

```
<filter>
      <filter-name>springSecurityFilterChain</filter-name>
      <filter-class>org.springframework.web.filter.DelegatingFilterProxy
      </filter-class>
</filter>
<filter-mapping>
      <filter-name>springSecurityFilterChain</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

Nastavení doby platnosti přihlášení:

```
<session-config>
      <session-timeout>60</session-timeout>
</session-config>
```

Uvítací stránka:

```
<welcome-file-list>
	<welcome-file>index.xhtml</welcome-file>
</welcome-file-list>
```

5.2 Servisní vrstva

Jádrem aplikace je servisní vrstva, která obsahuje vlastní logiku aplikace. Tato vrstva využívá služeb nižší, datové vrstvy a je naopak využívána vrstvou prezentační. Pracuje s doménovými objekty (též zvanými „business objects“), tedy třídami reprezentujícími objekty problémové domény. Ty jsou v našem případě umístěny v balíčku com.pavel.portaldoucovani.bo. Většina základních služeb pro manipulaci s daty v databázi je pak umístěna ve třídách com.pavel.portaldoucovani.service, kde jsou definována i rozhraní těchto služebních tříd.

Tento způsob rozdělení centrální vrstvy aplikace na „business objects“ a služby není příliš výhodný z toho pohledu, že je snadné sklouznout k návrhovému antivzoru „anemický doménový model“ [12], kdy dochází k rozdělení dat a metod k nim přistupujících, což je proti duchu objektově orientovaného návrhu. Na druhou stranu je nutno říci, že právě tento způsob implementace umožňuje použitému Spring frameworku injektovat třídy se službami a zajistit tak například transakční provedení a zabezpečení metod této vrstvy. Za tímto účelem ve třídách, kde chceme tyto služby používat, deklarujeme proměnnou typu požadovaného rozhraní a přidáme anotaci @Autowired. Spring pak do této proměnné přiřadí instanci odpovídající služební třídy, jejíž metody se ovšem musí řídit také anotacemi definovanými v daném rozhraní.

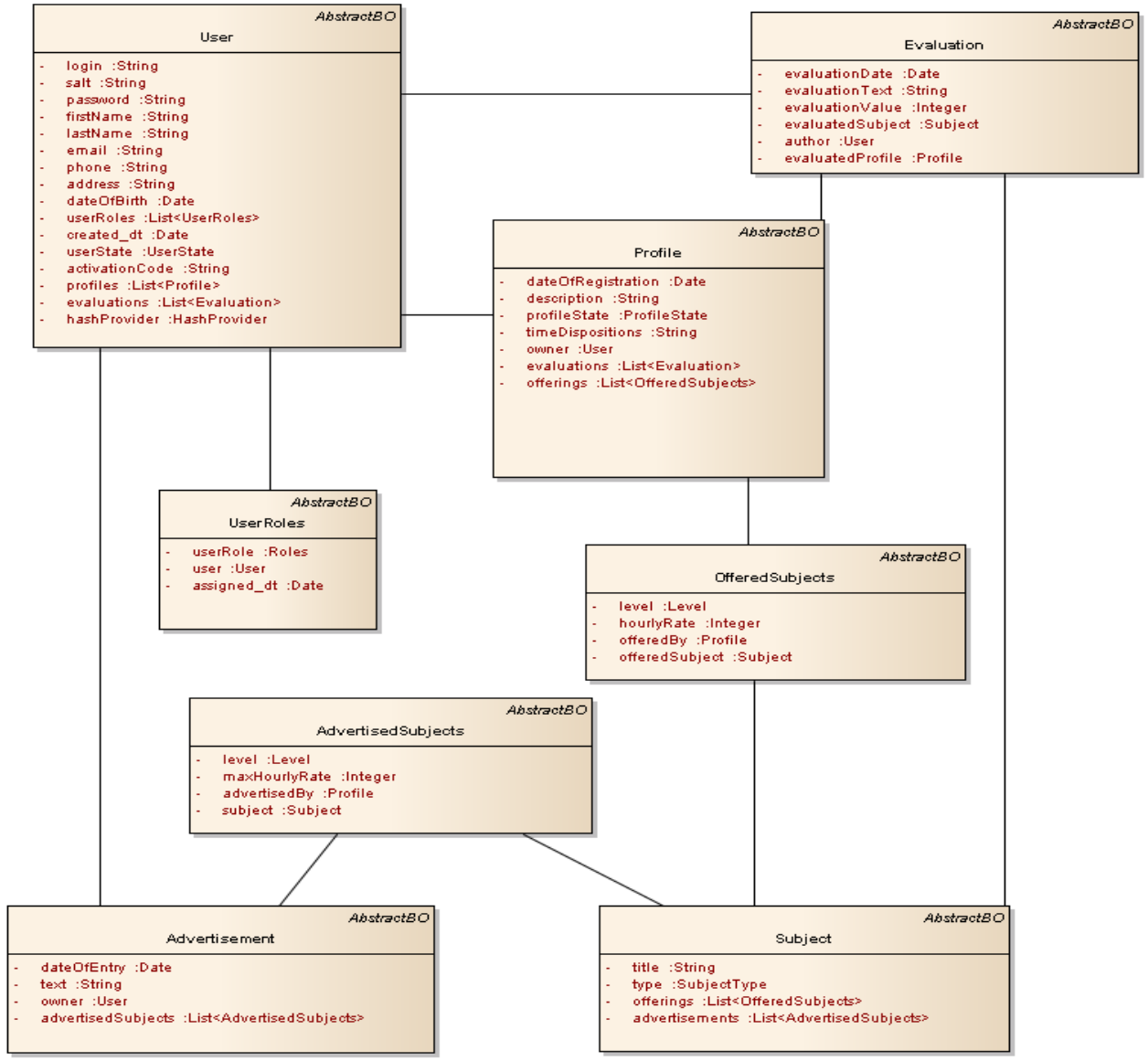
Do této vrstvy dále patří spíše pomocné třídy z balíčku com.pavel.portaldoucovani.provider a nejspíše bychom sem zařadili i výčtové číselníky z balíčku com.pavel.portaldoucovani.enums, i když k těm mají ostatní vrstvy aplikace volný přístup.

