Dokumentácia k inžinierskemu dielu po 8. šprinte

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Znalosti a zručnosti študentov

Tímový projekt

Dokumentácia k inžinierskemu dielu

Tím č.16 – WeKnowIT

Bc. Jakub Šalmík

Bc. Patrik Polakovič

Bc. Michal Chylik

Bc. Peter Ivanec

Bc. Anton Szórád

Vedúci tímového projektu: Ing. Michal Holub
Akademický rok: 2011/2012

Obsah

5. Piaty šprint – 40 ročná vojna

V rámci piateho šprintu sme riešili nasledovné používateľské príbehy:

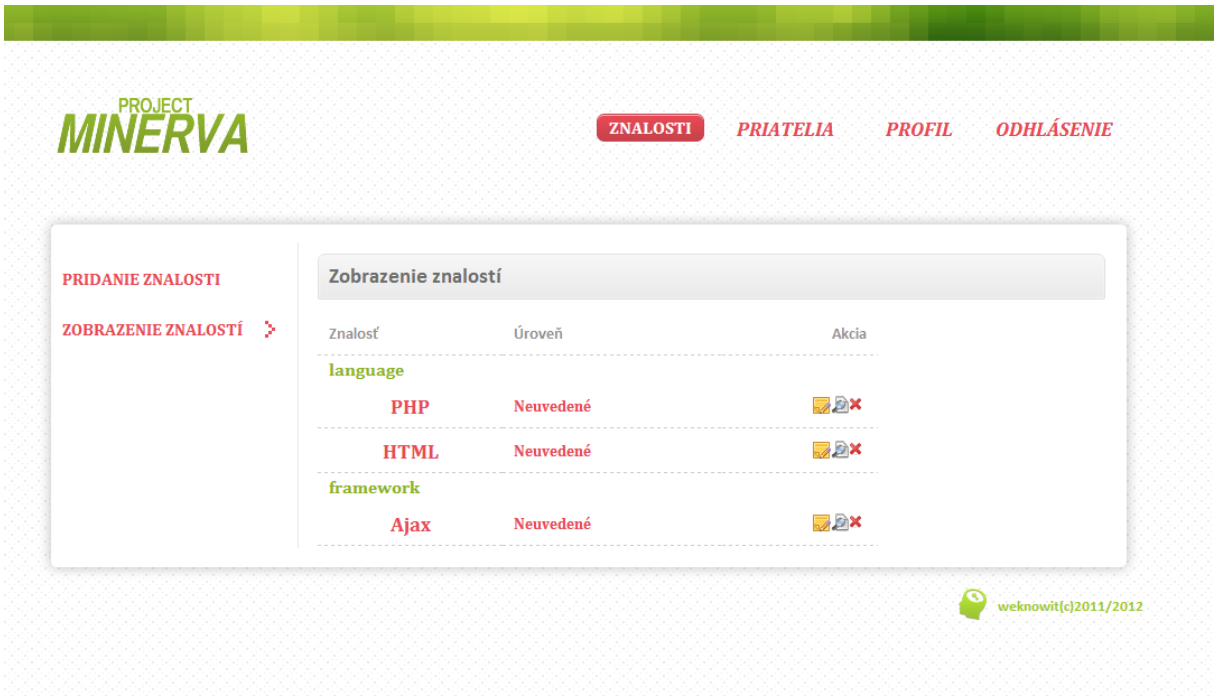
- US20 – Vymazávanie zručností,
- US21 – Voliteľná úroveň znalostí,
- US22 – Pridávanie poznámok.

5.1. US20 – Vymazávanie zručností

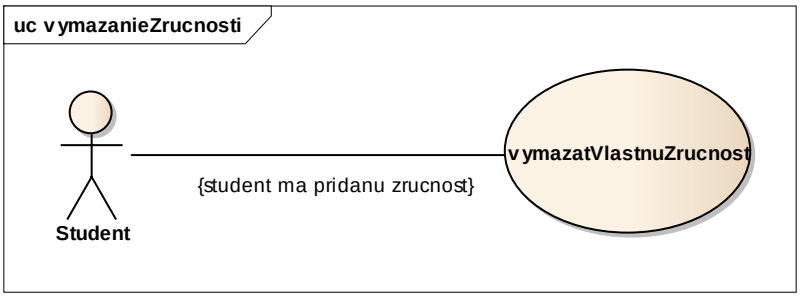
Cieľom používateľského príbehu je umožniť študentovi vymazať si zručnosť, ktorú má už pridanú. Na vymazanie slúži ikonka červeného krížika v zozname vlastných zručností.

