SLOVENSKÁ TECHNICKÁ UNIVERZITA
Fakulta informatiky a informačných technológií

BÁZA ZNALOSTÍ A ZRUČNOSTÍ ŠTUDENTOV

(Tímový projekt)

Dokumentácia k projektu

Tím č.10 – ČERNÉ OFCE:
Bc. Martin Macko
Bc. Martin Paulech
Bc. Peter Rada
Bc. Miroslava Romanová
Bc. Tibor Schvartz
Bc. Lukáš Slížik
Študijný odbor: Informačné systémy
Kontakt: team1078@gmail.com
Školský rok: 2007/2008
Dátum: 14. novembra 2007

OBSAH

1 ÚVOD

1.1 Cieľ

Predpokladá sa, že súčasný študent vysokej školy nadobudol pred svojím štúdium na nej určité znalosti a zručnosti, ktoré si počas svojho štúdia môže ďalej rozširovať a zlepšovať sa v nich alebo sa oboznámiť s novými vedomosťami, ktoré získava na škole. Úroveň ovládania znalostí a zručností študenta je dôležitou informáciou a aj poznatkom najmä pre pedagógov, ale aj pre samotných študentov, korí sa týmto spôsobom môžu porovnávať so svojimi spolužiakmi a zistiť na akej úrovni sa ich znalosti nachádzajú.

Cieľom tohto projektu je vytvorenie informačného systému, ktorý bude študentovi slúžiť ako nástroj na sebahodnotenie svojich nadobudnutých znalostí a môže mu pomôcť pri štúdiu napríklad prideľovaním zadaní projektov, ktoré budú preňho ľahšie zvládnuteľné, nakoľko bude ovládať vedomosti potrebné na vyriešenie zadania. Študentovi sa okrem toho môže naskytnúť šanca pracovať v niektorej z firiem, ktoré hľadajú práve takého ako je on sám, so znalosťami potrebnými na získanie danej pracovnej pozície s prislúchajúcim finančným ohodnotením.

Vytvorením takéhoto informačného systému získajú v neposlednom rade aj pedagógovia, ktorým umožní vytvárať si na študenta vďaka jeho znalostiam názor, pohľad, prípadne majú možnosť potvrdiť a zvýšiť preferenciu najlepších študentov, na základe svojich skúseností získaných počas ich vyučovania. Taktiež im umožní vyhľadávať najvhodnejších študentov pre svoje projekty, súťaže, práce, odborné praktiká a pod.

Všeobecnou celkovou podstatou projektu je vytvorenie systému, ktorý uľahčí život na fakulte pedagógom a aj nám študentom.

Tento dokument slúži ako dokumentácia k informačnému systému Bázy znalostí a zručností študentov. V časti Úvod sa nachádza okrem cieľu projektu podkapitola Referencie a Slovník pojmov. Ďalšia kapitola sa venuje špecifikácii požiadaviek systému. Kapitola Analýza je venovaná rozboru systémov s podobnou tematikou, fakultným systémom, ktoré môžu svojimi informáciami prispieť k tomuto informačnému systému a analýze možných technológií. Návrh systému sa venuje konkrétnemu popisu nami navrhovaných jednotlivých častí systému v podobe architektúry systému a logického modelu údajov databázy.

V kapitole venovanej dokumentácii k prototypu je bližšie popísaný cieľ prototypu, technická dokumentácia, používateľská príručka k prototypu a zhodnotenie prototypu, ktorým prispel k vývoju informačného systému.

1.2 Referencie

[1] Stránka tímu The Llama team, ktorý riešil rovnaký projekt.
http://www2.dcs.elf.stuba.sk/TeamProject/2005/team06/

[2] Stránka tímu _elf_, ktorý riešili rovnaký projekt.
http://www2.dcs.elf.stuba.sk/TeamProject/2005/team11/

[3] Stránka AIS – Akademický informačný systém.
https://is.stuba.sk/

[4] Stránka Yonban – systém na prideľovanie projektov.
http://www2.fiit.stuba.sk/yonban/index.jsp

[5] Stránka FIIT Moodle - nástroj na elektronické vzdelávanie.
http://moodle.fiit.stuba.sk/moodle/

1.3 Slovník pojmov

DAO (Data Access Object): návrhový vzor, ktorý poskytuje abstraktné rozhranie niektorým databázovým mechanizmom

JDBC API (Java Database Connectivity Application Programming Interface): rozhranie, ktoré definuje prístup klienta k databáze

JDO (Java Data Objects): špecifikácia perzistencie Java objektov

MVC (Model-View-Controller): architektonický vzor; v komplexných aplikáciách sa separujú údaje (model) a používateľské rozhranie (view), takže zmena jedného neovplyvní druhé

Objektovo - relačné mapovanie:	programovacia technika, ktorá slúži na konverziu údajov medzi databázou a objektovo - orientovanými programovacími jazykmi; vytvára sa tak dojem virtuálnej objektovej databázy
Oracle Database XE:	Oracle Database Express Edition
ORM:	Object - Relational Mapping, objektovo - relačné mapovanie
PHP:	PHP: Hypertext Preprocessor, programovací jazyk, pôvodne určený na tvorbu dynamických web stránok
XML:	Extensible Markup Language

2 ANALÝZA

Táto kapitola sa venuje analýze predchádzajúcich riešených systémov s podobným zadaním, analýze systémov používaných na našej fakulte, analýze technológií a ich výhod a nevýhod v súvislosti s realizáciou projektu.

2.1 Analýza predchádzajúcich systémov

V školskom roku 2005/2006 boli na našej škole realizované dva projekty s veľmi podobnom tematikou. V nasledujúcich dvoch podkapitolách sa venujeme ich bližšej analýze s dôrazom na ich klady a zápory, ktoré pri tvorbe použili.

2.1.1 Analýza systému vytvoreného tímom _elf_

Tím _elf_ vychádzal z dvoch exitujúcich podobných systémov, na ktorých niektorí jeho členovia v minulosti pracovali. Jednalo sa o systémy z oblasti poisťovníctva a z oblasti logistiky. Tento tím vytvoril informačný systém, ktorý získaval zoznam študentov z informačného systému Študent. Na zabezpečenie aktuálnosti a zároveň predchádzaniu zdvojeniu dát, import údajov sa vykonával manuálne na podnet administrátora na konci každého semestra. Študenti si mali vo vlastnom záujme dbať o svoj profil znalostí a zručností. Výstup získaných informácii mal byť použiteľný ďalej v iných informačných systémoch ako napríklad YonBan..

Používateľ systému (vyučujúci alebo externý systém) si mohol určiť kritériá a priority, podľa ktorých si zoradí vybraných študentov. Vyučujúcim teda mali pomáhať získané informácie o doterajších skúsenostiach, schopnostiach a záujmoch študenta. Systém však neposkytoval priame informácie o študijných výsledkoch ani osobné číslo spolu s menom študenta kvôli dodržiavaniu zákona o ochrane osobných údajov (Zákon . 363/2005 Z. z. o ochrane osobných údajov v znení neskorších predpisov).

Medzi požívateľov systému tohto tímu definovali jeho členovia rolu študenta, vyučujúceho, externého systému (poskytoval informáciu o študentovi alebo využíval výstup z informačného systému Znalosti), administrátora (staral sa o správu systému a jeho správny chod počas prevádzky) a konfigurátora (rozhodoval o tom, aké typy informácií budú prístupné používateľom informačného systému).

Do prípadov použitia tím _elf_ zahrnul napríklad pridanie certifikátu, zručnosti, znalosti, hodnotenia, poskytnutie indexovaného zoznamu študentov, komplexnej alebo čiastočnej informácie o študentovi, známky, import údajov, notifikáciu, nastavenie profilu a správu jednotlivých častí. Vstupy systému boli známky, znalosti, zručnosti, certifikáty, hodnotenia a informácie o predmetoch a výstupy zoznam študentov podľa výsledného indexu, komplexná informácia o študentovi a informácia o predmete.

Komunikáciu s okolím predstavovali webový formulár, web services a import údajov zo Študenta. Systém bol zabezpečený autentifikáciou (login a heslo), autorizáciou a protokolom HTTPS. Zloženie systému vytvoreného týmto tímom pozostávalo z dátovej, aplikačnej a prezentačnej vrstvy. Členovia tímu vytvorili JSP stránky a aplikácie s grafickým rozhraním.

V implementácii systému sa zamerali na využitie služieb aplikačného servera Apache Tomcat, knižnice Apache Axis a Apache Xerces, na zostavenie aplikácii použili Apache Maven a na vývoj prostredia Java Eclipse.

2.1.2 Analýza systému vytvoreného tímom The Llama team

Tento tím vytvoril informačný systém s názvom Pallas, ktorý bol taktiež zameraný na bázu znalostí a zručností študentov a jeho navrhnutie vychádzalo predovšetkým z poznatkov získaných z informačných systémov iných univerzít.

Používateľom systému vytvoreného týmto tímom boli priradené roly študent, administrátor, pedagóg a pracovník pedagogického oddelenia. V systéme členovia tímu využili automatické notifikácie (pravidelné notifikácie študentov, aby si aktualizovali údaje v systéme), možnosť tvorby nových formulárov a ukladanie fotografie, životopisu a ďalších dokumentov v profile študenta. Výstup zo systému sa dal použiť ako vstup pre iné systémy,

s ktorými bol prepojený, a poskytoval možnosť automatického vstupu do systému napríklad prostredníctvom SOAP/web services.

Tím poskytol študentovi, aby si svoje nadobudnuté znalosti z technológií v systéme aj otestoval formou testu, ktorý sa následne automaticky vyhodnotil, alebo si študent mohol dať vygenerovať vlastný životopis. Samozrejmosťou pre systém bolo vyhľadávanie a triedenie informácií o študentoch podľa rôznych kritérií a vkladanie poznámok od pedagógov.

Výsledný systém poskytoval nepriamu informáciu o študentovi a jeho študijných výsledkoch, a teda či sa nachádza medzi desiatimi percentami najlepších študentov daného predmetu. Členovia tímu uvažovali aj nad rozšírením systému o informácie týkajúce sa registrovania a prideľovania tém v súvislosti so systémom Yonban.

2.2 Analýza systémov na fakulte

Na našej fakulte existujú a používajú sa v súčasnosti tri informačné systémy, ktoré by mohli obsahovať údaje dôležité pre nami vytváraný systém. V nasledujúcich kapitolách sa nachádza ich bližší popis a informácie, ktoré by sme z nich chceli čerpať.

2.2.1 Akademický informačný systém (AIS)

AIS funguje na našej fakulte približne druhý rok, je to odkúpený elektronický systém zahŕňajúci všetky informácie týkajúce sa predovšetkým študentov a pedagógov navštevujúcich a vyučujúcich na Slovenskej technickej univerzite v Bratislave.

Pre náš systém sme sa rozhodli čerpať údaje najmä z tohto systému, ale dbať pritom na ochranu osobných údajov. Relevantnými informáciami pre projekt by malo byť osobné číslo, meno a priezvisko študenta, email študenta, predmety, ktoré vyštudoval alebo študuje, známky, ktoré sa však v hodnotení nášho systému nezobrazia, ale len použijú.

Nepredpokladáme, že by bolo v našom systéme potrebné uchovávať ešte ďalšie informácie o študentovi, keďže všetky tieto sú v AIS a tým by sme ich iba duplikovali.

Z dôvodu uchovávania osobného čísla študenta v našom systéme nebude pre nás problém v prípade potreby získať z AIS o študentovi ďalšie informácie.

Keďže systém je založený na databáze vytvorenej v Oracle, pre ľahšiu prácu s dátami sme si aj my zvolili implementáciu databázy práve v tomto prostredí.

2.2.2 Yonban

Existencia tohto systému sa datuje od roku 2002, kedy bol vytvorený študentmi v rámci predmetu Tímový projekt. Jeho prvoradým cieľom je poskytnúť študentom a pedagógom ucelený systém, zefektívňujúci prácu na projekte vo všetkých jeho životných cykloch.

Poskytuje študentom možnosť registrovať sa na jednotlivé témy dôležitých školských projektov (bakalárska práca, diplomová práca...), ktoré im môžu byť vedúcimi prác následne pridelené. Medzi ďalšie funkcie systému sa radí aj odovzdávanie vyriešených projektov a posudkov na ne.

Keďže sa v systéme nenachádza výsledné hodnotenie študenta za vypracovaný projekt (t. zn. známka, ktorú mu komisia udelila po jeho obhajobe, alebo iné relevantné informácie, ktoré by sme chceli z nášho hľadiska použiť pre náš systém), rozhodli sme sa, že informácie, ktoré sa nachádzajú v systéme Yonban, zatiaľ nepoužijeme pri riešení nášho projektu.

2.2.3 FIIT Moodle

Moodle je systém pre vytváranie kurzov založených na Internete a web stránkach. Je projektom určeným pre podporu sociálneho konštruktivistického rámca vyučovania. Slovo Moodle bolo pôvodne akronymom pre Modular Object Oriented Dynamic Learning Environment (modulárne objektovo orientované dynamické výukové prostredie).

V súčasnosti existuje na FIIT STU server s nainštalovaným Moodle. V rámci tohto systému sa nachádzajú len informácie o hodnotení študenta a ostatné informácie, ktoré sú irelevantné pre vyhodnocovanie znalostí študenta. Systém Moodle poskytuje možnosť vyexportovať hodnotenia študentov pre daný predmet, avšak tieto informácie sú taktiež dostupné prostredníctvom akademického informačného systému AIS STU. Z tohto dôvodu

považujeme Moodle ako nepodstatný zdroj informácií pre vyhodnocovanie znalostí študentov.

2.3 Analýza technológií

Táto kapitola sa zameriava na opis technológií, nad ktorými sme uvažovali, a na základe ich výhod príp. nevýhod sme sa rozhodli použiť najvhodnejšie z nich pri realizácii riešenia nášho projektu.

2.3.1 Apache Tomcat

Pri štúdiu technológií vhodných na realizáciu prezentačnej vrstvy projektu Báza znalostí a zručností študentov sme sa najmä kvôli zachovaniu čo najväčšej konzistentnosti a aj iným výhodám rozhodli využiť JSP technológiu. Na zobrazovanie stránok s aktívnym obsahom používateľom nášho systému je potrebné mať na strane serveru nainštalovaný Apache Tomcat.

Apache Tomcat ako technológia predstavuje akýsi „kontajner“ pre servlety, ktorý sa často používa pri implementácií Java Servletov a Java Server Pages technológií. Tomcat dokáže vystupovať aj ako plnohodnotný web server, ale k dispozícií sú aj konektory pre najpoužívanejšie web servery Apache a Microsoft IIS. Tieto konektory umožňujú transparentnú obsluhu požiadaviek na JSP stránky pomocou inštancie Tomcatu a obsluhu ostatného obsahu pôvodným web serverom. Zdrojové kódy sú vo všetkých verziách voľne dostupné na domovskej stránke.

2.3.2 Hibernate

Relačné databázy v súčasnosti nepopierateľne tvoria jadro moderných podnikov. Kým programovacie jazyky ako napríklad Java poskytujú intuitívny, objektovo-orientovaný prístup k jednotlivým entitám, podnikové informácie, ktoré sa skrývajú za týmito entitami sú silne

relačné vo svojej podstate. Hlavná výhoda relačných modelov spočíva v tom, že ich dizajn je oddelený od akéhokoľvek programovacieho prístupu.

Jednou zo súčasných technológií, ktorá umožňuje prepojenie relačných dát a objektovo-orientovanej paradigmy prostredníctvom objektovo/relačného mapovania (ORM) je Hibernate. Dokonca sa pár rokov po jeho objavení stal jednou z vedúcich ORM technológií súčasnosti. Veľkou výhodu Hibernate je fakt, že je to open source.

2.3.3 Java

Java ako programovací jazyk je voľne dostupný pre verejnosť, ktorý má širokú oblasť využitia, od osobných internetových stránok až po využívanie vo vesmírnom programe NASA. Java je od základov vytvorená ako čistý, prehľadný, bezpečný a objektovo orientovaný programovací jazyk. Syntax Javy sa odvodila z programovacieho jazyka C++, ale oproti tomuto jazyku má mnoho výhod, ktoré sú zamerané na dodržiavanie štyroch vyššie uvedených základných princípov.

Aplikácie a aplety napísané v Jave sa spúšťajú pomocou Java Virtual Machine (JVM), ktorý musí byť nainštalovaný na počítači používateľa. JVM vytvára prostredie, v ktorom sa vykonáva program napísaný v tomto jazyku.

2.3.4 J2EE

Java Enterprise Edition je priemyselný štandard pre vyvíjanie prenositeľných, robustných, škálovateľných a bezpečných Java aplikácií na strane servera. Je postavená na solídnych základoch Java SE (Standard Edition) a poskytuje webové služby, model komponentov, manažment a komunikáciu s API.

Hlavné výhody použitia J2EE:

- Jednoduchší vývoj. S pomocou J2EE technológie je vývojárom poskytnutá možnosť lepšieho a rýchlejšieho písania zdrojového kódu.
- EJB – jednoduchšie a lepšie využívanie JavaBeans technológie pomocou používania Plain Old Java Objects (POJO).

- Vylepšené web služby – ideálna implementačná platforma pre SOA (Service – Oriented Architecture)

2.3.5 JSP, PHP a ASP

PHP je „open source“ technológia, a teda jeho veľkou výhodou napríklad oproti ASP je cena. PHP je distribuované zadarmo a nezávislé na platforme. Funguje pod Windows, Linux, Solaris a mnohými inými systémami.

Medzi hlavné nevýhody PHP môžeme zaradiť nefunkčný objektový model. Knižnica základných funkcií PHP je totiž procedurálna a takmer vôbec nevyužíva objekty. Funkcie potom nevyhadzujú ani výnimky, čo v podstate znemožňuje objektové programovanie.

JSP je technológia, ktorá umožňuje vývojárom rýchlo vytvárať webové stránky a aplikácie. JSP je založené na objektovo orientovanom jazyku Java a umožňuje vytváranie robustných webových systémov.

Medzi najvýznamnejšie výhody JSP patrí:

- multi - platformová technológia,
- znovuvyužívanie existujúcich komponentov pomocou JavaBeans a Enterprise Java Beansov (EJB),
- výhoda Javy,
- prenositeľnosť JSP súborov na akúkoľvek inú platformu, web server alebo JSP servlet engine,
- HTML a grafika zobrazovaná v internetovom prehliadači tvorí prezentačnú vrstvu a Java kód (JSP) tvorí implementačnú vrstvu.

JSP a ASP sú svojou funkcionalitou takmer podobné. ASP je automaticky inštalované spolu so serverom IIS, ale nesie so sebou skryté výdavky. Ak potrebujete dodatočné služby ako napríklad možnosť uploadovať súbory, šifrovanie alebo správu e-mailov, je potrebné ich dokúpiť.

ASP aplikácie bežia len na vybraných operačných systémoch. ASP nedosahuje rýchlosť PHP a je založená na COM architektúre. Ak programátor používa VBScript, volá neustále COM objekty, čo zapríčiňuje nepríjemné spomalenie.

Porovnanie hlavných funkcií PHP, JSP a ASP:

Funkcia	ASP	PHP	JSP
Separácia medzi obsahom a logikou	Čiastočné	Čiastočné	Úplné
Cieľový používateľ	Programátor	Programátor	Web vývojár, autor stránky
Jazyk	VBScript	Pearl, C	Java
Rozšíriteľný pomocou third-party komponentov	Nie	Nie	Áno

2.3.6 Spring

Spring Framework je open source aplikácia pre vývoj softvéru. Skladá sa z kolekcie menších modulov, ktoré sú navrhnuté tak, aby pracovali nezávisle a poskytovali lepšiu funkcionalitu. To znamená, že komponenty sa vyvíjajú nezávisle a integráciu zabezpečuje jeden človek.

Medzi jeho ďalšie výhody rozhodne patrí značné zjednodušenie vývoja, nakoľko obsahuje podporu pre webové aplikácie (jednoduché vytvorenie prezentačnej vrstvy, flexibilný Model-View-Controller, podpora JSP, Velocity a ďalších), podporu pre objektovo-relačné mapovanie (štandardné aplikácie Hibernate, TopLink, JDO a ďalšie) alebo aspektovo-orientovanú funkcionalitu.

Modul DAO (Data Access Object) poskytuje abstraktnú vrstvu pre prácu s JDBC API (Java Database Connectivity Application Programming Interface). Jeho hlavné výhody sú, že odstraňuje veľa zbytočného kódu (získanie spojenia, zrušenie spojenia, iterovanie cez

výsledky a i.). Správa výnimiek je prevedená z *java.sql.SQLException* do inteligentnej hierarchie Runtime výnimiek, o ktoré sa nemusí starať programátor. Snahou je, aby sa programátor zameral len na prácu s SQL a extrahovanie výsledkov. Za jeho nevýhodu sa dá považovať nemožnosť priamo volať vzdialené objekty.

2.3.7 Oracle

Ide o technológiu ponúkajúcu databázový systém riadenia, umožňujúci transformáciu údajov a dát do informácií. Tento systém umožňuje používateľom vytvárať, aktualizovať a vyberať informácie z databázy. Oracle je v súčasnosti významným nástrojom pri práci s databázovými štruktúrami.

Výhody:

- jednoduchá inštalácia a manažment,
- intuitívne používateľské rozhranie (browser-based),
- kompatibilita s množstvom súčasných technológií (PHP, Java, .NET, XML).

Nevýhody:

- nutnosť zakúpenia komerčnej verzie pre projekty rozsiahlejšieho charakteru (voľne dostupné riešenie ponúka len Oracle Database XE),
- hardvérové obmedzenia v Oracle Database XE (max. 4GB používateľských údajov, max. 1GB RAM).

2.3.8 Zhodnotenie

Z uvedenej analýzy technológií sa nám javí ako najvhodnejšie riešenie použitie Javy, konkrétne J2EE. Vzhľadom na čo najväčšiu konzistentnosť vytváraného systému je výhodné vytvoriť prezentačné vrstvu pomocou JSP technológie, čím sa nám taktiež otvára možnosť efektívneho prepojenia s Oracle databázou a aplikačnou vrstvou vytváraného systému Bázy znalostí a zručností študentov. Už samotnú implementáciu prezentačnej, aplikačnej a dátovej vrstvy budeme riešiť pomocou Spring frameworku, ktorý zjednoduší a urýchli celý proces vytvárania systému. Spring sme zvolili kvôli modulom Hibernate, web services a iným, ktoré tento framework obsahuje.

3 ŠPECIFIKÁCIA POŽIADAVIEK

Táto kapitola sa zaoberá požiadavkami na vytváraný systém. Je rozdelená do štyroch častí. Prvá obsahuje charakteristiku jednotlivých používateľov systému, v druhej je uvedený diagram prípadov použitia, tretia sa venuje špecifikácii funkcií systému a ostatné požiadavky sa nachádzajú v poslednej časti.