5.2.1 Návrhový model doménových tříd

Tento model ukazuje, jaké atributy, metody a vazby mají doménové třídy. Oproti analytickému modelu již přesněji odpovídá navrhované aplikaci, například v tom, že složitější vazby jsou dekomponovány, anebo že byly doplněny i pomocné atributy .

Pro zjednodušení obrázku bylo vypuštěno několik prvků. Především se jedná o nadtřídu všech znázorněných tříd. Tato nadtřída je nazvána **AbstractBO** a obsahuje unikátní klíč (datový typ Long), příslušný getter a setter, překrývá metody equals a hashCode a implementuje rozhraní Serializable. Byla vytvořena kvůli ukládání doménových objektů do databáze a také kvůli správnému porovnávání objektů.

Dále bylo upuštěno od zobrazení všech getterů a setterů, které by celý obrázek pouze zkomplikovalo a nepřineslo žádnou podstatnou novou informaci.



Obr. 11: Návrhový model tříd

5.3 Datová vrstva

Tato vrstva aplikace se stará o propojení servisní vrstvy a relační databáze Apache Derby. Činí tak pomocí knihovny Hibernate a JPA anotací u jednotlivých perzistovaných tříd, podle nichž jsou při prvním spuštění v databázi vytvořeny příslušné tabulky. Přístup do nich je v této aplikaci řešen pomocí návrhového vzoru DAO – Data Access Object.

5.3.1 Konfigurace Hibernate

Hibernate je konfigurován především v souboru **applicationContext.xml**, který byl již zmiňován dříve (oddíl 5.1.2.1), a to včetně příslušné konfigurace.

5.3.2 Použité anotace

Základní mapování doménových tříd na tabulky databáze je nakonfigurováno pomocí anotací, které jsou umístěny přímo u těchto tříd. Využity jsou následující anotace:

Anotace	**Popis**
@Entity	základní anotace; označuje, že daná třída odpovídá tabulce v databázi
@Table	název tabulky; když u entity není, tabulka se bude jmenovat stejně jako entita
@Column	označuje, že atribut odpovídá sloupci tabulky
@Temporal	používá se pro označení atributů typu Date (datum)
@Transient	označuje atribut, který se nemá v databázi uchovávat
@OneToMany, @ManyToOne	označují strany vztahu typu 1:N; u @OneToMany se specifikuje, podle kterého atributu se má mapování provádět

Tab. 5: Použité JPA anotace

5.3.3 DAO – Interface a implementace

Pro komunikaci mezi běžící aplikací a databází byl vytvořen balíček com.pavel.portaldoucovani.dao, kde je jednak definováno DAO rozhraní pro využití v aplikaci, jednak specifická implementace pro knihovnu Hibernate, a to pouze za využití prostředků definovaných v JPA2[3]. Samotná implementace je pak částečně převzata z materiálů k předmětu AD7B39WPA [13].

3 knihovna Hibernate nabízí více funkcí než je definováno rozhraním JPA2. Jejich využití by však mohlo mít negativní vliv na kompatibilitu aplikace s jinými JPA2 poskytovateli, pokud bychom se za ně rozhodli Hibernate vyměnit.

5.4 Prezentační vrstva

Do této vrstvy patří jak samotné stránky, tak kód tzv. „backing beans“, což jsou javovské třídy sloužící jako kontrolery stránek. Pro tvorbu stránek a s ní spojenou problematiku je pak využit framework Java Server Faces 2 spolu s knihovnou AJAXových komponent PrimeFaces.

5.4.1 Třídy DTO

Pro komunikaci se servisní vrstvou slouží třídy z knihovny com.pavel.portaldoucovani.dto. Jedná se o implementace návrhového vzoru Přepravka pro každou z doménových tříd. Ty podle [4] i [14] pro tento účel nelze použít z několika důvodů:

- transakční kontext končí na servisní vrstvě
- mohou obsahovat citlivá data nevhodná pro přenos
- došlo by k odhalení vnitřního rozhraní aplikace
- nelze je přenést na jinou Java VM (kvůli injektovaným závislostem)

Třídy DTO obsahují k jednotlivým atributům pouze gettery a settery, namísto případných referencí pak obsahují jen identifikátory.

5.4.2 Konfigurace JSF

Základem frameworku JSF je tzv. FacesServlet, což je kontroler, který zpracovává vstupy od uživatele. Pro zpracování těchto vstupů používá mimo jiné metod výše zmíněných backing beans. Ty mohou být buď nakonfigurovány pomocí elementů v základním JSF konfiguračním souboru faces-config.xml, anebo pomocí anotací. Při použití spolu s frameworkem Spring mohou tuto roli hrát i třídy se springovými anotacemi @Component, @Controller, a dalšími. V této aplikaci je využita právě tato možnost, která je nakonfigurována pomocí následujícího záznamu v souboru faces-config.xml:

```
<application>
      <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-
                                                                        resolver>
</application>
```

Životnost jednotlivých instancí je pak definována též pomocí anotace, konkrétně @Scope(value="session") pro životnost po dobu celé relace, anebo @Scope(value = "request") pro životnost po dobu jednoho http dotazu. Mezi další možnosti patří životnost typu Jedináček (Singleton), která ovšem není v aplikaci využita.