5.1.1. Analýza

Najvhodnejším miestom na pridanie ikonky prostredníctvom ktorej môže používateľ vymazávať vlastné zručnosti je zoznam všetkých zručností. Tento zoznam sa nachádza na obrazovke znázornenej na obrázku číslo 5.1.1 - Zobrazenie zručností.



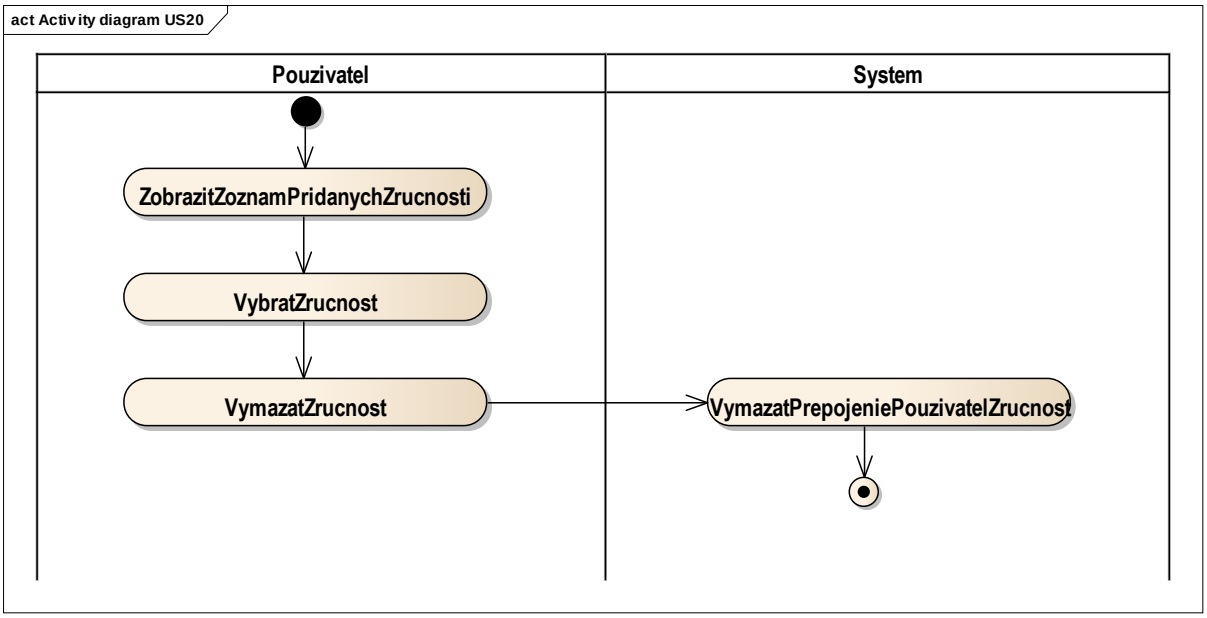
Obr. č. 5.1.1 – Obrazovka zobrazenia zručností



Obr. č. 5.1.2 – Prípad použitia vymazanieZručnosti

Prípad použitia	vymazanieZručnosti
ID	5.1
Stručný popis	Používateľ má možnosť vymazať si zručnosť, ktorú má pridanú.
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený, Používateľ je študent, Používateľ má pridanú zručnosť, ktorú chce vymazať
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky Zobrazenie zručností. 2. Používateľ klikne na ikonu červeného krížika pri zručnosti, ktorú chce vymazať. 3. Systém vymaže prepojenie používateľa s danou zručnosťou.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

5.1.2. Návrh



Obr. č. 5.1.3 - Diagram aktivít pre vymazanie zručnosti

5.1.3. Implementácia

Implementácia bola vykonaná v dvoch súboroch. V ovládači SkillController bola umiestnená metóda na vymazanie zručnosti používateľa.

```
//metoda vymaze zrucnost priradenu pouzivatelovi, id zrucnosti a id
pouzivatela su zadane ako parametre
      public function deleteSkillOfUser($id_skill, $id_user){

            $connection = Yii::app()->db;
            $connection->createCommand()->delete('user_skill',
'id_user=:id_user and id_skill=:id_skill', array(':id_user'=>$id_user,
':id_skill'=>$id_skill));

            return 1;

      } // end public function deleteSkillOfUser
```

