

Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

Báza znalostí a zručností študentov

Dokumentácia k inžinierskemu dielu



Tím č. 9

Vedúci projektu: RNDr. Valéria Šimáková

Predmet: Tvorba softvérového systému v tíme

Študijný program: Softvérové inžinierstvo

Ak. rok: 2008/2009

Bc. Bálint Farkas

Bc. Michal Holub

Bc. Juraj Kollár

Bc. Vojtech Villarís

Bc. Martin Virík

Obsah

ÚVOD	I
Účel dokumentu	i
Prehľad dokumentu	i
Zadanie projektu.....	i
Skratky	ii
Použitá notácia	iii
1 OPIS RIEŠENÉHO PROBLÉMU	1
1.1 Prehľad problémovej oblasti	1
1.2 Ciele projektu	1
1.3 Slovník pojmov problémovej oblasti	2
2 BIZNIS ANALÝZA	3
2.1 Tím Černé ofce.....	3
2.1.1 Identifikované roly v systéme	3
2.1.2 Údaje evidované systémom a spôsob ich získania	3
2.1.3 Funkcie systému.....	4
2.1.4 Architektúra systému	4
2.1.5 Zhodnotenie	5
2.2 Tím elf	5
2.2.1 Identifikované roly v systéme	5
2.2.2 Údaje evidované systémom a spôsob ich získania	5
2.2.3 Funkcie systému.....	6
2.2.4 Architektúra systému	6
2.2.5 Vlastnosti systému	7
2.2.6 Zhodnotenie	7
2.3 Analýza modelu používateľa	8
2.4 Analýza informačných systémov používaných na FIIT.....	8
2.4.1 Akademický informačný systém.....	8
2.4.2 Yonban.....	8
2.5 Zmena váh.....	9
2.6 Analýza faktov	9
3 ŠPECIFIKÁCIA POŽIADAVIEK NA INFORMAČNÝ SYSTÉM	10
3.1 Funkcionálne požiadavky na systém.....	10
3.2 Ďalšie požiadavky na systém	10
3.3 Prepojenie cieľov a požiadaviek	10
3.4 Charakteristika rolí.....	11
3.5 Diagramy prípadov použitia.....	11
3.5.1 Prípady použitia pre rolu študent	11
3.5.2 Prípady použitia pre rolu pedagóg	17
3.5.3 Prípady použitia pre rolu administrátor.....	20
3.5.4 Prípady použitia pre rolu externý používateľ	21
3.5.5 Prípady použitia pre rolu konfigurátor	21
3.6 Logický model údajov	25
4 HRUBÝ NÁVRH RIEŠENIA	27
4.1 Aplikácia na báze JEE.....	27
4.2 Prezentačná vrstva.....	27
4.3 Aplikačná vrstva	28
4.4 Objektovo-relačné mapovanie	28
4.5 Hodnotové objekty.....	29
4.6 Servlet kontajner	29
4.7 Moduly systému.....	29

5	PROTOTYP	31
5.1	Ciele prototypovania	31
5.2	Vybrané časti systému na prototypovanie.....	31
5.2.1	Scenár použitia systému realizovaný prototypom	31
5.2.2	Prípady použitia vybrané na prototypovanie	31
5.3	Aplikačná vrstva	32
5.3.1	FaktService	33
5.3.2	PouzivatelService.....	33
5.3.3	ZnalostService.....	33
5.4	Prezentačná vrstva.....	33
5.5	Testovanie	33
5.6	Dosiahnuté výsledky	34
	POUŽITÉ ZDROJE	35

Zoznam obrázkov

Obr. 1.1 Diagram biznis cieľov	1
Obr. 2.1 Diagram architektúry systému tímu Černé ofce [3]	4
Obr. 2.2 Diagram architektúry systému tímu _elf_ [2]	7
Obr. 2.3 Diagram faktov a ich atribútov	9
Obr. 3.1 Diagram prípadov použitia pre rolu študent	11
Obr. 3.2 Sekvenčný diagram pre prípad použitia UC03 – Pridanie faktu	14
Obr. 3.3 Diagram prípadov použitia pre rolu pedagóg	17
Obr. 3.4 Sekvenčný diagram pre prípad použitia UC10 – Vyhľadávanie študentov	19
Obr. 3.5 Diagram prípadov použitia pre rolu administrátor	20
Obr. 3.6 Diagram prípadov použitia pre rolu externý používateľ	21
Obr. 3.7 Diagram prípadov použitia pre rolu konfigurátor	22
Obr. 3.8 Sekvenčný diagram pre prípad použitia UC14 – Konfigurácia faktu	23
Obr. 3.9 Logický model údajov systému Znalec	25
Obr. 4.1 Návrh architektúry systému Znalec	27
Obr. 4.2 Diagram komponentov opisujúci moduly systému Znalec	29
Obr. 5.1 Diagram tried aplikačnej vrstvy	32

Zoznam tabuliek

Tab. 1 Použité skratky	ii
Tab. 2 Notácia pre diagram cieľov	iii
Tab. 3 Notácia pre diagram prípadov použitia	iii
Tab. 4 Notácia pre diagram logického modelu údajov	iii
Tab. 5 Notácia pre diagram tried	iv
Tab. 6 Notácia pre diagram tried	iv
Tab. 3.1 Prípad použitia UC01 – Správa osobného profilu	12
Tab. 3.2 Prípad použitia UC02 – Správa faktov	12
Tab. 3.3 Prípad použitia UC03 – Pridanie faktu	12
Tab. 3.4 Prípad použitia UC04 – Požiadavka na vytvorenie faktu	14
Tab. 3.5 Prípad použitia UC05 – Zobrazenie znalostí študenta	15
Tab. 3.6 Prípad použitia UC06 – Vyhodnotenie znalostí študenta	15
Tab. 3.7 Prípad použitia UC07 – Zobrazenie profilu študenta	16
Tab. 3.8 Prípad použitia UC08 – Zobrazenie faktov študenta	16
Tab. 3.9 Prípad použitia UC09 – Vytvorenie filtra pre vyhľadávanie	17
Tab. 3.10 Prípad použitia UC10 – Vyhľadávanie študentov	18
Tab. 3.11 Prípad použitia UC11 – Vyhľadávanie študentov podľa preddefinovaného filtra	19
Tab. 3.12 Prípad použitia UC12 – Správa používateľských účtov	20
Tab. 3.13 Prípad použitia UC13 – Import údajov z DB AISu	21
Tab. 3.14 Prípad použitia UC14 – Konfigurácia faktu	22
Tab. 3.15 Prípad použitia UC15 – Vytvorenie atribútu	23
Tab. 3.16 Prípad použitia UC16 – Konfigurácia vzorca	24
Tab. 3.17 Prípad použitia UC17 – Konfigurácia znalosti	24
Tab. 3.18 Prípad použitia UC18 – Konfigurácia kľúčových slov	24

Úvod

Účel dokumentu

Predkladaný dokument predstavuje technickú dokumentáciu k softvérovému systému Znalec, ktorý je výsledkom študentského projektu v predmete Tvorba softvérového systému v tíme na tému Báza znalostí a zručností študentov.

Prehľad dokumentu

Opis riešeného problému sa nachádza v kapitole 1. Obsahuje základný prehľad problémovej oblasti a ciele projektu. Pojednáva aj o postupe riešenia pri naplňaní jednotlivých cieľov.

Druhá kapitola obsahuje biznis analýzu. V tejto kapitole predstavujeme podklady, z ktorých pri riešení projektu vychádzame. Analyzujeme tu výsledky, ktoré tímy pri riešení tejto témy dosiahli v minulých rokoch. Ďalej analyzujeme všeobecné trendy pojednávaním o zaujímavých prácach zasahujúcich do oblasti znalostí a modelov používateľa. Takisto ponúkame stručný prehľad fakultných systémov, z ktorých by sme mohli čerpať údaje pre náš systém. Následne sa zamýšľame nad rôznymi typmi údajov a ich atribútmi, ktoré môžu ovplyvniť znalosti používateľa.

V kapitole 3 uvádzame špecifikáciu požiadaviek na vytváraný softvérový systém. Tá obsahuje špecifikáciu funkcií systému spolu s určením ich priorit, špecifikáciu správania systému a špecifikáciu údajov v systéme. Funkcie systému sú tu špecifikované ako jednotlivé prípady použitia. K vybraným prípadom použitia uvádzame aj sekvenčné diagramy na presnejší opis správania systému. Údaje v systéme sú špecifikované uvedením logického modelu údajov. V tejto kapitole tiež uvádzame špecifikáciu nefunkcionálnych požiadaviek.

Štvrtá kapitola obsahuje hrubý návrh architektúry systému. Návrh je spracovaný v diagrame zobrazujúcom jednotlivé moduly systému. Ich funkcionálna je následne opísaná slovnou.

V piatej kapitole sa venujeme prototypovaniu. Opisujeme tu ciele prototypovania a vybrané časti systému, ktoré sme sa rozhodli zahrnúť do prototypu. Následne uvádzame dosiahnuté výsledky.

Zadanie projektu

Obdobie štúdia na vysokej škole je zamerané na formovanie profesijného profilu, charakteru, zmyslu pre povinnosť a pre spoluprácu v tíme. Charakter a rozsah projektov, či už je to v komerčnej oblasti alebo vo vede a výskume, ukazuje, že schopnosť spolupráce je v mnohom rozhodujúca pre úspešnosť projektov, pracovných tímov v praxi. Spolupráca predpokladá určitú vyváženosť vedomostí, zručností a vzájomnej ústretovosti. Čas štúdia na vysokej škole dáva možnosť tieto schopnosti vysledovať a využiť pri výbere študentov do projektov, pri odporúčaní na študijné stáže a podobne. Tieto informácie sú dôležité a užitočné pre učiteľov, ale aj pre študentov, napríklad pri vytváraní tímov na študentské súťaže, projekty, stáže.

Úlohou tímov bude navrhnuť novú architektúru s využitím moderných technológií.

Požiadavky na výsledný produkt:

- Spolupráca so systémami používanými na FIIT (AIS, YONBAN).
- Získavanie a ukladanie znalostí o znalostiach študentov do bázy znalostí a informácií.
- Manipulácia so znalosťami, ich indexovanie a vyhľadávanie.
- Používateľský príjemná prezentácia informácií o znalostiach. Personifikovaný prístup k informáciám (učiteľ, študent...).
- Akceptácia externých informačných entít.


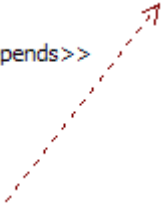
Skratky

Tab. 1 Použité skratky





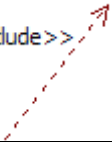

SKRATKA	VÝZNAM
AIS	Akademický informačný systém
API	Application Programming Interface
CRUD	Create, Read, Update, Delete (vytvorenie, načítanie, aktualizácia, vymazanie)
DAO	Data Access Object
DB	Databáza
FIIT	Fakulta informatiky a informačných technológií
HQL	Hibernate Query Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IoC	Inversion of Control
JEE	Java Enterprise Edition
Obr.	Obrázok
ORM	Objektovo-relačné mapovanie
OWL	Web Ontology Language
SQL	Structured Query Language
Tab.	Tabuľka
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

Použitá notácia




Tab. 2 Notácia pre diagram cieľov

 Vytvorenie bázy znalostí	Biznis cieľ
 <<depends>>	Asociácia vyjadrujúca závislosť

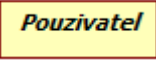

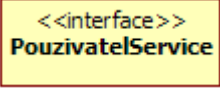


Tab. 3 Notácia pre diagram prípadov použitia

 Študent	Hráč
 UC01 Správa osobného profilu profilu	Prípad použitia
	Neorientovaná asociácia
	Orientovaná asociácia
 <<include>>	Asociácia vyjadrujúca zahrnutie jedného prípadu použitia v druhom
	Asociácia vyjadrujúca zovšeobecnenie

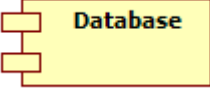



Tab. 4 Notácia pre diagram logického modelu údajov

	Entita logického modelu s niektorými atribútmi
	Neorientovaná asociácia medzi entitami
	Asociácia vyjadrujúca slabú agregáciu vrátane kardinalít

Tab. 5 Notácia pre diagram tried

	Abstraktná trieda
	Trieda
	Rozhranie
	Asociácia vyjadrujúca slabú agregáciu
	Asociácia vyjadrujúca vzťah dedenia

Tab. 6 Notácia pre diagram tried

	Komponent systému
	Rozhranie
	Asociácia vyjadrujúca závislosť
	Asociácia vyjadrujúca tok riadenia

1 Opis riešeného problému

Opis riešeného problému vychádza zo zadania uvedeného v úvode.

1.1 Prehľad problémovej oblasti

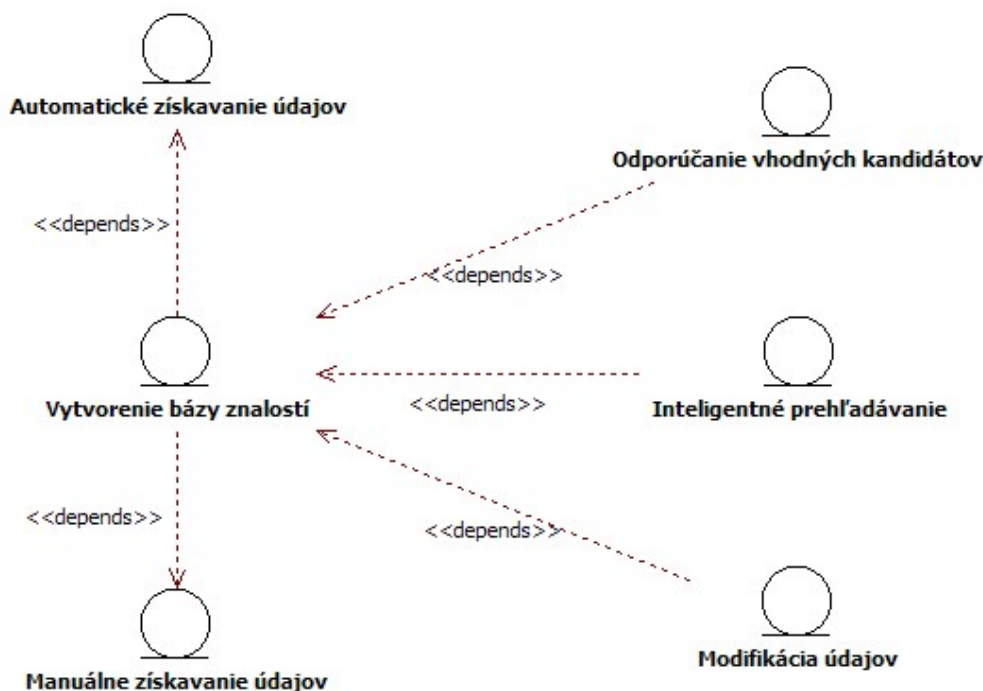
Fakulta informatiky a informačných technológií (FIIT) dostáva rozličné ponuky na účasť na stážach, konferenciách prípadne na spoluprácu na rozličných projektoch. Tieto ponuky sú často určené aj pre študentov. FIIT využíva na uchovávanie informácií o študentoch prevažne celouniverzitný systém AIS. V ňom sa nachádza množstvo údajov o jednotlivých študentoch, no medzi údajmi nie je zrejma súvislosť. AIS nedokáže z jednotlivých údajov sformovať znalosti daného študenta. Ten si navyše nemôže upravovať svoj osobný profil pridávaním informácií o absolvovaných seminároch, školeniach, praxi, ktoré vylepšujú jeho znalosti v jednotlivých odboroch. Niektoré údaje o študentoch sú uložené aj v iných systémoch, ktoré fakulta používa, no je snaha previesť všetky tieto údaje do AISu.