Metody a atributy těchto tříd lze pak využívat přímo v kódu jednotlivých JSF stránek s využitím tzv. Expression Language. Příklad takového volání:

```
<h:outputText id="adYear" value="#{viewMyAdsBean.selectedAd.dateOfEntry}"/>
```

Navigace se u tohoto frameworku dá řešit buď pomocí klasických odkazů, anebo pomocí definice navigačních pravidel v souboru faces-config.xml. Příklad použitého navigačního pravidla:

```
<navigation-rule>
      <from-view-id>/login.xhtml</from-view-id>
      <navigation-case>
            <from-outcome>register</from-outcome>
            <to-view-id>/register.xhtml</to-view-id>
            <redirect />
      </navigation-case>
</navigation-rule>
```

5.4.3 Konfigurace PrimeFaces

Komponenty z knihovny PrimeFaces jsou nadstavbou nad JSF komponentami. Mezi jejich hlavní výhody patří použití AJAXu, nenáročnost a možnost jednoduché konfigurace vzhledu.

Chceme-li tuto knihovnu v naší aplikaci využít, je třeba pouze nadefinovat v souboru pom.xml příslušnou závislost, a případně, pokud dosud použité repozitáře tuto knihovnu neobsahují, ještě nějaký vhodný repozitář.

Pro použití komponent této knihovny se pak používají html tagy z jmenného prostoru PrimeFaces (`xmlns:p="http://primefaces.org/ui"`).

Jak již bylo řečeno, jednou z velkých předností PrimeFaces je jednoduchá konfigurace vzhledu. Ta spočívá především v tom, že je k dispozici několik desítek různých propracovaných barevných témat, v nichž je definován vzhled všech komponent. Tato témata (Themes), a tím i vzhled celé aplikace, lze snadno měnit nastavením jedné konfigurační položky v souboru web.xml:

```
<context-param>
      <param-name>primefaces.THEME</param-name>
      <param-value>aristo</param-value>
</context-param>
```

Je dokonce možné barevné téma měnit i za běhu aplikace na základě uživatelova rozhodnutí, což ovšem není v naší aplikaci využito.

5.4.4 Stručně o návrhu uživatelského rozhraní

Uživatelské rozhraní bylo navrženo s využitím šablonovacího systému Facelets, který je součástí frameworku JSF. Jde o to, že si vytvoříme jednu základní stránku – šablonu, ve které pomocí faceletového tagu `<ui:insert>` vyhradíme oblast (případně i více oblastí), do které budou pomocí tagu `<ui:define>` vkládány stránky ostatní.

V této aplikaci byla jako šablona zvolena PrimeFaces komponenta Complex Layout, která stránku rozděluje na pět oblastí – uprostřed, nahoře, dole, vlevo, vpravo. Centrální část představuje místo do kterého se vždy „dodefinuje“ aktuální stránka, nahoře a vlevo jsou rozmístěny ovládací položky menu, dole je umístěn název a autor aplikace, část vpravo není zatím využita ani zobrazena.

Níže uvádím vybrané části kódu, pomocí nichž je tato šablona vytvořena:

```
...
<f:view contentType="text/html">
        ...
      <h:body>
            <p:layout fullPage="true" >
            ...
                  <p:layoutUnit id="center" position="center">
                        <ui:insert name="content">Put default content here, if
                        any.</ui:insert>
                  </p:layoutUnit>
            </p:layout>
      </h:body>
</f:view>
```

A použita:

```
<h:body>
      <ui:composition template= "./WEB-INF/includes-templates/layout-
                                                                  complex.xhtml">
            <ui:param name="pageTitle" value="#{msgs.login_title}" />
                  <ui:define name="content">
                  ...
                  </ui:define>
      </ui:composition>
</h:body>
```

Pro samotný náhled na vzhled aplikace odkazuji na Přílohu C – Uživatelská příručka, kde je vzhled aplikace s defaultním barevným tématem `aristo` zachycen na několika obrázcích.

5.4.5 Lokalizace

Ačkoliv v současnosti není předpokládáno nasazení této aplikace jinde než v České republice, je ze cvičných důvodů lokalizována kromě češtiny ještě do angličtiny. Všechny zprávy pro uživatele jsou obsaženy ve dvou textových souborech: **version_cs.properties** a **version_en.properties**. Tyto soubory mají na každé řádce uvedenu strukturu klíč=“hodnota“, přičemž množiny klíčů obou souborů by měly být stejné. „Hodnota“ představuje zprávu, která se uživateli zobrazí pro dané nastavení jazyka.

Aby JSF věděly, že mají tyto soubory využívat, kde je hledat a které jazyky umí, jsou v souboru **faces-config.xml** nastaveny následující elementy:

```
<application>
      <locale-config>
            <default-locale>en</default-locale>
            <supported-locale>cs</supported-locale>
            <supported-locale>en</supported-locale>
      </locale-config>
      <resource-bundle>
            <base-name>com.pavel.portaldoucovani.locale.version</base-name>
            <var>msgs</var>
      </resource-bundle>
</application>
```

V samotných JSF elementech je pak použití klíčů těchto souborů podobné jako při volání getteru managed beany, tzn. používají se s využitím Expression Language. Pro výše zmíněnou konfiguraci se tato „beana“ jmenuje msgs a použije se například takto:

```
<h:outputLabel for="adYear" value="#{msgs.ad_date}: "  />
```

To, která z podporovaných lokalizací bude použita v konkrétním případě, lze buď explicitně definovat v závislosti např. na vlastnostech přihlášeného uživatele, anebo se při výběru zobrazeného jazyka můžeme spolehnout na nastavení uživatelova prohlížeče. Tato druhá varianta byla zvolena i v naší aplikaci.