Metódu voláme nasledovne v súbore *showSkill.php*:

```
$this->deleteSkillOfUser($id_skill,$id_user);
```

5.1.4. Testovanie

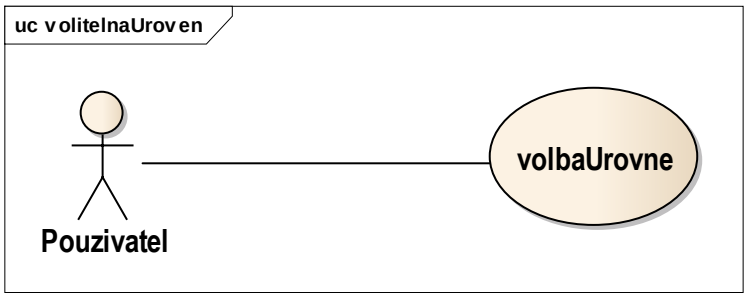
Názov	Úspešné vymazanie vlastnej zručnosti		**ID Testu**	5.1-01
Rozhranie	Obrazovka Zobrazenie zručností		**ID UC**	5.1
Účel	Vymazanie vlastnej zručnosti			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Klinutie na ikonku červeného krížika pri zručnosti v zozname.	Vymazanie prepojenia zručnosti na používateľa. Zručnosť zmizne zo zoznamu zručností používateľa.	Vymazalo sa prepojenia zručnosti na používateľa. Zručnosť zmizla zo zoznamu zručností používateľa.	

5.2. US21 – Voliteľná úroveň znalostí

Pri pridávaní znalostí chcem mať možnosť voľby, či chcem príslušnú znalosť ohodnotiť určitou úrovňou, na akej danú znalosť ovládam.

5.2.1. Analýza

Používateľ by mal mať pri pridávaní znalostí k svojej osobe možnosť nielen určitým spôsobom vyjadriť, do akej miery ovláda tú ktorú znalosť, ale sa aj rozhodnúť, či úroveň ovládania danej znalosti chce konkretizovať alebo nie.



Obr. č. 5.2.1 – Prípad použitia *voľba úrovne znalosti*

Prípad použitia	volbaUrovne
ID	5.2
Stručný popis	Používateľ má možnosť ohodnotiť znalosť úrovňou.
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny

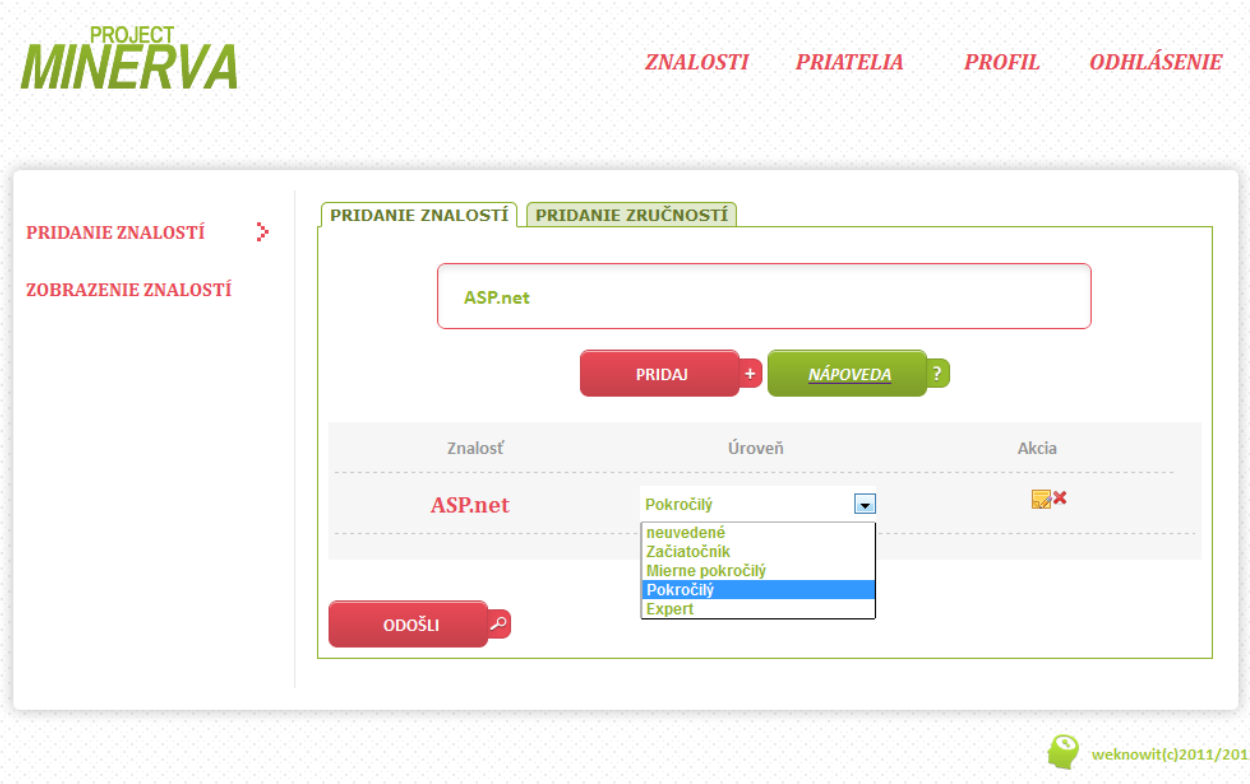
Vstupné podmienky	Používateľ je prihlásený, Používateľ má zvolenú znalosť, ktorú chce pridať
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky Pridanie znalostí. 2. Používateľ napíše časť alebo celý názov znalostí do formuláru. 3. Systém zobrazí ponuku znalostí vyhovujúcich vyhľadávaniu. 4. Používateľ pridá znalosť do radu. 5. Používateľ vyberie úroveň ovládania danej znalosti a potvrdí pridanie. 6. Systém zvaliduje požiadavku a pridá ohodnotenú znalosť používateľovi.
Výstupné podmienky	Žiadne
Alternatívne scenáre	5a. Používateľ si nevyberia žiadnu úroveň a potvrdí pridanie. 6a. Systém zvaliduje požiadavku a pridá znalosť bez hodnotenia používateľovi.

5.2.2. Návrh

Do existujúcej stupice štyroch úrovní sme sa rozhodli pridať piatu úroveň - *neuvedené*, ktorá neodráža používateľove schopnosti v danej znalosti.

Kým ostatné úrovne majú číselné ohodnotenia od 1 do 4, kde 1 je najhoršie a 4 najlepšie, číselná hodnota neuvedenej úrovne je 0, čo ju jasne identifikuje.

Na obrázku číslo 5.2.2 je znázornený grafický návrh zadávania úrovne ku príslušnej znalosti v systéme.



Obr. č. 5.2.2 – Obrazovka voliteľnej úrovne znalostí

5.2.3. Implementácia

Pri pridávaní znalostí sa vybrané znalosti najprv dynamicky začlenia do radu na obrazovke, až nakoniec sa hromadne pridajú ďalším potvrdením.

Táto funkcionalita je riešená pomocou javascriptu. Nasledujúci kód znázorňuje akým spôsobom sú zobrazené úrovne pri daných znalostiach.

```
var opt = document.createElement('option');
      opt.innerHTML = 'neuvedené';
      opt.setAttribute('value','0');
      sel.appendChild(opt);

      var opt = document.createElement('option');
      opt.innerHTML = 'Začiatočník';
      opt.setAttribute('value','1');
      sel.appendChild(opt);

      var opt = document.createElement('option');
      opt.innerHTML = 'Mierne pokročilý';
      opt.setAttribute('value','2');
      sel.appendChild(opt);

      var opt = document.createElement('option');
      opt.innerHTML = 'Pokročilý';
      opt.setAttribute('value','3');
      sel.appendChild(opt);

      var opt = document.createElement('option');
```

```
        opt.innerHTML = 'Expert';
        opt.setAttribute('value','4');
        sel.appendChild(opt);
```