Nakoľko na FIIT študuje veľké množstvo študentov, nie je v súčasnosti možné efektívne odporučiť vhodného študenta na odpovedanie na ponuku. Poverený pracovník tak musí prejsť množstvo údajov o študentoch a manuálne v nich hľadať tých, ktorí by mohli mať predpoklady a potenciálny záujem o danú ponuku. Pri tom stratí veľa času a môže sa stať, že niektorých vhodných kandidátov prehliadne.

1.2 Ciele projektu

Účelom nami vytváraného informačného systému je zjednodušenie a zefektívnenie procesu vyhľadania vhodných študentov, ktorí by mohli reagovať na ponuku. Z tohto dôvodu je nutné vytvoriť bazu znalostí a zručností študentov, a pravidelne ju obnovovať.

Pre analyzovaný informačný systém sme identifikovali tieto biznis ciele: vytvorenie bázy znalostí, manuálne a automatické získavanie údajov, odporúčanie vhodných kandidátov, inteligentné prehľadávanie, modifikácia údajov. Ich vzájomné vzťahy sú zobrazené na Obr. 1.1.



Obr. 1.1 Diagram biznis cieľov

1.3 Slovník pojmov problémovej oblasti

V tejto časti uvádzame vysvetlenie hlavných pojmov, ktoré používame v projekte.

Znalosť je súbor faktov a informácií, získaných skúsenosťami alebo štúdiom, poprepájaných vzťahmi, ktoré navzájom prinášajú známosť, vedomosť v oblasti poznávania^{1 2}.

Zručnosť je preukázaná osobná vlastnosť alebo preukázaná schopnosť aplikovať nadobudnuté znalosti pri praktickom riešení úloh³.

V našom systéme je potrebné dôsledne rozlišovať pojmy znalosť a zručnosť, hoci môžu byť vyjadrené rovnakými kľúčovými slovami, a teda sa na prvý pohľad javiť ako totožné. Znalosť chápeme ako zloženú vedomosť v oblasti poznania, ktorá sa skladá zo zručností. Ako príklad uvedieme znalosť jazyka Java. K tejto znalosti môžu prispieť aj zručnosti vyjadrené kľúčovými slovami Eclipse alebo Net Beans. Samotné kľúčové slovo Java však môže v inom kontexte predstavovať zručnosť. Tu sa jedná o elementárnu schopnosť. Zručnosť Java môže spolu so zručnosťou UML prispievať k znalosti objektovo-orientovaného programovania.

V systéme ďalej používame výraz fakt. Pod týmto pojmom rozumieme určitú vzdelávaciu jednotku, ktorá pomohla študentovi nadobudnúť zručnosti, ktoré sa následne premietnu do jeho znalostí. Faktom môže byť napr. absolvované školenie o programovaní v Jave. Študent na ňom mohol získať zručnosti Eclipse a Java, ktoré prispejú k jeho znalosti OOP.

Každý fakt má svoje atribúty. Pre uvedený príklad faktu školenie môžu byť atribúty názov školenia, dĺžka trvania, forma ukončenia.

Posledným dôležitým pojmom je hodnota atribútu faktu. Rozumieme pod ním konkrétne hodnoty jednotlivých atribútov faktov. V uvedenom príklade faktu školenie môže mať atribút dĺžka trvania hodnotu napr. 1 týždeň.

¹ <http://www.thefreedictionary.com>

² <http://www.translationbureau.gc.ca/index.php>

³ <http://www.ism3.com/index.php>

2 Biznis analýza

V tejto kapitole analyzujeme ako projekty zaoberajúce sa podobnou problematikou, tak aj postupy, ktoré by sme mohli uplatniť pri vytváraní nášho riešenia. Najprv uvádzame analýzu výsledkov tímov zaoberajúcich sa tvorbou bázy znalostí a zručností z minulých rokov.

2.1 Tím Černé ofce

V školskom roku 2007/2008 riešil tému Bába znalostí a zručností študentov tím č. 10 s názvom Černé ofce. Tento tím vytvoril systém s nasledujúcimi vlastnosťami [3].

2.1.1 Identifikované roly v systéme

Tím Černé ofce identifikoval nasledujúce roly:

- **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 bežnej prevádzky.
- **Externá osoba** – Využíva znalosti poskytované systémom po súhlase študenta.

2.1.2 Údaje evidované systémom a spôsob ich získania

Systém navrhnutý tímom Černé ofce eviduje nasledovné údaje:

- používateľ systému,
 - študent
 - pedagóg
 - administrátor
 - hosť
- predmety,
- kľúčové slová,
- váhy,
- hodnotenie študenta pedagógom a
- študentovo vlastné hodnotenie.

Informácie o študentoch sú importované z Akademického informačného systému, ostatné skupiny používateľov sú do systému pridané ručne. Predmety sú importované tiež z Akademického informačného systému. Sebahodnotenie vyplní prihlásený používateľ (študent). Obdobne, prihlásený pedagóg môže vyplniť hodnotenie jednotlivým študentom.

Samotný logický model databázy je jednoduchý a prehľadný. Je však otázne, do akej miery by ho bolo možné ďalej normalizovať. Bolo by napríklad možné zahrnúť do modelu databázy aj váhy sebahodnotenia a hodnotenia učiteľov. V dokumentácii sa nachádza zmienka o možnosti ďalšieho rozširovania hodnotení o započítavanie rôznych diplomov, súťaží, atď. Databázový model v existujúcej podobe je však málo flexibilný a aj algoritmy vyhodnocovania sú príliš viazané na jednotlivé entity.

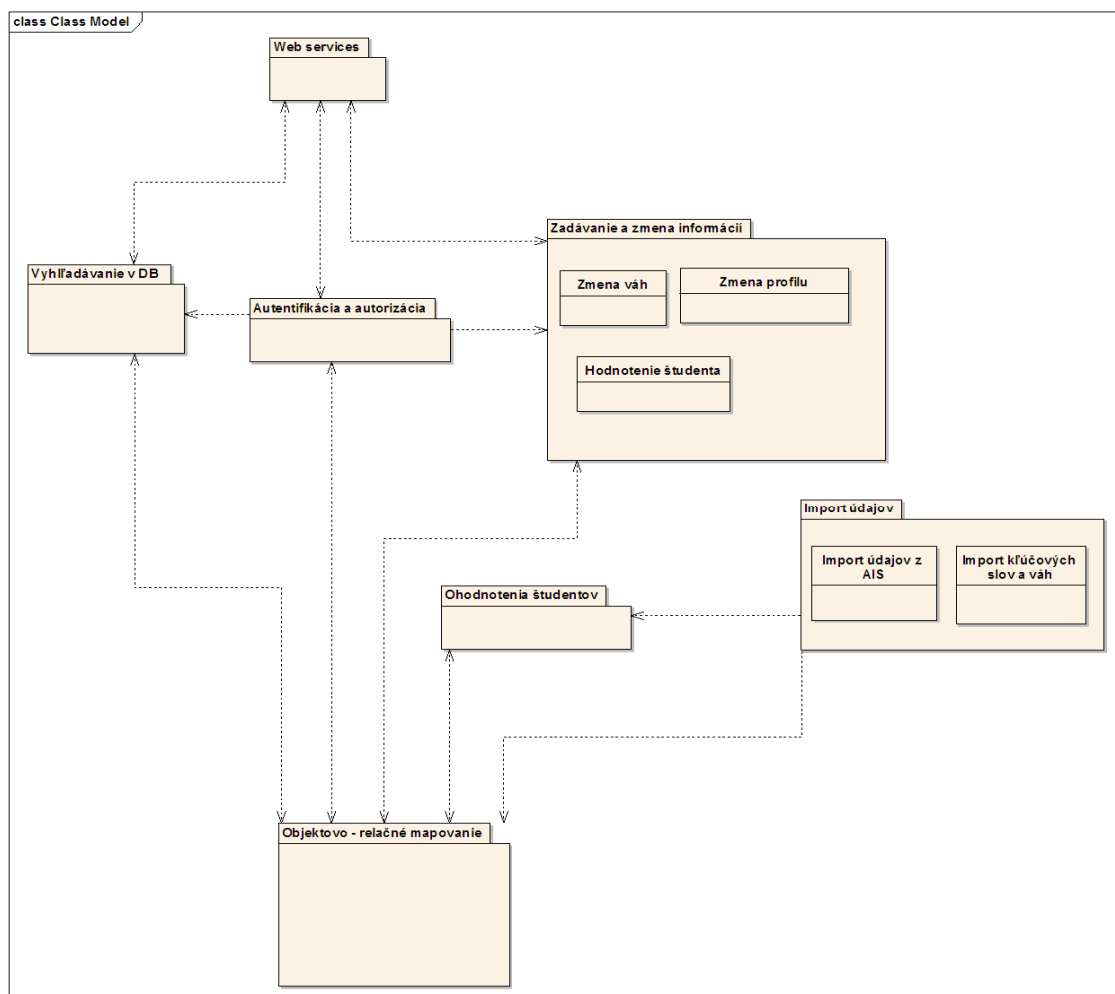
2.1.3 Funkcie systému

System, ktorý tím Černé ofce vytvoril, má nasledovné funkcie:

- Poskytnutie súhlasu pre tretie strany – rozhodnutia študenta zverejniť informácie o sebe aj tretej strane. Túto funkciu môže používať len študent.
- Zadávanie informácií o študentovi – vloženie informácií týkajúcich sa znalostí, zručností a študijných výsledkov študenta. Túto funkciu môže používať ako študent, tak aj pedagogický pracovník.
- Vyhľadanie študenta/študentov – systém nájde študenta/študentov na základe zadaných kritérií hľadania. Túto funkciu môže používať študent, pedagogický pracovník, ako aj externá osoba.
- Kontaktovanie študenta – spojenie sa so študentom využitím poskytnutého kontaktu (e-mail, tel. č., ICQ, ...). Túto funkciu môžu použiť všetci používatelia systému.
- Správa servera – zabezpečenie správneho chodu servera, rôzne nastavenia. Túto funkciu môže používať iba administrátor.
- Zmeny v databáze – uskutočnenie nevyhnutných zmien v databáze v prípade potreby. Ako predošlú funkciu, aj túto môže používať iba administrátor.

2.1.4 Architektúra systému

Na Obr. 2.1 je zobrazený diagram architektúry systému tímu Černé ofce.



Obr. 2.1 Diagram architektúry systému tímu Černé ofce [3]

Používateľ prístupuje 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. Zistí sa tiež, aké práva daný používateľ má. Následne, podľa prístupových práv, bude môcť používateľ používať funkcie systému.

Na implementáciu a chod systému použil tím Černé ofce nasledovné nástroje a technológie:

- aplikačný rámec Apache,
- databázový server PostgreSQL,
- vývojové prostredie NetBeans,
- objektovo-relačný mapovač Hibernate a
- Java Server Pages (JSP).

2.1.5 Zhodnotenie

Tím Černé ofce navrhol pomerne jednoduchý systém, ktorý však v dostatočnej miere umožňuje poskytovanie informácií o výsledkoch študentov. Tento systém následne úspešne implementovali. Nedostatok vidíme v nízkej flexibilitate systému, čo môže spôsobovať problémy pri snahe o rozšírenie systému. Systém je tiež úzko zameraný a je v ňom málo možností nastavenia.

2.2 Tím _elf_

V školskom roku 2005/2006 riešil tému Bába znalostí a zručností študentov tím č. 11 s názvom _elf_. Tento tím vďaka skúsenostiam z praxe úspešne vyhodnotil existujúce systémy v danej oblasti. Na základe týchto poznatkov navrhol a implementoval vlastný systém s nasledujúcimi vlastnosťami [2].

2.2.1 Identifikované roly v systéme

Tím _elf_ identifikoval nasledovné roly:

- **Študent** – Zadáva do systému informácie o vlastných znalostiach a zručnostiach.
- **Vyučujúci** – Zadáva do systému informácie o študentoch, požaduje od systému znalosti o študentoch.
- **Externý systém** – Využíva znalosti poskytnuté navrhovaným systémom na ich ďalšie spracovanie alebo priamu prezentáciu, alebo poskytuje informácie o študentovi (ako sú známky, vážený študijný priemer, a pod.).
- **Administrátor** – Spravuje systém a má na starosti jeho správny chod počas rutínnej prevádzky.
- **Konfigurátor** – Nastavuje parametre systému, určuje typy informácií prístupné používateľom.

2.2.2 Údaje evidované systémom a spôsob ich získania

Systém navrhnutý tímom _elf_ eviduje nasledovné údaje:

- používatelia systému a ich roly,
- študenti a učitelia,
- predmety,
- kľúčové slová,
- znalosti – znalosti, zručnosti, certifikáty, známky z predmetov, hodnotenia od učiteľov,
- konfiguračné parametre systému a
- balíčky – preddefinované kritériá vyhľadávania.

Údaje o študentoch, učiteľoch a predmetoch získava systém importom dát zo systému ŠTUDENT, ako aj zo vstupov od používateľov. Ostatné údaje sa do systému dostávajú ako vstupy od rôznych typov používateľov, teda nie len od študentov, ale aj od učiteľov, administrátorov, konfiguratorov systému a od externých systémov.

Logická štruktúra uložených údajov je vhodná. Je navrhnutá tak, aby sa systém dal dostatočne konfigurovať a prispôbovať, a aby sa s ním dalo efektívne pracovať. Umožňuje napríklad definovať tzv. balíčky, ktoré by sa dali charakterizovať ako vopred definované kritéria pre vyhľadávanie študentov. Existujú globálne balíčky, ktoré môžu znovu použiť učitelia, prípadne si ich môžu modifikovať a uložiť do svojich vlastných preddefinovaných balíčkov.

Údaje, ktoré sa týkajú znalostí, sú váhované, vďaka čomu sa dá lepšie opísať miera danej znalosti. Umožňuje to aj presnejšie vyhľadávanie.

2.2.3 Funkcie systému

Role **študent** poskytuje systém nasledovnú funkcionality:

- poskytovanie detailných informácií o svojom profile,
- zobrazenie informácií o predmetoch,
- zobrazenie znalostí, zručností, atď. uchovávaných v systéme,
- zmena osobného profilu a
- notifikácia študenta, aby si obnovil svoj profil.

Role **vyučujúci** poskytuje systém nasledovnú funkcionality:

- zobrazenie informácií o predmetoch,
- zobrazenie znalostí, zručností, atď. uchovávaných v systéme,
- prehliadanie informácií o študentoch,
- správa osobných balíčkov a
- vyhľadávanie študentov na základe balíčkov.

Role **administrátor** poskytuje systém nasledovnú funkcionality:

- import údajov zo systému ŠTUDENT,
- správa používateľov systému a
- poslanie notifikácie študentom, aby si obnovili profil.