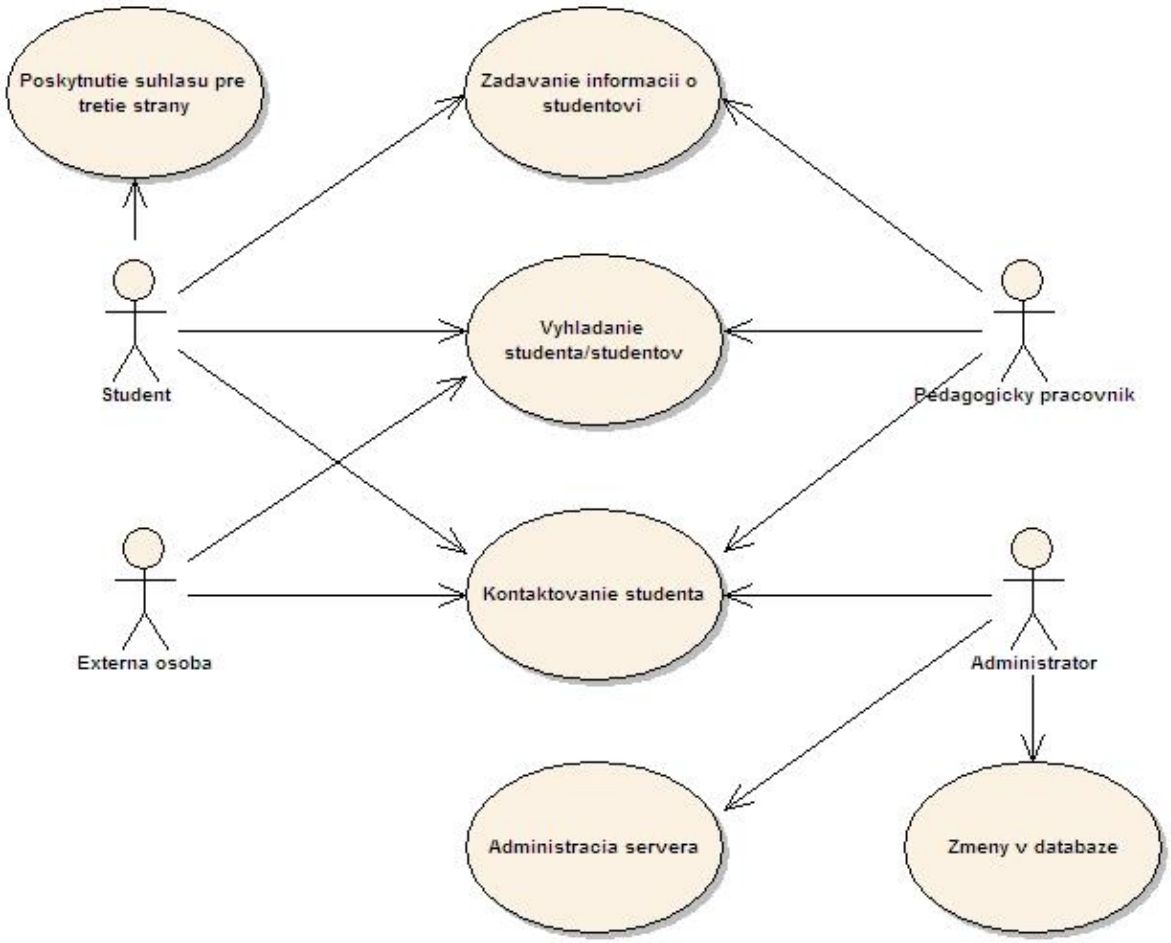
3.1 Charakteristiky používateľov

Z hľadiska rolí používateľov systému sú identifikovaní nasledovní používatelia:

Študent	Zadáva do systému informácie o vlastných znalostiach a zručnostiach.
Pedagogický pracovník	Zadáva do systému informácie o študentoch. Požaduje od systému znalosti o študentoch.
Administrátor	Spravuje systém a má na starosti jeho správny chod počas rutinnej prevádzky.
Externá osoba	Využíva znalosti poskytované systémom po súhlase študenta.

3.2 Diagram prípadov použitia

Na obrázku č.1 je znázornený diagram prípadov použitia, ktorý poskytuje prehľadnú informáciu o poskytovanej funkcionalite vo vzťahu k jednotlivým používateľom.



Obr.1: Diagram prípadov použitia

3.3 Požiadavky na funkcie systému

3.3.1 UC - Poskytnutie súhlasu pre tretie strany

Vstup: súhlas/nesúhlas

Výstup: uloženie vstupu do databázy (upravená databáza)

Používatelia: Študent

Opis: evidovanie rozhodnutia študenta zverejniť informácie o sebe aj tretej strane

3.3.2 UC - Zadávanie informácií o študentovi

Vstup: znalosti, zručnosti, študijné výsledky...

Výstup: uloženie vstupov do databázy (upravená databáza)

Používatelia: Študent, Pedagogický pracovník

Opis: vloženie informácií týkajúcich sa znalostí, zručností a študijných výsledkov študenta

3.3.3 UC - Vyhľadanie študenta/študentov

Vstup: požiadavky na študenta

Výstup: študent/zoznam študentov

Používatelia: Študent, Pedagogický pracovník, Externá osoba

Opis: systém nájde študenta/študentov na základe zadaných kritérií hľadania

3.3.4 UC - Kontaktovanie študenta

Vstup: požiadavka na študenta

Výstup: spojenie so študentom

Používatelia: Študent, Pedagogický pracovník, Administrátor, Externá osoba

Opis: pri potrebe spojenia sa so študentom z rôznych príčin s využitím kontaktu na študenta (mail, tel. č., icq, ...)

3.3.5 UC - Administrácia servera

Vstup: údaje týkajúce sa administrácie (napr. prístupové práva používateľov)

Výstup: úprava údajov, nastavenia servera

Používatelia: Administrátor

Opis: zabezpečenie správneho chodu servera

3.3.6 UC - Zmeny v databáze

Vstup: správna alebo chýbajúca informácia v databáze

Výstup: upravená databáza

Používatelia: Administrátor

Opis: v prípade nekonzistentnosti dát v databáze uskutoční administrátor nevyhnutné zmeny

3.4 Ďalšie požiadavky

Bezpečnosť a ochrana informácií

Vzhľadom na to, že systém spracováva a uchováva údaje chránené zákonom o ochrane osobných údajov, musia byť tieto chránené pred náhodným ako aj nezákonným poškodením a zničením, náhodnou stratou, zmenou, nedovoleným prístupom a sprístupnením ako aj pred akýmikoľvek inými neprípustnými formami spracúvania. Systém by mal preto obsahovať štandardné spôsoby ochrany autentifikáciu a autorizáciu.

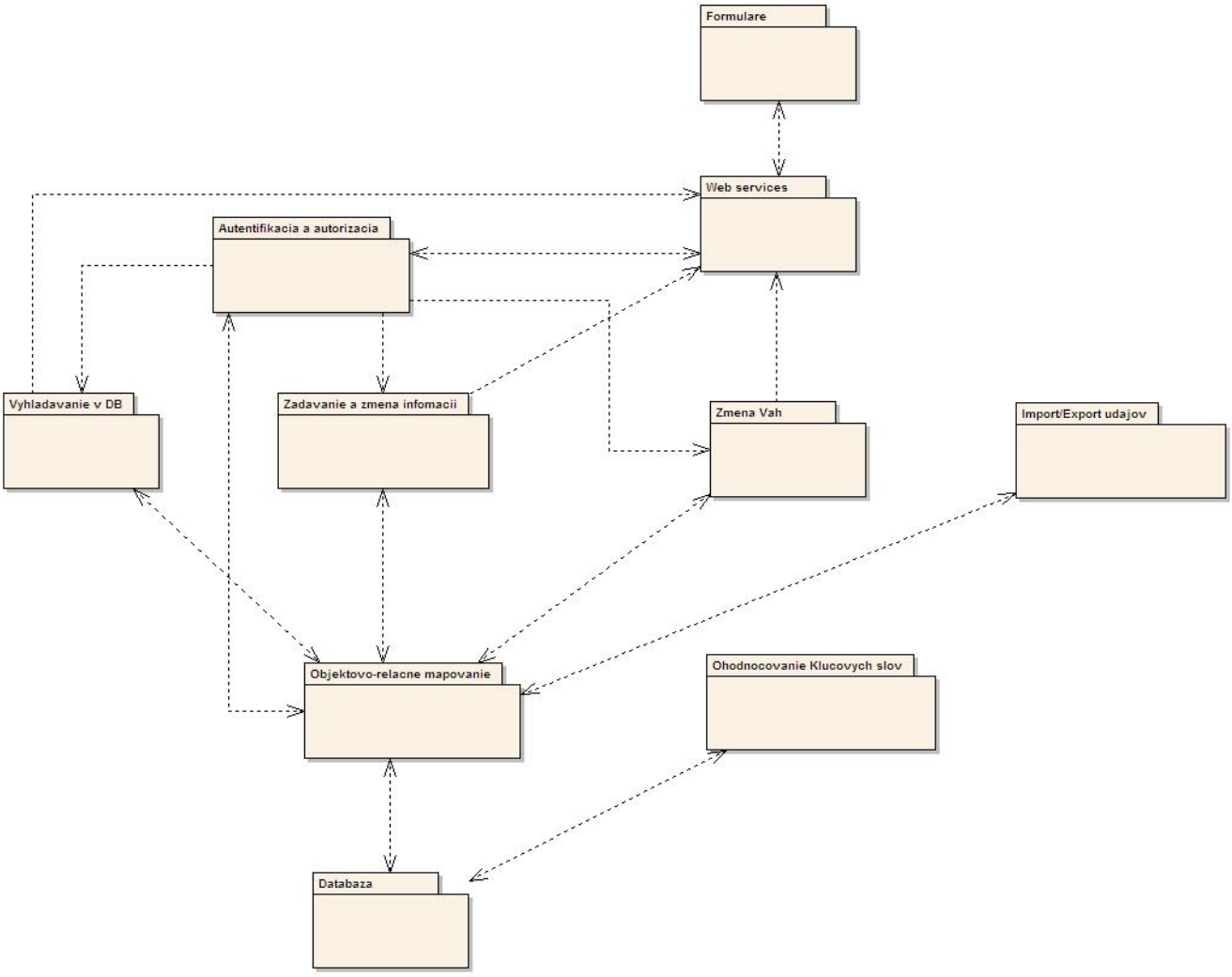
Používateľ, ktorý chce pracovať so systémom, sa musí najskôr prihlásiť pomocou prihlasovacieho formulára. Zadaním a potvrdením mena a hesla sa spustí proces autentifikácie. Zadané údaje sa porovnajú s údajmi, ktoré sú uložené v databáze. Pokiaľ sa nájde zhoda, používateľovi je umožnené prihlásenie sa do systému. Meno a heslo sú pri tomto procese posielané v šifrovanej podobe.

Každý používateľ je zaradený do jednej alebo viacerých skupín definujúcich jeho práva na používanie jednotlivých funkcií systému. Na základe členstva v skupinách sa mu sprístupnia dané funkcie.

4 NÁVRH

4.1 Architektúra systému

4.1.1 Diagram architektúry systému



Obr. 2: Architektúra systému

Používateľ pristupuje do systému cez formuláre (webové stránky). Bude vyzvaný, aby zadal svoje prístupové meno a heslo. Tieto údaje sa cez webové služby pošlú do modulu autentifikácie a autorizácie, kde sa zistí korektnosť zadaných údajov a aké práva daný

používateľ má. Následne bude môcť používateľ vykonávať v systéme akcie (podľa prístupových práv).

Vyhľadávanie v DB - autorizovaný používateľ si bude môcť vyhľadať v databáze potrebné informácie. Požiadavka používateľa sa odošle z formuláru do modulu webových služieb, kde sa identifikuje typ požiadavky, zistia sa práva používateľa (modul autentifikácie a autorizácie) a ak používateľ môže v databáze danú akciu vykonať, tak sa táto požiadavka pošle do modulu vyhľadávania v DB, ktorý ju ďalej spracuje a odošle modulu objektovo-relačného mapovania. Tento modul skonvertuje požiadavku do príkazov zrozumiteľných databáze. Databáza požiadavku spracuje a vráti modulu objektovo-relačného mapovania výsledok, ktorý ho skonvertuje do vhodnej formy a odošle modulu vyhľadávania informácií, kde sa spracuje do prezentačnej formy, odošle modulu webových služieb a ten sa postará o správne zobrazenie požadovaného výsledku vo formulári.

Zadávanie a zmena informácií - cez tento modul môže používateľ zadávať a modifikovať informácie v databáze. Používateľ sa najprv autentifikuje a autorizuje v systéme a následne bude môcť zadávať alebo meniť informácie, pre ktoré má práva. Zadá do formuláru požiadavku, ktorá sa odošle modulu webových služieb. Tu sa identifikuje typ požiadavky, oprávnenia a následne sa odošle modulu zadávania a zmeny informácií. V tomto module sa požiadavka spracuje a odošle sa modulu objektovo-relačného mapovania, ktorý požiadavku skonvertuje do jazyka zrozumiteľného databáze. Po odoslaní takto spracovanej požiadavky do databázy sa v nej vykonajú požadované zmeny a výsledok operácie (úspech/neúspech) sa cez modul zadávania a zmeny informácií a modul webových služieb prešíri do formuláru, kde sa zobrazí.

4.1.2 Popis modulov architektúry

Webové služby

Webové služby majú na starosti prekladanie požiadaviek zadaných prostredníctvom formulárov a naopak, prekladanie získaných informácií do webových formulárov.

Medzi webové služby patrí aj samotný proces prihlásenia sa do systému. Webové služby následne poskytujú informácie (login a heslo) o prihlásení pre modul autentifikácie a autorizácie.

Import/export údajov o študentovi

Modul importu/exportu údajov buď získava informácie o študentovi z relevantných zdrojov, ktoré sú spomenuté v časti Analýza systémov na fakulte a posiela ich modulu objektovo-relačného mapovania na spracovanie, alebo už takto spracované informácie o študentovi odosiela do systému Yonban. Toto získavanie informácií sa vykonáva v pravidelných intervaloch (napríklad jedenkrát za semester).

Zmena váh

V rámci procesu vzdelávania občas nastávajú zmeny v systéme výučby a vtedy môže dôjsť aj k zmenám váh, ktoré ovplyvňujú kľúčové slova uchované v databáze. Tieto kľúčové slová sú potrebné pre určenie znalosti študenta. Aby systém mohol reagovať na takéto zmeny a zakaždým určoval čo najpresnejšie znalosti študenta, musí obsahovať modul, pomocou ktorého budú tieto váhy upravené.

Ohodnocovanie kľúčových slov

Určitý používatelia, ktorý úspešne prejdú autentifikáciou majú možnosť vytvárať a modifikovať mieru akou jednotlivé kľúčové slová prispievajú k jednotlivým zručnostiam a znalostiam.

Formuláre

Formuláre tvoria prezentačnú vrstvu systému, zobrazovanú prostredníctvom používateľovho webového prehliadača. Slúžia ako vstupné rozhranie na zadávanie informácií potrebných pre vyhľadávanie študentov s konkrétnymi znalosťami a ako výstupné rozhranie pre zobrazenie informácií o hľadaných študentoch.

Zadávanie a zmena informácií

Poskytuje používateľovi možnosť zmeny určitých údajov v databáze (zmena osobných údajov, pridávanie poznámok pedagóga, zmeny váh predmetov, ...). Na základe úspešnej autentifikácie systém určí prístupové práva pre daného používateľa, rozsah a typ informácií, ktoré môže modifikovať.

Vyhľadávanie v DB

Databáza poskytuje zmysluplné a relevantné informácie na základe používateľských požiadaviek. Vyhľadávanie je podmienené úspešnou autorizáciou používateľa. Zjednodušene povedané, tento modul vytvára zoznam študentov zoradených podľa zadefinovaných požiadaviek.

Autentifikácia a autorizácia

Tento modul vyhodnocuje informácie, ktoré mu boli poslané z webových služieb. Na základe týchto informácii zistí totožnosť používateľa a podľa totožnosti určí, aké má práva v systéme.

Objektovo – relačné mapovanie

Predstavuje rozhranie medzi ostatnými modulmi a samotnou databázou. Smerom do databázy sú požiadavky transformované do príkazov zrozumiteľných pre konkrétne databázové prostredie (jazyk SQL) a smerom von z databázy sú naopak tieto informácie transformované do objektov (jazyk Java). Tento modul sa v systéme nachádza z dôvodu zvýšenia efektívnosti a jednoduchosti, ktorú prináša objektovo - orientovaná paradigma a práca s objektmi. Viac o objektovo-relačnom mapovaní je popísané v časti Analýza technológií (Hibernate).

Databáza

Tvorí jadro celého systému, pretože obsahuje informácie o študentoch (osobné číslo, meno, ročník, e-mail), ich výsledkoch, zručnostiach a znalostiach. Okrem týchto informácií sú

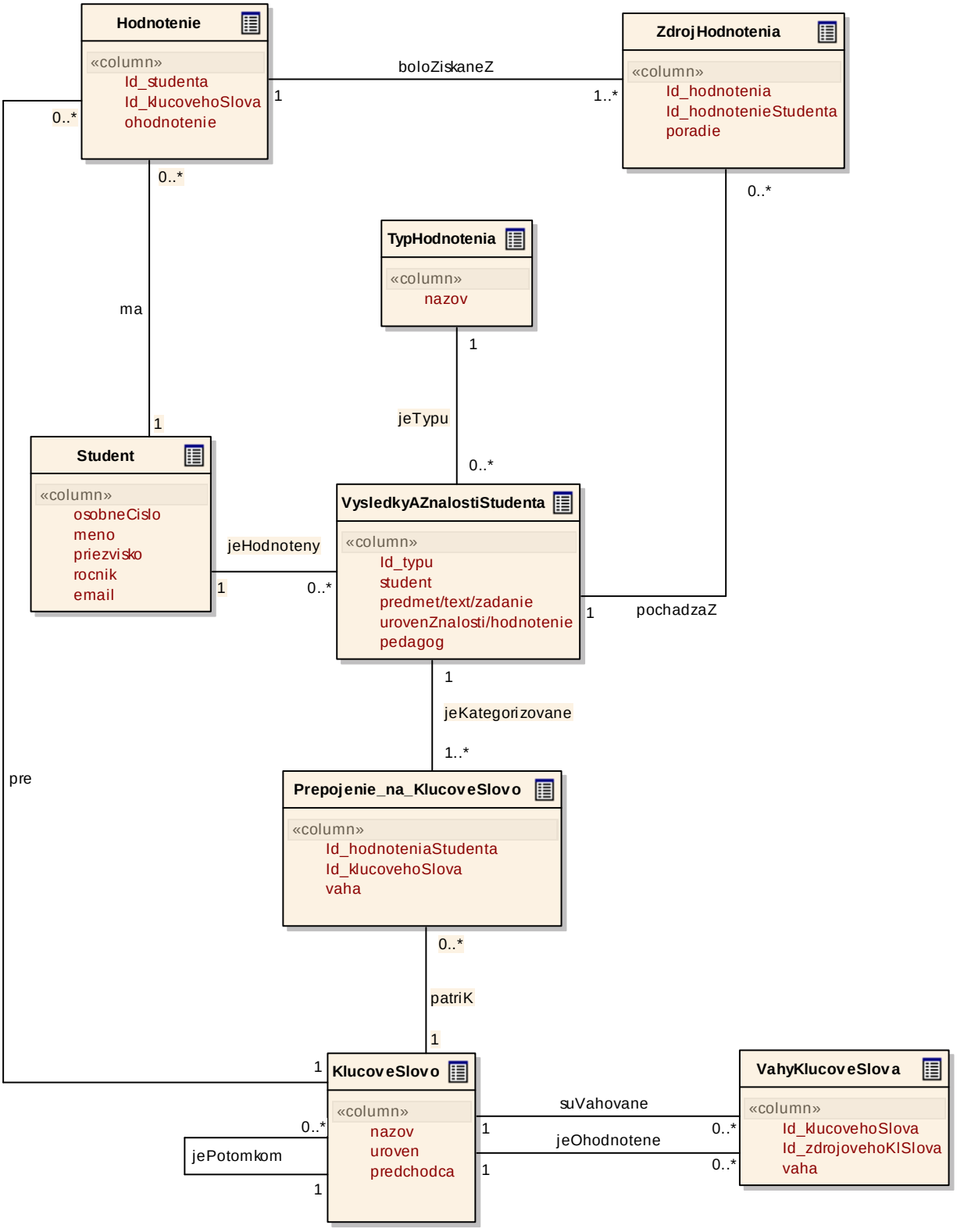
v databáze uložené aj prihlasovacie údaje pre registrovaných používateľov systému. Pri autentifikácií sú tieto údaje overované s údajmi v databáze a v prípade ich správnosti je používateľovi umožnená práca so systémov v rozsahu jeho právomocí.

4.2 Databáza

Táto kapitola obsahuje logický pohľad na uchovávané údaje. Vytvorený diagram modelu údajov identifikuje základné entity, ich atribúty a vzťahy medzi nimi. Taktiež Je v tejto kapitole uvedený príklad výpočtu ohodnotenia kľúčových slov.

4.2.1 Diagram modelu údajov databázy (logická úroveň)

Každá entita má svoje atribúty a je zviazaná s inými entitami pomocou vzájomných vzťahov. Na obrázku č.3 je znázornený logický model údajov navrhovanej databázy.



Obr. 3: Logický model údajov

Kľúčové slovo

Celá databáza je postavená na entite KľúčovéSlovo. Táto entita uchováva informácie o hierarchii kľúčových slov (tieto informácie sú uložené v atribútoch *úroveň* a *predchodca,* kde atribút *predchodca* je odkaz na kľúčové slovo, ktoré je priamym predchodcom daného kľúčového slova a kľúčové slovo, ktoré je na najvyššej úrovni bude mať tento atribút vynulovaný). Jedno kľúčové slovo má práve jedného predchodcu (kardinalita 1) a jedno kľúčové slovo môže byť predchodcom žiadneho (nemá nasledovníkov) alebo viacerých kľúčových slov (kardinalita 0..n).

VáhyKľúčovéSlová

Ak je kľúčové slovo predchodcom (má nasledovníkov) iného kľúčového slova, tak existuje väzba medzi entitami KľúčovéSlovo a VáhyKľúčovéSlová. V entite VáhyKľúčovéSlová je uchovaná informácia o tom, s akou váhou prispieva ohodnotenie daného kľúčového slova na ohodnotenie jeho predchodcu.

Väzba *jeOhodnotené:* jedna váha kľúčového slova prislúcha práve k jednému kľúčovému slovu (kardinalita 1) a jedno kľúčové slovo môže byť ohodnotené skrz váhy viacerých kľúčových slov (kardinalita 0..n).

Väzba *súVáhované*: jeden zdroj vo váhe kľúčového slova prislúcha k jednému kľúčovému slovu (kardinalita 1) a jedno kľúčové slovo môže byť zdrojom pre viacero kľúčových slov (kardinalita 0..n).

Študent

V entite Študent sú uchované informácie o jednotlivých študentoch (osobné číslo študenta, jeho meno a priezvisko, v ktorom ročníku práve študuje a jeho kontaktný e-mail).

VýsledkyAZnalostiŠtudenta

V tejto entite budú uložené všetky študijné výsledky a znalosti študenta. V atribúte *Id_typu* je vyjadrené o aký typ ide napríklad:

hodnotenie predmetu - v treťom atribúte entity bude identifikované o aký predmet ide a štvrtý atribút bude prázdny, lebo hodnotenie predmetu pre daného študenta sa bude

importovať z externého systému a v databáze sa ukladať nebude a taktiež aj piaty atribút bude prázdny

sebahodnotenie študenta - vtedy tretí atribút bude vyjadrovať čo študent na sebe hodnotí - akú znalosť, vo štvrtom atribúte bude uložená úroveň znalosti, ktorú študent dosiahol a piaty atribút bude prázdny

hodnotenie študenta pedagógom - tretí atribút vyjadruje znalosť, ktorú pedagóg na študentovi hodnotí, štvrtý vyjadruje úroveň znalosti a v piatom atribúte je identifikovaný pedagóg, ktorý hodnotenie do systému vložil

hodnotenie študentského projektu - v treťom atribúte je názov projektu, ktorú študent vypracoval, vo štvrtom je hodnotenie projektu a piaty atribút je prázdny.