5.2.4. Testovanie

Názov	Zvolenie úrovne znalosti		**ID Testu**	5.2-01
Rozhranie	Obrazovka pridávania znalostí		**ID UC**	5.2
Účel	Nastavenie úrovne danej znalosti			
Vstupné podmienky	Vybratá znalosť do radu.			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Zobrazenie úrovní pre znalosť.	Po kliknutí sa nám zobrazia úrovne pre danú znalosť.	Po kliknutí na combobox sa zobrazil zoznam úrovní pre zvolenú znalosť.	
2.	Potvrdenie pridania znalosti.	Po potvrdení pridania znalostí sa uloží do databázy aj úroveň.	Úroveň bola uložená k príslušnému používateľovi a znalosti.	

5.3. US22 – Pridávanie poznámok

Ako študent chcem pridať poznámku k zadanej znalosti. Poznámka je voľný text, jej dĺžka môže byť obmedzená (napr. 160 znakov ako SMS). K jednému záznamu o znalosti môžem pridávať ľubovoľný počet poznámok, pričom sa ukladá čas ich pridania.

5.3.1. Analýza

Pridávanie poznámok ku samotným znalostiam môže zlepšiť obraz o tom ako študent danú znalosť ovláda. Môže uviesť projekty alebo práce, ktoré s danou znalosťou vypracoval a pridaním viacerých poznámok môže uviesť pokrok, ktorý v danej znalosti dosiahol. Pri poznámkach sa objaví dátum a čas kedy tieto poznámky pridal.

Prípad použitia	**PridávaniePoznámok**
ID	5.3
Stručný popis	Používateľ si pridáva poznámky ku znalostiam pri ich pridávaní.
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny

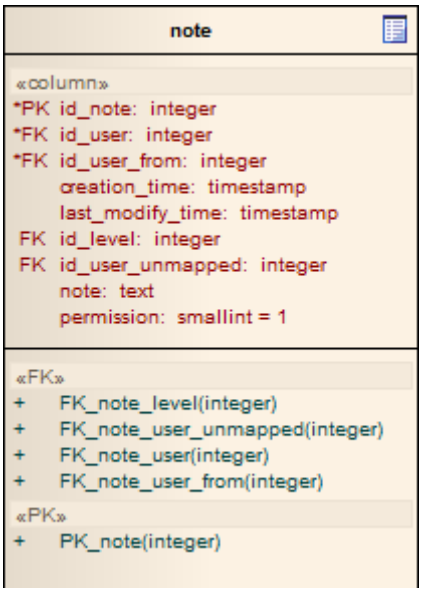
Vstupné podmienky	Používateľ je prihlásený
Hlavný scenár	1. Prípad použitia začína po zobrazení formuláru na pridávanie znalostí. 2. Používateľ si vyberie znalosť ktorú chce pridať. 3. Klikne na ikonku poznámky a následne sa mu zobrazí textové pole, kam môže písať poznámku k znalosti. 4. Používateľ odošle údaje a tým aj poznámku ku znalosti. 5. Systém spracuje formulár a uloží údaje do databázy.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

5.3.2. Návrh

Na obrázku 5.3.1 je zobrazená obrazovka pridania poznámky k vlastnej znalosti. Textové pole sa zobrazí po kliknutí na ikonku poznámky.

Obr. č. 5.3.1 – Obrazovka pridania poznámky ku znalosti

Pre ukladanie poznámky bola vytvorená entita *note:*



Obr. č. 5.3.2 – Databázová tabuľka *Note*

Používané atribúty sú nasledovné:

Názov atribútu	Popis
id_note	PK
id_user	ID používateľa, ktorému bola poznámka priradená
id_user_from	ID používateľa, ktorý poznámku priradil
creation_time	čas vytvorenia poznámky
id_level	ID vybranej znalosti/zručnosti, ku ktorej sa poznámka vzťahuje
id_user_unmapped	ID neznámej znalosti/zručnosti, ku ktorej sa poznámka vzťahuje
note	samotný text poznámky

5.3.3. Implementácia

Podstatná časť kódu je naprogramovaná v javascripte. Ide o používateľsky príjemné zobrazenie dialógu pre zadanie poznámky, kde sa dynamicky vytvárajú elementy pre zobrazenie poznámky.