Role **konfigurátor** poskytuje systém nasledovnú funkcionality:

- správa globálnych balíčkov,
- správa zoznamu zručností, znalostí, certifikátov a
- konfigurácia interných nastavení systému.

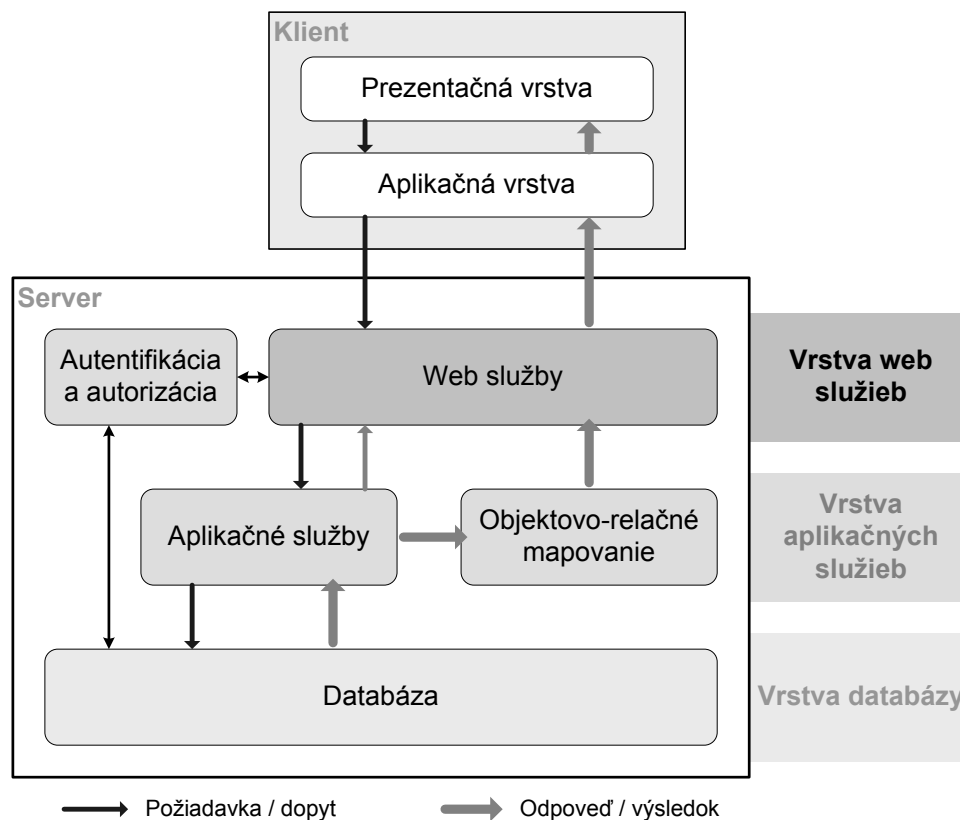
Role **externý systém** poskytuje systém nasledovnú funkcionality:

- správa balíčkov,
- poskytnutie známky a
- poskytnutie indexovaného zoznamu študentov.

2.2.4 Architektúra systému

Architektúra systému je postavená na báze klient – server. Vonkajšie rozhranie je vytvorené pomocou webových služieb. Ďalej je systém rozdelený na jednotlivé vrstvy. Klientskú časť tvoria prezentačná a aplikačná vrstva. Serverová časť systému má na starosti webové služby, autentifikáciu a autorizáciu,

aplikačné služby, objektovo-relačné mapovanie a prístup do databázy. Architektúra tohto systému je zobrazená na Obr. 2.2.



Obr. 2.2 Diagram architektúry systému tímu _elf_ [2]

Systém je implementovaný v jazyku Java. Na vývoj použil tím _elf_ integrované prostredie Eclipse. Ako databázový systém bol zvolený PostgreSQL. Na tvorbu webových služieb využili aplikačný rámec Apache AXIS. Prístup do databázy bol zjednodušený vlastnou implementáciou objektovo-relačného mapovania. Na vytvorenie prezentačnej vrstvy tím _elf_ použil rámec Struts, ktorého výstupom boli Java Server Pages (JSP) stránky. Aplikácia bola nasadená na aplikačný server Tomcat. Zostavovanie bolo zabezpečené pomocou nástroja Maven. Na logovanie dôležitých činností systému bol použitý nástroj Log4J. Systém bol testovaný pomocou knižnice JUnit. Serializácia XML dokumentov bola zabezpečená pomocou nástroja XStream.

2.2.5 Vlastnosti systému

Systém implementovaný tímom _elf_ má nasledovné vlastnosti:

- komunikácia s ostatnými systémami pomocou webových služieb (angl. *web services*),
- dôraz na bezpečnosť – autorizácia a autentifikácia,
- komunikácia pomocou zabezpečeného protokolu HTTPS,
- ochrana osobných údajov študentov a
- prepracovaný algoritmus na vyhľadávanie študentov na základe zadaných kritérií (balíčkov a skupín).

2.2.6 Zhodnotenie

Tímu _elf_ sa podarilo navrhnuť a implementovať fungujúci systém na evidenciu znalostí a zručností študentov. Vytvorený logický model má veľmi dobrú štruktúru, ktorá umožňuje systém dostatočne

konfigurovať. Niektoré údaje sa však evidujú zbytočne, napr. údaje o predmetoch alebo o učiteľoch neprinášajú nijakú špeciálnu hodnotu pre používateľa a sú už evidované v systéme AIS.

Vytvorený systém je dostatočne zabezpečený a chráni osobné údaje študentov v zmysle zákona. Ochrana osobných údajov je však na niektorých miestach až prehnaná, napr. zaraďovanie výsledného zoznamu študentov do tried na základe štatistických metód.

Navrhnutý algoritmus na vyhľadávanie študentov je dostatočne prepracovaný a umožňuje učiteľom pohodlne a efektívne vyhľadávať študentov podľa ich potreby.

Vytvorený systém spolupracuje so systémom ŠTUDENT, importuje z neho informácie o študentoch a predmetoch. Tento systém v súčasnosti nahradil na celej univerzite systém AIS.

2.3 Analýza modelu používateľa

Na reprezentáciu znalostí a zručností študenta potrebujeme model, ktorý nám umožní uchovávať a ohodnocovať jeho rozličné schopnosti. Najlepšie sa nám javí rozdeliť model na 3 časti:

- všeobecnú časť - charakteristiky používateľa ako napr. vek, pohlavie, atď.
- doménovo nezávislú časť – charakteristiky používateľa nezávisiace od domény, tieto charakteristiky sa menia len sporadicky. Patria sem dosiahnuté vzdelanie, odbor, atď.
- špecifickú časť

V [1] navrhli autori systém, v ktorom špecifická časť obsahuje schopnosti používateľa ohodnotené váhami v intervale $\langle 0;100 \rangle$. Následne pri zadávaní požiadavky na vhodných používateľov zadávateľ určil požadované schopnosti, ktoré tiež ohodnotil váhami. Na základe toho systém vybral vhodných kandidátov.

V systéme boli vytvorené stereotypy. Stereotyp je množina ľudí s rovnakými charakteristikami, napr. návrhár, analytik, programátor, tester, atď. Každý stereotyp má určené schopnosti, ktoré človek musí na zaradenie do tohto stereotypu spĺňať, a tiež mieru tohto splnenia v percentách. Jeden stereotyp bol úplne bez určených schopností, aby bol model všeobecný (do tohto stereotypu boli zaradení používatelia bez akýchkoľvek znalostí).

Model používateľa bol uchovaný vo forme ontológie, v jazyku OWL. Systém bol vytvorený v jazyku Java pomocou rámca Sesame (www.openrdf.org).

2.4 Analýza informačných systémov používaných na FIIT

2.4.1 Akademický informačný systém

AIS je komplexný celouniverzitný informačný systém evidujúci informácie o študentoch, predmetoch, vyučujúcich a pod. Tento systém je postavený na databázovom systéme Oracle. V súčasnej dobe prebieha vytváranie pohľadu do tejto databázy, ktorý bude obsahovať:

- údaje o študentoch (ID, meno, adresa, št. program, ročník, semester, predmety, rozvrh a informácie o štúdiu) a
- údaje o pedagógoch (predmety, rozvrh, pracovisko, rola v danom predmete).

Náš systém bude čerpať všetky údaje práve z AISu.

2.4.2 Yonban

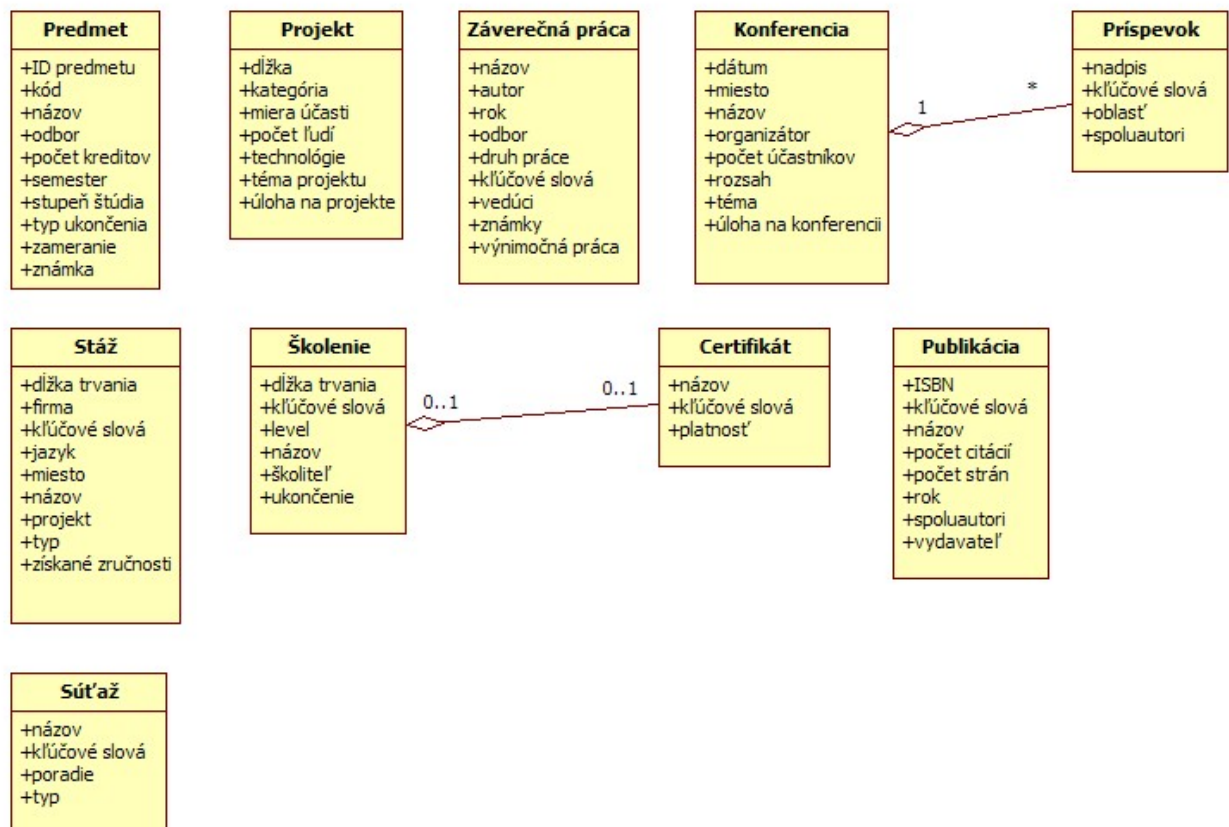
Yonban slúži na evidenciu bakalárskych, diplomových a iných záverečných prác. Všetky informácie v ňom evidované sa nachádzajú aj v systéme AIS, takže ho nebudeme v našom projekte využívať.

2.5 Zmena váh

V rámci procesu vzdelávania občas nastávajú zmeny v systéme výučby a vtedy môže dôjsť aj k zmenám váh, ktoré ovplyvňujú kľúčové slová uchované v databáze. Tieto kľúčové slová sú potrebné pre určenie znalostí študenta. Aby systém mohol reagovať na takéto zmeny a zakaždým určoval čo najpresnejšie znalosti študenta, musí obsahovať modul, pomocou ktorého budú tieto váhy upravené. Takýto modul navrhol v svojom projekte z roku 2007/2008 tím Černé ofce.

2.6 Analýza faktov

Identifikovali sme 8 druhov znalostí, z ktorých môže náš systém pri zostavovaní modelu študenta čerpať. Diagram zobrazujúci tieto fakty je na Obr. 2.3. Ku každému faktovi sme identifikovali niekoľko jeho atribútov. Nie všetky atribúty sú pre náš systém dôležité. V budúcnosti môže vzniknúť potreba definovať ďalšie typy faktov. Nami navrhovaný systém je všeobecný a umožňuje jednoduché definovanie nových faktov spolu s ich atribútmi. Tým je zabezpečená jeho univerzálnosť.



Obr. 2.3 Diagram faktov a ich atribútov

3 Špecifikácia požiadaviek na informačný systém

V tejto kapitole uvádzame špecifikáciu požiadaviek na vytváraný informačný systém Znalec.

3.1 Funkcionálne požiadavky na systém

Na základe analýzy a komunikácie so zákazníkom sme identifikovali funkcionálne požiadavky. Tie sme roztriedili podľa priority nasledovne:

- vysoká priorita
 - evidencia študentov
 - pridávanie konkrétnych faktov študentmi a ich ohodnotenie váhami
 - vyhodnotenie znalosti študenta na základe faktov
 - vyhľadávanie študentov na základe znalostí
- stredná priorita
 - získavanie informácií o študentoch zo systému AIS
 - vysoká konfigurovateľnosť systému
 - vyplňanie osobných profilov študentov
- nízka priorita
 - prezeranie študentských profilov osobami mimo fakulty
 - zaradovanie študentov do skupín podľa znalostí
 - vytváranie preddefinovaných pravidiel vyhľadávania študentov

3.2 Ďalšie požiadavky na systém

Okrem funkcionálnych požiadaviek sme identifikovali aj ďalšie požiadavky, týkajúce sa najmä prevádzky a údržby systému:

- Bezpečnosť – Prístup k databáze je možný len prostredníctvom nášho systému. Používateľom nebude umožnený priamy prístup k databáze. Každý používateľ systému sa musí prihlásiť pomocou svojho prihlasovacieho mena a hesla, ktoré sú uložené v šifrovanej podobe. Prihlasovacie heslo musí byť dostatočne robustné, čo systém overuje pri jeho vytváraní (musí obsahovať určitý minimálny počet alfanumerických a špeciálnych znakov).
- Spoľahlivosť – Systém monitoruje vykonané zmeny a pravidelne vykonáva zálohu databázy.
- Korektnosť údajov – Systém je zabezpečený zámkami, ktoré zaručujú, že jednu entitu (napr. profil študenta) môže naraz editovať len jeden používateľ a nenastal tak konflikt.

3.3 Prepojenie cieľov a požiadaviek

Jednotlivé funkcionálne požiadavky na systém, špecifikované v predchádzajúcej podkapitole, predstavujú biznis prípady použitia, ktoré realizujú ciele definované v kapitole 1.2. Nasleduje sumár prepojenia biznis prípadov použitia na ciele, ktoré realizujú.