Kapitola 6 – Testování

Testování aplikace tvoří důležitou součást jejího vývoje a vlastně i celého jejího životního cyklu. Cíle testování jsou v různých fázích vývoje aplikace různé. Na nižší, programátorské úrovni jde hlavně o to najít v kódu aplikace co nejvíce chyb, které pak mohou být odstraněny. Přitom je důležité si uvědomit, že sebevětší množství testů nám bezchybnost aplikace nezaručí, můžeme nicméně alespoň zvyšovat pravděpodobnost odhalení přítomných chyb.

Na vyšší, uživatelské úrovni pak jde v testování především o to, zda aplikace dělá to, co od ní její uživatelé očekávají. V případě projektu na zakázku zde přichází na řadu tzv. akceptační testy, tedy testování zákazníkem. To ovšem není náš případ, jelikož my vyvíjíme produkt pro sebe, a jiní lidé k němu budou mít přístup ne jakožto naši zákazníci, ale jakožto jeho uživatelé. Proto zvolíme metodiku testování použitelnosti (Usability Testing, viz např. [15]). Kromě toho sem ale patří i testování parametrů techničtějšího charakteru, jako jsou zátěžové testy, testování kompatibility, testy odolnosti proti hardwarovým výpadkům a podobně.

Ze všech těchto jmenovaných technik a metodik jich při testování naší aplikace bylo použito jen několik, které jsou i se svými výsledky blíže popsány níže. Ostatní by vzhledem k rozsahu a charakteru aplikace neměly smysl.

6.1 Testování kódu

6.1.1 Statické testování

Statickým testováním rozumíme analýzu samotného kódu aplikace, kdy aplikaci samotnou nespouštíme. Prvním stupněm tohoto testování je podpora automatické kontroly zdrojového kódu přímo při jeho psaní, která je v použitém vývojovém prostředí NetBeans 7.4 již poměrně vyspělá a na velké množství potenciálních chyb programátora upozorní záhy poté, co se objeví.

Dalším stupněm tohoto testování je použití specializovaného nástroje FindBugs, který v javovském kódu pátrá i po subtilnějších chybách jako je vystavení vnitřní reprezentace proměnné, podobné názvy metod lišící se jen velikostí některých písmen, serializace potenciálně neinicializované reference, přiřazení nikdy nepoužité hodnoty, a jiných. Tento nástroj lze do NetBeans integrovat jako plugin a jeho používání je tak pohodlné. Navíc poskytuje dobrá vodítka pro zlepšování kvality kódu, kterých se ovšem není třeba ortodoxně držet a snažit se za každou cenu počítadlo nalezených upozornění vynulovat. V některých fázích vývoje jsem jeho výstupy používal i při hledání příčin konkrétních zapeklitých chyb, které byly nalezeny jinými technikami, např. ve fázi integračního testování.

6.1.2 Jednotkové a integrační testy

Nyní již přecházíme k dynamickým testům, tedy k testům za běhu aplikace. Jedním ze základních typů dynamických testů jsou testy jednotkové, jejichž hlavní myšlenkou je, že pokud se nemá vyskytnout chyba ve výsledné funkcionalitě, musí být správně implementovány jednotlivé metody (obrácená implikace neplatí). Jednotkové testy tedy testují tyto jednotlivé metody a pomocí

tzv. „assert“ tvrzení zkouší, zda jsou jejich výsledky takové, jak je očekáváno.

O stupeň výše jsou testy komponent a integrační testy, kdy testujeme správnou funkčnost jednotlivých komponent a komunikaci mezi těmito komponentami uvnitř aplikace.

Musím říci, že pro samotné jednotkové testy jsem v implementované aplikaci neviděl místo, a to především proto, že jednotlivé metody v mé aplikaci jsou povětšinou příliš jednoduché, aby se psaní testů vyplatilo. Řídil jsem se zde zásadou „Testujte, dokud se strach z chyby nepromění v nudu.“ (citováno dle [16])

To už ovšem neplatí pro testy komponent a integrační testy, které mi poskytly nedocenitelná vodítka pro hledání různých chyb, často velice subtilních. Zde byly testovány především kombinace funkcí servisní vrstvy, primárním cílem byl konkrétně balíček com.pavel.portaldoucovani.service, který aplikaci poskytuje většinu funkcionalit. Při používání jeho metod jsou navíc přímo i nepřímo volány metody doménových tříd a poskytovatele DAO, které zase volá metody knihovny Hibernate a data jsou tak ve výsledku ukládána či získávána z databáze.

Pro implementaci těchto testů byl použit framework JUnit, který se používá i pro jednotkové testy. V kombinaci s vhodným nastavením frameworku Spring a buildovacího nástroje Maven bylo možné docílit i transakčního zpracování testů, kdy databáze není uvedena do nekonzistentního stavu ani když testy havarují a navíc je po jejich provedení uvedena do téměř[4] původního stavu.

Část nastavení v pom.xml, zajišťující kontext pro provedení testů při buildu aplikace:

```
<build>
	<testResources>
		<testResource>
			<directory>src/test/resources</directory>
			<filtering>true</filtering>
		</testResource>

		<testResource>
			<directory>src/main/webapp</directory>
			<filtering>true</filtering>
			<includes>
				<include>**/context/applicationContext*.xml</include>
				<include>**/properties/*.properties</include>
			</includes>
		</testResource>
	</testResources>
</build>
```

Příklad testu, který je potomkem abstraktního **AbstractServiceTest**, ve kterém je zajištěno transakční zpracování testů:

```
public class UserDetailsServiceTest extends AbstractServiceTest {
	@Autowired
	private UserDetailsIface userDetailsService;

	@Test
	    public void testAddAndRemoveUser() {
```

4 „Téměř“ znamená, že data samotná ani tabulky ovlivněny nejsou, ale například počítadla automaticky generovaných identifikátorů navýšena jsou, a pokud by bylo zapnuté auditování, byl by uložen záznam o provedených operacích.