Databázovou časťou bolo použitie *CActiveRecord* modelu pre uloženie získaného obsahu od používateľa:

```
if($_POST[$note_name][$index] != '') {
	$note = new Note;
	$note->id_user_from = Yii::app()->user->getId();
	$note->id_user = $id;
	$note->creation_time = new CDbExpression('now()::timestamp');
	$note->last_modify_time = new CDbExpression('now()::timestamp');
	$note->note = $_POST[$note_name][$index];
	$note->id_level = $user_level->id_level;
	$note->save();
}
```

5.3.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zapísanie poznámky ku znalosti	**ID Testu**	5.3-01
Rozhranie	Obrazovka pridávanie znalostí	**ID UC**	5.3
Účel	Zistenie, či sa poznámka zapíše k znalosti		
Vstupné podmienky	Napísaná poznámka v textovom poli		
Výstupné podmienky	Žiadne		

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**
1.	Kliknutie na ikonku poznámky.	Zobrazenie textového poľa na vkladanie poznámky.	Textové pole na pridanie poznámky zobrazené.
2.	Zadanie poznámky do textového poľa.	Zobrazenie textu poznámky v textovom poli.	Text poznámky je zobrazený v textovom poli.
3.	Kliknutie na tlačidlo “Pridaj”	Zobrazenie oznamu že znalosť bola úspešne zapísaná do systému.	Oznam o zapísaní do systému zobrazený.

6. Šiesty šprint – Husitská vojna

V rámci šiesteho šprintu sme riešili nasledovné používateľské príbehy:

- US23 – Poznámkovanie znalostí iného používateľa,
- US24 – Komentovanie používateľov,
- US25 – Zručnosti.

6.1. US23 – Poznámkovanie znalostí iného používateľa

Ako používateľ chcem mať možnosť pridať poznámku ku znalosti iného študenta.

6.1.1. Analýza

Pridávanie poznámok ku znalostiam iba samotným študentom môže byť v mnohých prípadoch neobjektívne a skreslené. Pridaním možnosti pridania poznámky buď vyučujúcim alebo iným študentom poskytneme možnosť zobjektívniť názor na skutočné ovládanie znalostí, ktoré si student zadal.

Prípad použitia	**PridaniePoznámkyInémuPoužívateľovi**
ID	6.1
Stručný popis	Používateľ pomocou textového poľa zadá poznámku k príslušnej znalosti. Textové pole sa mu zobrazí po kliknutí na ikonu.
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Prihlásený používateľ
Hlavný scenár	1. Prípad použitia začína pri zobrazení zoznamu znalostí používateľa. 2. Používateľ klikne na ikonu poznámky, čím sa mu zobrazí textove pole. 3. Používateľ zadá poznámku do poľa. 4. Používateľ odošle poznámku. 5. Systém spracuje poznámku a zapíše ju do systému.
Výstupné	Prihlásený používateľ.

podmienky	
Alternatívne scenáre	žiadne

6.1.2. Návrh

Na obrázku číslo 6.1.1 je zobrazená obrazovka pre pridanie poznámky inému používateľovi. Ikona poznámky sa zobrazí iba pri zobrazení cudzieho profilu a nie vlastného.

Obr. č. 6.1.1 – Obrazovka pridania poznámky ku znalosti iného používateľa

6.1.3. Implementácia

Pre príjemné a jednoduché pridávanie poznámok ku znalostiam iného používateľa bol implementovaný ajax formulár. Ten odosiela zadané poznámky na server bez potreby opätovného načítania stránky.

```
<?php echo CHtml::beginForm(); ?>
<?php echo CHtml::textArea('note_content', '', array('class' =>
'profile_note')); ?>
      <input name="CommentLevelId" type="hidden" value="<?php echo
$subdata['id_level']; ?>" />
      <br />
```

```
      <?php echo CHtml::ajaxSubmitButton(
            "Odošli",
            array('ajax/storeNote', 'id' => $_GET['id'], 'lid' =>
$subdata['id_level'], 'unmp' => 'false'),
            array('success'=>'js:function(data){
                  var newnodes =
$(data).appendTo("#tr_'.$subdata['id_level'].'")
                  showNote("note_'.$subdata['id_level'].'");
                  unhideComments("comm_'.$subdata['id_level'].'");

      $("#tr_'.$subdata['id_level'].'").treeview({add:newnodes});
            }'),
            array('class'=>'note_send')
      );
?>
```

Na strane servera sa vykonalo len jednoduché uloženie poznámky do databázy.