Automatické získavanie údajov: získavanie informácií o študentoch zo systému AIS

Manuálne získavanie údajov: pridávanie konkrétnych faktov študentmi a ich ohodnotenie váhami

Vytvorenie bázy znalostí: evidencia študentov, o vyhodnotenie znalosti študenta na základe faktov

Odporúčanie vhodných kandidátov: vyhľadávanie študentov na základe znalostí, zaraďovanie študentov do skupín podľa znalostí

Inteligentné prehľadávanie: prezeranie študentských profilov osobami mimo fakulty, vytváranie preddefinovaných pravidiel vyhľadávania študentov

Modifikácia údajov: vysoká konfigurovateľnosť systému, vyplňanie osobných profilov študentov

3.4 Charakteristika rolí

V systéme sme identifikovali nasledujúce roly:

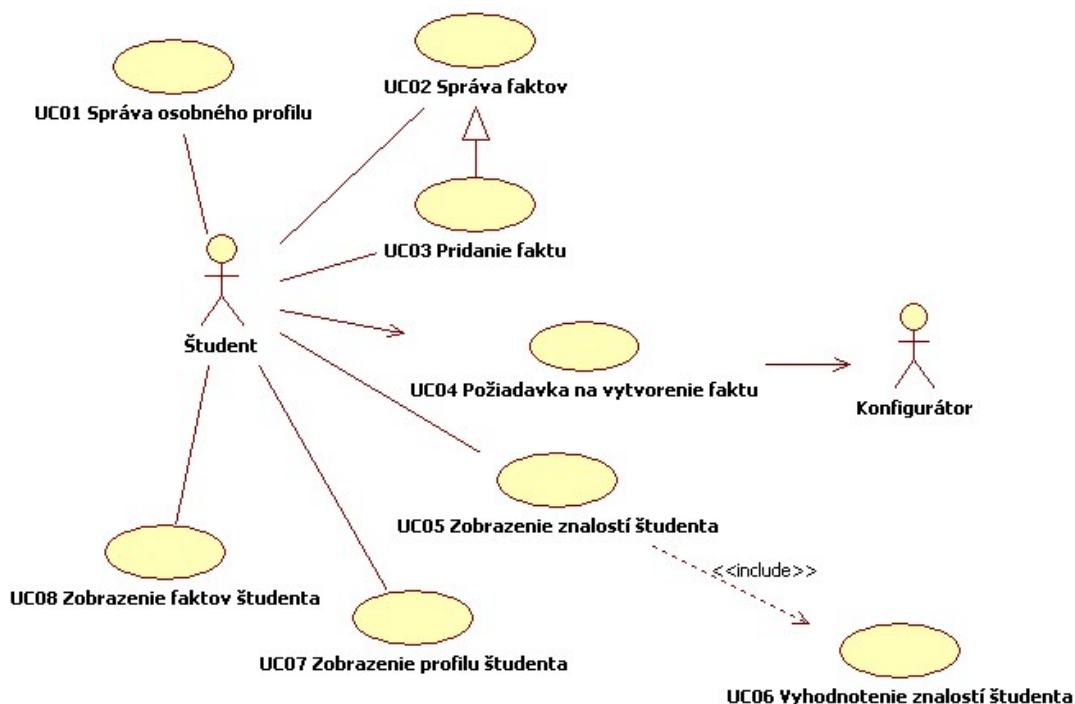
- **Študent** – riadne zapísaný študent
- **Pedagóg** – pedagogický zamestnanec školy
- **Administrátor** – osoba poverená správou systému
- **Externý používateľ** – osoba mimo fakulty, musí však mať v systéme účet
- **Konfigurátor** – osoba zodpovedná za konfiguráciu faktov, znalostí a kľúčových slov, a vzťahov medzi nimi

3.5 Diagramy prípadov použitia

Na základe uvedených funkcionálnych požiadaviek sme vytvorili diagramy prípadov použitia nášho systému. Každý diagram zobrazuje jednu rolu a prípady použitia, ktoré k nej prislúchajú. Následne je každý prípad použitia opísaný v samostatnej tabuľke.

3.5.1 Prípady použitia pre rolu študent

Pre rolu študent sme identifikovali prípady použitia zobrazeného na Obr. 3.1.



Obr. 3.1 Diagram prípadov použitia pre rolu študent

V nasledujúcich tabuľkách uvádzame podrobný opis prípadov použitia pre rolu študent.

Tab. 3.1 Prípad použitia UC01 – Správa osobného profilu

Správa osobného profilu	
Identifikátor	UC01
Názov	Správa osobného profilu
Opis	CRUD operácie nad osobným profilom študenta
Priorita	stredná
Vstupné podmienky	
Výstupné podmienky	Uložené zmeny sú perzistentné.
Používatelia	Študent

Tab. 3.2 Prípad použitia UC02 – Správa faktov

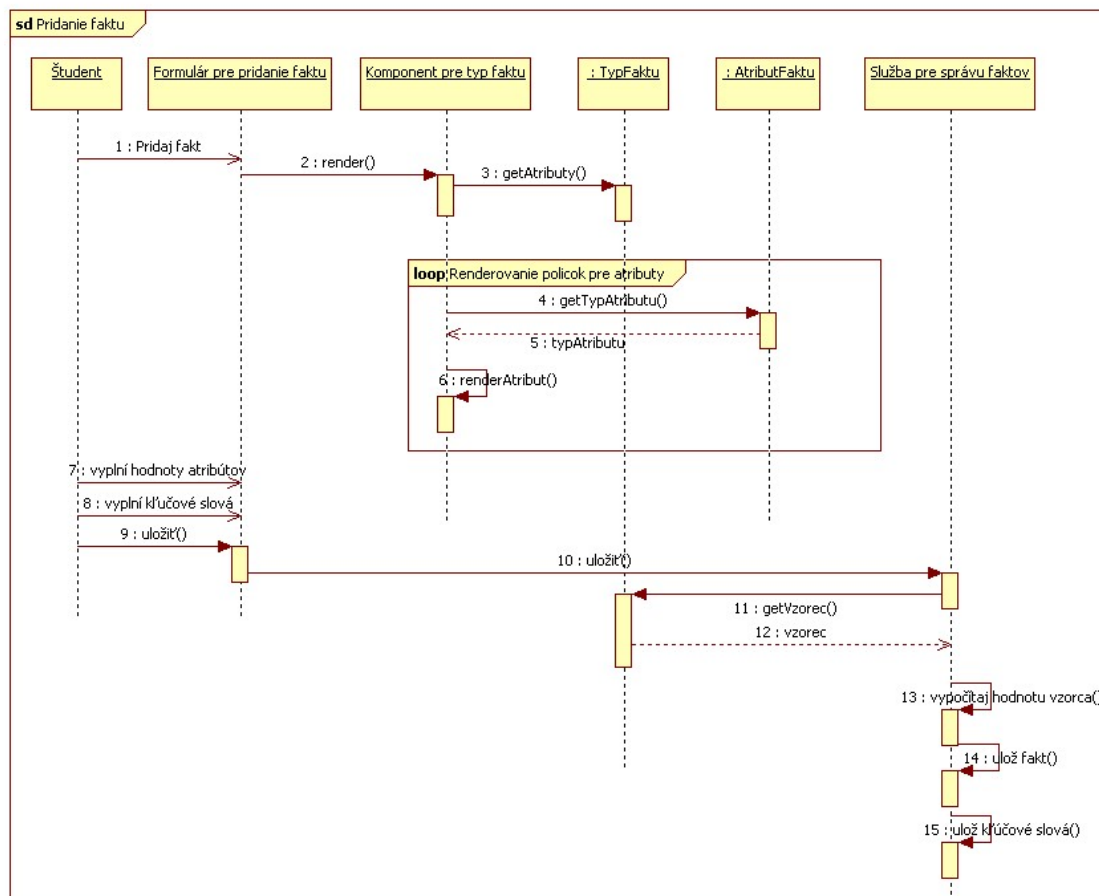
Správa faktov	
Identifikátor	UC02
Názov	Správa faktov
Opis	CRUD operácie nad faktami študenta
Priorita	vysoká
Vstupné podmienky	
Výstupné podmienky	Uložené zmeny sú perzistentné.
Používatelia	Študent

Tab. 3.3 Prípad použitia UC03 – Pridanie faktu

Pridanie faktu	
Identifikátor	UC03
Názov	Pridanie faktu
Opis	Umožňuje pridať fakt k študentovi. Fakt môže pridať prihlásený študent len sebe. Prihlásený pedagóg môže pridať fakt ľubovoľnému študentovi. Fakt môže pridať aj systém pri importe údajov z AISu.
Priorita	vysoká
Vstupné podmienky	Používateľ je prihlásený ako študent alebo používateľ je prihlásený ako pedagóg, alebo používateľ je prihlásený ako administrátor.
Výstupné podmienky	Pridaný fakt je perzistentný.
Používatelia	Študent, Pedagóg, Administrátor

Postupnosť pre študenta	Krok	Činnosť
	1	Študent zvolí typ faktu, ktorý chce pridať.
	2	Systém zobrazí formulár pre zadanie hodnôt atribútov daného faktu definovaných typom faktu.
	3	Študent vyplní položky formulára. Vyplní aj kľúčové slová, ktoré s faktom súvisia a určí ich váhy podľa dôležitosti v rámci daného faktu.
	4	Študent potvrdí pridanie faktu.
	5	Systém uloží zadané údaje do databázy. Ak sa zadané kľúčové slová ešte nenachádzajú v databáze, tak sa do nej pridajú. Ak sa nachádzajú, tak sa len prepoja s daným faktom a priradia sa im zadané váhy.
Postupnosť pre pedagóga	Krok	Činnosť
	1	Pedagóg zvolí študenta, ktorému chce pridať nový fakt.
	2	Ďalšie kroky sú rovnaké ako pri študentovi.
Postupnosť pri importe údajov z AISu	1	Systém pridáva relevantné fakty (absolvovanie predmetov, záverečné práce a pod.) k študentom na základe údajov z AISu. Váhy sa nastavujú na implicitnú hodnotu.
Poznámky	Tento prípad použitia je špecializáciou prípadu použitia správa faktov. Dôvod jeho vyčlenenia je ten, že pri ňom vystupujú viacerí hráči a aj okolnosti jeho vykonania sú špeciálnejšie.	

Na Obr. 3.2 je zobrazený sekvenčný diagram pre prípad použitia UC03 – Pridanie faktu. Študent najskôr zvolí, že chce pridať fakt. Prítom zadá aj aký typ faktu chce pridať. Systém mu ponúkne stránku s formulárom pre zadanie hodnôt atribútov pridávaného faktu. Formulár sa kreslí genericky v závislosti od typu faktu. Grafický komponent požiadava entitu typ faktu, aby mu vrátila zoznam atribútov. Tento komponent potom v závislosti od typu atribútu vykreslí príslušné vstupné pole formulára. Po vykreslení formulára študent vyplní jeho vstupné polia, zadá kľúčové slová a uloží ho. Služba pre správu faktov vypočíta hodnotu faktu na základe vzorca a vyplnených atribútov. Táto služba nakoniec uloží fakt s jeho vyplnenými atribútmi a kľúčové slová do databázy.



Obr. 3.2 Sekvenčný diagram pre prípad použitia UC03 – Pridanie faktu

Tab. 3.4 Prípad použitia UC04 – Požiadavka na vytvorenie faktu

Požiadavka na vytvorenie faktu		
Identifikátor	UC04	
Názov	Požiadavka na vytvorenie faktu	
Opis	Umožňuje upozorniť konfigurátora, aby pridal do systému nový typ faktu.	
Priorita	nízka	
Vstupné podmienky		
Výstupné podmienky	Konfigurátor bol notifikovaný e-mailom.	
Používatelia	Študent, Pedagóg	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ zvolí, že chce pridať nový fakt
	2	System zobrazí formulár pre zadanie opisu žiadaného faktu.
	3	Používateľ vyplní položky formulára.
	4	System pošle e-mail s požiadavkou na vytvorenie faktu s údajmi zadanými používateľom.

Tab. 3.5 Prípád použitia UC05 – Zobrazenie znalostí študenta

Zobrazenie znalostí študenta		
Identifikátor	UC05	
Názov	Zobrazenie znalostí študenta	
Opis	Zobrazí používateľovi zistené znalosti daného študenta.	
Priorita	vysoká	
Vstupy	Študent, ktorého znalosti sa budú vyhodnocovať a zobrazovať	
Výstupy	Zistené znalosti o danom študentovi	
Vstupné podmienky	Používateľ je prihlásený a má aspoň jednu z rolí Študent, Pedagóg. Ak je používateľ prihlásený ako Externý používateľ, tak daný študent musí mať povolené zobrazovanie informácií pre verejnosť.	
Výstupné podmienky		
Používatelia	Študent, Pedagóg, Externý používateľ	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ zvolí študenta, ktorého znalosti chce zobraziť.
	2	Spustí sa prípad použitia Vyhodnotenie znalostí študenta. Na jeho vstup pôjde meno daného študenta.
	3	Systém zobrazí výstup prípadu použitia Vyhodnotenie znalostí študenta

Tab. 3.6 Prípád použitia UC06 – Vyhodnotenie znalostí študenta

Vyhodnotenie znalostí študenta		
Identifikátor	UC06	
Názov	Vyhodnotenie znalostí študenta	
Opis	Vyhodnotí znalosti študenta na základe faktov, priradených tomuto študentovi.	
Priorita	vysoká	
Vstupy	Študent, ktorého znalosti sa budú vyhodnocovať	
Výstupy	Usporiadané dvojice (znalosť, úroveň znalosti)	
Vstupné podmienky	Študent má priradené nejaké fakty. Tieto fakty majú priradené kľúčové slová a ich váhy. V systéme sú evidované znalosti, ku ktorým sú priradené kľúčové slová a ich váhy.	
Výstupné podmienky		
Používatelia		
Postupnosť krokov	Krok	Činnosť
	1	Iterujeme cez všetky fakty daného študenta.
	2	Pre každý fakt nájdeme znalosti, s ktorými súvisí. To zistíme na základe prieniku kľúčových slov, ktoré definujú daný fakt a zároveň znalosť.
	3	K zatiaľ vypočítanej úrovni znalosti pripočítame hodnotu: znalosť.vahaZnalosti * fakt.vahaFaktu.
	4	Algoritmus vráti usporiadané dvojice (znalosť, úroveň znalosti)