Jeden výsledok/znalosť patrí práve k jednému študentovi (kardinalita 1) a jeden študent môže mať viacero výsledkov/znalostí (kardinalita 0..n).

Prepojenie_na_KľúčovéSlovo

V tejto entite sú uchované informácie, ku ktorým kľúčovým slovám daný výsledok/znalosť študenta patrí a s akou váhou ovplyvňuje ohodnotenie kľúčového slova.

Vzťah *VýsledkyAZnalstiŠtudenta-Prepojenie_na_KľúčovéSlovo:* jeden výsledok/znalosť študenta môže byť priradené k viacerým kľúčovým slovám (kardinalita 1..n) a jeden záznam tejto entity patrí k jednému výsledku/znalosti študenta.

Vzťah: *Prepojenie_na_KľúčovéSlovo -KľúčovéSlovo*: záznam tejto entity prislúcha k jednému kľúčovému slovu (kardinalita 1) a jedno kľúčové slovo môže mať viacero záznamov tejto entity (kardinalita 0..n).

Hodnotenie

V tejto entite sú uchované informácie aké má študent ohodnotenie kľúčových slov. Sú tu informácie ako ku ktorému študentovi a ktorému kľúčovému slovu dané hodnotenie patrí a samotné ohodnotenie.

Vzťah *Hodnotenie-Študent*: Jeden jedno hodnotenie patrí jednému študentovi (kardinalita 1) a jeden študent má viacero hodnotení (kardinalita 0..n).

Vzťah *Hodnotenie-KľúčovéSlovo:* jedno hodnotenie prislúcha k jednému kľúčovému slovu (kardinalita 1) a jedno kľúčové slovo môže mať viacero hodnotení (kardinalita 0..n).

ZdrojHodnotenia

Entita na uchovanie toho, ktoré výsledky/znalosti študenta najviac prispeli k ohodnoteniu kľúčového slova pre daného študenta. Atribútom je poradie - hodnotu 1 bude mať ten zdroj, ktorý najviac prispel k ohodnoteniu kľúčového slova.

Vzťah *ZdrojHodnotenia- VýsledkyAZnalstiŠtudenta*: Jeden zdroj prislúcha k jednému výsledku/znalosti študenta (kardinalita 1) a jeden výsledok/znalosť študenta môže byť vo viacerých zdrojoch (kardinalita 0..n).

Vzťah *ZdrojHodnotenia-Hodnotenie*: jeden zdroj patrí k jednému hodnoteniu (kardinalita 1) a jedno hodnotenie má viacero zdrojov (kardinalita 0..n).

Typ

Táto entita uchováva typy výsledku/znalosti študenta. Môže nadobúdať hodnoty ako napr. "Sebahodnotenie", "ŠtudentskéPráce" atď. Jeden výsledok/znalosť študenta je jedného typu (kardinalita 1) a jeden typ môže byť vo viacerých hodnoteniach (kardinalita 0..n).

4.2.2 Príklad výpočtu základného ohodnotenia kľúčových slov

Majme v tabuľke *Študent* (Tab.1) nasledovný záznam:

osobné č.	*meno*	*priezvisko*	*ročník*	*e-mail*
20851	Ján	Malý	3.	janko@it.sk

Tab.1: Študent

a pre tohto študenta budeme chcieť vypočítať ohodnotenia kľúčových slov, ktoré máme v tabuľke *KľúčovéSlová* (Tab.2)*:*

názov	*úroveň*	*predchodca*
programovacie jazyky	1	-
architektúra systémov	1	-
procedurálne prog. jazyky	2	programovacie jazyky
objektovo orientované prog. jazyky	2	programovacie jazyky
funkcionálne prog. jazyky	2	programovacie jazyky

Tab.2: KľúčovéSlová

V nasledujúcej tabuľke (Tab.3) sú váhy, ktorými prispievajú ohodnotenia jednotlivých predmetov k ohodnoteniu kľúčových slov:

kľúčové slovo	*predmet*	*váha*
architektúra systémov	architektúra informačných systémov	0,6
architektúra systémov	manažment v softvérovom inžinierstve	0,1
procedurálne prog. jazyky	algoritmizácia a programovanie	0,5
procedurálne prog. jazyky	dátové štruktúry a algoritmy	0,2
objektovo orientované prog. jazyky	objektovo orientované programovanie	0,4
objektovo orientované prog. jazyky	vývoj programov pre platformu Java2	0,4
funkcionálne prog. jazyky	funkcionálne a logické programovanie	0,8

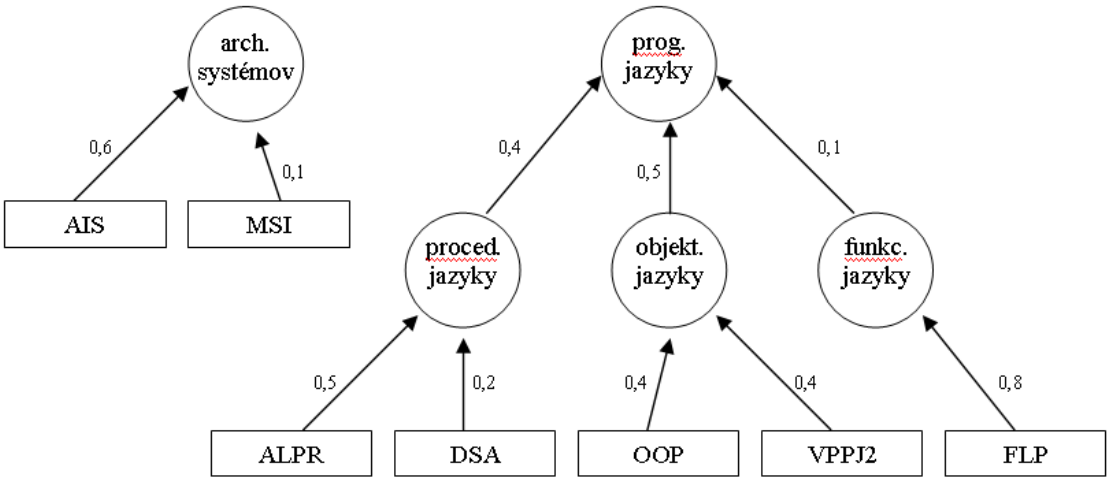
Tab.3: Váhy na ohodnotenie predmetov

a v tabuľke váh, ktorými prispievajú ohodnotenia kľúčových slov k ohodnoteniu ich predchodcov sú tieto záznamy (Tab.4):

kľúčové slovo	zdrojové kľúčové slovo	váha
programovacie jazyky	procedurálne prog. jazyky	0,4
programovacie jazyky	objektovo orientované prog. jazyky	0,5
programovacie jazyky	funkcionálne prog. jazyky	0,1

Tab.4: Váhy

Obrázok č. 4 znázorňuje hierarchiu kľúčových slov spolu s prepojeniami na predmety (názvy predmetov sú uvedené v skratkách).



Obr. 4: Hierarchia kľúčových slov

Teraz si môžeme ukázať ako vypočítať ohodnotenia kľúčových slov pre študenta Jána Malého.

Vezmeme si tabuľku kľúčových slov (Tab. 2). Zoberieme prvé kľúčové slovo ("programovacie jazyky") a vyhľadáme všetky predmety, ktoré prispievajú k ohodnoteniu

tohto kľúčového slova (v Tab. 3). Zistíme, že taký predmet neexistuje (k ohodnoteniu tohto kľúčového slova prispievajú len iné kľúčové slová, ale žiaden predmet). Toto slovo si teda zatiaľ odložíme bokom (spracujeme ho neskôr).

Zoberieme druhé slovo ("architektúra systémov") a vyhľadáme predmety (v Tab. 3). V tomto prípade nájdeme dva, ktoré ovplyvňujú hodnotenie tohto slova, a to "architektúra informačných systémov" s váhou 0,6 a "manažment v softvérovom inžinierstve" s váhou 0,1. Teraz potrebujeme zistiť ohodnotenie daných predmetov (AIS a MSI) pre študenta Jána Malého (túto informáciu dostaneme z externého systému - odtiaľ sme zistili že predmet AIS mal ohodnotený na 78 bodov a predmet MSI na 98 bodov).

Keď máme tieto informácie, tak môžeme kľúčové slovo "architektúra systémov" ohodnotiť. Jeho hodnotenie bude váženým súčtom zistených bodov podľa váh: hodnotenie prvého predmetu * váha, s ktorou prispieva prvý predmet k ohodnoteniu kľúčového slova + hodnotenie druhého predmetu * váha, s ktorou prispieva druhý predmet k ohodnoteniu kľúčového slova + ...

Čiže výsledné hodnotenie kľúčového slova "architektúra systémov" pre študenta Jána Malého bude 78*0,6 + 98*0,1=56,6 lebo predmet AIS mal Ján Malý ohodnotený na 78 bodov a toto hodnotenie prispieva k ohodnoteniu kľúčového slova váhou 0,6 a podobne pre predmet MSI je to 98*0,1. Takto prejdeme všetky ostatné kľúčové slová, ktoré máme v Tab. 3. Dostali by sme napr. takéto výsledky: "procedurálne prog. jazyky" má ohodnotenie 52,1; " objektovo orientované prog. jazyky" má 68,7 a "funkcionálne prog. jazyky " má 34,9.

Nakoniec nám ešte zostanú slová, ktoré sme si odložili bokom, lebo sme nenašli žiaden predmet, ktorý by prispieval k ich ohodnoteniu (v našom prípade je to len kľúčové slovo "programovacie jazyky"). Vezmeme tieto slová a hľadáme, ktoré kľúčové slová prispievajú k ich ohodnoteniu (Tab. 4). V našom príklade pre kľúčové slovo "programovacie jazyky" to sú "procedurálne programovacie jazyky", "objektovo orientované programovacie jazyky" a "funkcionálne programovacie jazyky".

Ohodnotenia týchto kľúčových slov sme už vypočítali a ohodnotenie kľúčového slova "programovacie jazyky" bude váženým súčtom daných kľúčových slov, teda v našom príklade by to bolo 52,1*0,4 + 68,7*0,5 + 34,9*0,1 = 58,7.

4.2.3 Hierarchia kľúčových slov

Ako hierarchiu kľúčových slov budeme využívať ACM klasifikáciu - http://oldwww.acm.org/class/1998/. Keďže je táto klasifikácia veľmi rozsiahla, použijeme z nej len vhodnú časť. To znamená, že niektoré kategórie použijeme do hĺbky 3 a niektoré iba do hĺbky 2, prípadne len do hĺbky 1, podľa toho, ako veľmi sú dané kategórie relevantné s obsahom výučby na fakulte. V prílohe A sú uvedené prvé dve úrovne ACM klasifikácie.

5 Prototyp

5.1 Cieľ prototypu

Cieľom tvorby nášho prototypu je oboznámiť sa s technológiami, s ktorými máme najmenšie skúsenosti a ktoré sú pre náš informačný systém kľúčové. Ďalším cieľom je implementácia týchto technológií v jednoduchom prostredí, ktoré je podobné nášmu informačnému systému. Za kľúčové technológie považujeme Hibernate a JSP, ktoré sme implementovali v prostredí JavaSE. Oproti návrhu systému sme urobili viacero zmien, z ktorých najpodstatnejšia spočíva v zmene databázového prostredia, kde nepoužívame databázový systém Oracle, ale jednoduchší systém PostgreSQL. Táto zmena nastala z dôvodu, že náš informačný systém nie je pre databázový systém Oracle dostatočne rozsiahly a preto by bolo nasadenie tohto databázového systému mrhaním systémovými prostriedkami.

Implementovaný prototyp nemá slúžiť na následný vývoj informačného systému, ale je to viac-menej "prototyp na zahodenie". Ide hlavne o získanie skúseností zo spomínaných technológií.

5.2 Technická dokumentácia

V prototype sú implementované základné funkcie z technológií Hibernate a JSP, ktoré ale budú vo finálnom informačnom systéme využívané veľmi často.

Z technológie Hibernate sú to:

- výber atribútov tabuľky na základe definovanej podmienky (prihlasovanie do systému, výpis všetkých študentov)
- vkladanie nových informácií do tabuľky (pridávanie používateľa)

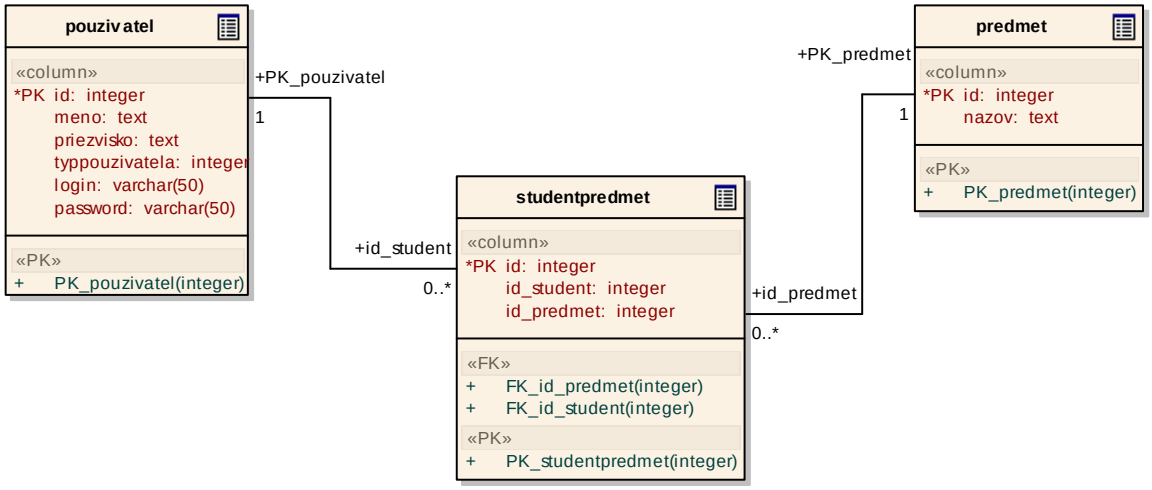
- výber informácií z tabuliek, ktoré sú vo vzťahu M:N (výpis predmetov zvoleného študenta)

Z technológie JSP sú to:

- vytváranie formulárov pre získanie potrebných informácií (pomocou listboxov, textových polí, atď.)
- zaslanie požiadavky a prípadných potrebných údajov pre Hibernate na vykonanie používateľom požadovanej funkcie
- spracovanie a výpis informácií, ktoré pošle Hibernate ako odpoveď na požadovanú funkciu

5.2.1 Databáza prototypu

Na obrázku č.5 je znázornený fyzický model databázy prototypu.



Obr.č.5: Fyzický model databázy prototypu

Databáza obsahuje 2 hlavné tabuľky, ktorými sú *pouzivatel* a *predmet*.

Tabuľka *pouzivatel* zoskupuje hlavné informácie o jednotlivých používateľoch. Týmito informáciami sú meno a priezvisko používateľa, jeho prihlasovacie meno a heslo a typ o akého používateľa ide.

Tabuľka *predmet* obsahuje informácie o vyučovacích predmetoch. Pre jednoduchosť je predmet popísaný iba jeho názvom.

Tabuľka *studentpredmet*, vyjadruje že ktoré predmety má ktorý používateľ (študent) zapísané a slúži na rozbitie vzťahu M:N, ktorý je medzi tabuľkami *pouzivatel* a *predmet*.

5.2.2 Objektovo relačné mapovanie

Pomocou Hibernate-u je možné namapovať jednotlivé tabuľky databázy na k nim prislúchajúce triedy v programe. Ale najprv je potrebné povedať Hibernate-u, kde má hľadať databázu, aká je to databáza a ako sa k nej pripojí. Pre tento účel je vytvorený jeden XML súbor, ktorý obsahuje práve tieto údaje a vyzerá takto:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
        <property name="connection.url">jdbc:postgresql://localhost/Baza</property>
        <property name="connection.username">Username</property>
        <property name="connection.driver_class">org.postgresql.Driver</property>
        <property name="dialect">org.hibernate.dialect.PostgreSQLDialect</property>
        <property name="connection.password">password</property>
        <property name= "transaction.factory_class"> org.hibernate.transaction.JDBCTransactionFactory
                </property>
<!--  thread is the short name for org.hibernate.context.ThreadLocalSessionContext and let Hibernate bind the
session automatically to the thread -->
         <property name="current_session_context_class">thread</property>

        <!-- mapping files -->
        <mapping resource="Pouzivatel.hbm.xml" />
        <mapping resource="Predmet.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

V časti session-factory v tag-och <property name=...> sú uvedené jednotlivé informácie o databáze aby sa k nej bolo možné pripojiť. Meno tag-u určuje čo daný tag popisuje a najdôležitejšie sú:

- "connection.url" - popisuje o akú databázu ide a kde je uložená
- "connection.driver_class" - popisuje, kde sa nachádza ovládač databázy, pomocou ktorého je možné s databázou manipulovať
- "connection.dialect" - definuje spôsob komunikácie s databázou
- "connection.username" - špecifikuje prihlasovacie meno do databázy
- "connection.password" - špecifikuje prihlasovacie heslo do databázy

Ďalším dôležitým tag-om je <mapping resource=...>, ktorý specifikuje meno súboru, v ktorom sa nachádza popis mapovania tabuľky databázy na triedy v aplikácii. V našom prototype máme dva súbory, ktoré popisujú mapovanie dvoch hlavných tried databázy. Tieto mapovania sa nachádzajú v súboroch s názvami *Pouzivatel.hbm.xml* a *Predmet.hbm.xml.* Obsah súboru *Pouzivatel.hbm.xml* je uvedený nižšie (obsah súboru *Predmet.hbm.xml má obdobnú štruktúru*):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >
<hibernate-mapping>

 <class name="Pouzivatel" table="pouzivatel">
        <id name="id" column="id" type="java.lang.Integer">
                <generator class="sequence">
                        <param name="sequence">student_id_seq</param>
                </generator>
        </id>
        <property name="meno" column="meno" type="java.lang.String" />
        <property name="priezvisko" column="priezvisko" type="java.lang.String" />
        <property name="typPouzivatela" column="typPouzivatela" type="java.lang.Integer" />
        <property name="login" column="login" type="java.lang.String" />
        <property name="password" column="password" type="java.lang.String" />
```

```
        <set name="predmety" table="StudentPredmet">
                <key>
                        <column name="id_student" not-null="true" />
                </key>
                <many-to-many entity-name="Predmet">
                        <column name="id_predmet" not-null="true" />
                </many-to-many>
        </set>
</class>
```

Štruktúra XML súboru:

Tag <class>

name - identifikuje triedu v aplikácii, na ktorú sa namapuje tabuľka z databázy

table - identifikuje tabuľku z databázy

Tag <property>

name - identifikuje premennú v triede

class - identifikuje atribút tabuľky, ktorý sa namapuje do premennej

type - identifikuje akého typu je daný atribút tabuľky

Tag <id> - identifikuje primárny kľúč tabuľky, má obdobnú štruktúru ako <property>

Tag <set> - premennú typu množina v triede v aplikácii umožní prepojiť s hodnotami v inej tabuľke na základe rovnosti kľúčov (primárneho v jednej a cudzieho v druhej tabuľke). Vzťah daných dvoch tabuliek je 1:N.

name - identifikuje premennú triedy

table - identifikuje názov tabuľky, na ktorú je daná tabuľka (ktorá je namapovaná na danú triedu) prepojená

<*key*>-určuje, ktorý atribút z tabuľky, na ktorú sa prepája, musí byť zhodný s primárnym kľúčom danej tabuľky

<*many-to-may*> - umožňuje aby bola premenná priamo prepojená (cez spojovaciu

tabuľku definovanú v tag-u <set>) na ďalšiu tabuľku, ktorá má z

pôvodnou vzťah M:N.

5.2.3 Práca s databázou cez Hibernate

Po namapovaní tabuliek je možné s nimi pracovať ako s objektmi.
Príklad:

```
List Student = session.createQuery("from Pouzivatel p where p.typPouzivatela=1").list();
        //toto query vyberie z databázy všetkých používateľov, ktorý sú typu 1 - študenti
        //premenná session udržuje prepojenie s databázou
Iterator iter = Student.iterator();
        //cez iterátor budeme môcť pristupovať k jednotlivým objektom
for (Iterator iter = Student.iterator(); iter.hasNext();)
{       Pouzivatel element = (Pouzivatel) iter.next();
                //premenná element je naplnená informáciami z databázy
                //ďalej s ňou pracujeme ako so štandardným objektom
}
```

5.2.4 Komunikácia JSP a Hibernate

Komunikácia medzi JSP a Hibernate je riešená ako preposielanie textových reťazcov medzi jednotlivými vrstvami. Tieto reťazce v sebe obsahuju XML štruktúru, ktorej koreňový element obsahuje názov funkcie požadovanej používateľom a zoznamom

prípadných parametrov alebo údajov potrebných pre vyhľadávanie informácií v databáze. Príklad správy pre Hibernate môže vyzerať následujúco:

```
// funkcia na vypisanie zoznamu predmetov pre konkretného študenta
    StringBuffer line = new StringBuffer("");
    //textový reťazec, ktorý sa odošle Hibernatu
    line.append("<?xml version='1.0' encoding='UTF-8'?>\n");
    line.append("<subjList>\n");
        //koreňový element, ktorý hovorí Hibernatu, akú akciu požaduje používateľ
    line.append("<name>");
    line.append(request.getParameter("studenti"));
        //obsahuje meno študenta z databázy
    line.append("</name>\n");
    line.append("</subjList>\n");
```

Po prijatí informácii z Hibernatu je potrebné tieto informácie spracovať:

```
//metóda na spracovanie XML prijatého z Hibernatu
private boolean readMessage(BufferedReader br) throws IOException {
     String line;
     line=br.readLine();
        // volanie jednotlivých metód (podľa toho aká odpoveď príde zo servera – XML
        vygenerované Hibernatom)
     while (line != "") {
        if (line.indexOf("<login_answ>")>-1){
           return vypisLoginInfo(br);
           //volanie metódy pre spracovanie XML s údajmi o prihlasovaní používateľov
        }
        if (line.indexOf("<studList_answ>")>-1){
           return vypisZozStud(br);
           //volanie metódy pre spracovanie XML s údajmi pre výpis všetkých študentov v
databáze
        }
        if (line.indexOf("<subjList_answ>")>-1){
           return vypisZozPredmetov(br);
           //volanie metódy pre spracovanie XML s údajmi o všetkých predmetoch, ktoré študent
```