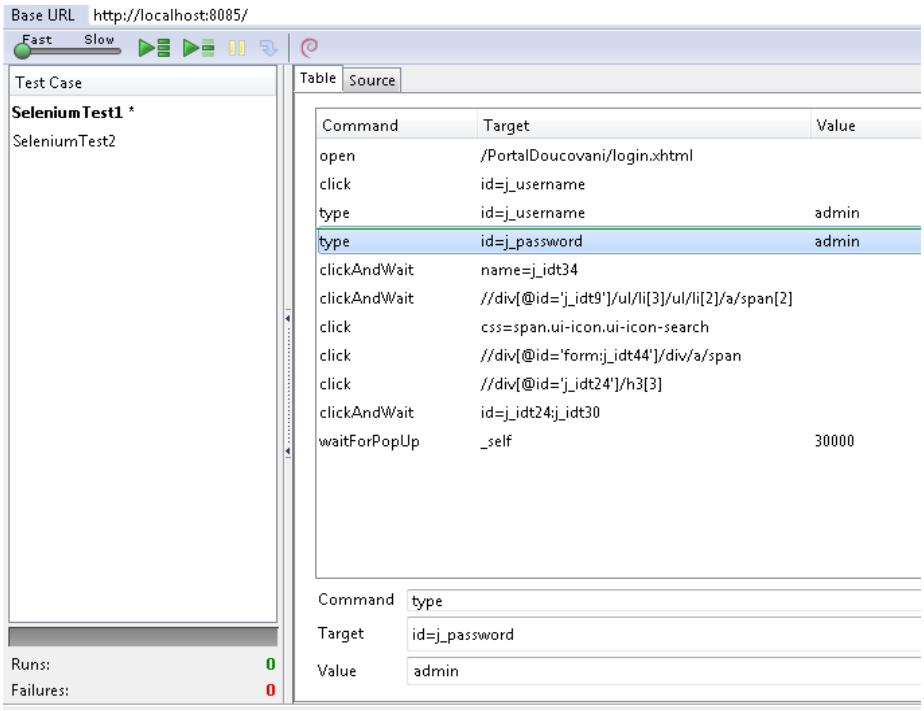
```
            Long userID = userDetailsService.addUser
                              ("test","test2","test3","test4","test5",date);
            assertTrue(userID ==userDetailsService.getUserById(userID).getId());
            assertTrue(userDetailsService.getUserById(userID) != null);
            userDetailsService.deleteUser(userID);
            assertTrue(userDetailsService.getUserById(userID) == null);
        }
}
```

6.1.3 Systémové testy

Další oblastí, kterou je vhodné testovat, je to, jak se aplikací generované stránky zobrazují ve webových prohlížečích. I když je totiž samotná aplikace napsána správně, může se stát, že kvůli použitému prohlížeči, nastavení webového serveru, anebo i z jiných důvodů nebude správně zobrazovat to, co by zobrazovat měla. Zde přichází ke slovu nástroj Selenium, pomocí něhož lze automatizovat procházení webovou aplikací a simulovat tak chování potenciálního uživatele. Bez tohoto nástroje bychom byli nuceni aplikaci po každé úpravě znovu „proklikat“, abychom ověřili, že provedené změny neovlivnily správnost zobrazení.

Selenium nám umožňuje takové procházení naší aplikací nahrát jako makro, případně zapsat pomocí speciálního skriptovacího jazyka, anebo dokonce i v našem programovacím jazyce (podrobný seznam viz [17], Java je podporována). Testy pak můžeme spouštět manuálně či automaticky po každém buildu bez našeho zásahu, případně je exportovat do formátu kompatibilního s použitým frameworkem JUnit.

Příklad testu v rozšíření prohlížeče Firefox Selenium IDE:



Obr. 12: Příklad testu v Selenium IDE

6.1.4 Testy uživatelského rozhraní

Tyto testy vypovídají o kvalitě návrhu uživatelského rozhraní z hlediska uživatele. Toto rozhraní by mělo být intuitivní, mělo by mít samopopisné vlastnosti, odpovídat očekávání uživatele. Dále by mělo být vhodné pro daný úkol, kontrolovatelné uživatelem a mít určitou toleranci k jeho chybným volbám, tj. možnost rozhodnutí vrátit, anebo si před provedením nezvratného úkonu alespoň vyžádat výslovný souhlas. [18]

Uživatelské rozhraní se nejlépe testuje empirickým průzkumem mezi předtím nezainteresovanými uživateli, lze se ale též vydat cestou heuristické analýzy či kognitivního průchodu. Vhledem k omezenému počtu implementovaných funkcionalit mojí aplikace se mi tyto testy jevily prozatím zbytečné, při budoucím vývoji a přidávání nových funkcionalit by však bylo vhodné k nim přikročit.

6.1.5 Testovací data

Pro všechny nestatické kategorie testů bylo třeba naplnit databázi testovacími daty, nad kterými bylo teprve možné testy provádět. SQL výrazy pro vložení těchto dat se nacházejí v souboru **data.sql** a jedná se typicky o 20-30 záznamů na 1 tabulku. Data neobsahují žádné podstatné údaje osobního charakteru, s výjimkou mého jména a testovací emailové adresy, která byla vytvořena přímo za účelem testování.

Kapitola 7 – Závěr

V této práci jsem se zabýval analýzou, návrhem a implementací webové aplikace zaměřené na zprostředkování doučování. Byla provedena rešerše řešení existujících na českém internetu, díky které jsem získal podrobnější představu o tom, co tato řešení svým uživatelům nabízejí a co jim podle mého názoru chybí. Z rešerše vycházela analýza, do níž jsem zahrnul i další požadavky na aplikaci, které sice nevyplývaly přímo ze zadání, ale které by při úvahách o reálném nasazení aplikace její součástí určitě být měly. Tyto funkcionality byly zatím implementovány jen zčásti.