Tab. 3.7 Prípád použitia UC07 – Zobrazenie profilu študenta

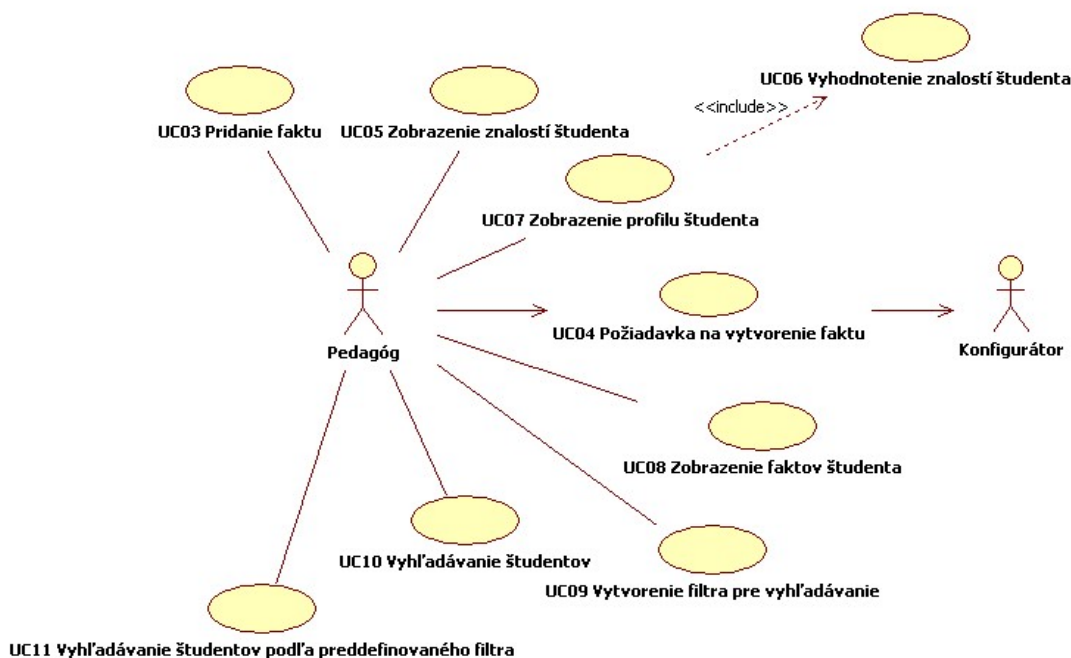
Zobrazenie profilu študenta		
Identifikátor	UC07	
Názov	Zobrazenie profilu študenta	
Opis	Zobrazí osobný profil daného študenta. Osobný profil obsahuje informácie ako meno, stupeň štúdia, odbor, fotka, informácie o sebe, e-mail, odkaz na svoj web a pod.	
Priorita	vysoká	
Vstupy	Identifikátor študenta, ktorého profil sa má zobraziť.	
Výstupy	Osobný profil študenta.	
Vstupné podmienky	Používateľ je prihlásený a má aspoň jednu z rolí Študent, Pedagóg. Ak má rolu Externý používateľ, tak daný študent musí mať povolené zobrazovanie informácií pre verejnosť.	
Výstupné podmienky		
Používatelia	Študent, Pedagóg, Externý používateľ	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ zvolí študenta, ktorého profil chce zobraziť.
	2	Systém zobrazí osobný profil študenta.

Tab. 3.8 Prípád použitia UC08 – Zobrazenie faktov študenta

Zobrazenie faktov študenta		
Identifikátor	UC08	
Názov	Zobrazenie faktov študenta	
Opis	Zobrazí fakty priradené danému študentovi.	
Priorita	vysoká	
Vstupy	Študent, ktorého fakty sa majú zobraziť.	
Výstupy	Fakty študenta.	
Vstupné podmienky	Používateľ je prihlásený a má aspoň jednu z rolí Študent, Pedagóg. Ak je používateľ prihlásený ako Externý používateľ, tak daný študent musí mať povolené zobrazovanie informácií pre verejnosť.	
Výstupné podmienky		
Používatelia	Študent, Pedagóg, Externý používateľ	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ zvolí študenta, ktorého fakty chce zobraziť.
	2	Systém zobrazí fakty študenta rozdelené do skupín podľa faktov. Pri jednotlivých faktoch zobrazí aj kľúčové slová a ich váhy.

3.5.2 Prípady použitia pre rolu pedagóg

Pre rolu pedagóg sme identifikovali prípady použitia zobrazené na Obr. 3.2.



Obr. 3.3 Diagram prípadov použitia pre rolu pedagóg

V nasledujúcich tabuľkách uvádzame podrobný opis prípadov použitia pre rolu pedagóg.

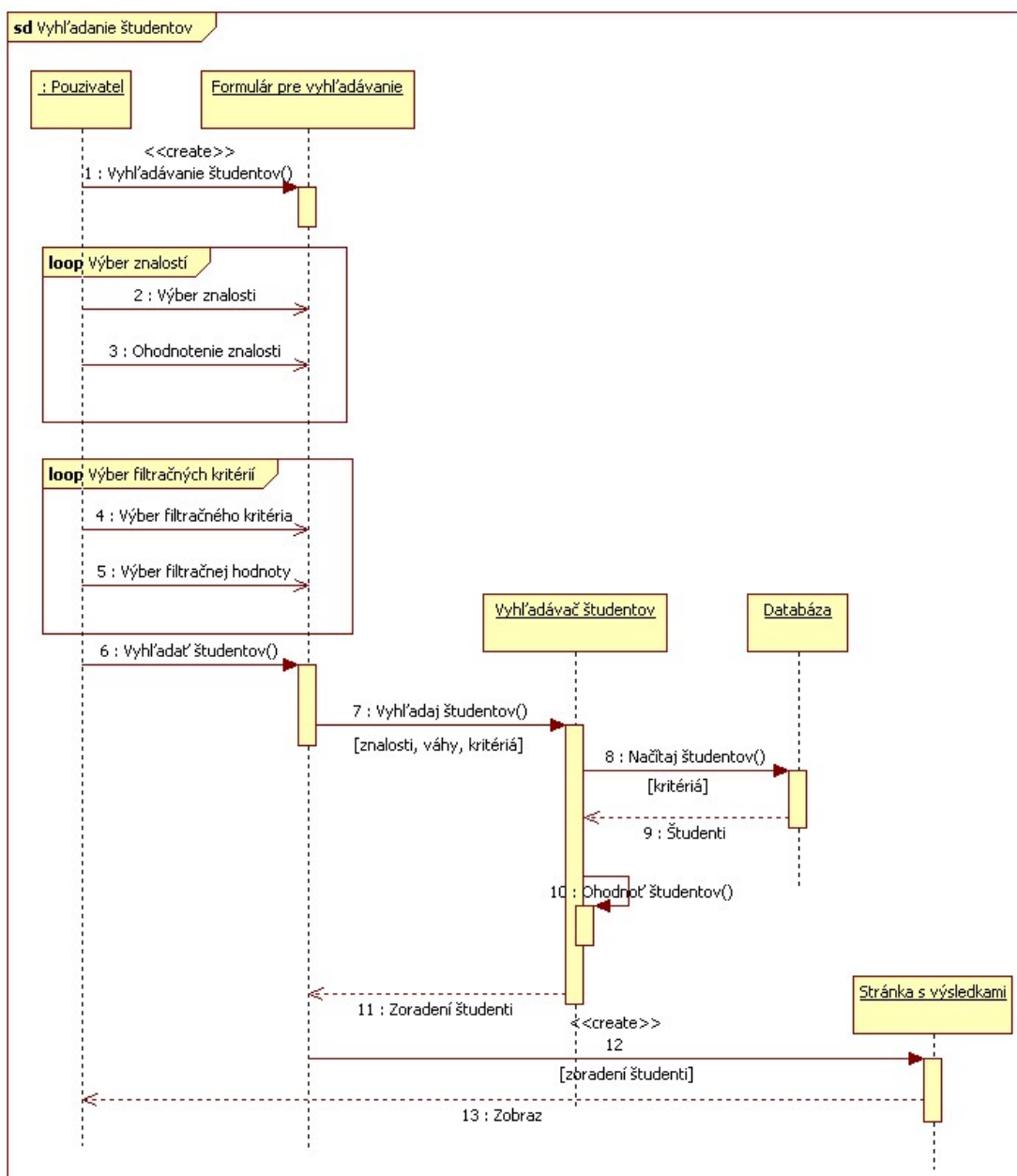
Tab. 3.9 Prípad použitia UC09 – Vytvorenie filtra pre vyhľadávanie

Vytvorenie filtra pre vyhľadávanie		
Identifikátor	UC09	
Názov	Vytvorenie filtra pre vyhľadávanie	
Opis	Umožňuje vytvoriť preddefinované vyhľadávacie pravidlá.	
Priorita	nízka	
Vstupy	Kritéria pre filtrovanie výsledkov. Kritéria, podľa ktorých sa majú zobrazené výsledky zotriediť.	
Výstupy		
Vstupné podmienky		
Výstupné podmienky	Filter je perzistentný.	
Používatelia	Pedagóg, Konfigurátor	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ vyberie, že chce vytvoriť nový filter.
	2	Systém mu zobrazí formulár s položkami pre filtrovanie a zoradovanie výsledkov
	3	Používateľ vyberie položky, podľa ktorých chce filtrovať a triediť.
	4	Používateľ zvolí filtrovacie a triediace kritériá (<, >, =, má, nemá, ...)
	5	Systém uloží tento filter do databázy.

Tab. 3.10 Prípád použitia UC10 – Vyhľadávanie študentov

Vyhľadávanie študentov		
Identifikátor	UC10	
Názov	Vyhľadávanie študentov	
Opis	Umožňuje vyhľadávať študentov podľa zadaných kritérií.	
Priorita	vysoká	
Vstupy	Kritéria pre filtrovanie výsledkov. Kritéria, podľa ktorých sa majú zobrazené výsledky zotriediť.	
Výstupy	Zoznam študentov, spĺňajúcich dané kritériá, zoradených podľa triediacich kritérií. V prípade, že používateľom je externý používateľ, výsledný zoznam neobsahuje študentov, ktorí nemajú povolené zobrazovanie informácií pre verejnosť.	
Vstupné podmienky	Prihlásený používateľ má rolu Pedagóg alebo Externý používateľ.	
Výstupné podmienky		
Používatelia	Pedagóg, Externý používateľ	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ vyberie, že chce vyhľadávať študentov.
	2	Systém mu zobrazí formulár s položkami pre filtrovanie a zoradovanie výsledkov.
	3	Používateľ vyberie položky, podľa ktorých chce filtrovať a triediť výsledky.
	4	Používateľ zvolí filtrovacie a triediace kritériá (<, >, =, má, nemá, ...)
	5	Systém zobrazí zoznam študentov, spĺňajúcich dané kritériá, zoradených podľa triediacich kritérií. V prípade, že používateľom je externý používateľ, výsledný zoznam neobsahuje študentov, ktorí nemajú povolené zobrazovanie informácií pre verejnosť

Na Obr. 3.4 je zobrazený sekvenčný diagram pre prípad použitia UC10 – Vyhľadávanie študentov.



Obr. 3.4 Sekvenčný diagram pre prípad použitia UC10 – Vyhľadávanie študentov

Tab. 3.11 Prípad použitia UC11 – Vyhľadávanie študentov podľa preddefinovaného filtra

Vyhľadávanie študentov podľa preddefinovaného filtra	
Identifikátor	UC11
Názov	Vyhľadávanie študentov podľa preddefinovaného filtra
Opis	Umožňuje vyhľadávať študentov podľa preddefinovaného filtra.
Priorita	nízka
Vstupy	Preddefinovaný filter.
Výstupy	Zoznam študentov, spĺňajúcich dané kritériá, zoradených podľa triediacich kritérií.
Používatelia	Pedagóg, Externý používateľ

Postupnosť krokov	Krok	Činnosť
	1	Používateľ vyberie, že chce vyhľadávať študentov podľa preddefinovaného filtra.
	2	Systém mu ponúkne preddefinované filtre v systéme.
	3	Používateľ vyberie filter ktorý chce použiť.
	4	Používateľ môže upraviť filtrovacie kritériá
	5	Systém zobrazí zoznam študentov, spĺňajúcich dané kritériá, zoradených podľa triediacich kritérií.

3.5.3 Prípady použitia pre rolu administrátor

Pre rolu administrátor sme identifikovali prípady použitia zobrazené na Obr. 3.3.



Obr. 3.5 Diagram prípadov použitia pre rolu administrátor

V nasledujúcich tabuľkách uvádzame podrobný opis prípadov použitia pre rolu administrátor.

Tab. 3.12 Prípad použitia UC12 – Správa používateľských účtov

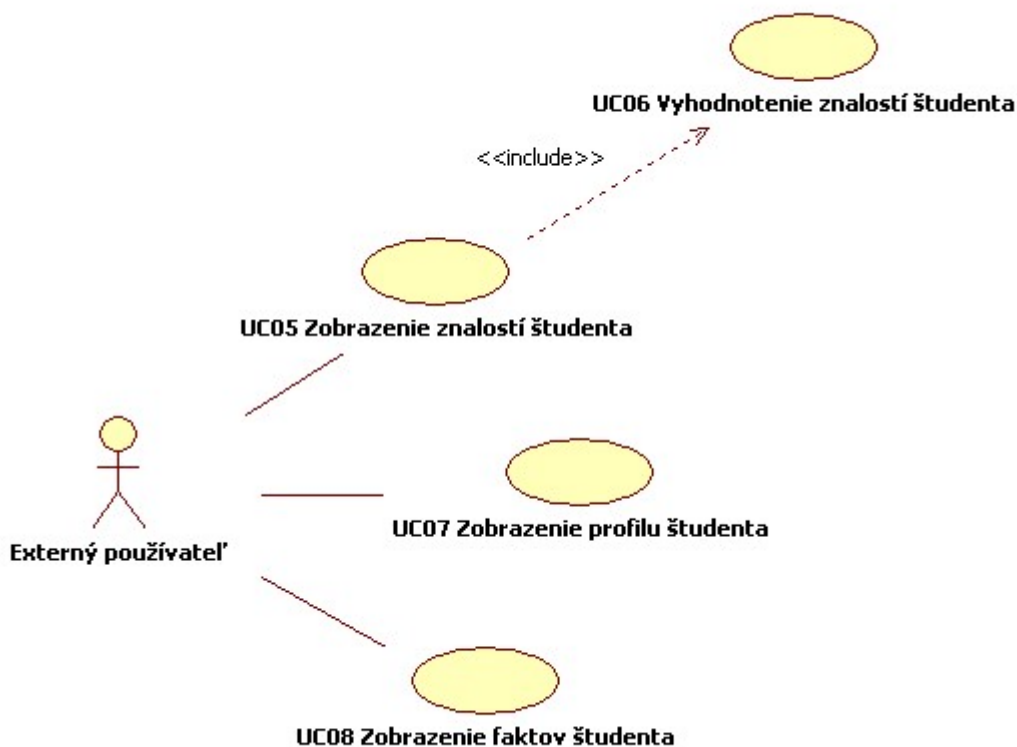
Správa používateľských účtov	
Identifikátor	UC12
Názov	Správa používateľských účtov
Opis	CRUD operácie nad účtami používateľov. Tento prípad použitia umožňuje odstraňovať nesprávne, nepravdivé alebo nevhodné fakty o študentoch. Tento prípad použitia tiež umožňuje vygenerovanie nového hesla k používateľskému účtu.
Priorita	stredná
Vstupné podmienky	
Výstupné podmienky	Uložené zmeny sú perzistentné.
Používatelia	Administrátor

Tab. 3.13 Prípady použitia UC13 – Import údajov z DB AISu

Import údajov z DB AISu		
Identifikátor	UC13	
Názov	Import údajov z DB AISu	
Opis	Importuje do systému údaje z akademického informačného systému.	
Priorita	stredná	
Vstupné podmienky	Dá sa pripojiť do databázy AIS.	
Výstupné podmienky	Údaje z AISu sú importované a transformované do databázy nášho systému vo forme faktov. Pôvodné fakty v systéme sú nezmenené (za predpokladu, že sa nezmenili v AISe).	
Používatelia	Administrátor	
Postupnosť krokov	Krok	Činnosť
	1	Používateľ zvolí, že chce importovať údaje z AISu.
	2	Systém sa pripojí do databázy AISu.
	3	Systém zistí, ktoré údaje sa zmenili a ktoré pribudli od posledného importovania. Tieto údaje transformuje do faktov a pridá ich do systému použitím prípadu použitia Pridanie faktu.