```
            študuje
        }
        line=br.readLine();
    }
    return false;
}
```

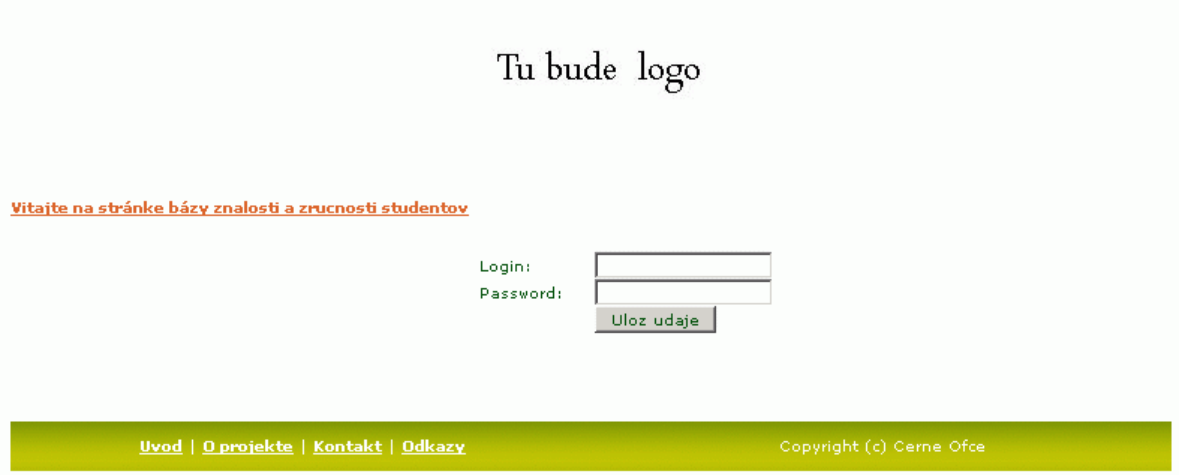
Vytváraný systém by v prípade úspešnej implementácie mal byť nasadený na jednom z fakultných serverov, a práve z tohto dôvodu sme sa rozhodli kontaktovať administrátora Ing. Steinmüllera, ktorý nám prisľúbil pomoc a možnosť nasadenia systému v prípade, že bude spĺňať určité požiadavky.

5.2.5 Použité knižnice

Na spustenie a správny beh technológie Hibernate je potrebných okrem samotnej hibernate knižnice ešte viacero ďalších knižníc. Medzi nimi je jednou z najdôležitejších knižnica, ktorá obsahuje ovládač databázy. Ďalšími knižnicami sú napr dom4j, log4j, jdbc, jta a ďalšie.

5.3 Používateľská príručka

Okno - Prihlásenie

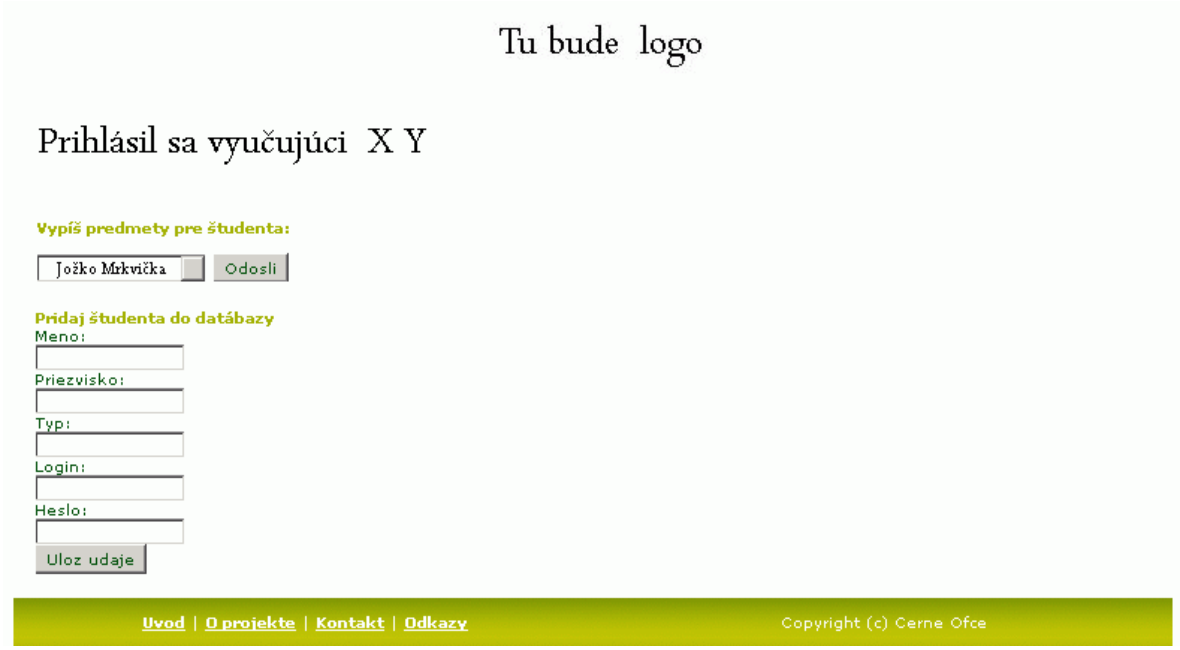


Okno pre prihlásenie užívateľov do systému bázy znalosti. Po zadaní prihlasovacích údajov do systému prebehne autentifikácia používateľa a systém zistí, či sa do systému prihlásil študent alebo vyučujúci.

Okno – Študent

Po prihlásení sa študenta do prototypu systému Bázy znalostí a zručností sa na stránke zobrazia údaje o prihlásenom študentovi.

Okno - Vyučujúci



Po prihlásení sa vyučujúceho do prototypu systému Bázy znalostí a zručností sa scroll box naplní menami všetkých študentov nachádzajúcich sa v databáze. Po zvolení konkrétneho mena vyučujúcim sa zobrazia všetky predmety, ktoré vybraný študent študuje. Vyučujúci má taktiež možnosť pridávať nových študentov do databázy pod podmienkou zadania všetkých potrebných údajov.

Okno – Výpis predmetov

Po zvolení konkrétneho študenta a odoslaní žiadosti o výpis predmetov tohto študenta sa zobrazí okno, v ktorom sa vypíšu všetky predmety, ktoré vybraný študent študuje.

5.4 Zhodnotenie

Cieľom tohto prototypu nebolo vytvoriť plne využiteľný systém. Vytvorený prototyp je založený len na komunikácií klienta so serverom s využitím objektovo-relačného mapovania pomocou knižnice Hibernate. Táto časť predstavuje jadro celého systému a keďže sme predpokladali, že konfigurácia tejto komunikácie a samotného mapovania bude to najkomplikovanejšie vo vytváranom systéme, rozhodli sme sa implementovať do prototypu práve túto časť.

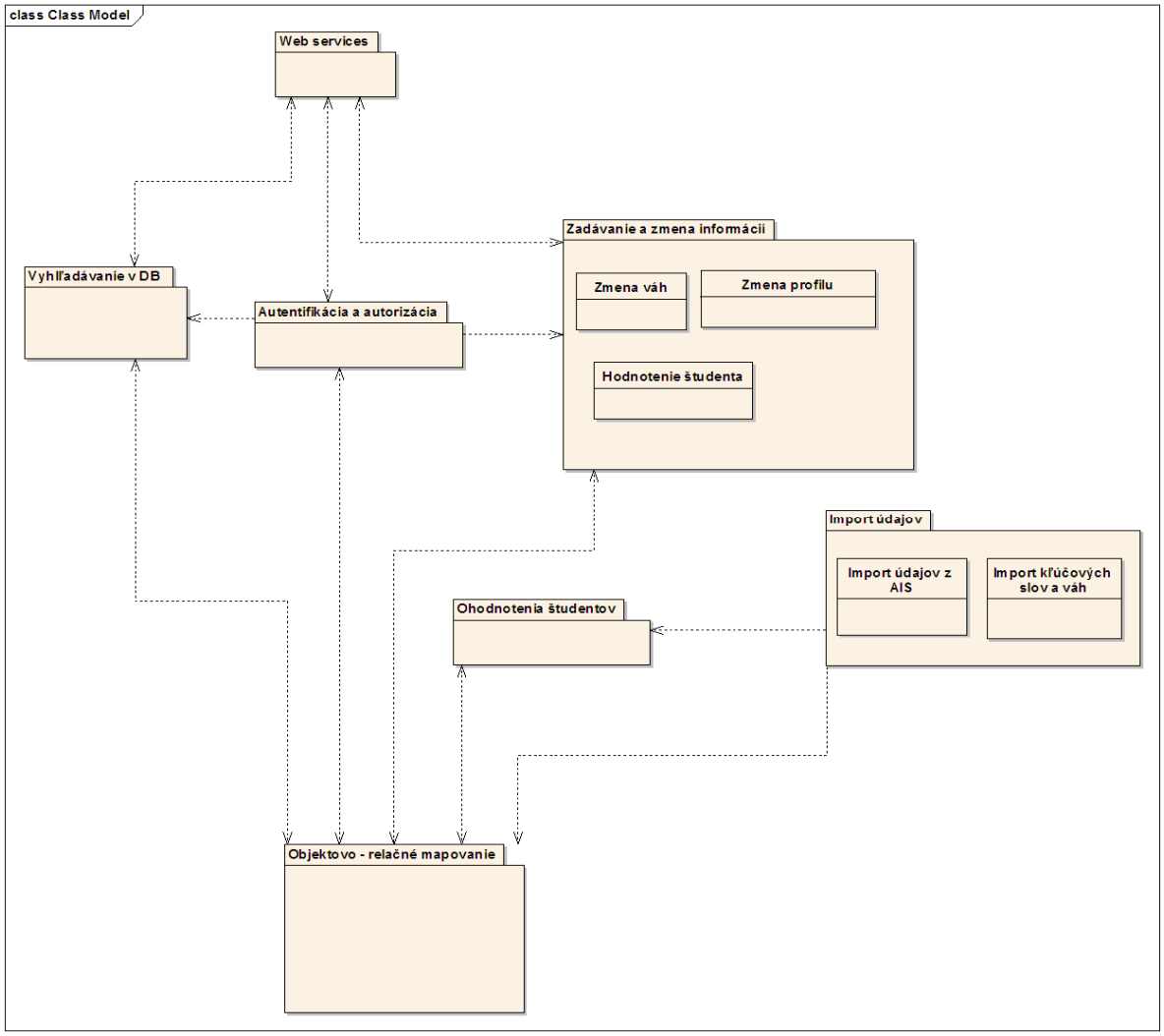
Celkovo prototyp stanovený cieľ splnil, podarilo sa nám úspešne implementovať komunikáciu medzi klientom a serverom.

6. Zmeny oproti pôvodnej špecifikácii a návrhu systému

6.1. Architektúra systému

Táto kapitola popisuje architektúru nášho systému.

6.1.1. Diagram architektúry systému



Obr. 6 Architektúra systému

Používateľ pristupuje do systému cez používateľské prostredie, ktoré je generované webovými službami. Bude vyzvaný, aby zadal svoje prístupové meno a heslo. Tieto údaje sa cez webové služby pošlú do modulu autentifikácie a autorizácie, kde sa zistí korektnosť zadaných údajov a aké práva daný používateľ má. Následne bude môcť používateľ vykonávať v systéme akcie (podľa prístupových práv).

6.1.2. Popis modulov architektúry

Webové služby

Webové služby majú na starosti generovanie používateľského prostredia, proces prihlasovania sa do systému.

Webové služby následne poskytujú informácie (login a heslo) o prihlásení pre modul autentifikácie a autorizácie.

Import údajov o študentovi

Modul importu údajov sa skladá z podmodulov import údajov z AIS (nami navrhnutého) a import kľúčových slov a váh z XML súborov. Toto získavanie informácií sa vykonáva v pravidelných intervaloch (napríklad jedenkrát za semester).

Zadávanie a zmena informácií

Modul Zadávanie a zmena informácií sa skladá z podmodulov Zmena váh, Zmena profilu a Hodnotenie študenta. Zmena profilu umožňuje (zmenu osobných údajov, zmena hesla, ...). Zmena váh umožňuje zmeniť váh kľúčových slov. Hodnotenie študenta slúži na to, aby mohol pedagóg ohodnotiť študenta a študent sa môže ohodnotiť aj sám.

Ohodnotenia študentov

Modul spracúva výsledky predmetov študenta z databázy AIS a ohodnotí ich podľa váh kľúčových slov.

Vyhľadávanie v DB

Databáza poskytuje vyhľadávanie študentov na základe používateľských požiadaviek. Vyhľadávanie je podmienené úspešnou autorizáciou používateľa. Zjednodušene povedané, tento modul vytvára zoznam študentov zoradených podľa zadefinovaných požiadaviek.

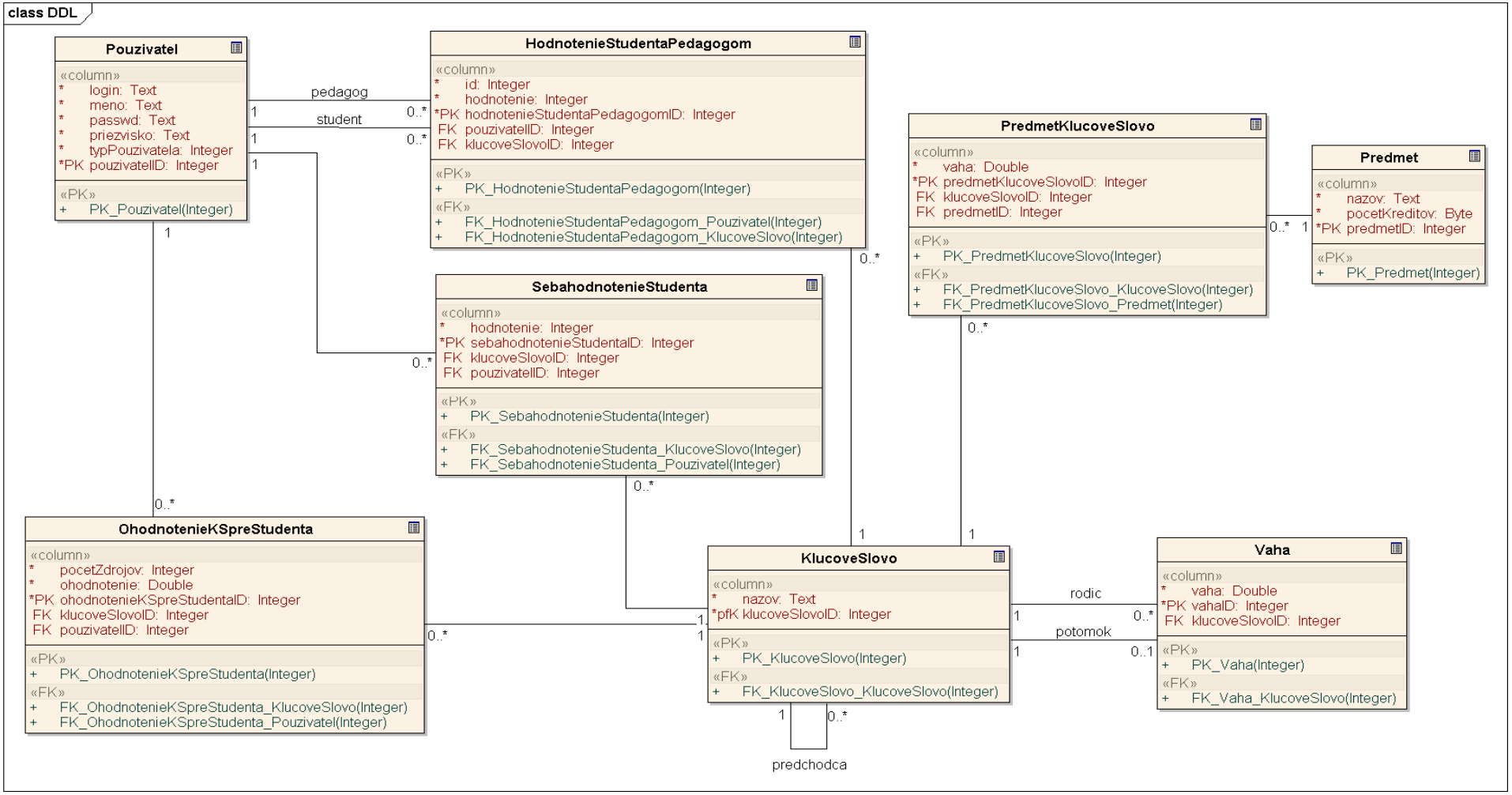
Autentifikácia a autorizácia

Tento modul vyhodnocuje informácie, ktoré mu boli poslané z webových služieb. Na základe týchto informácii zistí totožnosť používateľa a podľa totožnosti určí, aké má práva v systéme.

Objektovo – relačné mapovanie

Predstavuje rozhranie medzi ostatnými modulmi a samotnou databázou. Smerom do databázy sú požiadavky transformované do príkazov zrozumiteľných pre konkrétne databázové prostredie (jazyk SQL) a smerom von z databázy sú naopak tieto informácie transformované do objektov (jazyk Java). Tento modul sa v systéme nachádza z dôvodu zvýšenia efektívnosti a jednoduchosti, ktorú prináša objektovo - orientovaná paradigma a práca s objektmi. Viac o objektovo-relačnom mapovaní je popísané v časti Analýza technológií (Hibernate).

6.2. Fyzický model



Obr. 7 Fyzický model

6.2.1. Popis entít

Táto časť popisuje jednotlivé entity.

Kľúčové slovo
Celá databáza je postavená na entite KľúčovéSlovo. Táto entita uchováva informácie o hierarchii kľúčových slov (jedno z kľúčových slov je "root", ktoré tvorí koreň stromu kľúčových slov). Väzba predchodca určuje kľúčové slovo, ktoré je "rodičom" daného kľúčového slova - je v strome o jednu úroveň vyššie. Jedno kľúčové slovo má práve jedného predchodcu (kardinalita 1) a jedno kľúčové slovo môže byť predchodcom ("rodičom") žiadneho (nemá nasledovníkov) alebo viacerých kľúčových slov (kardinalita 0..n).

Váha
Ak je kľúčové slovo predchodcom (má nasledovníkov) iného kľúčového slova, tak existuje väzba medzi entitami KľúčovéSlovo a Váha. V entite Váha je uchovaná informácia o tom, s akou váhou prispieva ohodnotenie daného kľúčového slova na ohodnotenie jeho predchodcu.

Väzba rodič: k ohodnoteniu jedného kľúčového slova môžu prispievať ohodnotenia viacerých kľúčových slov, ale aj žiadne - prípad kľúčového slova, ktoré nemá potomkov (kardinalita 0..n) a v entite Váha je definovaný len jeden rodič (kardinalita 1).
Väzba potomok: jeden potomok môže prispievať k ohodnoteniu maximálne jedného kľúčového slova, lebo má iba jedného rodiča a koreň stromu neprispieva k ohodnoteniu žiadneho kľúčového slova (kardinalita 0..1). Entita Váha definuje iba jedného potomka (kardinalita 1).

PredmetKúčovéSlovo
Táto entita je spojovacou tabuľkou medzi entitami Predmet a KľúčovéSlovo a vyjadruje váhu, ktorou prispieva hodnotenie predmetu, ktoré študent na fakulte dosiahol, k ohodnoteniu kľúčového slova. K ohodnoteniu kľúčového slova môžu prispievať viaceré predmety a principiálne aj žiadne (kardinalita 0..n) a jeden záznam v entite PedmetKľúčovéSlovo prislúcha k jednému kľúčovému slovu (kardinalita 1).

Predmet
V tejto entite sa uchovávajú informácie o predmetoch, ktoré sa na fakulte vyučujú. Tieto informácie sú importované z Akademického Informačného Systému. Vzťah s entitou PredmetKľúčovéSlovo: Predmet prispieva k ohodnoteniu viacerých kľúčových slov, ale principiálne nemusí k žiadnemu (kardinalita 0..n) a záznam v entite PredmetKľúčovéSlovo prislúcha k jednému predmetu (kardinalita 1).

Používateľ
Entita používateľ v sebe obsahuje informácie o jednotlivých používateľoch systému. Každý používateľ má svoje prihlasovacie meno a heslo, ktorým sa autentifikuje pri prihlasovaní do systému. Jednotliví používatelia sú rozlíšený atribútom typPoužívateľa na skupiny: študent, pedagóg, administrátor a hosť. Informácie o študentoch sú importované z Akademického Informačného Systému, ostatné skupiny používateľov sú do systému pridané ručne.

SebahodnotenieStudenta
Ak si v systéme používateľ typu študent vyplní sebahodnotenie, tak tieto informácie sa uchovajú v tejto entite. Táto entita má spojenie s entitou Používateľ - jeden používateľ (študent) si môže vyplniť viacero sebahodnotení (kardinalita 0..n) a jedno sebahodnotenie patrí k jednému používateľovi (študentovi). Druhým prepojením je prepojenie na entitu KľúčovéSlovo, kde sebahodnotenie prislúcha k jednému kľúčovému slovu (kardinalita 1) a k jednému kľúčovému slovu môžu patriť viaceré sebahodnotenia (kardinalita 0..n)

HodnotenieŠtudentaPedagógom
Podobne ako sebahodnotenie, ale v tomto prípade dané hodnotenie zadáva používateľ typu pedagóg, preto je medzi entitami Používateľ a HodnotenieStudentaPedagogom ešte ďalšie spojenie, ktoré určuje pedagóga, ktorý hodnotenie zadal. Kardinalita je 1:0..n, lebo jedno hodnotenie patrí k jednému pedagógovi a pedagóg môže zadať viaceré hodnotenia.

OhodnotenieKSpreŠtudenta
V tejto entite sa uchovávajú výsledné ohodnotenia kľúčových slov pre jednotlivých používateľov-študentov. Ohodnotenie sa skladá z dvoch atribútov a to atribútu ohodnotenie, ktoré vyjadruje vážený súčet jednotlivých hodnotení, ktoré prispeli k ohodnoteniu daného kľúčového slova a druhým atribútom je početZdrojov, ktorý uvádza z koľkých zdrojov sa dané ohodnotenie vypočítalo (počet predmetov, ktoré prispeli k ohodnoteniu + sebahodnotenie + hodnotenie pedagógom). Prepojenie na ostatné entity je rovnaké ako pri entite Sebahodnotenie.

6.3. Návrh algoritmov spracovania

6.3.1. Algoritmus výpočtu ohodnotenia kľúčových slov

Uvedieme si postup pre ohodnotenie jedného kľúčového slova. Čiže vyberieme z databázy jedno kľúčové slovo (tabuľka KlucoveSlovo). Ďalej si z databázy zistíme všetky predmety, ktorých hodnotenie prispieva k ohodnoteniu tohto kľúčového slova (tabuľka PredmetKlucoveSlovo). Dané kľúčové slovo ideme ohodnotiť pre všetkých študentov, preto z databázy načítame všetkých študentov (tabuľka Pouzivatel, typPouzivatela=1). Po jednom prechádzame tento zoznam študentov a pre každého študenta hľadáme v externej databáze známky z predmetov, ktoré prispievajú k ohodnoteniu daného kľúčového slova. Vypočítame vážený súčet týchto hodnotení predmetov, výsledok zapíšeme do databázy (tabuľka OhodnotenieKSpreStudenta) a ideme vypočítať ohodnotenie kľúčového slova pre ďalšieho študenta. Takto spočítame ohodnotenia všetkých kľúčových slov, ku ktorým prispievajú hodnotenia predmetov (to sú všetky listy stromu kľúčových slov). Následne vypočítame hodnotenia všetkých vnútorných uzlov stromu ako vážený súčet ohodnotení kľúčových slov, ktoré sú priamymi potomkami daného kľúčového slova. Týmto sme skončili prvú etapu ohodnocovania stromu kľúčových slov. V druhej etape sa k existujúcemu (práve vypočítanému) ohodnoteniu pripočítajú sebahodnotenia študentov (tabuľka SebahodnotenieStudenta) a hodnotenia študentov pedagógmi (tabuľka pedagogHodnotenieStudenta). Toto pripočítavanie sa vykoná nasledovne: pre sebahodnotenia študentov sa zoberú jednotlivé záznamy z danej tabuľky a hodnotenie daného študenta pre dané kľúčové slovo sa prenásobené definovanou váhou (0.1) pripočíta k prislúchajúcemu záznamu tabuľke OhodnotenieKSpreStudenta. Ak taký záznam neexistuje, tak sa vytvorí nový (s hodnotou nula) ku ktorému sa daná hodnota pripočíta. Pri hodnoteniach študentov pedagógmi je situácie trochu zložitejšia, z dôvodu, že jedného študenta pre jedno kľúčové slovo mohlo hodnotiť viacero pedagógov. V tom prípade sa najprv vypočíta priemerná známka z hodnotení všetkých pedagógov, ktorí daného študenta hodnotili v rámci daného kľúčového slova a tento priemer sa použije ako základ pre pripočítavanie. Tento základ sa ešte prenásobí definovanou váhou (0.5) a potom sa pripočíta k ohodnoteniu kľúčového slova pre daného študenta rovnako ako pri sebahodnotení študenta.

6.3.2. Algoritmus vyhľadávania podľa kľúčových slov

Od používateľa dostane funkcia vyhľadávania zoznam preferencií kľúčových slov - sú to dvojice kľúčové slovo-priorita. Funkcia vezme tieto preferencie a prepočíta ich tak, aby ich súčet dával hodnotu 1, ale aby zostali zachované pomery medzi jednotlivými prioritami (každú preferenciu vydelí sumou všetkých preferencií). Následne sa vo funkcii vyhľadajú všetci študenti, ktorí majú ohodnotené aspoň jedno kľúčové slovo zo vstupného zoznamu. Pre každého študenta sa spočíta vážený súčet ohodnotení kľúčových slov (podľa modifikovaných priorít) a zoznam študentov sa zoradí podľa tohto váženého súčtu.

7. Realizácia návrhu

Z analyzovaných technológií v predchádzajúcom semestri sme nakoniec použili nasledovné implementačné nástroje a prostriedky.

7.1. Hibernate

Dátová perzistencia a objektovo-relačné mapovanie sú moderné technológie, ktoré významným spôsobom uľahčujú prácu s dátami (ich uskladnením a spracovaním). Aj keď ich počiatočné zvládnutie vyžaduje zvýšenú dávku úsilia, po určitom čase sa dostavia výhody v podobe omnoho jednoduchšej manipulácie. Hibernate umožňuje pracovať s databázou na vyššej úrovni ako samotné JDBC (navyše pri JDBC používateľ musí poznať jazyk SQL). Aj keď pri práci s Hibernate musí používateľ poznať jazyk HQL (Hibernate Query Language), tým že ide o vyššiu úroveň jazyka SQL, je omnoho jednoduchší a jeho používanie je intuitívnejšie.

príklad:

Tento príklad uvádza dopyt vykonávajúci tú istú funkcionalitu (pripomínam, že programátor pracuje s jazykom HQL, ktorý je potom automaticky prekladaný do jazyka SQL - databáza pracuje totižto len s SQL).

dopyt v jazyku HQL: `from Pouzivatel p where p.typPouzivatela=1`
dopyt v jazyku SQL: `select * form Pouzivatel where typPouzivatela=1`

Samozrejme pri práci s jazykmi a technológiami vyššej úrovne prichádza na rad problém optimalizácie. Hibernate totižto prekladá požiadavky napísané používateľom v jazyku HQL na požiadavky jazyka SQL. Otázka znie, či aplikačný programátor nie je schopný napísať efektívnejší SQL kód, ako ten ktorý vygeneruje Hibernate. Skúsení SQL programátor dokáže v určitých prípadoch napísať efektívnejší kód ako ten ktorý by sa automaticky vygeneroval, no ide o špeciálne prípady (zložité dopyty, zápis obrovského množstva dát, ...) a v našom systéme pracujeme len s jednoduchými operáciami, pri ktorých by práca s SQL nepriniesla žiadne výhody, čo sa optimalizácie týka.

Práve tento fakt viedol k výberu technológie ORM, konkrétna voľba nástroja Hibernate potom už vychádzala z faktu, že v súčasnosti ide o najrozšírenejší nástroj v tejto oblasti.

7.1.1. Mapovacie XML

Mapovací XML súbor slúži k vytvoreniu prepojenia triedy v jazyku Java a príslušnej tabuľky v databázovom systéme (v našom prípade PostgreSQL). Nižšie je uvedený príklad toho, ako takýto súbor vyzerá (spolu aj s komentármi k jednotlivým elementom). Podobný súbor je potrebné vytvoriť ku každej triede, ktorá má byť perzistentná (teda každá trieda, ktorá sa má ukladať do databázy).

príklad:

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-mapping>
   <class name="entities.Predmet" table="PREDMET">                  // trieda Predmet je naviazaná na tabuľku
                                                                    // PREDMET
      <id name="id" column="PREDMET_ID">                            // primárny kľúč je PREDMET_ID
         <generator class="increment"/>                             // a je generovaný automaticky
      </id>

      <property name="nazov" column="PREDMET_NAZOV"/>               // atribút nazov je naviazany na stlpec
                                                                    // PREDMET_NAZOV
      <property name="pocetKreditov" column="PREDMET_POCETKREDITOV"/>
                                                                    // atribút pocetKreditov je naviazaný na
                                                                    // stĺpec
                                                                    // PREDMET_POCETKREDITOV

      <set name="cieleOhodnotenia" table="predmetklucoveslovo">     // vytvorenie cudzieho kľúča
         <key column="idPredmet"/>                                  // FK
         <one-to-many class="entities.PredmetKlucoveslovo"/>        // vzťah 1:N
      </set>

   </class>
</hibernate-mapping>
```