Aplikace byla testována na několika různých úrovních a případné nálezy byly postupně opravovány podle toho, jak kritický byl jejich dopad na aplikaci a jak složitá byla jejich náprava.

Všechny cíle definované v zadání byly splněny.

Co se týče budoucího vývoje aplikace, za nejbližší další kroky považuji:

- doplnit implementaci zbývajících funkcionalit dle analýzy
- provést analýzu odolnosti aplikace proti různým bezpečnostním hrozbám a podle možností ji zabezpečit
- podrobit uživatelské rozhraní empirickým testům za účelem jeho vylepšení a zpříjemnění pro potenciální uživatele
- nasadit aplikaci do zkušebního provozu na některé nízkonákladové platformě
- začít analyzovat možnosti nových funkcionalit a poptávku po nich; mezi takové funkcionality by mohla patřit podpora online doučování či umožnění plateb za kurzy

Osobní přínos této práce pro mě spatřuji v tom, že jsem se naučil zacházet s množstvím frameworků a knihoven, se kterými jsem předtím neměl žádné zkušenosti. Donutit všechny tyto dílčí složky aplikace uspokojivě spolupracovat a komunikovat nebylo vždy jednoduché, o to cennější jsou ale získané zkušenosti. Co se týče analytické fáze práce, jako určitou nevýhodu vnímám to, že jsem aplikaci vyvíjel „pro sebe“, a nemusel jsem tedy analýzu ani implementaci probírat a dohadovat se o ní se zákazníkem či nadřízeným, a tím pádem jsem se musel postarat i o vstupy, které by obvykle byly dodány jedním z nich.

Při implementaci jsem si opět ověřil, jak je při každém projektu důležité plánování času, kterého i při těch nejlepších úmyslech bývá v závěrečných fázích akutní nedostatek. Za úplně největší zkušenost pak považuji to, že jsem si v plném rozsahu uvědomil význam často opakované poučky o tom, jak je důležité snažit se minimalizovat provázanost mezi různými částmi aplikace. To platí tím více, čím více roste velikost vyvíjené aplikace, a u rozsáhlejších aplikací by programátor tuto poučku měl mít neustále na zřeteli, protože čas, který vydá na řízení se podle ní, se mu později mnohonásobně vrátí ve formě ušetřených hodin při testování, hledání chyb a dalším rozšiřování.

Použitá literatura

[1] PRESSMAN, Roger S. *Software engineering: a practitioner's approach*. 6. ed. New York: McGraw-Hill, 2005. International Edition. ISBN 00-712-3840-9.
[2] ARLOW, Jim. *UML 2 and the unified process: practical object-oriented analysis and design*. Vyd. 1. Boston: Addison-Wesley, 2005, 592 s. ISBN 03-213-2127-8.
[3] MARTIN, Robert C. *UML for Java programmers*. Upper Saddle River, NJ: Prentice Hall PTR, c2003, xxiv, 249 p. ISBN 01-314-2848-9.
[4] PECINOVSKÝ, Rudolf. *Návrhové vzory*. Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN 978-80-251-1582-4.
[5] Přispěvatelé Wikipedie. *Java EE* [online]. 2014. [citováno 14.5.2014] Dostupné z: <http://cs.wikipedia.org/wiki/Java_EE>
[6] Pavel Mička. *Materiály k předmětu AD7B39WPA – Spring Framework* [online]. 2013. [citováno 14.5.2014] Dostupné z: <https://cw.felk.cvut.cz/wiki/_media/courses/a7b39wpa/spring1.pdf>
[7] Apache. *What is Apache Maven* [online]. 2014. [citováno 14.5.2014] Dostupné z: <http://maven.apache.org/what-is-maven.html>
[8] Sun Microsystems. *Java Persistence API FAQ* [online]. 2008. [citováno 14.5.2014] Dostupné z: <http://web.archive.org/web/20080822023926/http://java.sun.com/javaee/overview/faq/persistence.jsp>
[9] Oracle. *JavaServer Faces Technology* [online]. 2014. [citováno 14.5.2014] Dostupné z: <http://docs.oracle.com/javaee/7/tutorial/doc/jsf-intro.htm>
[10] Petr Aubrecht. *Materiály k předmětu AD7B39WPA – Java Server Faces* [online]. 2013. [citováno 14.5.2014] Dostupné z: <https://cw.felk.cvut.cz/wiki/_media/courses/a7b39wpa/12z__prednaska5-jsf.pdf>
[11] Thoughtworks. *Technology Radar* [online]. 2014. [citováno 14.5.2014] Dostupné z: <http://thoughtworks.fileburst.com/assets/technology-radar-jan-2014-en.pdf>
[12] Martin Fowler. *Anemic Domain Model* [online]. 2003 [citováno 14.5.2014] Dostupné z: <http://martinfowler.com/bliki/AnemicDomainModel.html>
[13] Bogdan Kostov. *Materiály k předmětu AD7B39WPA – příklad aplikace postavené na Java EE* [online] 2014. [citováno 16.5.2014] Dostupné z: <https://cw.felk.cvut.cz/wiki/_media/courses/a7b39wpa/bookstore-jpa-spring-junit-jsf.zip>
[14] Pavel Mička. *Materiály k předmětu AD7B39WPA – Architektura aplikací* [online]. 2013. [citováno 16.5.2014] Dostupné z: <https://cw.felk.cvut.cz/wiki/_media/courses/a7b39wpa/architektura.pdf>
[15] MYERS, Glenford J. *The art of software testing*. 2nd ed. Hoboken: Wiley, c2004, xv, 234 s. ISBN 978-0-471-46912-4.
[16] RAINSBERGER, J a Scott STIRLING. *JUnit recipes: practical methods for programmer testing*. Greenwich: Manning, c2005, xxx, 721 s. ISBN 19-323-9423-0.