3.5.4 Prípady použitia pre rolu externý používateľ

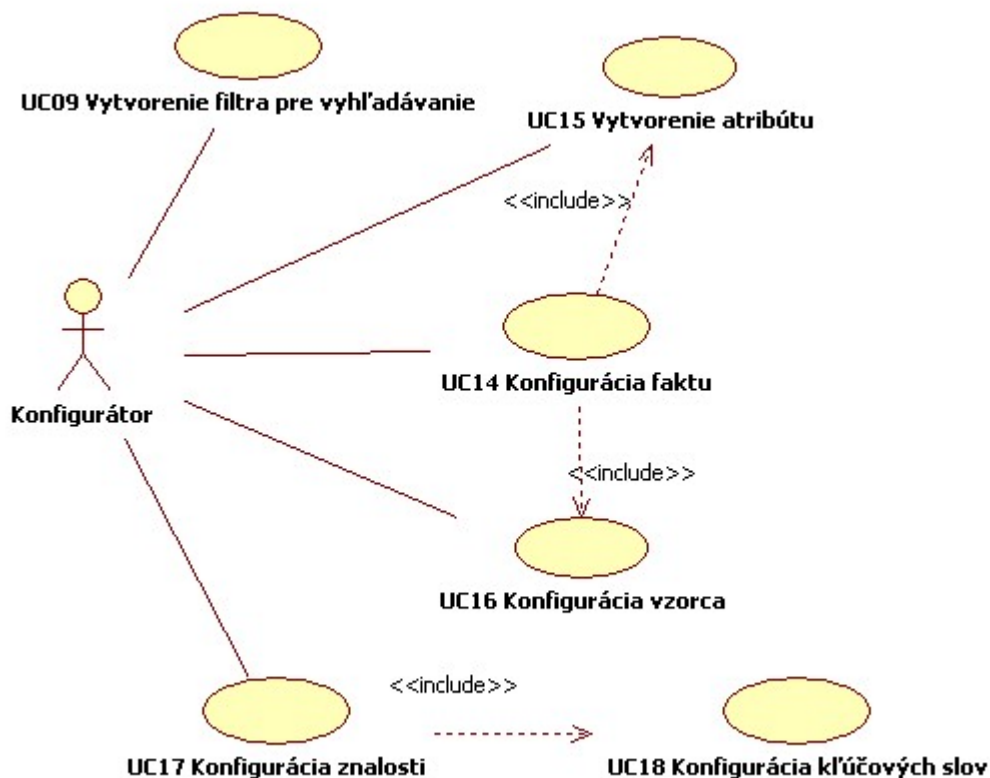
Pre rolu externý používateľ sme identifikovali prípady použitia zobrazené na Obr. 3.4.



Obr. 3.6 Diagram prípadov použitia pre rolu externý používateľ

3.5.5 Prípady použitia pre rolu konfigurátor

Pre rolu konfigurátor sme identifikovali prípady použitia zobrazené na Obr. 3.5.



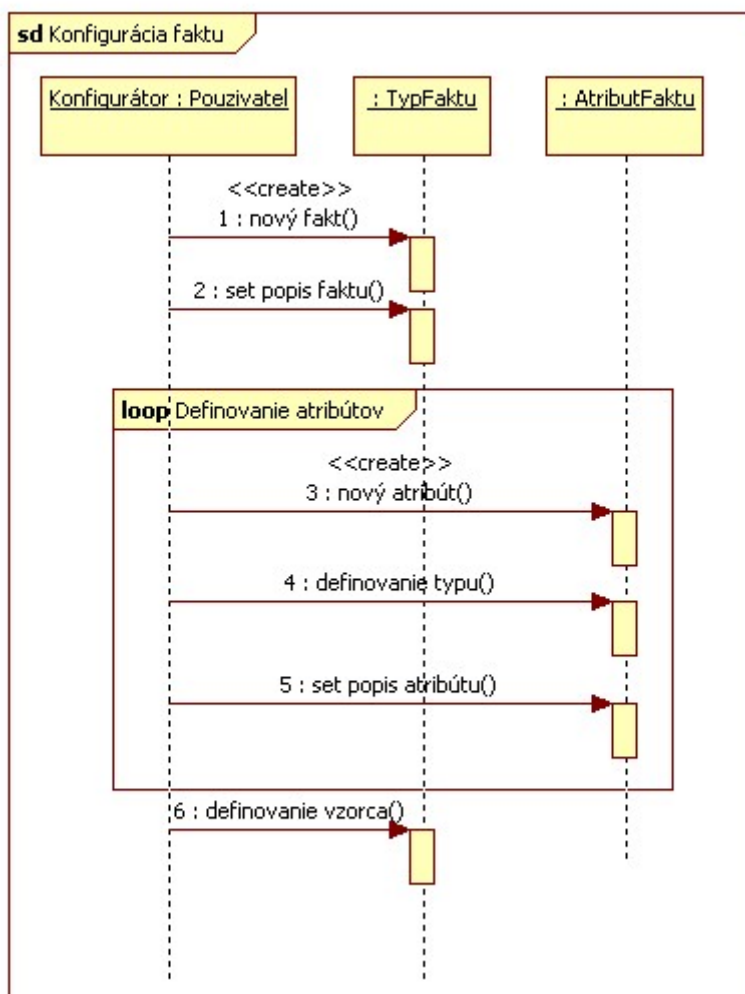
Obr. 3.7 Diagram prípadov použitia pre rolu konfigurátor

V nasledujúcich tabuľkách uvádzame podrobný opis prípadov použitia pre rolu konfigurátor.

Tab. 3.14 Prípady použitia UC14 – Konfigurácia faktu

Konfigurácia faktu		
Identifikátor	UC14	
Názov	Konfigurácia faktu	
Opis	Umožňuje pridávať a upravovať typy faktov.	
Priorita	stredná	
Vstupy	Názov a popis faktu.	
Výstupy	Nové fakty v databáze.	
Používatelia	Konfigurátor	
Postupnosť krokov pre pridanie nového faktu	Krok	Činnosť
	1	Používateľ vyberie, že chce pridať nový typ faktu.
	2	Používateľ pomenuje a popíše tento typ faktu. Zadefinuje atribúty faktu pomocou prípadu použitia UC15 Konfigurácia atribútov
	3	Používateľ vytvorí vzorec pre výpočet hodnotnosti faktu pomocou prípadu použitia UC16 Konfigurácia vzorca.
	4	Systém uloží nový typ faktu do databázy.

Na Obr. 3.8 je zobrazený sekvenčný diagram pre prípad použitia UC14 – Konfigurácia faktu.



Obr. 3.8 Sekvenčný diagram pre prípad použitia UC14 – Konfigurácia faktu

Tab. 3.15 Prípad použitia UC15 – Vytvorenie atribútu

Konfigurácia atribútu		
Identifikátor	UC15	
Názov	Vytvorenie atribútu	
Opis	Umožňuje pridávať a upravovať atribúty faktu.	
Priorita	stredná	
Vstupy	Názov a popis a dátový typ atribútu.	
Výstupy	Nové atribút v databáze.	
Používatelia	Konfigurátor	
Postupnosť krokov pre pridanie nového atribútu	Krok	Činnosť
	1	Používateľ vyberie, že chce pridať nový atribút.
	2	Používateľ pomenuje a popíše tento atribút. Zadefinuje dátový typ tohto atribútu
	3	Systém uloží nový atribút do databázy.

Tab. 3.16 Prípád použitia UC16 – Konfigurácia vzorca

Konfigurácia vzorca		
Identifikátor	UC16	
Názov	Konfigurácia vzorca	
Opis	Umožňuje vytvoriť vzorec pre výpočet hodnotnosti faktu.	
Priorita	nízka	
Vstupy	Atribúty faktu, pre ktorý sa vzorec vytvára.	
Výstupy	Vytvorený vzorec	
Používatelia	Konfigurátor	
Postupnosť krokov pre pridanie novej znalosti	Krok	Činnosť
	1	Používateľ vyberie, že chce pridať novú znalosť.
	2	Používateľ pomocou rozhrania vytvorí vzorec pre výpočet hodnotnosti faktu z jeho atribútov.
	3	Systém uloží daný vzorec k vstupnému faktu.

Tab. 3.17 Prípád použitia UC17 – Konfigurácia znalosti

Konfigurácia znalosti		
Identifikátor	UC17	
Názov	Konfigurácia znalosti	
Opis	Umožňuje pridávať a upravovať znalosti v systéme.	
Priorita	nízka	
Vstupy	Názov a popis znalosti. Kľúčové slová, viažuce sa na túto znalosť a ich váhy.	
Výstupy	Nová znalosť v databáze.	
Používatelia	Konfigurátor	
Postupnosť krokov pre pridanie novej znalosti	Krok	Činnosť
	1	Používateľ vyberie, že chce pridať novú znalosť.
	2	Používateľ pomenuje a popíše túto znalosť. Opíše túto znalosť pomocou kľúčových slov, ktorým pridelí váhy pomocou prípadu použitia Konfigurácia kľúčových slov.
	3	Systém uloží novú znalosť do databázy.

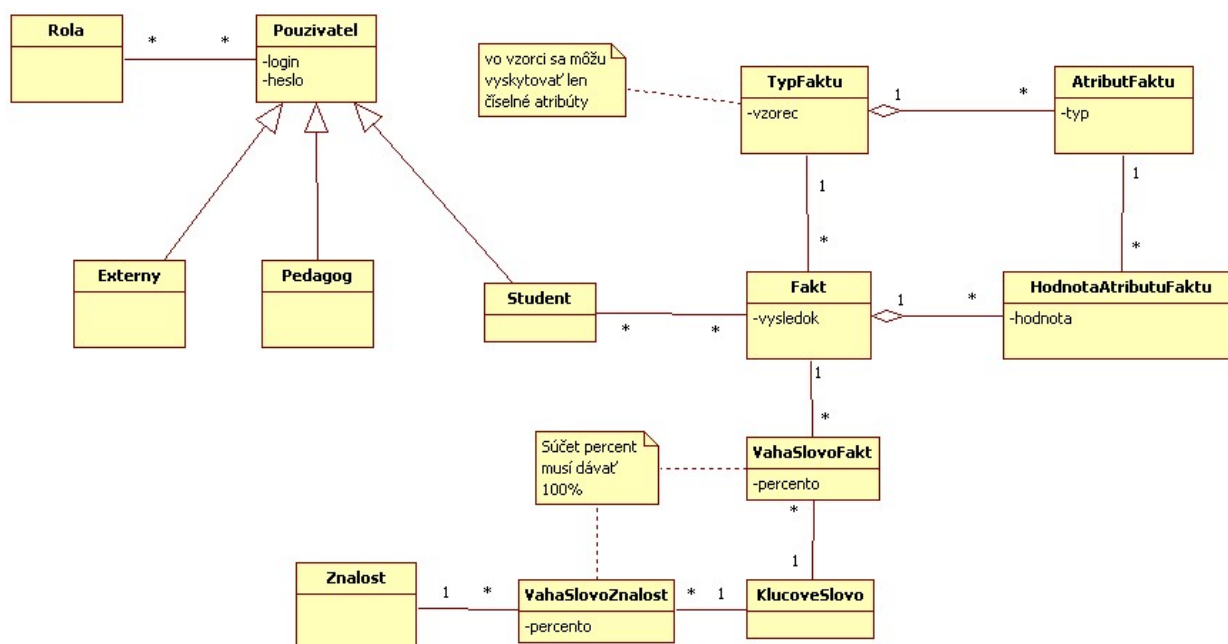
Tab. 3.18 Prípád použitia UC18 – Konfigurácia kľúčových slov

Konfigurácia kľúčových slov	
Identifikátor	UC18
Názov	Konfigurácia kľúčových slov
Opis	Umožňuje pridávať a upravovať kľúčové slová a ich váhy k danej znalosti.
Priorita	nízka
Vstupy	Identifikátor znalosti.
Výstupy	
Používatelia	Konfigurátor

Postupnosť krokov pre pridanie novej znalosti	Krok	Činnosť
	1	Používateľ vyberie zadá kľúčové slová a ich váhy smerom ku vstupnej znalosti.
	2	Systém uloží kľúčové slová a ich váhy do databázy.

3.6 Logický model údajov

Logický model nášho systému je možné rozdeliť do troch častí, z ktorých každá pozostáva z viaceru navzájom súvisiacich entít. Prvá množina slúži na uchovávanie používateľov, druhá na evidenciu a pridávanie faktov o študentoch, a tretia na evidenciu znalostí. Diagram logického modelu údajov je zobrazený na Obr. 3.6.



Obr. 3.9 Logický model údajov systému Znalec

Základnou entitou prvej množiny je abstraktná entita **Pouzivatel**, ktorej hlavnými atribútmi sú **prihlasovacie meno** a **heslo**, ktoré slúžia na autorizáciu vstupu používateľa do systému. Používateľia môžu mať priradené viaceré roly, opísané entitou **Rola**. Od entity **Pouzivatel** sú odvodené jednotlivé skupiny používateľov. Prvou je entita **Externy**, ktorá reprezentuje externého používateľa. Ním môže byť napríklad pracovník externej firmy hľadajúci budúcich zamestnancov. Druhou je **Pedagog**, ktorá slúži na uchovávanie pedagógov. Poslednou entitou odvodenou od **Pouzivatel**-a je **Student**. Študent je špeciálne vyčlenený, aby sme mohli evidovať, s akými faktami je asociovaný.

Druhá množina entít opisuje fakty súvisiace so študentmi. Entita **TypFaktu** určuje druh faktu, ktorý môžeme o študentovi evidovať, ako napríklad účasť na projekte, konferencií alebo jeho záverečné práce. Každý takýto **TypFaktu** sa dá opísať viacerými atribútmi. Pomocou číselných atribútov je definovaný vzorec pre výpočet „kvality“ daného faktu. Atribút typu faktu je reprezentovaný entitou **AtributFaktu** (napr. pre typ faktu Projekt to môžu byť dĺžka projektu, počet zúčastnených ľudí, rola v tíme, programovací jazyk). Entita **AtributFaktu** má atribút **typ** na určenie, aký dátový typ reprezentuje. Entita **Fakt** predstavuje inštanciu typu faktu (napr. účasť na projekte XYZ). Táto inštancia je tvorená hodnotami jej atribútov. Hodnoty sú uložené pomocou entity **HodnotaAtributuFaktu**, ktorá reprezentuje hodnotu typu atribútu (napr. dĺžka projektu – 5 mesiacov). Pomocou vzorca pre príslušný typ faktu sa z nich počíta miera vplyvu daného faktu

na znalosti študenta. Táto miera sa ukladá v atribúte *vysledok*. Takáto abstrakcia typov faktov a ich atribútov nám umožňuje konfigurovať systém bez potreby programovania.