7.1.2. Connection Pool Management

Pri každej operácií, ktorá je vykonávaná nad databázou si musí používateľ najskôr vytvoriť spojenie (connection). Problém spočíva v tom, že vytvorenie nového spojenia je pomerne drahá záležitosť, čo sa výpočtových prostriedkov týka. Aplikácia by z toho dôvodu mala využívať tzv. connection pool ("bazén" spojení). Ide o inteligentný manažment všetkých otvorených pripojení k databáze, kde používateľ pri prístupe do databázy dostane pridelené spojenie z connecion pool a po skončení ho vráti nazad. Hlavná výhoda connection pool management spočíva v tom, že optimalizuje prácu s otvorenými spojeniami (po určitom čase nepoužívané spojenia automaticky zatvára).
V našom systéme sme využívali knižnicu C3P0, ktorá predstavuje API pre prácu s manažmentom pripojení. Rôzne nastavenie manažmentu sa ukladajú do externého XML súboru. Príklad takýchto základných nastavení je uvedený nižšie:

```
<property name="hibernate.c3p0.min_size">5</property>          // minimálny počet otvorených pripojení
<property name="hibernate.c3p0.max_size">20</property>         // maximálny počet otvorených pripojení
<property name="hibernate.c3p0.timeout">300</property>         // počet sekúnd, koľko je pripojení držané v
                                                               // "bazéne"
```

7.1.3. Hibernate nástroje

Nižšie sú vysvetlené jednotlivé nástroje, s ktorými sme sa pri vývoji stretli.

hbm2ddl (Hibernate to DDL - data definition language)
Tento nástroj automaticky generuje z mapovacieho XML súboru SQL kód (DDL).

XML -> (hbm2ddl) -> DDL

hbm2hbmxml (Hibernate to XML)
Tento nástroj automaticky generuje z databázovej schémy (DDL) mapovacie XML.

DDL -> (hbm2hbmxml) -> XML

hmb2java (Hibernate to Java)
Tento nástroj automaticky generuje z mapovacieho XML súboru zdrojový kód v Jave.

XML -> (hmb2hbmxml) -> Java

7.1.4. Vývojové prístupy

Top-down prístup

Tento prístup je spomedzi ostatných najpriamočiarejší a po skúsenostiach s prototypom, kde sme využívali prístup "**Meet in the middle**" (popísaný nižšie) sme sa ho rozhodli aplikovať do finálnej verzie nášho systému.

postup:

- programátor napíše zdrojový kód v jazyku Java
- programátor napíše mapovacie XML, v ktorom namapuje jednotlivé triedy na tabuľky v databáze
- nástroj hbm2ddl automaticky vygeneruje databázovú schému (ddl)

Java (perzistentná trieda) + mapovacie XML -> hbm2ddl -> DB schéma (ddl)

Bottom-up prístup

Ide o prístup využívajúci reverzné inžinierstvo (opačný prístup ako pri Top-down vývoji).

postup:

- programátor vytvorí databázovú schému (ddl)
- nástroj hbm2hbmxml automaticky vygeneruje mapovacie XML
- nástroj hbm2java automaticky vygeneruje zdrojový kód v jazyku Java

DB schéma (ddl) -> hbm2hbmxml -> mapovacie XML -> hbm2java -> Java

Middle-out prístup

Použitie tohto prístupu sa odporúča len skúseným expertom. Programátorovi stačí napísať mapovacie XML a Hibernate nástroje automaticky vygenerujú zdrojové kódy programu a databázovú schému. V praxi, ale býva mapovacie XML dosť stručné a vygenerované kódy zväčša potrebujú refaktoring. O použití tohto prístup sme vzhľadom na naše skúsenosti s technológiou Hibernate neuvažovali.

postup:

- programátor napíše mapovacie XML, v ktorom namapuje jednotlivé triedy na tabuľky v database
- nástroj hbm2java automaticky vygeneruje zdrojový kód v jazyku Java
- nástroj hbm2ddl automaticky vygeneruje databázovú schému (ddl)

mapovacie XML -> hbm2ddl -> DB schéma (ddl)
mapovacie XML -> hbm2java -> Java (perzistentná trieda)

Meet in the middle prístup

V podstate ide o najkomplikovanejší možný prístup spomedzi ostatných spomenutých. Ako neskúsení používatelia Hibernate sme si tento prístup zvolili v prototype, no poučení z chyby sme sa rozhodli vo finálnej verzii aplikovať prístup **Top-down** (popísaný vyššie). Výhoda tohto prístupu spočíva podľa nášho názoru v jeho priamočiarosti, pretože používateľ si sám naprogramuje zdrojový kód a sám si vytvorí databázovú schému (ddl). Nevýhoda na druhej strane spočíva v tom, že si musí manuálne vytvoriť aj mapovacie XML, ktoré treba doslovne "napasovať" medzi program a databázu, ide teda o problémy pri konfigurácií.

postup:

- programátor vytvorí databázovú schému (ddl)
- programátor napíše zdrojový kód v jazyku Java
- programátor napíše mapovacie XML, v ktorom namapuje jednotlivé triedy na tabuľky v databáze

Z postupu je zrejmé, že je minimum toho, čo môžu Hibernate nástroje v tomto smere pre programátora urobiť.

7.2. PostgreSQL

Pre databázový systém PostgreSQL sme sa rozhodli hlavne na základe podnetu Ing. Filkorna. V našom projekte nie je problém použiť aj iný databázový systém, kedže využívame technológiu objektovo-relačného mapovania, pomocou nástroja Hibernate. Zmena databázového systému je otázkou zmeny niekoľkých konfiguračných nastavení.

7.3. Apache

Apache v súčasnosti predstavuje najpoužívanejší server pre JSP technológiu.

Projekt Apache Tomcat je robustný, rozšíriteľný, komerčne využiteľný a voľne dostupný zdrojový kód implementovaný pomocou Java Sevlet a JavaServer Pages technológie. Tento projekt je spravovaný skupinou dobrovoľníkov z celého sveta, ktorý navzájom komunikujú pomocou internetu. Projekt Apache HTTP Server je pod záštitou Apache Software Foundation.

Hlavnou úlohou web servera je obsluhovať požiadavky, ktoré prídu prostredníctvom HTTP protokolu. Zvyčajne server dostane požiadavku na nejaký zdroj a ako odpoveď dostane konkrétny zdroj. Za zdroje sa väčšinou považujú súbory a obsah adresárov. Klient môže požadovať aj program a v takom prípade je úlohou servera tento program spustiť a ako odpoveď vrátiť výstup daného programu. Inými slovami server dostane požiadavku, spracuje, získa zdroje a odošle naspäť klientovi. Server sa takisto stará aj o kontrolu prístupu a autorizáciu.

Funkcionalita Apache servera sa môže jednoducho zmeniť vytvorením nových modulov, ktoré doplnia alebo úplne nahradia pôvodne moduly. Server je taktiež vysoko konfigurovateľný a moduly môžu definovať svoje vlastné konfiguračné príkazy, ktoré sú uvedené v konfiguračných XML súboroch.

Ďalšou alternatívou pre webový server bolo využitie SunOne servera, ktorý je však založený na jadre Apache Tomcat 4.0.4 a teda poskytuje takmer rovnaké služby ako Apache Tomcat.

7.4. Java Server Pages (JSP)

Na tvorbu dynamických (stránok prispôsobujúcich sa používateľovi, stránok interaktívnych, reagujúcich na impulzy používateľa), slúži viacero programovacích jazykov. My sme sa rozhodli pre jazyk JSP.

Metóda programovania JSP spočíva v kombinácii jazyka HTML na tvorenie statického obsahu stránky a programovacieho jazyka Java, pomocou ktorého môže programátor vytvárať dynamické súčasti stránky.

Java Server Pages je serverová technológia. To znamená, že všetky požiadavky používateľa, či príkazy programátora sa vykonávajú a spracovávajú na strane servera (serverovom počítači, ktorý bol počas riešenia tímového projektu umiestnený v softvérovom štúdiu FIIT). Výhodou je, že takto vytvorené dokumenty môžu zdieľať viacerí používatelia prostredníctvom nástroja pre správu verzií.

V porovnaní so statickými stránkami nám JSP poskytuje širšie možnosti, ako sú napríklad:

- vytváranie stránok prispôsobených špeciálnym požiadavkám jednotlivých používateľov
- získavanie údajov z formulárov a ich ďalšie spracovanie, a teda interakcia s používateľom prostredníctvom formulárov
- spolupráca s databázou
- kontrola toku spracovania dát

Technológia JSP nám umožňuje plne využívať programovací jazyk Java a to hlavne pomocou objektov JavaBeans, čo sú v podstate vopred pripravené javovské triedy.
Technológia JSP je len jedným z prostriedkov na tvorbu dynamických stránok. Porovnajme výhody aj nevýhody jednotlivých prostriedkov.

Výhodou JSP proti ASP (Active Server Pages) je, že dynamická časť kódu je písaná v jazyku Java a nie vo VBScripte alebo inom jazyku určenom pre ASP. Java je komplexnejší jazyk a je prenositeľná, teda neexistuje obmedzenie vzhľadom na použitý operačný systém či server. Výhodou ASP je vysoká podpora komerčných programov a jednoduchá tvorba stránok.

Výhodou JSP proti PHP je, že môžeme niektoré súčasti kódu (triedy) používať aj v iných programoch napísaných v Jave. Pokiaľ si niekto vyberá, ktorú z týchto dvoch technológií použiť, tak v prípade, že má s Javou skúsenosti, pri prechode na JSP by nemal mať žiadne problémy. Jazyk PHP je potrebné sa učiť od základov, aj keď niektoré základné programové štruktúry sú veľmi podobné jazyku Java. Jednou z užitočných vlastností je, že jazyk PHP nevyžaduje deklarovanie typov premenných, a teda nevyžaduje pretypovávanie a konvertovanie.

Výhodou JSP oproti Servletom je, že oddeľuje statický a dynamický obsah stránky. Kód JSP je tým pádom aj oveľa kratší a prehľadnejší. Pri spúšťaní JSP stránky sa jej obsah najprv preloží do servletu a až potom sa vykoná. JSP je vhodné použiť v prípade, že sa nemení celý obsah stránky. Je potrebné povedať, že to, čo sa dá vytvoriť v JSP, dá sa vytvoriť aj v Servletoch.

7.5. NetBeans

NetBeans je úspešný Open Source projekt s veľmi rozsiahlou používateľskou základňou, rastúcou komunitou vývojárov a partnermi po celom svete. Firma Sun Microsystems založila Open Source projekt NetBeans v júni 2000 a je zároveň aj hlavným sponzorom celého projektu.

Dnes existujú dva produkty: vývojové prostredie NetBeans (NetBeans IDE) a vývojová platforma NetBeans (The NetBeans Platform). Pričom pre potreby tímového projektu sme využili iba vývojové prostredie NetBeans. Oba produkty sú vyvíjané pod licenciou Open Source a je ich možné bezplatne používať v komerčnom a nekomerčnom prostredí. Zdrojový kód je dostupný pod licenciou Common Development and Distribution License (CDDL).

Vývojové prostredie NetBeans IDE je nástroj, pomocou ktorého programátori môžu vytvárať, prekladať a ladiť aplikácie. Okrem toho existuje veľké množstvo modulov, ktoré toto vývojové prostredie rozširuje. Vývojové prostredie NetBeans je bezplatne šírený produkt, ktorý je možné používať bez akýchkoľvek obmedzení.

Najväčším konkurentom NetBeans na poli IDE pre Javu je Eclipse, pôvodne vyvíjaný spoločnosťou IBM. Eclipse, tak ako aj NetBeans, je napísané v Jave a teda je taktiež multiplatformové. K väčšej rozšírenosti Eclipse však prispelo množstvo rozšírení od tretích strán, tento deficit sa snaží dohnať NetBeans pridaním nových modulov. Do verzie 6.0 zakomponovali vývojári zlepšenú podporu skriptovacích jazykov, rozšírenie je napríklad k dispozícii pre PHP. Priamo v základnej inštalácii je pridaná podpora WYSIWYG editora, ktorý zjednodušuje tvorbu webových stránok. Na testovanie web aplikácií sú k IDE pribalené aplikačné servery GlassFish a Tomcat.

8. Testovanie

Testovacie údaje v systéme predstavujú 23 náhodne vybraných študentov s vymyslený známkami a študijnými programami, pričom všetky predmety uložené v databáze systéme pochádzajú z reálnych študijných programov bakalárskeho a inžinierskeho štúdia (označenie predmetov zodpovedá označeniu predmetov v AIS, odkiaľ boli použité informácie aj ohľadom počtu kreditov, príslušného semestra a spôsobu ukončenia predmetu). Použité údaje o predmetoch sú aktuálne k mesiacu apríli 2008.

Systém bol testovaný už počas fázy implementácie, kedy bola odhalená väčšina chýb systému. Pri postupnom odstraňovaní týchto chýb vznikali postupne jednotlivé verzie systému. V tejto fáze vývoja testoval každý člen tímu časť zdrojového kódu, ktorý vytvoril (Odhalené významné odhalené chyby boli napríklad: chyba pri zmene hesla v profile používateľa, chyba pri zabránení prístupu k obsahu stránky pri stlačení tlačidla späť v prehliadači, a pod.).
Cieľom bolo overiť základnú funkčnosť, ale aj reakcie na výnimočné situácie. Na tejto úrovni sme nepoužívali žiaden konkrétny nástroj na testovanie.

Výsledný produkt bol testovaný aj vedúcimi nášho tímového projektu - Ing.Považanovou a Ing. Šimákovou. V zásade ako používateľkám systému (vo funkcii pedagóga) sa im funcionalita systému páčila, mali výhradu skôr k výzoru používateľského prostredia.

Počas testovania boli úspešne opravené všetky odhalené chyby, avšak pre úplne otestovanie je potrebné zaviesť systém do dlhšej testovacej prevádzky a taktiež vykonať záťažové testy systému.

Kritické chyby odhalené v procese testovania

1. Otestovanie autentifikácie a autorizácie používateľov
   - Odhalená chyba prístupu k obsahu stránky pri stlačení tlačidla späť v prehliadač.
   - Stav: Opravená
   - Zobrazenie stránky pri manuálnom zadaní URL adresy bez vykonania úspešnej autorizácie.
   - Stav: Opravená

2. Otestovanie navigácie v systéme:
   - Pri prihlásení pedagóga bola neošetrená situácia dvojitého výberu položky menu "Ohodnotenie študentov"
   - Stav: Opravená
   - Manuálne zadanie URL adresy prihláseným používateľom
   - Stav: Stránky ktoré vyžadujú parametre formulárov vrátia neošetrenú výnimku.

3. Neplatné vstupy pri vyhľávanívaní
   - chyba pri nezadaní kľúčových slov
   - Stav: Opravená

4. Problémy s nekonzistenciou kódovania stránok
   - Stav: Opravená a nastavené na kódovanie Windows-1250

9 Celkové zhodnotenie

Našou prioritou pri vývoji informačného systému bolo vytvoriť funkčný produkt. Niektoré z požiadaviek, ktoré sme si určili v zimnom semestri, sme nerealizovali vzhľadom na krátke časové obdobie. Naše nedostatočné skúsenosti pri vývoji rozsiahlejších systémov nás pri návrhu viedli k rôznym funkcionálnym možnostiam, ktorých náročnosť sme nedokázali správne odhadnúť. Vzhľadom na komplikovanosť nových technológií (objektovo-relačné mapovanie, dátová perzistencia) časť funkcionality zostala len pri návrhu. Zoznam funkcionality, ktorá nebola v porovnaní s návrhom implementovaná je uvedená nižšie:

- **sofistikovaná metóda ohodnocovania**: táto problematika je natoľko komplexná, že ju v rámci svojho diplomového projektu rieši náš kolega Tibor Schvartz. Náš systém implementuje len základnú metódu, ktorá je popísaná v predchádzajúcich kapitolách
- **zmeny váh**: plánovali sme implementovať aj XML editor, ktorý by používateľovi umožnil jednoduchým spôsobom distribuovať váhy medzi jednotlivými predmetmi a kľúčovými slovami a tiež medzi klúčovými slovami rôznych úrovní
- **možnosť pridávania nových váh**: implementovaná je len možnosť zmeny už existujúcich váh
- **možnosť spracovania rôznych dokumentov**: študenti počas svojho štúdia môžu preukázať vysokú mieru znalostí a prípadné certifikáty, zahraničné stáže, vedecké konferencie, prípadne ocenenia pridávajú na hodnote. V pôvodnom návrhu sme počítali s prednastavenými certifikátmi a dokumentami, ktoré by boli tiež váhované, nakoľko niektoré môžu byť prestížnejšie a významnejšie ako iné. Uvažovali sme aj o možnosti pridania akéhokoľvek vlastného dokumentu, no problémom by potom bolo jeho spracovanie a ohodnotenie
- **uskladnenie zdrojov ohodnotenia**: v návrhu sme uvažovali nad uchovávaním zdrojov, ktoré najviac prispeli k ohodnoteniu kľúčového slova. Pri vyhľadávaní by si používateľ zadal požadované znalosti a zručnosti. Systém by vyhľadal relevantných študentov a zapamätal si všetky zdroje, ktoré prispeli do výsledného hodnotenia študenta. Okrem známok to mohli byť poznámky od pedagógov, vyššie spomínané certifikáty, účasť na konferenciách, ocenenia
- **fulltextové vyhľadávanie študentov**: pre neskúseného používateľa je prehľadávanie v strome uľahčenie, ale napriek tomu je potrebné mať aké-také znalosti o

kategorizovaní kľúčových slov v strome. Skúsený používateľ by naopak mohol využiť možnosť fulltextového vyhľadávania. Do textového poľa by zadal kľúčové slová, prípadne zložitejšie výrazy. Problémom by v tomto prípade bolo ošetrovanie neexistujúcich kľúčových slov

- **história kľúčových slov:** uchovávali by sa všetky zmeny, ktoré študent v systéme urobil. Bolo by možné teda zistiť, akým spôsobom sa jeho znalosti a zručnosti vyvíjali

Medzi zlepšováky, ktoré mohli byť do výslednej aplikácie zahrnuté, patrí napríklad tabuľka s už existujúcimi hodnoteniami znalostí a zručností študenta, ktorá by sa zobrazila pri hodnotení znalostí a zručností študentom vedľa stromu. Odpadla by tak nutnosť jeho rozbaľovania, aby sa študent pozrel, ako má jednotlivé znalosti ohodnotené. Ďalšou dôležitou funkciou, by bola indexácia databáz. Toto by značne urýchlilo výpočtové časy pri výpočtoch a vyhľadávaní.