[17] Selenium HQ. *Selenium Documentation* [online] 2014. [citováno 19.5.2014] Dostupné z: <http://docs.seleniumhq.org/docs/index.jsp>
[18] Přispěvatelé Wikipedie. *User interface design* [online]. 2014. [citováno 19.5.2014] Dostupné z: <http://en.wikipedia.org/wiki/User_interface_design>

Příloha A – Instalační příručka

Doporučená konfigurace:

- JRE 6, JDK 6
- Apache Tomcat 7.0.41
- NetBeans IDE 7.4
- Apache Derby 1.7.0.45

Nejprve si vytvoříme databázi přímo v NetBeans IDE. Defaultní paramatry jsou:

- název databáze: portaldoucovani
- uživatelské jméno: app
- heslo: application
- port: 1527

Tato nastavení se hodí pro testovací provoz, v reálném nasazení by samozřejmě použité parametry (zvláště uživatelské jméno a heslo) byly jiné.

Poté otevřeme mavenovský projekt, zvolíme „Run“ a vybereme Apache Tomcat jako „deployment server“.

Pokud bychom chtěli využít dodaných testovacích dat, využijeme soubor **data.sql**.

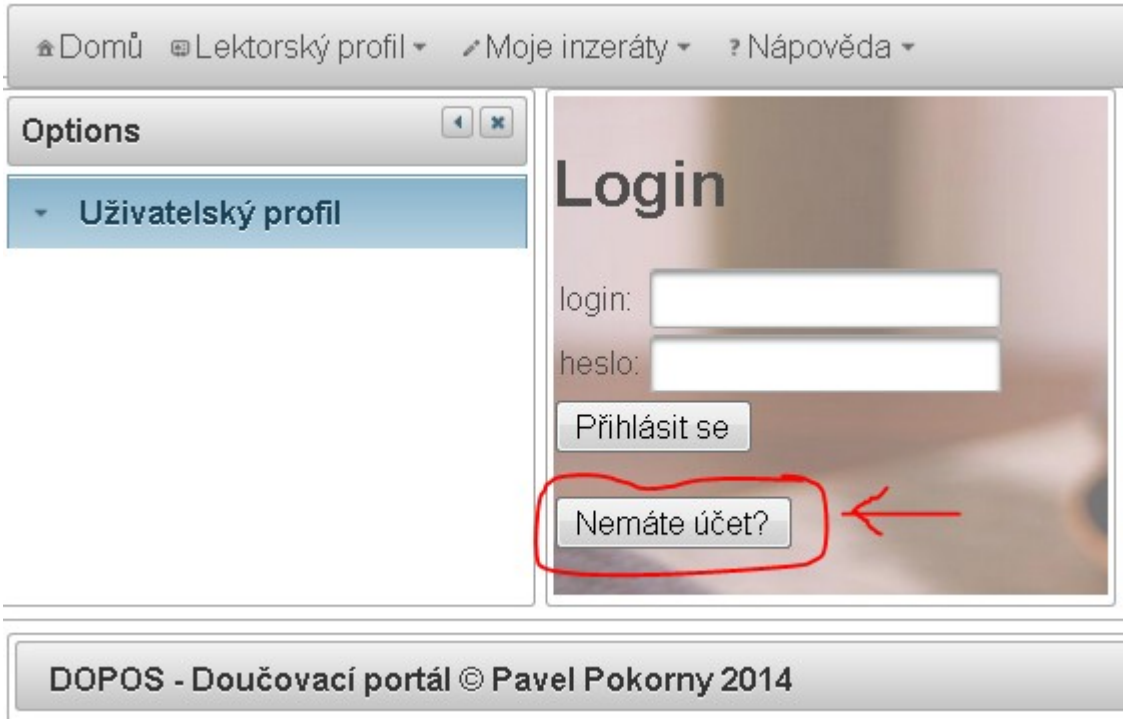
Příloha B – Uživatelská příručka

Tato příručka popisuje pouze registraci nového uživatele a přihlášení. Další základní úkony, které může uživatel v aplikaci provádět, jsou popsány v detailnější příručce, která je uložena ve složce **Guide** na přiloženém CD. Tato příručka je dostupná také ze sekce „Nápověda“ přímo v aplikaci.