Entita **Fakt** je prepojená s entitou **Student** s kardinalitou M ku N, pretože k študentovi sa môže viazať viacero faktov a jeden fakt sa môže viazať k viacerým študentom (napr. účasť na spoločnom školskom projekte). Entita **KlucoveSlovo** predstavuje evidované kľúčové slovo. Pomocou väzobnej entity **VahaSlovoFakt** sú namapované kľúčové slová na jednotlivé fakty pomocou percent. Súčet percent kľúčových slov daného faktu musí dávať 100 %.

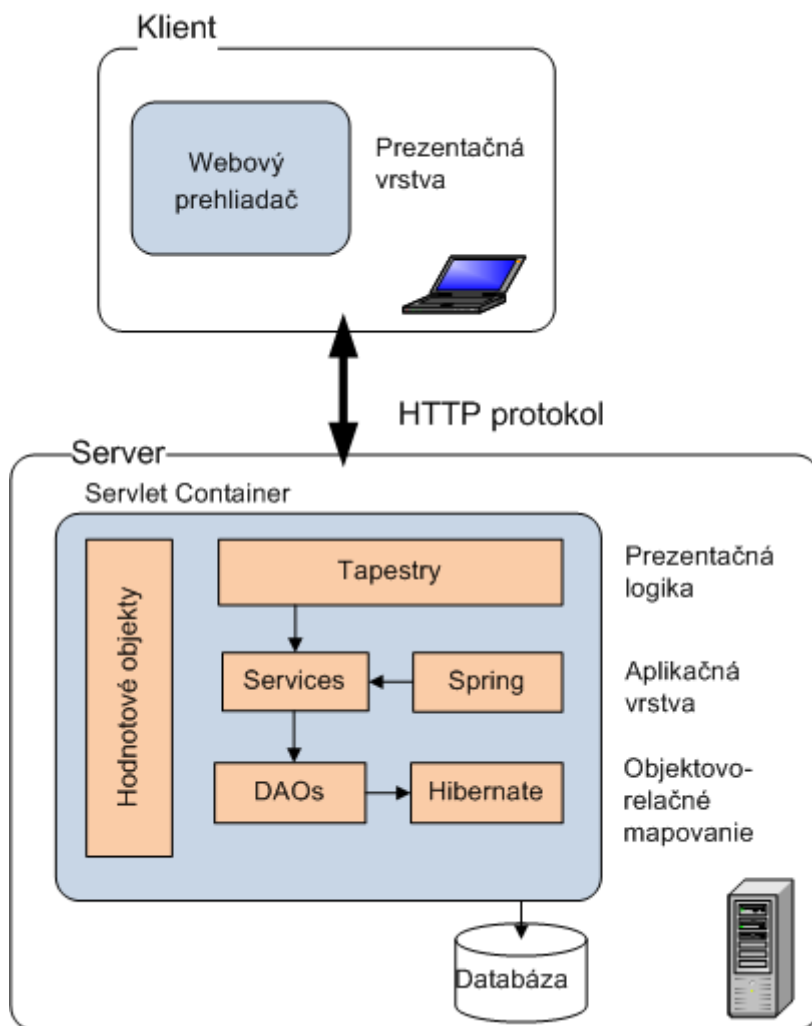
Posledné skupina pozostáva z entít **KlucoveSlovo**, **Znalost** a **VahaSlovoZnalost**. **VahaSlovoZnalost** slúži na prepojenie kľúčových slov so znalosťami a každému takémuto prepojeniu ešte pridáva aj percento, akým sa to-ktoré kľúčové slovo podieľa na jednotlivých znalostiach. Opäť platí, že súčet musí dávať 100 %.

4 Hrubý návrh riešenia

V tejto kapitole opisujeme návrh architektúry navrhovaného systému pre bázu znalostí a zručností študentov.

4.1 Aplikácia na báze JEE

Navrhovaný systém sme sa rozhodli implementovať ako webovú aplikáciu na platforme Java. Systém je založený na trojvrstvovej architektúre JEE. Na implementáciu tejto architektúry použijeme voľne dostupné aplikačné rámce a knižnice s otvoreným zdrojovým kódom. Návrh architektúry je uvedený na Obr. 5.1.



Obr. 4.1 Návrh architektúry systému Znalec

Základnou architektúrou je klient-server. Používateľovi je „viditeľný“ len tzv. tenký klient, ktorý sa skladá z prezentačnej vrstvy čo predstavuje webový prehliadač. Oveľa zaujímavejšou je serverová časť, ktorá sa skladá z prezentačnej logiky, aplikačnej vrstvy a vrstvy pre objektovo-relačné mapovanie. V nasledujúcich kapitolách opíšeme jednotlivé vrstvy.

4.2 Prezentačná vrstva

Úlohou prezentačnej vrstvy je poskytnúť používateľovi rozhranie na prácu s aplikáciou. Ako prezentačnú vrstvu sme si zvolili webové rozhranie. Na implementáciu tejto vrstvy sme sa rozhodli použiť voľne

dostupný javovský rámec s otvoreným zdrojovým kódom Tapestry⁴. Je to rámec na tvorbu dynamických webových aplikácií založený na jazyku Java. Webová aplikácia je rozdelená na stránky a tie sa skladajú z komponentov. To umožňuje vysokú znovupoužitelnosť. Tapestry disponuje sadou často používaných komponentov, je však rozširiteľný, takže každý si môže prispôsobiť existujúce komponenty alebo vytvoriť vlastné. Tento rámec je plne objektovo orientovaný, takže každá stránka a každý komponent je reprezentovaný Java triedou. To umožňuje programátorom sústrediť sa na funkcionality a nemusia sa starať o nízkoúrovňové procesy ako parsovanie parametrov z URL alebo vytváranie HTTP požiadaviek a odpovedí. Stránky a komponenty sú založené na HTML. HTML slúži však iba ako šablóna, ktorá sa za behu aplikácie môže meniť (a spravidla mení). Previazanie medzi HTML šablónou a Java kódom je zabezpečené pomocou metadát. Tieto metadáta môžu mať viacero foriem: XML dokument, špeciálne značky a atribúty v HTML alebo Java anotácie v kóde.

Tapestry zabraňuje „znovuvynachádzaniu“ kolesa tým, že zabezpečuje často sa opakujúce činnosti ako validáciu vstupu od používateľa alebo internacionalizáciu aplikácie. Trochu nezvyčajnou črtou Tapestry je princíp inverznej kontroly (angl. *Inversion of Control*). Je to opačný princíp ako pri bežných knižniciach, kde programátor volá služby knižnice. Pri rámcoch je to naopak – rámec „volá“ služby (objekty, metódy), ktoré napísal programátor. Preto väčšina Tapestry tried, ktoré budeme písať bude abstraktná, pretože Tapestry od nich za behu dedí a dopĺňa ďalšie vlastnosti.

Prezentačná vrstva len sprostredkuje interakciu s používateľom, aplikačná logika sa však skrýva v aplikačnej vrstve, ktorej služby používa prezentačná vrstva.

4.3 Aplikačná vrstva

Aplikačná vrstva predstavuje aplikačnú logiku, voľne povedané, v nej sa nachádza to dôležité, o čom vlastne aplikácia je. Táto vrstva sa dá reprezentovať ako množina služieb, ktoré poskytuje prezentačnej vrstve. Ide teda o použitie vzoru Facade. Služby sú prezentované ako metódy, ktoré sú logicky zoskupené do celku, ktorý je opísaný pomocou Java rozhrania. Ako príklad môže slúžiť bankový účet. Služby bankového účtu môžu byť opísané Java rozhraním *AccountService*, ktoré má metódy ako *createAccount*, *deposit*, *withdraw* a pod. Prezentačná vrstva „vidí“ len tieto rozhrania, ich implementácia je však vnútornou záležitosťou aplikačnej vrstvy. Tento princíp sa nazýva programovanie voči rozhraniam a umožňuje vymieňanie implementácie bez toho, aby si to klient (používateľ rozhraní) „všimol“. To napríklad zjednodušuje testovanie, kedy sa aplikácia môže javiť ako plne funkčná a pri tom napr. vôbec nemusí byť pripojená do databázy.

Na túto činnosť slúži aplikačný rámec Spring⁵. Pomocou neho môžeme konfigurovať celú aplikáciu a spájať jednotlivé moduly. Spring je veľmi silný nástroj a poskytuje veľa užitočných funkcií. My ho však v našom systéme budeme najviac využívať na konfiguráciu servisných služieb, ich prepojenie na objektovo-relačný mapovač a na testovanie modulov (angl. *unit testing*). Na konfiguráciu sa používa, ako je to pri JEE zvykom, formát XML.

4.4 Objektovo-relačné mapovanie

Pod týmto pojmom sa skrýva pomerne jednoduchá vec. Ide o mapovanie objektov do relačnej databázy. Ako nástroj na toto mapovanie sme si zvolili Hibernate⁶, ktorý sa v tejto oblasti stal už de facto štandardom na platforme Java. Zjednodušene sa dá povedať, že každá Java trieda je reprezentovaná tabuľkou v relačnej databáze, každý stĺpec tabuľky reprezentuje atribút triedy a jeden riadok predstavuje konkrétny objekt, teda inštanciu Java triedy. Opäť, na takéto mapovanie, sa používajú metadáta. Tie sú najčastejšie vo forme XML, ale môžeme použiť aj Java anotácie.

Výhodou tohto prístupu je, že programátor pracuje s Java triedami a nemusí sa zaoberať písaním SQL selectov na bežné CRUD operácie. Používatelia Hibernate síce nepoužívajú jazyk SQL, ale jazyk HQL, ktorý je SQL veľmi podobný. Výhodou je, že sa používa tzv. bodková notácia namiesto SQL príkazov join.

⁴ <http://tapestry.apache.org>

⁵ <http://www.springframework.org>

⁶ <http://www.hibernate.org>

Tieto dopyty podporujú aj polymorfizmus, čo sa dá niekedy s výhodou využiť. Ďalšou výhodou je, že Hibernate ukrýva databázový systém pred programátorom, takže ten vlastne ani nemusí vedieť, na akom databázovom systéme je aplikácia nasadená. Rovnako to umožňuje (teoreticky) vymeniť databázový systém bez toho, aby to nejakovo ovplyvnilo aplikáciu. Opäť ide o akúsi transparentnosť keď jedna vrstva skrýva svoje implementačné detaily a poskytuje svojmu klientovi len služby.

Ďalšou abstraktnou vrstvou je použitie Java EE vzoru DAO. DAO je trieda, ktorá predstavuje abstrakciu CRUD operácií, typické metódy DAO sú *save*, *update*, *delete*, *executeQuery* a pod.

4.5 Hodnotové objekty

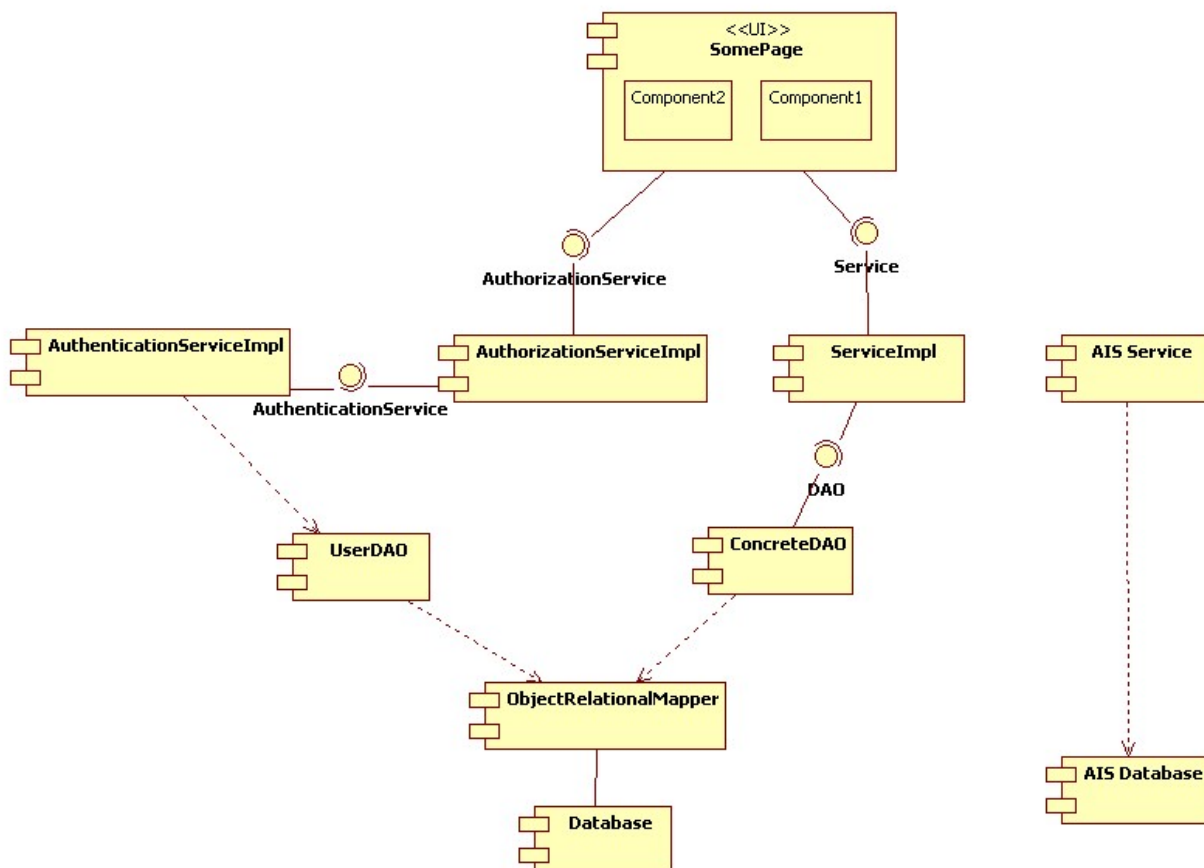
Hodnotové objekty (angl. *value objects*) tvoria doménový model aplikačnej oblasti. Sú to teda objekty aplikačnej domény, ktorú modelujeme. Ide napr. o objekty ako zmluva, faktúra, ale aj abstraktnejšie ako proces, udalosť a pod. Tieto objekty sa používajú naprieč celou aplikáciou, t.j. vo všetkých vrstvách. Tieto objekty sa ukladajú do databázy.

4.6 Servlet kontajner

Webové aplikácie na platforme Java sú väčšinou postavené na Servlet API, čo je zjednodušene povedané obsluha HTTP požiadaviek a vytváranie HTTP odpovedí. Každý servlet, musí bežať v nejakom „prostredí“. Toto prostredie sa nazýva servlet kontajner (angl. *servlet container*). Takýmto servlet kontajnerom je napríklad voľne dostupný aplikačný server Tomcat, ten poskytuje aj ďalšie služby.

4.7 Moduly systému

Na Obr. 5.2 obrázku opisujeme moduly systému a ich závislosti pomocou diagramu komponentov.