Práca na tomto projekte bola veľmi zaujímavá. Zadanie bolo pre nás motivujúce a snažili sme sa ho vypracovať čo najlepšie. Viedlo nás to k intenívnemu štúdiu technológií a postupov pre vývoj informačných systémov. Naučili sme sa pracovať s niektorými technológiami. Tieto skúsenosti sú pre náš veľmi cenné. Mali sme možnosť oboznámiť sa s technológiami JSP, XML, Hibernate, Apache. Pracovali sme s rôznymi databázovými prostrediami (Oracle, PostgreSQL, MySQL), pri manažovaní práce v tíme sa používal systém dotProject a na verziovanie systém CVS. Pri pohľade z druhej strany (netechnologickej) sme sa naučili spolupracovať a komunikovať nielen v rámci tímu, ale takisto aj v rámci fakulty (napríklad pri riešení problémov zo serverom, pri vyhľadávaní informácií o podobných systémoch a pri špecifikácií požiadaviek na náš systém zo strany vedenia fakulty). Svoj systém sme tiež sami testovali a odladovali ho.

Pochopili sme, že prvoradý cieľ predmetu Vývoj informačného systému v tíme nespočíval vo vytvorení dokonalého systému. Vďaka simulácií reálneho pracovného prostredia, bolo cieľom naučiť nás pracovať a komunikovať medzi sebou a využívať pritom nástroje pre manažment činnosti. Väčšina doterajších projektov na našej fakulte, ktoré sa riešili samostatne, boli významné z dôvodu organizácie práce a času. Nenaučili nás však to najkomplikovanejšie, a to pracovať s inými ľudmi. Aj keď v rámci štúdia sme sa stretli s projektmi, ktoré sa riešili tímovo, tieto predmety svojím rozsahom nedosahovali úroveň tímového projektu, kde sme mohli využiť prevažnú väčšinu vedomostí získaných počas štúdia o tvorbe informačného systému. Práve z tohto dôvodu považujeme prínos tohto predmetu za neoceniteľný a systém, ktorý sme počas tohto obdobia špecifikovali, navrhli, implementovali a ostestovali považujeme za dobrý základ pre ďalší vývoj.

Prílohy

Príloha A – ACM klasifikácia

- **A. General Literature**
  - A.0 GENERAL
  - A.1 INTRODUCTORY AND SURVEY
  - A.2 REFERENCE (e.g., dictionaries, encyclopedias, glossaries)
  - A.m MISCELLANEOUS
- **B. Hardware**
  - B.0 GENERAL
  - B.1 CONTROL STRUCTURES AND MICROPROGRAMMING (D.3.2)
  - B.2 ARITHMETIC AND LOGIC STRUCTURES
  - B.3 MEMORY STRUCTURES
  - B.4 INPUT/OUTPUT AND DATA COMMUNICATIONS
  - B.5 REGISTER-TRANSFER-LEVEL IMPLEMENTATION
  - B.6 LOGIC DESIGN
  - B.7 INTEGRATED CIRCUITS
  - B.8 PERFORMANCE AND RELIABILITY (C.4)
  - B.m MISCELLANEOUS
- **C. Computer Systems Organization**
  - C.0 GENERAL
  - C.1 PROCESSOR ARCHITECTURES
  - C.2 COMPUTER-COMMUNICATION NETWORKS
  - C.3 SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS (J.7)
  - C.4 PERFORMANCE OF SYSTEMS
  - C.5 COMPUTER SYSTEM IMPLEMENTATION
  - C.m MISCELLANEOUS
- **D. Software**
  - D.0 GENERAL
  - D.1 PROGRAMMING TECHNIQUES (E)
  - D.2 SOFTWARE ENGINEERING (K.6.3)
  - D.3 PROGRAMMING LANGUAGES
  - D.4 OPERATING SYSTEMS (C)

- D.m MISCELLANEOUS

- **E. Data**
  - E.0 GENERAL
  - E.1 DATA STRUCTURES
  - E.2 DATA STORAGE REPRESENTATIONS
  - E.3 DATA ENCRYPTION
  - E.4 CODING AND INFORMATION THEORY (H.1.1)
  - E.5 FILES (D.4.3, F.2.2, H.2)
  - E.m MISCELLANEOUS
- **F. Theory of Computation**
  - F.0 GENERAL
  - F.1 COMPUTATION BY ABSTRACT DEVICES
  - F.2 ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY (B.6, B.7, F.1.3)
  - F.3 LOGICS AND MEANINGS OF PROGRAMS
  - F.4 MATHEMATICAL LOGIC AND FORMAL LANGUAGES
  - F.m MISCELLANEOUS
- **G. Mathematics of Computing**
  - G.0 GENERAL
  - G.1 NUMERICAL ANALYSIS
  - G.2 DISCRETE MATHEMATICS
  - G.3 PROBABILITY AND STATISTICS
  - G.4 MATHEMATICAL SOFTWARE
  - G.m MISCELLANEOUS
- **H. Information Systems**
  - H.0 GENERAL
  - H.1 MODELS AND PRINCIPLES
  - H.2 DATABASE MANAGEMENT (E.5)
  - H.3 INFORMATION STORAGE AND RETRIEVAL
  - H.4 INFORMATION SYSTEMS APPLICATIONS
  - H.5 INFORMATION INTERFACES AND PRESENTATION (e.g., HCI) (I.7)
  - H.m MISCELLANEOUS
- **I. Computing Methodologies**
  - I.0 GENERAL

- I.1 SYMBOLIC AND ALGEBRAIC MANIPULATION
- I.2 ARTIFICIAL INTELLIGENCE
- I.3 COMPUTER GRAPHICS
- I.4 IMAGE PROCESSING AND COMPUTER VISION
- I.5 PATTERN RECOGNITION
- I.6 SIMULATION AND MODELING (G.3)
- I.7 DOCUMENT AND TEXT PROCESSING (H.4, H.5)
- I.m MISCELLANEOUS

- **J. Computer Applications**
  - J.0 GENERAL
  - J.1 ADMINISTRATIVE DATA PROCESSING
  - J.2 PHYSICAL SCIENCES AND ENGINEERING
  - J.3 LIFE AND MEDICAL SCIENCES
  - J.4 SOCIAL AND BEHAVIORAL SCIENCES
  - J.5 ARTS AND HUMANITIES
  - J.6 COMPUTER-AIDED ENGINEERING
  - J.7 COMPUTERS IN OTHER SYSTEMS (C.3)
  - J.m MISCELLANEOUS
- **K. Computing Milieux**
  - K.0 GENERAL
  - K.1 THE COMPUTER INDUSTRY
  - K.2 HISTORY OF COMPUTING
  - K.3 COMPUTERS AND EDUCATION
  - K.4 COMPUTERS AND SOCIETY
  - K.5 LEGAL ASPECTS OF COMPUTING
  - K.6 MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS
  - K.7 THE COMPUTING PROFESSION
  - K.8 PERSONAL COMPUTING
  - K.m MISCELLANEOUS

Príloha B – Zmenená ACM klasifikácia

1. Hardware
1.1 Kontrolné štruktúry a mikroprogramovania - **Mikropočítače**
1.2 Aritmetické a logické štruktúry - **Návrh digitálnych systémov, Rekonfigurovateľné digitálne systémy**
1.3 Vstupná/výstupná a dátová komunikácia - **Teória komunikácie, Komunikačné služby a siete, Periférne zariadenia, Komunikačné systémy, Digitálne spracovanie signálov, WAN technológie**
1.4 Logický dizajn - **Rekonfigurovateľné digitálne systémy**
1.5 Integrované obvody – **Logické obvody, Elektrotechnika, Elektronika, Fyzika**, **Programovateľné obvody**
1.6 Výkon a spoľahlivosť – **Meranie, Diagnostika a spoľahlivosť digitálnych systémov**
1.7 Rôzne - **Testovanie digitálnych systémov**

2. Organizácia počítačových systémov
2.1 Architektúra procesorov - **Architektúra počítačov, Distribuované počítačové systémy, Návrh digitálnych systémov, Architektúra počítačových systémov**
2.2 Počítačovo-komunikačné siete – **Počítačové siete I, Počítačové siete II, Šifrovanie v počítačových sieťach, WAN technológie**
2.3 Výkon systémov – **Meranie, Vnorené systémy, Diagnostika a spoľahlivosť digitálnych systémov**
2.4 Implementácia počítačových systémov – **Projektovanie aplikácií počítačov, Distribuované počítačové systémy, Vnorené systémy**
2.5 Rôzne - **Bezpečnosť počítačových systémov, Bezpečnosť v Internete**

3. Software
3.1 Programovacie techniky – **Dátové štruktúry a algoritmy**
3.2 Softvérové inžinierstvo – **Manažment v softvérovom inžinierstve, Princípy softvérového inžinierstva**
3.3 Programovacie jazyky – funkcionálne (lisp) – **Funkcionálne a logické programovanie**
- logické (prolog) - **Funkcionálne a logické programovanie, Umelá inteligencia**
- procedurálne (C) – **Stavba operačných systémov, Procedurálne programovanie**
- objektovo-orientované (Java, C++) – **Vývoj programov pre platformu Java 2, Objektovo-orientované programovanie**
- skriptovacie (cshell) – **Operačné systémy**
- aspektovo - orientované (AspectJ) – **Aspektovo - orientovaný vývoj softvéru**

3.4 Operačné systémy (Unix, Linux, windows) – **Operačné systémy, Stavba operačných systémov, Distribuované operačné systémy**
3.5 Rôzne - **Opis a preklad programovacích jazykov, Návrh prekladačov, Objektovo orientovaná analýza a návrh softvéru, Znalostné systémy, Kvalita programových a informačných systémov**

4. Dáta
4.1 Dátové štruktúry – **Dátové štruktúry a algoritmy**
4.2 Reprezentácie a uskladnenia dát – **Pokročilé databázové technológie, Databázové systémy, Inteligentná analýza údajov**
4.3 Kryptovanie dát – **Základy kryptológie, Šifrovanie v počítačových sieťach**
4.4 Kódovanie a informačná teória – **Metódy inžinierskej práce, Teoretické základy informatiky, Kódovanie, Komunikačné systémy**
4.5 Súbory – **Operačné systémy, Asemblery a systémové programovanie**

5. Výpočtová teória
5.1 Analýza a zložitosť problémov systémov – **Analýza a zložitosť algoritmov, Evolučné algoritmy, Metódy rozhodovania a teória hier, Tvorba efektívnych algoritmov a programov**
5.2 Matematické logické a formálne jazyky – **Teoretické základy informatiky, Metódy a prostriedky špecifikácie, Špecifikačné a opisné jazyky, Matematická logika I, Matematická logika II, Formálne modely digitálnych systémov**

6. Matematika výpočtov
6.1 Numerická analýza – **Numerické a štatistické výpočty**
6.2 Diskrétna matematika – **Algebra a diskrétna matematika**
6.3 Pravdepodobnosť a štatistika – **Pravdepodobnosť a štatistika**
6.4 Matematický software – **Maticové algoritmy, Numerické a štatistické výpočty**

6.5 Rôzne – **Fuzzy systémy, Grafy, Matematická analýza I, Matematická analýza II**

7. Informačné systémy
7.1 Modely a princípy – **Architektúra informačných systémov, Princípy informačných systémov I, Princípy informačných systémov II, Princípy softvérového inžinierstva, Princípy webového inžinierstva, Tvorba softvérových systémov, Architektonické a návrhové vzory pre programové informačné systémy, Modelovanie ekonomických systémov, Formálne modely digitálnych systémov, Objektovo orientovaná analýza a návrh softvéru**
7.2 Databázový manažment – **Databázové systémy, Pokročilé databázové technológie**
7.3 Získavanie a ukladanie informácií – **Pokročilé databázové technológie, Vyhľadávanie informácií, Získavanie znalostí**
7.4 Aplikácie informačných systémov – **Architektúra informačných systémov**
7.5 Informačné rozhrania a prezentácie – **Interakcia človek – počítač**
7.6 Rôzne - **Počítačové multimediálne systémy, Kvalita programových a informačných systémov**

8. Výpočtová metodológia
8.1 Symbolická a algebraická manipulácia – **Algebry a grafy, Algebra, Algebra a diskrétna matematika**
8.2 Umelá inteligencia – **Umelá inteligencia, Neurónové siete**
8.3 Počítačová grafika – **Počítačová grafika, Interakcia človek – počítač, Počítačové multimediálne systémy**
8.4 Rozpoznávanie vzorov – **Architektonické a návrhové vzory pre programové informačné systémy, Architektúra softvérových systémov**
8.5 Modelovanie a simulácia – **Modelovanie a simulácia**
8.6 Spracovanie textu a dokumentov – **Vyhľadávanie informácií, Metódy inžinierskej práce**

9. Počítačové aplikácie
9.1 Sociálne a kognitívne vedy – **Interakcia človek – počítač, Spoločenské súvislosti informatiky a informačných a komunikačných technológií**
9.2 Počítačovo podporované inžinierstvo – **Interakcia človek – počítač, Projektovanie aplikácií počítačov**

10. Informatika v sociálnych vedách
10.1 Počítačový priemysel – **Marketing, Finančný manažment, Účtovníctvo**
10.2 História výpočtov – **Metódy inžinierskej práce, Architektúra počítačov**
10.3 Počítače, vzdelávanie a spoločnosť - **Spoločenské súvislosti informatiky a informačných a komunikačných technológií, Komunikácia v dejinách kultúry, Anglický jazyk + iné jazyky**
10.4 Právne aspekty v informačných technológiách - **Právo – vybrané problémy, Právo informačných a komunikačných technológií**
10.5 Manažment v informačných systémoch – **Manažment v softvérovom inžinierstve, Manažérska ekonómia, Podnikanie a manažment, Manažment bezpečnosti informačných technológií, Manažment kvality, Manažment sociálnych systémov, Bezpečnosť a manažment informačných systémov**
10.6 Rôzne – **Dejiny dizajnu**

Príloha C - XML s hierarchiou kľúčových slov a s definovaním a váhami predmetov, ktoré prispievajú k jednotlivým kľúčovým slovám

Štruktúra:

<?xml version="1.0" encoding="ISO-8859-1"?>
identifikácia XML súboru
<rozdelenie>
identifikácia začiatku definície kľúčových slov a váh
<kategoria id="1" nazov="Data">
identifikácia začiatku definície kľúčového slova prvej úrovne (priamy nasledovník kľúčového slova "root"). Toto kľúčové slovo je nazvané "Data". Za týmto tagom nasledujú buď definície kľúčových slov, ktoré sú nasledovníkmi tohto kľúčového slova, alebo zoznam predmetov, ktoré prispievajú k ohodnoteniu tohto kľúčového slova.
<podkategoria id="1.1" meno="Subory">
identifikácia kľúčového slova druhej úrovne s názvom " Subory". Keďže sa nachádza v rámci definície kľúčového slova "Data", tak je priamym potomkom tohto kľúčového slova.
<predmet>Operacne systemy<vaha>1.0</vaha></predmet>
identifikácia predmetu, ktorý prispieva k ohodnoteniu kľúčového slova. Tento tag sa nachádza v definícií kľúčového slova " Subory" a preto pirspieva k ohodnoteniu práve tohto kľúčového slova. V rámci tagu <predmet> je aj párový tag <vaha>, ktorý definuje akou váhou prispieva hodnotenie predmetu k ohodnoteniu kľúčového slova.
<vaha>0.3</vaha>
pokiaľ tag <vaha> je priamo v tagu <podkategoria> a nie je v tagu <predmet>, tak definuje váhu, ktorou prispieva ohodnotenie daného kľúčového slova, definovaného v tagu <podkategoria>, k ohodnoteniu jeho nadradeného kľúčového slova, definovaného v tagu <kategoria>.
</podkategoria>
</kategoria>
</rozdelenie>

Zhrnutie:

V našom prípade máme dve kľúčové slová a to "Data" a "Subory ", kde "Data" je nadradeným kľúčovým slovom (rodičom) kľúčového slova "Subory". K ohodnoteniu kľúčového slova "Subory" prispieva hodnotenie predmetu "Operacne systemy" váhou 1.0 (čiže hodnotenie predmetu sa priamo pripočítava k ohodnoteniu kľúčového slova) a ohodnotenie kľúčového slova "Subory" prispieva k ohodnoteniu kľúčového slova "Data" váhou 0.3 (čiže k ohodnoteniu "Data" sa pripočítava 0.3*hodnotenie kľúčového slova "Subory")

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rozdelenie>
<kategoria id="1" nazov="Hardware">
   <podkategoria  id="1.1" meno="Kontrolne struktury a mikroprogramovania">
      <predmet>Mikropocitace<vaha>0.5</vaha><vaha>1.0</vaha></predmet>
      <vaha>0.3</vaha>
   </podkategoria>
   <podkategoria  id="1.2" meno="Aritmeticke a logicke struktury">
      <predmet>Navrh digitalnych systemov<vaha>0.4</vaha></predmet>
      <predmet>Rekonfigurovatelne digitalne systemy<vaha>0.3</vaha></predmet>
      <vaha>0.1</vaha>
   </podkategoria>
   <podkategoria id="1.3" meno="Vstupna-vystupna a datova komunikacia">
      <predmet>Teoria komunikacie<vaha>0.5</vaha></predmet>
      <predmet>Komunikacne sluzby a siete<vaha>0.6</vaha></predmet>
      <predmet>Periferne zariadenia<vaha>0.7</vaha></predmet>
      <predmet>Komunikacne systemy<vaha>0.3</vaha></predmet>
      <predmet>Digitalne spracovanie signalov<vaha>0.2</vaha></predmet>
      <predmet>WAN technologie<vaha>0.4</vaha></predmet>
      <vaha>0.3</vaha>
   </podkategoria>
   <podkategoria  id="1.4" meno="Logicky dizajn">
      <predmet>Rekonfigurovatelne digitalne systemy<vaha>0.4</vaha></predmet>
      <vaha>0.1</vaha>
   </podkategoria>
   <podkategoria  id="1.5" meno="Integrovane obvody">
      <predmet>Logicke obvody<vaha>0.8</vaha></predmet>
      <predmet>Elektrotechnika<vaha>0.3</vaha></predmet>
      <predmet>Elektronika<vaha>0.3</vaha></predmet>
      <predmet>Fyzika<vaha>0.2</vaha></predmet>
      <predmet>Programovatelne obvody<vaha>0.5</vaha></predmet>
      <vaha>0.6</vaha>
   </podkategoria>
   <podkategoria  id="1.6" meno="Vykon a spolahlivost">
      <predmet>Meranie<vaha>0.4</vaha></predmet>
      <predmet>Diagnostika a spolahlivost digitalnych systemov<vaha>0.7</vaha></predmet>
      <vaha>0.2</vaha>
   </podkategoria>
   <podkategoria  id="1.7"  meno="Rozne">
      <predmet>Testovanie digitalnych systemov<vaha>0.2</vaha></predmet>
      <vaha>0.2</vaha>
   </podkategoria>
</kategoria>
<kategoria  id="2" nazov="Organizacia pocitacovych systemov">
   <podkategoria id="2.1"  meno="Architektura procesorov">
      <predmet>Architektura pocitacov<vaha>0.9</vaha></predmet>
      <predmet>Distribuovane pocitacove systemy<vaha>0.2</vaha></predmet>
      <predmet>Navrh digitalnych systemov<vaha>0.3</vaha></predmet>
      <predmet>Architektura pocitacovych systemov<vaha>0.7</vaha></predmet>
      <vaha>0.8</vaha>
   </podkategoria>
   <podkategoria  id="2.2" meno="Pocitacovo-komunikacne siete">
      <predmet>Pocitacove siete I<vaha>1.0</vaha></predmet>
      <predmet>Pocitacove siete II<vaha>1.0</vaha></predmet>
      <predmet>Sifrovanie v pocitacovych sietach<vaha>0.3</vaha></predmet>
      <predmet>WAN technologie<vaha>0.7</vaha></predmet>
      <vaha>0.5</vaha>
   </podkategoria>
   <podkategoria  id="2.3" meno="Vykon systemov">
      <predmet>Meranie<vaha>0.3</vaha></predmet>
      <predmet>Vnorene systemy<vaha>0.2</vaha></predmet>
      <predmet>Diagnostika a spolahlivost digitalnych systemov<vaha>0.2</vaha></predmet>
      <vaha>0.2</vaha>
   </podkategoria>
   <podkategoria  id="2.4" meno="Implementacia pocitacovych systemov">
      <predmet>Projektovanie aplikacii pocitacov<vaha>0.9</vaha></predmet>
      <predmet>Distribuovane pocitacove systemy<vaha>0.2</vaha></predmet>
```

```
      <predmet>Vnorene systemy<vaha>0.1</vaha></predmet>
      <vaha>0.4</vaha>
    </podkategoria>
    <podkategoria  id="2.5" meno="Rozne">
      <predmet>Bezpecnost pocitacovych systemov<vaha>0.4</vaha></predmet>
      <predmet>Bezpecnost v Internete<vaha>0.4</vaha></predmet>
      <vaha>0.1</vaha>
    </podkategoria>
  </kategoria>
  <kategoria  id="3" nazov="Software">
    <podkategoria id="3.1" meno="Programovacie techniky">
      <predmet>Datove struktury a algoritmy<vaha>0.8</vaha></predmet>
      <vaha>0.4</vaha>
    </podkategoria>
    <podkategoria  id="3.2" meno="Softverove inzinierstvo">
      <predmet>Manazment v softverovom inzinierstve<vaha>0.6</vaha></predmet>
      <predmet>Principy softveroveho inzinierstva<vaha>0.8</vaha></predmet>
      <vaha>0.7</vaha>
    </podkategoria>
    <podkategoria  id="3.3" meno="Programovacie jazyky">
      <podpodkategoria id="3.3.1" meno="Funkcionalne">
        <predmet>Funkcionalne a logicke programovanie<vaha>0.5</vaha></predmet>
        <vaha>0.3</vaha>
      </podpodkategoria>
      <podpodkategoria  id="3.3.2" meno="Logicke">
        <predmet>Funkcionalne a logicke programovanie<vaha>0.5</vaha></predmet>
        <predmet>Umela inteligencia<vaha>0.4</vaha></predmet>
        <vaha>0.3</vaha>
      </podpodkategoria>
      <podpodkategoria id="3.3.3" meno="Proceduralne">
        <predmet>Stavba operacnych systemov<vaha>0.2</vaha></predmet>
        <predmet>Proceduralne programovanie<vaha>1.0</vaha></predmet>
        <vaha>0.6</vaha>
      </podpodkategoria>
      <podpodkategoria  id="3.3.4" meno="Objektovo-orientovane">
        <predmet>Vyvoj programov pre platformu Java 2<vaha>1.0</vaha></predmet>
        <predmet>Objektovo-orientovane programovanie<vaha>1.0</vaha></predmet>
        <vaha>0.7</vaha>
      </podpodkategoria>
      <podpodkategoria  id="3.3.5" meno="Skriptovacie">
        <predmet>Operacne systemy<vaha>0.5</vaha></predmet>
        <vaha>0.1</vaha>
      </podpodkategoria>
      <podpodkategoria  id="3.3.6" meno="Aspektovo-orientovane">
        <predmet>Aspektovo-orientovany vyvoj softveru<vaha>0.7</vaha></predmet>
        <vaha>0.1</vaha>
      </podpodkategoria>
      <vaha>0.6</vaha>
    </podkategoria>
    <podkategoria  id="3.4" meno="Operacne systemy">
      <predmet>Operacne systemy<vaha>1.0</vaha></predmet>
      <predmet>Stavba operacnych systemov<vaha>0.8</vaha></predmet>
      <predmet>Distribuovane operacne systemy<vaha>0.7</vaha></predmet>
      <vaha>0.8</vaha>
    </podkategoria>
    <podkategoria  id="3.5" meno="Rozne">
      <predmet>Opis a preklad programovacich jazykov<vaha>0.6</vaha></predmet>
      <predmet>Navrh prekladacov<vaha>0.2</vaha></predmet>
      <predmet>Objektovo orientovana analyza a navrh softveru<vaha>0.5</vaha></predmet>
      <predmet>Znalostne systemy<vaha>0.2</vaha></predmet>
      <predmet>Kvalita programovych a informacnych systemov<vaha>0.1</vaha></predmet>
      <vaha>0.3</vaha>
    </podkategoria>
  </kategoria>
  <kategoria  id="4" nazov="Data">
    <podkategoria id="4.1" meno="Datove struktury">
      <predmet>Datove struktury a algoritmy<vaha>1.0</vaha></predmet>
```