Registrace

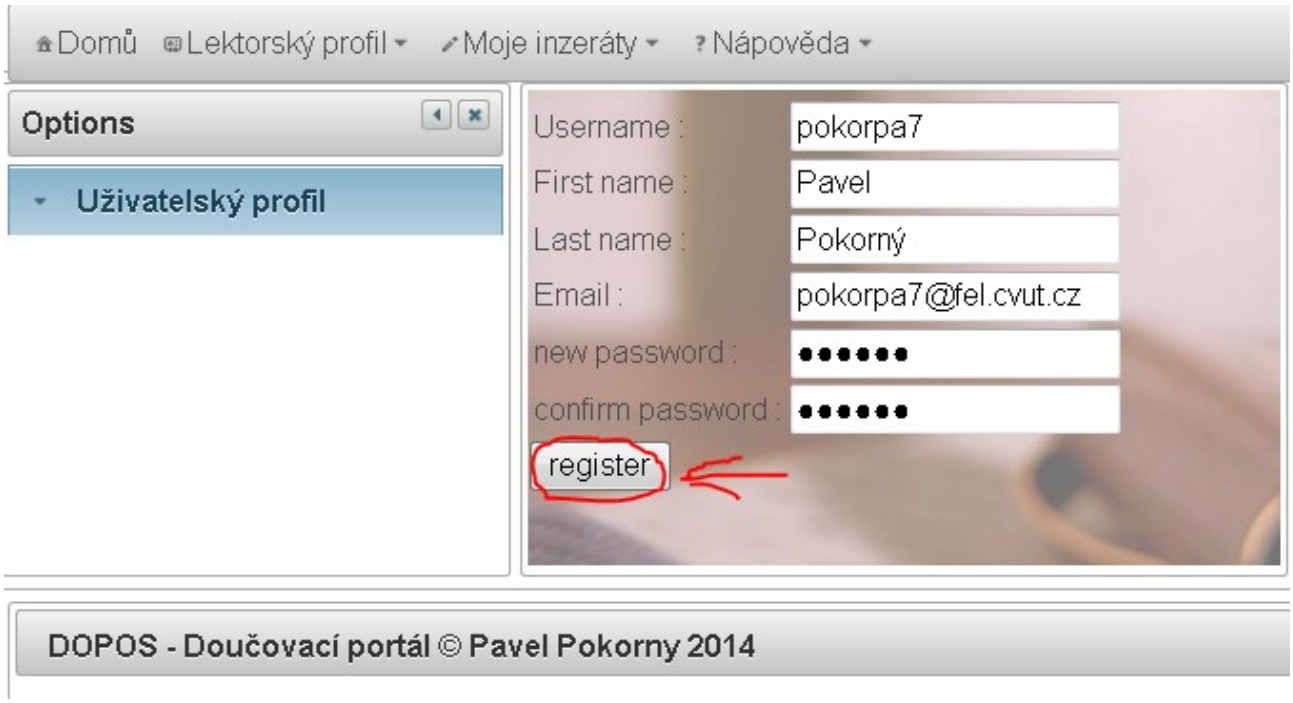
Po spuštění je aplikace dostupná na adrese http://localhost:8085/PortalDoucovani_/login.xhtml, případně podle adresy serveru, na němž je spuštěna.

Na uvítací obrazovce je třeba kliknout na „Nemáte účet?“



Obr. 13: Nový uživatel

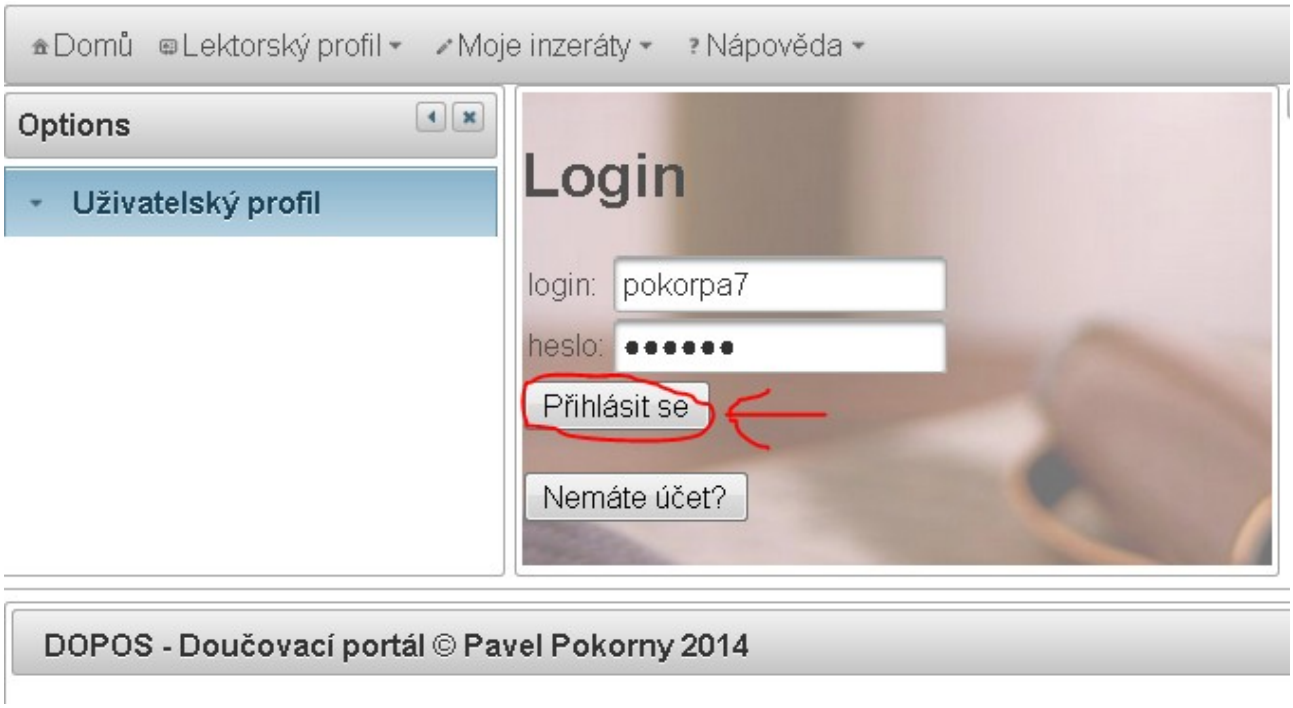
Zde vyplníme požadované údaje ve správném formátu a stiskneme tlačítko „Registrovat“:



Obr. 14: Registrace nového uživatele

Přihlášení

Pro přihlášení vyplníme na úvodní obrazovce naše uživatelské jméno a heslo:



Obr. 15: Přihlášení uživatele

Příloha C – Obsah přiloženého CD

Přiložené CD obsahuje tyto složky:

Složka	**Obsah**
bc	text samotné bakalářské práce ve formátu pdf; přílohy; zadání bakalářské práce
app	aplikace Portálu pro doučování; soubor s testovacími daty
source	zdrojové soubory aplikace
guide	uživatelská příručka
project	obsahuje .ea soubor s projektem v programu Enterprise Architect
project/html	obsahuje soubory s EA projektem exportovaným do formátu html

Tab. 6: Obsah přiloženého CD

Příloha D – Seznam obrázků

Seznam obrázků

Příloha E – Seznam tabulek

Seznam tabulek