Obr. 4.2 Diagram komponentov opisujúci moduly systému Znalec

Na najvyššej úrovni je webová stránka prezentovaná Java triedou `SomePage`. Stránky však poskytujú len používateľské rozhranie, ale ich činnosť závisí od služieb. Služby modelujeme pomocou rozhrania `Service`. V aplikačnej vrstve sa nachádza konkrétna implementácia tejto služby `ServiceImpl`. Služba väčšinou potrebuje pristupovať do databázy, preto závisí od rozhrania `DAO`. `DAO` poskytuje služby databázové služby. Konkrétne `DAO` poskytuje implementáciu týchto služieb. Na implementáciu týchto služieb používa objektovo-relačný mapovač, ktorý pristupuje do databázy. Na zistenie, či má nejaký používateľ práva vidieť nejakú stránku ho potrebujeme autorizovať. Na tu slúži modul `AuthorizationService`, implementuje ho `AuthorizationServiceImpl`. Na to, aby sme mohli používateľa autorizovať, potrebujeme ho najskôr autentifikovať, teda zistiť, kto to vlastne je. Preto Autorizačná služba závisí od Autentifikačnej. Tá zisťuje používateľove role z databázy. Na to využíva `UserDAO`, ktorý do nej pristupuje pomocou `ORM`.

Na import údajov z AIS-u slúži modul `AIS Service`. Ten pristupuje priamo do databázy AIS-u.

5 Prototyp

V tejto časti opisujeme prototyp vytvorený v zimnom semestri.

5.1 Ciele prototypovania

Pri tvorbe prototypu sme si stanovili nasledovné ciele:

- vyskúšať tvorbu dynamických stránok pomocou Tapestry,
- overiť možnosť implementovania entít navrhnutých v logickom modeli,
- vytvoriť služby aplikačnej vrstvy zabezpečujúce biznis logiku a
- implementovať základný scenár použitia systému.

Výsledkom prototypovania by mali byť nájdené nedostatky v špecifikácii a návrhu systému. Tiež by sme chceli získať praktické poznatky o rozsahu systému a množstve práce potrebnej na dokončenie implementácie finálnej verzie systému v letnom semestri.

5.2 Vybrané časti systému na prototypovanie

Rozhodli sme sa vytvoriť evolučný prototyp, t. j. taký, ktorý v letnom semestri rozšírime do výsledného systému. V tejto časti sme sa rozhodli prototypovať aplikačnú vrstvu zabezpečujúcu biznis logiku a prezentačnú vrstvu umožňujúcu používateľovi prístup do systému. Prototyp nespôlpracuje s databázou, všetky údaje sú vytvárané a uchovávané počas vykonávania v pamäti. Taktiež zatiaľ nespôlpracuje so systémom AIS, nakoľko prístup k tomuto systému sme dostali až tesne pred koncom semestra. Prototyp funguje len lokálne na našich vývojových počítačoch. Nie je zatiaľ nasadený na aplikačnom serveri. Výsledný systém bude nasadený serveri v rámci fakulty.

5.2.1 Scenár použitia systému realizovaný prototypom

Vytvorili sme základný scenár použitia výsledného systému, ktorého jednotlivé kroky sme sa rozhodli implementovať v prototypu. Tento scenár nám poskytol užitočný pohľad na systém ako celok. Scenár pozostáva z nasledujúcich krokov:

1. Používateľ si zobrazí úvodnú stránku systému a prihlási sa.
2. Jeho domovská stránka mu poskytne menu pozostávajúce z nasledovných položiek: pridanie nového používateľa, editovanie profilu, správa faktov, zobrazenie znalostí, odhlásenie.
3. Prvá položka umožní pridať do systému nového používateľa vyplnením základných údajov.
4. V časti editovanie profilu má prihlásený používateľ možnosť upraviť svoje údaje.
5. V časti správa faktov si môže používateľ vytvárať, editovať a mazať svoje fakty.
6. Dôležitou položkou je tiež zobrazenie znalostí, ktorá používateľovi ukáže zoznam znalostí, ktoré mu systém na základe vyplnených faktov pridelil.

V tomto scenári ešte nerozlišujeme medzi používateľmi, t. j. neberieme do úvahy ich roly. Prototyp zatiaľ nie je konfigurovateľný, typy faktov ako aj znalosti sú napevno nastavené v zdrojovom kóde.

5.2.2 Prípady použitia vybrané na prototypovanie

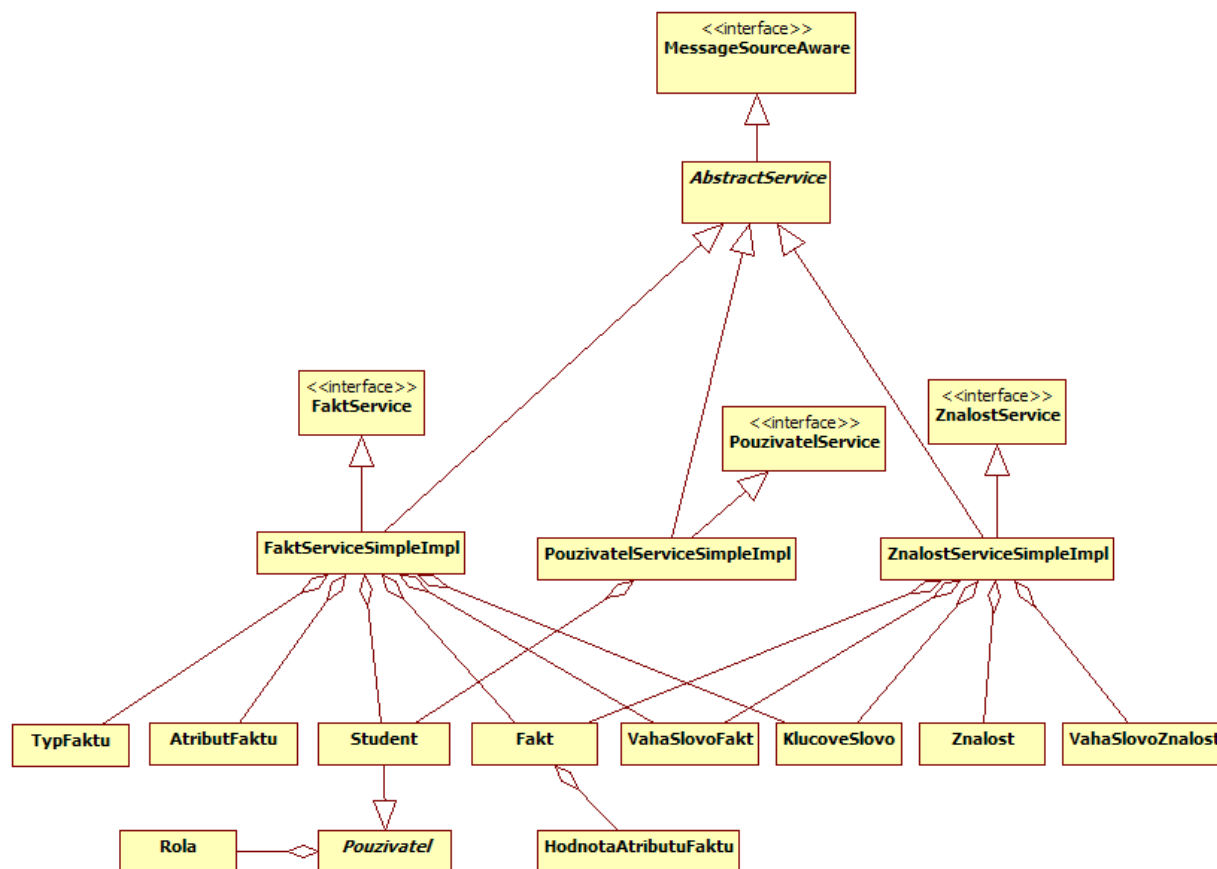
Pri vyberaní častí systému do prototypu sme postupovali podľa definovaných prípadov použitia. Vybrali sme tie, ktoré najviac demonštrujú zaujímavé vlastnosti systému. Niektoré z týchto prípadov použitia sme neimplementovali úplne, napr. z dôvodu, že obsahujú aj prístup do databázy alebo prístup k údajom

z externého fakultného systému. Tieto funkcie sme nahradili dočasnými údajmi, ktoré sme zapísali priamo v zdrojovom kóde prototypu. Na prototypovanie sme vybrali nasledujúce prípady použitia:

- UC01 Správa osobného profilu
- UC02 Správa faktov
- UC03 Pridanie faktu
- UC05 Zobrazenie znalostí študenta
- UC06 Vyhodnotenie znalostí študenta
- UC07 Zobrazenie profilu študenta
- UC08 Zobrazenie faktov študenta

5.3 Aplikačná vrstva

Ťažiskom prototypu je aplikačná vrstva. V nej sme sa snažili implementovať hlavné služby, ktoré táto vrstva poskytuje prezentačnej vrstve. Služby sme implementovali ako rozhrania a od nich od nich odvodené triedy, ktoré zabezpečujú potrebnú funkcionálnosť. Na implementáciu služieb sme použili rámec Spring, ktorý umožňuje jednoducho spájať vytvorené moduly do výsledného systému. Služby sme navrhli tak, aby bolo v ďalšej fáze možné nahradiť statické údaje údajmi z databázy. Na to bude stačiť implementovať príslušné metódy, ktoré vytiahnu údaje z databázy. Spracovanie údajov ani ich poskytnutie prezentačnej vrstve sa však nezmení. Na Obr. 5.1 je zobrazený diagram tried aplikačnej vrstvy a ich závislosti. Hlavné časti tvoria triedy implementujúce rozhrania služieb: FaktServiceSimpleImpl, PouzivatelServiceSimpleImpl a ZnalostiServiceSimpleImpl. Tieto služby v sebe agregujú ďalšie triedy (TypFaktu, AtributFaktu, Student, Fakt, VahaSlovoFakt, KlucoveSlovo, Znalost, VahaSlovoZnalost), ktoré predstavujú implementáciu logického modelu údajov.



Obr. 5.1 Diagram tried aplikačnej vrstvy

Tieto služby sú cez svoje rozhrania sprístupnené prezentačnej vrstve. Nasleduje stručný opis jednotlivých rozhraní.

5.3.1 FaktService

Toto rozhranie slúži na manipuláciu s typmi faktov v systéme, najmä na ich pridávanie. Umožňuje tiež pridávať a editovať fakty jednotlivým študentom. Prezentačnej vrstve poskytuje zoznam všetkých typov faktov, ktoré sú v systéme, ako aj zoznam faktov zadaného študenta. Fakty sú v tomto prípade zoskupené podľa typu.

5.3.2 PouzivatelService

Rozhranie PouzivatelService slúži na manipuláciu s používateľmi. Umožňuje pridať nového používateľa do systému a upraviť profil existujúceho používateľa. Taktiež poskytuje funkcionality autentifikácie používateľa, ktorá sa využíva pri procese prihlasovania sa do systému.

5.3.3 ZnalostService

Toto rozhranie slúži na získanie znalostí zadaného študenta. Porovnáva pri tom všetky kľúčové slová jednotlivých faktov študenta s kľúčovými slovami definovanými v znalostiach. Pri zhode priradí túto znalosť študentovi. V ďalšej časti implementácie dopracujeme výpočet miery získania danej znalosti, kedy budú do úvahy brané aj typy jednotlivých faktov a ich príspevky k znalostiam. Študent potom nebude mať pridelenú iba konkrétnu znalosť, ale aj percentuálnu mieru jej splnenia. To umožní usporadúvať študentov.

5.4 Prezentačná vrstva

Prezentačnú vrstvu tvoria dynamické HTML stránky, ktoré zobrazujú používateľovi informácie a prijímajú od neho príkazy a zadané údaje. Prezentačná vrstva využíva verejné metódy služieb z aplikačnej vrstvy, aby jej tak odovzdala údaje na spracovanie a dostala od nej výsledky. Využitím rámca Tapestry je z týchto výsledkov vygenerovaná dynamická stránka, ktorá je prostredníctvom aplikačného servera Tomcat zobrazená používateľovi.

5.5 Testovanie

V rámci testovania prototypu sme uplatnili dva prístupy:

- testovanie typu čierna skrinka a
- regresné testovanie.

V rámci testovania typu čierna skrinka sme postupne vykonávali jednotlivé kroky špecifikované v prototypovaných prípadoch použitia (s ohľadom na scenár prototypu). Takto sme odhalili chyby v dizajne, ako aj v aplikačnej logike, ktoré sme následne odstraňovali.

V rámci regresného testovania sme využili možnosti nástroja JUnit. Vytvorili sme niekoľko testov, ktoré testujú služby aplikačnej vrstvy. JUnit test vykonáva metódy vybranej služby s predpripravenými testovacími údajmi. Výsledné údaje, ktoré mu služba poskytne, porovnáva s očakávanými údajmi. Ak sa nezhodujú, skončí test s chybou. Nástroj JUnit umožňuje vytvoriť ľubovoľný počet testov a následne ich spúšťať buď jednotlivito alebo viaceru naraz. Výhoda týchto testov spočíva v tom, že ich môžeme bezo zmien použiť pri implementovaní prístupu systému k databáze. Pri jeho implementovaní doplníme metódy služieb o získavanie údajov z databázy, vstupné a výstupné údaje sa však nezmenia. Toto považujeme za veľkú výhodu JUnit testov, pre ktorú sme sa ich rozhodli využiť.

5.6 Dosiahnuté výsledky

Môžeme skonštatovať, že sa nám podarilo splniť všetky ciele, ktoré sme si v úvode prototypovania stanovili. Podarilo sa nám vytvoriť funkčný prototyp demonštrujúci základné funkcie budúceho systému. Úspešne sme implementovali dynamické stránky prezentačnej vrstvy pomocou rámca Tapestry. Vytvorili sme hlavné služby aplikačnej vrstvy, ktoré v ďalšej fáze vývoja doplníme o prístup k údajom v databáze. Úspešne sme tiež implementovali jednotlivé entity logického modelu údajov. Jediný problém, ktorý sme v tejto časti identifikovali, spočíva v návrhu previazania znalostí a faktov na kľúčové slová prostredníctvom váh. V ďalšej časti budeme musieť rozhodnúť, ako má byť toto previazanie realizované (v prototypu sme ho zatiaľ vyriešili staticky pri zadávaní znalostí). Taktiež sme narazili na problém identifikovania kľúčových slov z jednotlivých faktov. Oba problémy však pokladáme za ľahko vyriešiteľné v letnom semestri.

Použité zdroje

1. Šimún, M., Andrejko, A., Bieliková, M.: Inicializácia a aktualizácia modelu používateľa pri hľadaní pracovných ponúk na webe. In Znalosti 2007, P. Mikulecký, J. Dvorský, M. Krátký (Eds.), Ostrava, Czech Republic (2007) 190-120.
2. Tím_elf_: Bába znalostí a zručností študentov – Projektová dokumentácia. FIIT STU, máj 2006 [cit. 2008-10-20]. Dostupný z WWW: <http://labss2.fiit.stuba.sk/TeamProject/2005/team11/documents/_elf_LS_Projekt_v1.0.doc>.
3. Tím Černé ofce: Bába znalostí a zručností študentov – Dokumentácia k projektu. FIIT STU, máj 2008 [cit. 2008-10-20]. Dostupný z WWW: <<http://labss2.fiit.stuba.sk/TeamProject/2007/team10is-si/files/dok-leto.pdf>>.