```
      <vaha>0.9</vaha>
    </podkategoria>
    <podkategoria  id="4.2" meno="Reprezentacia a uskladnenie dat">
      <predmet>Pokrocile databazove technologie<vaha>0.7</vaha></predmet>
      <predmet>Databazove systemy<vaha>0.5</vaha></predmet>
      <predmet>Inteligentna analyza udajov<vaha>0.2</vaha></predmet>
      <vaha>0.8</vaha>
    </podkategoria>
    <podkategoria id="4.3" meno="Kryptovanie dat">
      <predmet>Zaklady kryptologie<vaha>0.4</vaha></predmet>
      <predmet>Sifrovanie v pocitacovych sietach<vaha>0.5</vaha></predmet>
      <vaha>0.6</vaha>
    </podkategoria>
    <podkategoria id="4.4" meno="Kodovanie a informacna teoria">
      <predmet>Metody inzinierskej prace<vaha>0.3</vaha></predmet>
      <predmet>Teoreticke zaklady informatiky<vaha>0.5</vaha></predmet>
      <predmet>Kodovanie<vaha>1.0</vaha></predmet>
      <predmet>Komunikacne systemy<vaha>0.6</vaha></predmet>
      <vaha>0.4</vaha>
    </podkategoria>
    <podkategoria id="4.5" meno="Subory">
      <predmet>Operacne systemy<vaha>0.4</vaha></predmet>
      <predmet>Asemblery a systemove programovanie<vaha>0.3</vaha></predmet>
      <vaha>0.5</vaha>
    </podkategoria>
  </kategoria>
  <kategoria id="5" nazov="Vypoctova teoria">
    <podkategoria id="5.1"  meno="Analyza a zlozitost problemov systemov">
      <predmet>Analyza a zlozitost algoritmov<vaha>0.8</vaha></predmet>
      <predmet>Evolucne algoritmy<vaha>0.4</vaha></predmet>
      <predmet>Metody rozhodovania a teoria hier<vaha>0.3</vaha></predmet>
      <predmet>Tvorba efektivnych algoritmov a programov<vaha>0.3</vaha></predmet>
      <vaha>0.4</vaha>
    </podkategoria>
    <podkategoria  id="5.2" meno="Matematicke logicke a formalne jazyky">
      <predmet>Teoreticke zaklady informatiky<vaha>0.5</vaha></predmet>
      <predmet>Metody a prostriedky specifikacie<vaha>0.7</vaha></predmet>
      <predmet>Specifikacne a opisne jazyky<vaha>0.8</vaha></predmet>
      <predmet>Matematicka logika I<vaha>0.7</vaha></predmet>
      <predmet>Matematicka logika II<vaha>0.3</vaha></predmet>
      <predmet>Formalne modely digitalnych systemov<vaha>0.1</vaha></predmet>
      <vaha>0.3</vaha>
    </podkategoria>
  </kategoria>
  <kategoria id="6" nazov="Matematika vypoctov">
    <podkategoria id="6.1" meno="Numericka analyza">
      <predmet>Numericke a statisticke vypocty<vaha>0.8</vaha></predmet>
      <vaha>0.6</vaha>
    </podkategoria>
    <podkategoria  id="6.2" meno="Diskretna matematika">
      <predmet>Algebra a diskretna matematika<vaha>0.5</vaha></predmet>
      <vaha>0.3</vaha>
    </podkategoria>
    <podkategoria  id="6.3" meno="Pravdepodobnost a statistika">
      <predmet>Pravdepodobnost a statistika<vaha>1.0</vaha></predmet>
      <vaha>0.8</vaha>
    </podkategoria>
    <podkategoria  id="6.4"  meno="Matematicky software">
      <predmet>Maticove algoritmy<vaha>0.6</vaha></predmet>
      <predmet>Numericke a statisticke vypocty<vaha>0.4</vaha></predmet>
      <vaha>0.7</vaha>
    </podkategoria>
    <podkategoria id="6.5"  meno="Rozne">
      <predmet>Fuzzy systemy<vaha>0.3</vaha></predmet>
      <predmet>Grafy<vaha>0.2</vaha></predmet>
      <predmet>Matematicka analyza I<vaha>0.1</vaha></predmet>
      <predmet>Matematicka analyza II<vaha>0.1</vaha></predmet>
```

```
      <vaha>0.2</vaha>
    </podkategoria>
  </kategoria>
  <kategoria id="7" nazov="Informacne systemy">
    <podkategoria id="7.1" meno="Modely a principy">
      <predmet>Architektura informacnych systemov<vaha>0.6</vaha></predmet>
      <predmet>Principy informacnych systemov I<vaha>0.8</vaha></predmet>
      <predmet>Principy informacnych systemov II<vaha>0.7</vaha></predmet>
      <predmet>Principy softveroveho inzinierstva<vaha>0.9</vaha></predmet>
      <predmet>Principy weboveho inzinierstva<vaha>0.4</vaha></predmet>
      <predmet>Tvorba softverovych systemov<vaha>0.3</vaha></predmet>
      <predmet>Architektonicke a navrhove vzory pre programove informacne systemy<vaha>0.8</vaha></predmet>
      <predmet>Modelovanie ekonomickych systemov<vaha>0.3</vaha></predmet>
      <predmet>Formalne modely digitalnych systemov<vaha>0.2</vaha></predmet>
      <predmet>Objektovo orientovana analyza a navrh softveru<vaha>0.2</vaha></predmet>
      <vaha>0.6</vaha>
    </podkategoria>
    <podkategoria id="7.2" meno="Databazovy manazment">
      <predmet>Databazove systemy<vaha>0.4</vaha></predmet>
      <predmet>Pokrocile databazove technologie<vaha>0.4</vaha></predmet>
      <vaha>0.2</vaha>
    </podkategoria>
    <podkategoria  id="7.3" meno="Ziskavanie a ukladanie informacii">
      <predmet>Pokrocile databazove technologie<vaha>0.3</vaha></predmet>
      <predmet>Vyhladavanie informacii<vaha>0.8</vaha></predmet>
      <predmet>Ziskavanie znalosti<vaha>0.6</vaha></predmet>
      <vaha>0.5</vaha>
    </podkategoria>
    <podkategoria id="7.4" meno="Aplikacie informacnych systemov">
      <predmet>Architektura informacnych systemov<vaha>0.4</vaha></predmet>
      <vaha>0.7</vaha>
    </podkategoria>
    <podkategoria  id="7.5"  meno="Informacne rozhrania a prezentacie">
      <predmet>Interakcia cloveka s pocitacom<vaha>0.9</vaha></predmet>
      <vaha>0.3</vaha>
    </podkategoria>
    <podkategoria  id="7.6" meno="Rozne">
      <predmet>Pocitacove multimedialne systemy<vaha>0.2</vaha></predmet>
      <predmet>Kvalita programovych a informacnych systemov<vaha>0.4</vaha></predmet>
      <vaha>0.3</vaha>
    </podkategoria>
  </kategoria>
  <kategoria  id="8" nazov="Vypoctova metodologia">
    <podkategoria id="8.1" meno="Symbolicka a algbraicka manipulacia">
      <predmet>Algebry a grafy<vaha>0.4</vaha></predmet>
      <predmet>Algebra<vaha>0.7</vaha></predmet>
      <predmet>Algebra a diskretna matematika<vaha>0.3</vaha></predmet>
      <vaha>0.5</vaha>
    </podkategoria>
    <podkategoria  id="8.2"  meno="Umela inteligencia">
      <predmet>Umela inteligencia<vaha>1.0</vaha></predmet>
      <predmet>Neuronove siete<vaha>0.4</vaha></predmet>
      <vaha>0.4</vaha>
    </podkategoria>
    <podkategoria  id="8.3" meno="Pocitacova grafika">
      <predmet>Pocitacova grafika<vaha>1.0</vaha></predmet>
      <predmet>Interakcia cloveka s pocitacom<vaha>0.7</vaha></predmet>
      <predmet>Pocitacove multimedialne systemy<vaha>0.3</vaha></predmet>
      <vaha>0.4</vaha>
    </podkategoria>
    <podkategoria  id="8.4"  meno="Rozpoznavanie vzorov">
      <predmet>Architektonicke a navrhove vzory pre programove informacne systemy<vaha>0.7</vaha></predmet>
      <predmet>Architektura softverovych systemov<vaha>0.2</vaha></predmet>
      <vaha>0.2</vaha>
    </podkategoria>
    <podkategoria  id="8.5" meno="Modelovanie a simulacia">
      <predmet>Modelovanie a simulacia<vaha>1.0</vaha></predmet>
```

```
      <vaha>0.3</vaha>
   </podkategoria>
   <podkategoria  id="8.6"  meno="Spracovanie textu a dokumentov">
      <predmet>Vyhladavanie informacii<vaha>0.8</vaha></predmet>
      <predmet>Metody inzinierskej prace<vaha>0.3</vaha></predmet>
      <vaha>0.2</vaha>
   </podkategoria>
</kategoria>
<kategoria id="9" nazov="Pocitacove aplikacie">
   <podkategoria id="9.1" meno="Socialne a kognitivne vedy">
      <predmet>Interakcia cloveka s pocitacom<vaha>0.2</vaha></predmet>
      <predmet>Spolocenske suvislosti informatiky a informacnych a komunikacnych
technologii<vaha>0.7</vaha></predmet>
      <vaha>0.3</vaha>
   </podkategoria>
   <podkategoria id="9.2" meno="Pocitacovo podporovane inzinierstvo">
      <predmet>Interakcia cloveka s pocitacom<vaha>0.3</vaha></predmet>
      <predmet>Projektovanie aplikacii pocitacov<vaha>0.5</vaha></predmet>
      <vaha>0.4</vaha>
   </podkategoria>
</kategoria>
<kategoria id="10"  nazov="Informatika v socialnych vedach">
   <podkategoria  id="10.1"  meno="Pocitacovy priemysel">
      <predmet>Marketing<vaha>0.3</vaha></predmet>
      <predmet>Financny manazment<vaha>0.3</vaha></predmet>
      <predmet>Uctovnictvo<vaha>0.3</vaha></predmet>
      <vaha>0.5</vaha>
   </podkategoria>
   <podkategoria  id="10.2"  meno="Historia vypoctov">
      <predmet>Metody inzinierskej prace<vaha>0.4</vaha></predmet>
      <predmet>Architektura pocitacov<vaha>0.5</vaha></predmet>
      <vaha>0.1</vaha>
   </podkategoria>
   <podkategoria id="10.3" meno="Pocitace. vzdelavanie a spolocnost">
      <predmet>Spolocenske suvislosti informatiky a informacnych a komunikacnych
technologii<vaha>1.0</vaha></predmet>
      <predmet>Komunikacia v dejinach kultury<vaha>0.2</vaha></predmet>
      <predmet>Anglicky jazyk<vaha>0.1</vaha></predmet>
      <predmet>Cudzi jazyk<vaha>0.1</vaha></predmet>
      <vaha>0.9</vaha>
   </podkategoria>
   <podkategoria id="10.4" meno="Pravne aspekty v informacnych technologiach">
      <predmet>Pravo-vybrane problemy<vaha>0.8</vaha></predmet>
      <predmet>Pravo informacnych a komunikacnych technologii<vaha>1.0</vaha></predmet>
      <vaha>0.6</vaha>
   </podkategoria>
   <podkategoria  id="10.5" meno="Manazment v informacnych systemoch">
      <predmet>Manazment v softverovom inzinierstve<vaha>0.6</vaha></predmet>
      <predmet>Manazerska ekonomia<vaha>0.1</vaha></predmet>
      <predmet>Podnikanie a manazment<vaha>0.3</vaha></predmet>
      <predmet>Manazment bezpecnosti informacnych technologii<vaha>0.6</vaha></predmet>
      <predmet>Manazment kvality<vaha>0.4</vaha></predmet>
      <predmet>Manazment socialnych systemov<vaha>0.1</vaha></predmet>
      <predmet>Bezpecnost a manazment informacnych systemov<vaha>0.7</vaha></predmet>
      <vaha>0.7</vaha>
   </podkategoria>
   <podkategoria  id="10.6" meno="Rozne">
      <predmet>Dejiny dizajnu<vaha>0.1</vaha></predmet>
      <vaha>0.3</vaha>
   </podkategoria>
</kategoria>
</rozdelenie>
```

Príloha D – Inštalačná a používateľská príručka

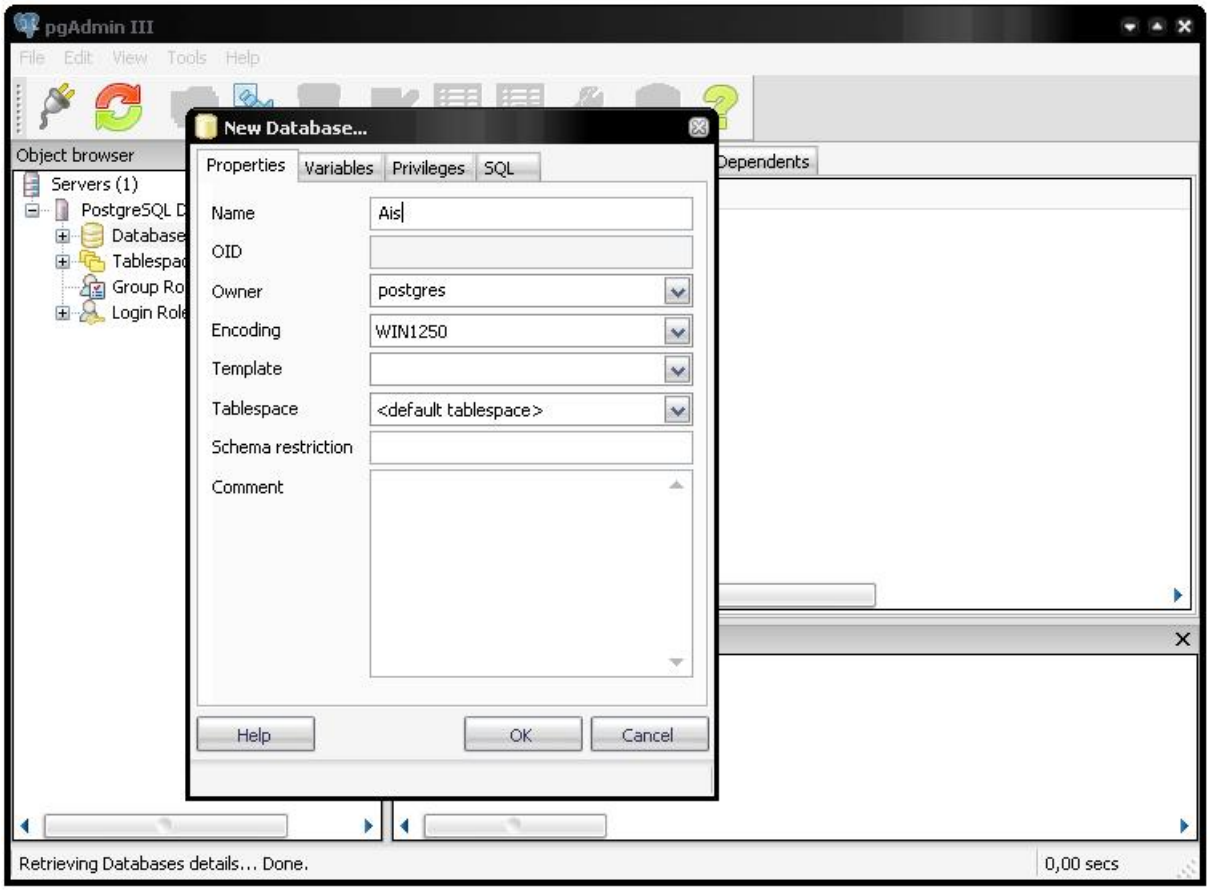
D.1 Inštalácia informačného systému BZZ

V prípade, že chce používateľ nainštalovať svoju vlastnú verziu informačného systému BZZ, je potrebné vykonať kroky opísané v častiach *6.1, 6.2, 6.3, 7*. Okrem toho je na školskom serveri spustená on-line verzia systému BZZ. Návod na jej spustenie sa nachádza v časti *8*. V tom prípade nie je potrebné inštalovať žiadne aplikácie. Potrebná je len aplikácia **putty** a prístup na školský server **labss**.

D.1.1 Databázový systém PostgreSQL

Informačný systém pracuje s dynamickými údajmi, ktoré sú uložené v databáze - PostgreSQL. Pre korektnú prevádzku odporúčame použiť verziu 8.3 alebo vyššiu. Aktuálnu verziu je možné stiahnuť na **http://www.postgresql.org/download/**. Databázový systém nainštalujeme spolu s používateľským rozhraním pgAdmin (potrebné kroky sú popísané v príslušnej dokumentácii - dostupnej tiež na **http://www.postgresql.org/docs/**) a aplikáciou na spravovanie databázy – pgAdminIII. Pri inštalácii je ponúknutá možnosť voľby hesla, to zvolíme napríklad „postgres“. Postupujeme podľa nasledujúcich krokov.

1. Spustíme aplikáciu pgAdminIII. Ak chceme vytvoriť nového používateľa, označíme aktuálny databázový server, z ponuky **Edit** vyberieme **New Object → New Login Role**. Do kolóniek **Role name** a **Password** zadáme meno a heslo, výber potvrdíme stlačením tlačidla OK. Vytvoriť nového používateľa však nie je nutné.
2. Pripojíme sa k serveru: z ponuky vyberieme **Tools → Connect**, zádáme heslo (ktoré sa zadávalo pri inštalácii).
3. Vytvoríme prázdnu databázu: označíme priečinok **Databases** a z ponuky **Edit** vyberieme **New Object → New Database**. Do kolónky **Name** vložíme “Bzz“. Z ponuky **Owner** vyberieme „postgres“ a z ponuky **Encoding** vyberieme „WIN1250“. Nakoniec potvrdíme výber stlačením tlačidla OK (Obr.6).
4. Vytvoríme druhú databázu. Do kolónky **Name** vložíme “Ais“. Z ponuky **Owner** vyberieme „postgres“ a z ponuky **Encoding** vyberieme „WIN1250“. Nakoniec potvrdíme výber stlačením tlačidla OK.



Obr.6: Prostredie aplikácie pgAdminIII, vytváranie novej databázy.

D.1.2 Rozhranie pre Java aplikácie

Je potrebné, aby na serveri bola nainštalovaná Java, najlepšie vo verzii 6.0 a vyššej. Odporúčame nainštalovať balíček Java 6 SE dostupný na:
http://java.sun.com/javase/downloads/.

D.1.3 Apache Tomcat

Apache Tomcat je server, ktorý je potrebný na spúšťanie Java servletov a stránok JSP (JavaServer Pages). Viac o tomto nástroji nájdete na **http://tomcat.apache.org/**.

D.2 Konfigurácia systému BZZ

D.2.1 Konfigurácia XML súborov

D.2.1.1 Konfigurácia súboru hibernate.cfg.xml

Tento súbor slúži na nastavenie spojenia s databázou. Pre nás sú dôležité parametre v párových značkách *property* označené a), b), c)

```
<hibernate-configuration>
    <session-factory>

        <property name="hibernate.connection.driver_class">  org.postgresql.Driver</property>
        <property name="hibernate.connection.url">jdbc:postgresql://localhost/Bzz</property>          a)
        <property name="hibernate.connection.username">postgres</property>                            b)
        <property name="hibernate.connection.password">postgres</property>                            c)
        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
        <property name="show_sql"> false </property>
        <property name="format_sql"> true </property>
        <property name="use_sql_comments"> false </property>

        <mapping resource="entities/Predmet.hbm.xml" />
        <mapping resource="entities/Pouzivatel.hbm.xml" />
        <mapping resource="entities/KlucoveSlovo.hbm.xml" />
        <mapping resource="entities/PredmetKlucoveslovo.hbm.xml" />
        <mapping resource="entities/OhodnotenieKSPreStudenta.hbm.xml" />
        <mapping resource="entities/Vaha.hbm.xml" />
        <mapping resource="entities/SebahodnotenieStudenta.hbm.xml" />
        <mapping resource="entities/pedagogHodnotenieStudenta.hbm.xml" />

    </session-factory>
</hibernate-configuration>
```

a) určuje názov serveru a názov databázy, všeobecný zápis má tvar:

jdbc:postgresql://názov_serveru/názov_databázy

V prípade, že PostgreSQL nebeží na štandardnom porte 5432, je nutné číslo portu

uviesť. Zápis vyzerá nasledovne:

jdbc:postgresql://názov_serveru:číslo_portu/názov_databázy

Názov servera sa musí zhodovať s názvom servera, na ktorom je spustená naša databáza a názov databázy musí byť totožný s našou databázou.

```
<property name="hibernate.connection.url">jdbc:postgresql://localhost/Bzz</property>
```

b) určuje prístupové meno k databáze. V časti *6.1 Databázový systém PostgreSQL* sme si popísali vytvorenie nového používateľa databázy a následné aplikovanie vlastníka (používateľa) databázy. Parameter v bode b) v konfiguračnom súbore by mal mať zhodný názov, ako vlastník našej databázy.

```
<property name="hibernate.connection.username">postgres</property>
```

c) určuje prístupové heslo k našej databáze. Toto heslo musí byť totožné s heslom majiteľa databázy (popísané v časti *6.1 Databázový systém PostgreSQL*).

```
<property name="hibernate.connection.password">postgres</property>
```

D.2.1.2 Konfigurácia súboru hibernateAis.cfg.xml

Tento súbor slúži na nastavenie pripojenia k databáze Ais. Pre nás sú dôležité parametre v párových značkách *property* označené a), b), c).

```
<hibernate-configuration>
  <session-factory>

    <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
    <property name="hibernate.connection.url">jdbc:postgresql://localhost/Ais</property>          a)
    <property name="hibernate.connection.username">postgres</property>                            b)
    <property name="hibernate.connection.password">postgres</property>                            c)
    <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
    <property name="show_sql"> false </property>
    <property name="format_sql"> true </property>
    <property name="use_sql_comments">false</property>
```

```
        <mapping resource="entitiesAIS/PouzivatelAIS.hbm.xml"/>
        <mapping resource="entitiesAIS/StudiumAIS.hbm.xml"/>
        <mapping resource="entitiesAIS/PredmetAIS.hbm.xml"/>
        <mapping resource="entitiesAIS/StudentPredmetAIS.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

a) určuje názov serveru a názov databázy.

```
<property name="hibernate.connection.url">jdbc:postgresql://localhost/Ais</property>
```

b) určuje prístupové meno k databáze.

```
<property name="hibernate.connection.username">postgres</property>
```

c) určuje prístupové heslo k našej databáze.

```
<property name="hibernate.connection.password">postgres</property>
```

D.2.2 Súbor Bzz_rozdelenie.xml

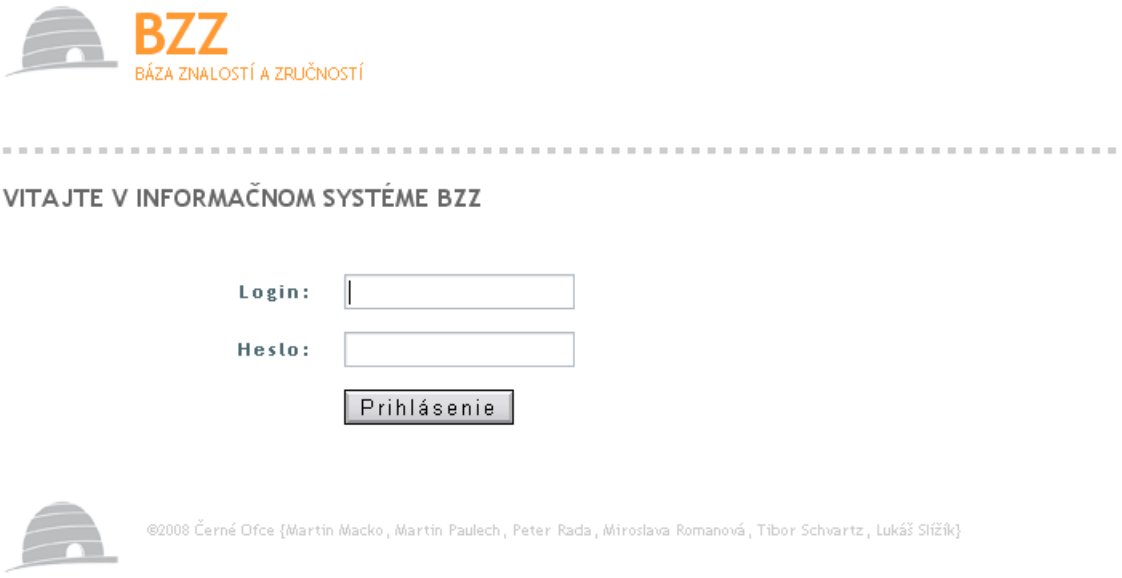
Súbor Bzz_rozdelenie.xml treba umiestniť do adresára **bin**, ktorý sa nachádza v domovskom adresári aplikácie Apache.

D.3 Spustenie a používanie systému BZZ

Do systému sa prihlásime na adrese: **http://localhost:8080/Final_v0.0/index.jsp** (v prípade, že port je nastavený na hodnotu 8080)**.** V prípade spúšťania on-line verzie systému postupujeme nasledovne:

1. otvoríme Putty
2. do poľa **Host name** nepíšeme: **labss2.fiit.stuba.sk**
3. zvolíme port **22**
4. v záložke **Connection -> SSH -> Tunnels** zadáme ako **Source port** číslo **8080** a ako **Destination** zvolíme **192.168.62.210:8080**. Portvrdíme tlačítkom **Add**
5. stlačíme tlačidlo **Open**
6. vyplníme prihlasovacie meno a heslo na server labss
7. v internetovom prehliadači si otvoríme úvodnú stránku **http://127.0.0.1:8080/Bzz/**

Na Obr.7 je zobrazená úvodná obrazovka informačného systému BZZ. Po prihlásení sa do systému pomocou loginu a hesla sa zobrazí hlavná obrazovka, ktorá je prispôsobená používateľovi v závislosti od jeho právomocí. Vzhľad a funkcie systému pre jednotlivých používateľov sú popísane v ďalších častiach dokumentu.



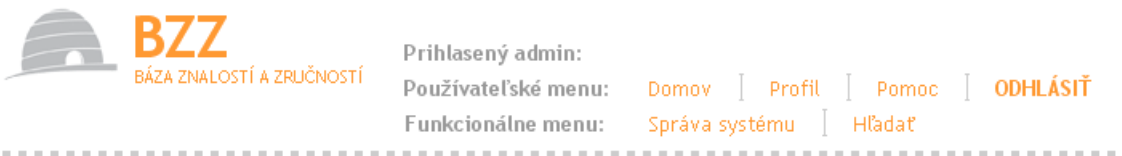
Obr.7: Úvodná obrazovka informačného systému BZZ.

D.3.1 Používatelia systému a ich funkcie

D.3.1.1 Administrátor

Po prihlásení sa administrátora do systému sa zobrazí okno ako na Obr.8. V hornej časti sa nachádza menu, skladajúce sa z dvoch častí. V jednej sa nachádzajú používateľské funkcie, v druhej funkcionálne.

V používateľskom menu sa nachádzajú odkazy na zobrazenie hlavnej stránky, na zobrazenie profilu s osobnými údajmi, na zobrazenie pomocných informácií a odhlásenie zo systému. Z funkcionálneho menu sa môžeme dostať k správe systému a vyhľadávaniu študentov.



Obr.8: Administrátorove hlavné menu, skladá sa z používateľského a funkcionálneho.

Hlavné okno (Domov)

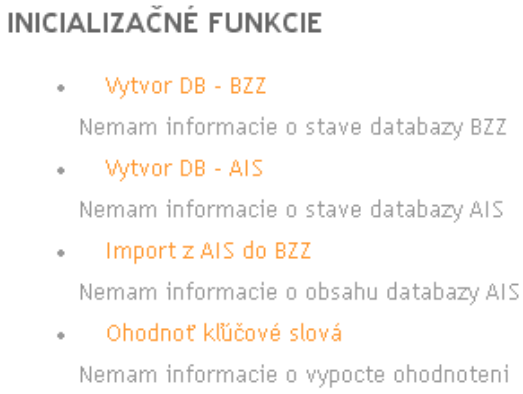
V strednej časti hlavného okna sa nachádzajú 4 funkcie (Obr.9). Prvou je vytvorenie databázy BZZ. Ako sme spomínali v kapitole *6.1 Databázový systém PostgreSQL*, databázy Bzz aj Ais boli vytvorené v prostredí pgAdminIII.

Vytvor DB - BZZ teda slúži na vytvorenie tabuliek a ich naplnenie údajmi.

Vytvor DB - AIS slúži na vytvorenie databázy Ais, do ktorej sú zároveň uložené potrebné údaje o študentoch. Databáza Ais je zredukovanou a nami upravenou verziou databázy AIS (Akademický informačný systém používaný na STU).

Import z AIS do BZZ slúži na importovanie potrebných údajov o študentovi z databázy Ais do Bzz. Známky z Ais do Bzz sa neprenášajú, slúžia len vo výpočte hodnoty pomocou kľúčových slov.

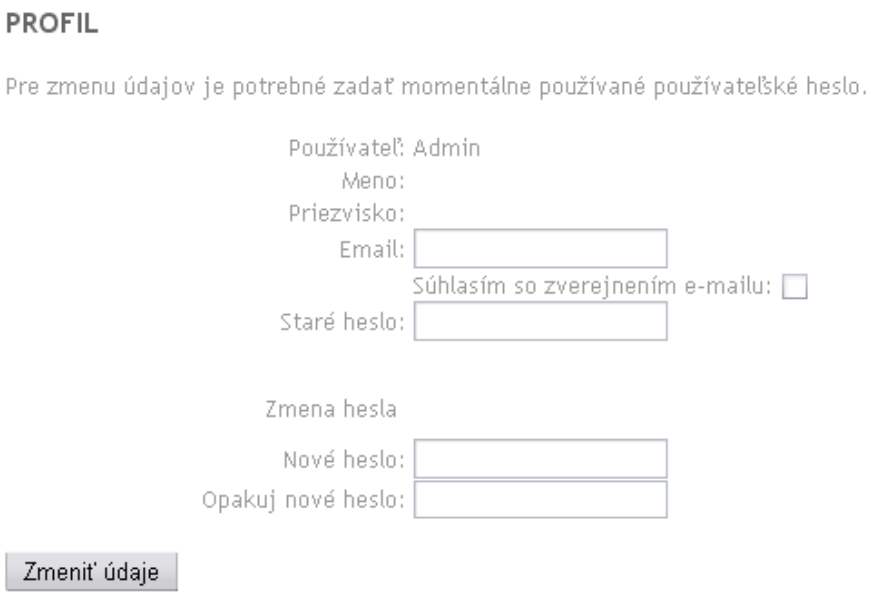
Ohodnoť kľúčové slová slúži na ohodnotenie študentov pomocou vyhodnocovacej funkcie, ktorá zoberie známku študenta z predmetov a vyhodnotí ju podľa kľúčových slov. Vypočítanú hodnotu zapíše do databázy Bzz.



Obr.9: Inicializačné funkcie administrátora informačného systému BZZ.

Profil

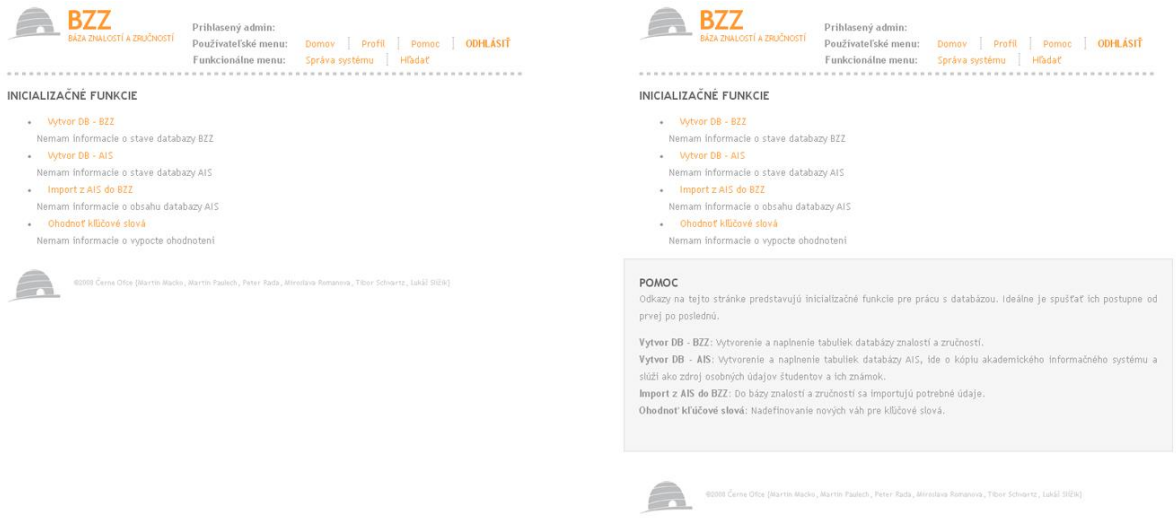
Slúži na zobrazenie informácií o práve prihlásenom používateľovi. Je tu napríklad aj možnosť zmeniť svoje prihlasovacie heslo (Obr.10).



Obr.10: Profil administrátora.

Pomoc

Slúži na zobrazenie pomocných informácií, týkajúcich sa práve zobrazených funkcií systému. Po kliknutí na Pomoc sa tieto informácie zobrazia v šedom rámčeku (Obr.11).



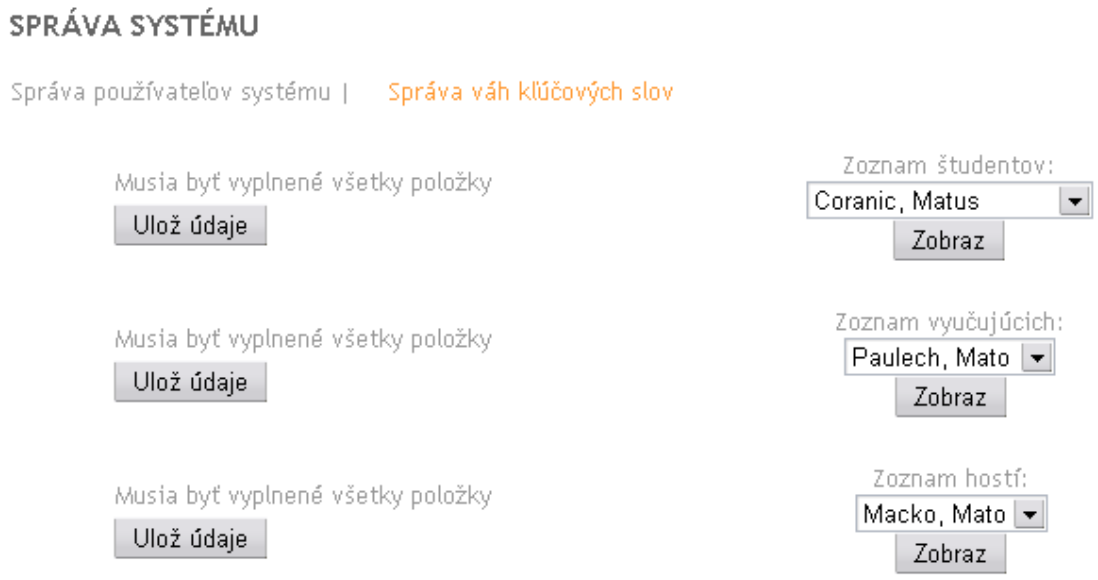
Obr.11: Zobrazenie pomoci sa vykonáva v rámci toho istého okna. Po kliknutí na Pomoc sa zobrazí šedý rámček (*vpravo*).

Správa systému

Tu sú zahrnuté funkcie pre správu používateľov systému BZZ a pre správu váh kľúčových slov.

Pri správe používateľov je možné zobraziť študentov, profesorov alebo hostí systému spolu s ich osobnými údajmi (Obr.12). Tieto údaje môže administrátor editovať.

V správe váh kľúčových slov má administrátor možnosť editovať jednotlivé váhy. Kvôli prehľadnosti sme použili kategorizované zobrazenie v podobe stromu (Obr.13). Jednotlivé kategórie (označené symbolom +) je možné kliknutím na ne rozbaľovať a zobrazovať tak ďalšie úrovne (podkategórie). Po kliknutí na tlačítko **Zobraz váhy** sa zobrazí tabuľka s vybranými váhami a tie je tu možné meniť(Obr.14). Každé kľúčové slovo je ohodnotiené stupňom od 1 (najhoršie) po 5 (najlepšie).



Obr.12: Správa používateľov systému.

Obr.13: Správa váh kľúčových slov, *vľavo* vidíme možnosť zvoliť úpravu vzhľadom na rodičov a potomkov alebo vzhľadom na predmet. *Vpravo* je strom s rozbalenými niektorými kategóriami.

Obr.14: Editovanie váh kľúčových slov.

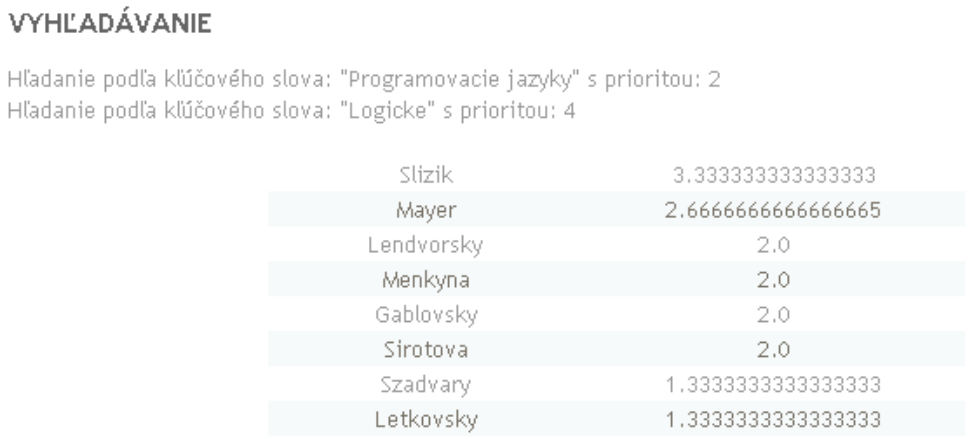
Hľadať

Jednou z najdôležitejších funkcií systému BZZ je vyhľadávanie študentov podľa zadaných kritérií. Pri vyhľadávaní si používateľ najskôr zvolí kritériá, ktoré má hľadaný

študent spĺňať. V našom systéme sa kritéria zobrazia v podobe stromu, v ktorom sú načítané jednotlivé kategórie kľúčových slov. Strom sa prehľadáva veľmi intuitívne, po kliknutí na kategóriu, pri ktorom je zobrazená ikona +, sa označená kategória rozbalí (Obr.15).

Používateľ si požadované kľúčové slová označí a zadá váhy jednotlivých slov (1 je menej podstatná znalosť, 5 je najpodstatnejšia). Následne po potvrdení sa mu zobrazí výstup v podobe tabuľky, v ktorej sú zoradení relevantní študenti (Obr.16).

Obr.15: Vyhľadávanie študentov, v strome si zvolím jednotlivé váhy kľúčových slov.



Obr.16: Vyhľadaní študenti podľa kľúčových slov.

D.3.1.2 Študent

Po prihlásení sa študenta do systému sa zobrazí okno ako na Obr.17. V hornej časti sa nachádza menu, skladajúce sa z dvoch častí. V jednej sa nachádzajú používateľské funkcie, v druhej funkcionálne.

V používateľskom menu sa nachádzajú odkazy na zobrazenie hlavnej stránky, na zobrazenie profilu s osobnými údajmi, na zobrazenie pomocných informácií a odhlásenie zo systému. Z funkcionálneho menu sa môžeme dostať k funkcii editovania znalostí a zručností študenta (sebahodnoteniu) a vyhľadávaniu iných študentov.

Obr.17: Študentovi je ponúknuté takéto menu.

Profil

Slúži na zobrazenie informácií o práve prihlásenom používateľovi. Je tu napríklad aj možnosť zmeniť svoje prihlasovacie heslo (Obr.10).

Pomoc

Slúži na zobrazenie pomocných informácií. Po kliknutí na Pomoc sa tieto informácie zobrazia v šedom rámčeku (Obr.11).

Sebahodnotenie

Jednotlivé znalosti a zručnosti prihláseného študenta sa nastavujú podobným štýlom ako je nastavovanie váh pre kľúčové slová (Obr.15). To znamená formou stromu. Každé kľúčové slovo reprezentujúce znalosť je možné ohodnotiť stupňom od 1 (najhoršie) po 5 (najlepšie).

Hľadať

Používateľ si najskôr zvolí kritériá, ktoré má hľadaný študent spĺňať. V našom systéme sa kritéria zobrazia v podobe stromu, v ktorom sú načítané jednotlivé kategórie kľúčových slov. Po kliknutí na kategóriu, pri ktorom je zobrazená ikona + (to znamená, že daná kategória obsahuje podkategórie), sa označená kategória rozbalí (Obr.15).

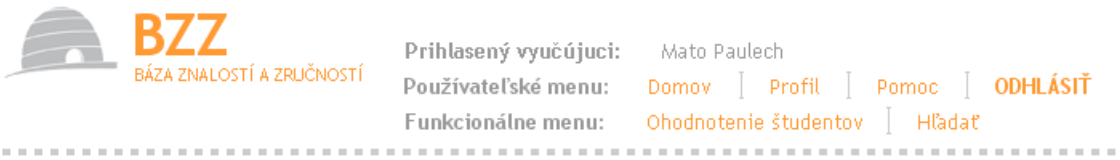
Používateľ si požadované kľúčové slová označí a zadá váhy jednotlivých slov (1 je menej podstatná znalosť, 5 je najpodstatnejšia). Následne po potvrdení sa mu zobrazí výstup v podobe tabuľky, v ktorej sú zoradení relevantní študenti (Obr.16).

D.3.1.3 Pedagóg

Po prihlásení sa pedagóga do systému sa zobrazí okno ako na Obr.18. V hornej časti sa nachádza menu, skladajúce sa z dvoch častí. V jednej sa nachádzajú používateľské funkcie, v druhej funkcionálne.

V používateľskom menu sa nachádzajú odkazy na zobrazenie hlavnej stránky, na zobrazenie profilu s osobnými údajmi, na zobrazenie pomocných informácií a odhlásenie zo systému. Z funkcionálneho menu sa môžeme dostať k funkcii ohodnoteniu študentov a vyhľadávaniu iných študentov. Ohodnotenie študentov slúži na dodatočné ohodnotenie znalostí študentov, ktorí napríklad nedosiahli vynikajúce výsledky, ale napriek tomu

preukázali počas semestra dostatočné znalosti. Takto sa im zvýši výsledný index pre túto znalosť.



Obr.18: Hlavné menu prihláseného pedagóga.

Profil

Slúži na zobrazenie informácií o práve prihlásenom používateľovi. Je tu napríklad aj možnosť zmeniť svoje prihlasovacie heslo (Obr.10).

Pomoc

Slúži na zobrazenie pomocných informácií. Po kliknutí na Pomoc sa tieto informácie zobrazia v šedom rámčeku (Obr.11).

Ohodnotenie študenta

Prihlásený pedagóg najskôr zo zoznamu vyberie študenta, ktorému hodlá vylepšiť hodnotenie. Jednotlivé znalosti a zručnosti prihláseného študenta sa potom nastavujú v strome, podobne ako na Obr.15. Každé kľúčové slovo reprezentujúce znalosť je možné ohodnotiť stupňom od 1 (najhoršie) po 5 (najlepšie).

D.3.1.4 Hosť

V role hosťa vystupujú rôzne externé spoločnosti, ktorých jedinou funkciou je vyhľadávanie študentov. Dôvodom môžu byť napríklad rôzne pracovné ponuky. Pri vyhľadávaní sa však v rámci ochrany osobných údajov nezverejnia žiadne dodatočné osobné údaje.

Príloha E – Zoznam používateľov systému na testovacie účely

V tabuľke sú vypísaní všetci zadefinovaní používatelia systému BZZ a ich prihlasovacie mená. Upozorňujeme, že je potrebné brať do úvahy veľké a malé písmená.

Typ používateľa	Login	Heslo
administrátor	admin	admin
študent	Coranic04	8601011111
študent	Danilova04	8601011111
študent	Gablovsky04	8601011111
študent	Koscak04	8601011111
študent	Lendvorsky04	8601011111
študent	Letkovsky04	8601011111
študent	Macko04	8601011111
študent	Mayer04	8601011111
študent	Menkyna04	8601011111
študent	Molnar04	8601011111
študent	Nagy04	8601011111
študent	Ochotnicky04	8601011111
študent	Paroulek04	8601011111
študent	Paulech04	8601011111
študent	Pavlovcin04	8601011111
študent	Pokorny04	8601011111
študent	Polorecka04	8601011111
študent	Rada04	8601011111
študent	Romanova04	8601011111
študent	Schvartz04	8601011111
študent	Sirotova04	8601011111
študent	Slizik04	8601011111
študent	Szadvary04	8601011111
pedagóg	pedagog	pedagog
pedagóg	Povazanova	anicka
pedagóg	Simakova	valika
hosť	guest	guest