6.1.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zapísanie poznámky ku znalosti iného používateľa	**ID Testu**	6.1-01
Rozhranie	Obrazovka znalostí iného používateľa	**ID UC**	6.1
Účel	Zistenie, či sa poznámka zapíše k znalosti iného používateľa		
Vstupné podmienky	Napísaná poznámka v textovom poli		
Výstupné podmienky	Žiadne		

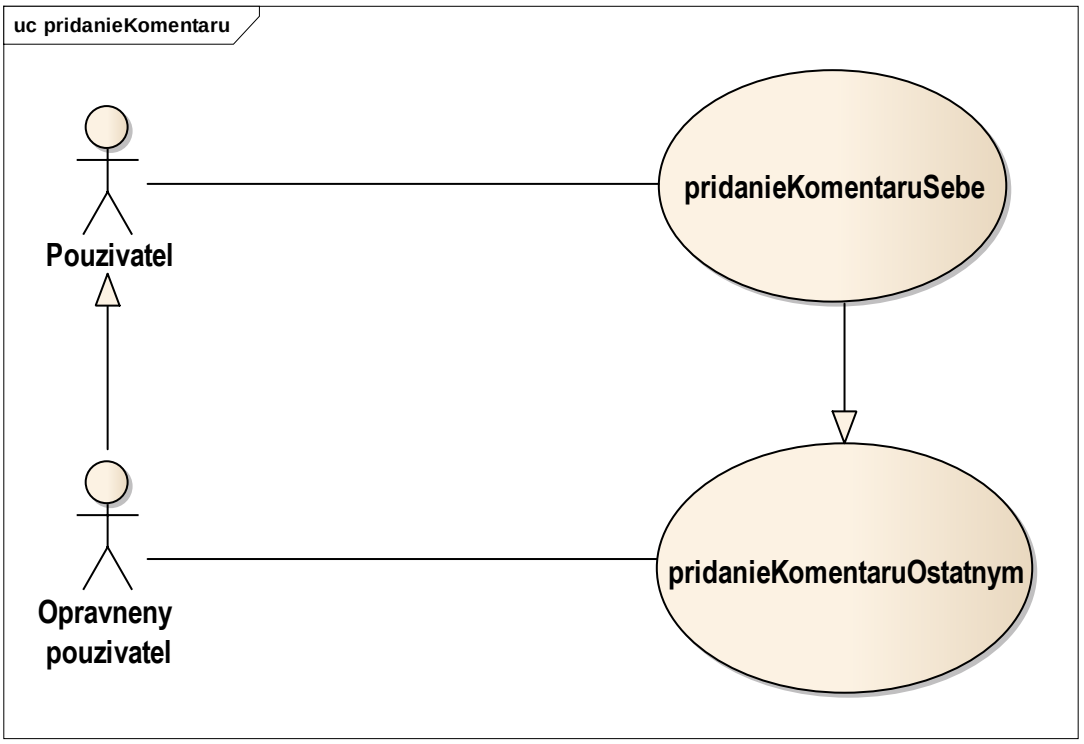
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**
1.	Kliknutie na ikonku poznámky.	Zobrazenie textového poľa na vkladanie poznámky.	Tetxtové pole na pridanie poznámky je zobrazené.
2.	Zadanie poznámky do textového poľa.	Zobrazenie textu poznámky v textovom poli.	Text poznámky je zobrazený v textovom poli.
3.	Kliknutie na tlačidlo “Odošli”	Zobrazená poznámka pri znalosti.	Poznámka zobrazená pri znalosti.

6.2. US24 – Komentovanie používateľov

Ako oprávnený používateľ systému chcem mať možnosť okomentovať seba alebo iného používateľa.

6.2.1. Analýza

Možnosť vyjadriť sa k svojej osobe, alebo inému používateľovi systému môže mať dôležitú informačnú hodnotu, ktorú by používateľ nemal kam inam zadať. Môže sem zadať svoje koníčky, záľuby, výhody, v podstate všetky *soft skills*.



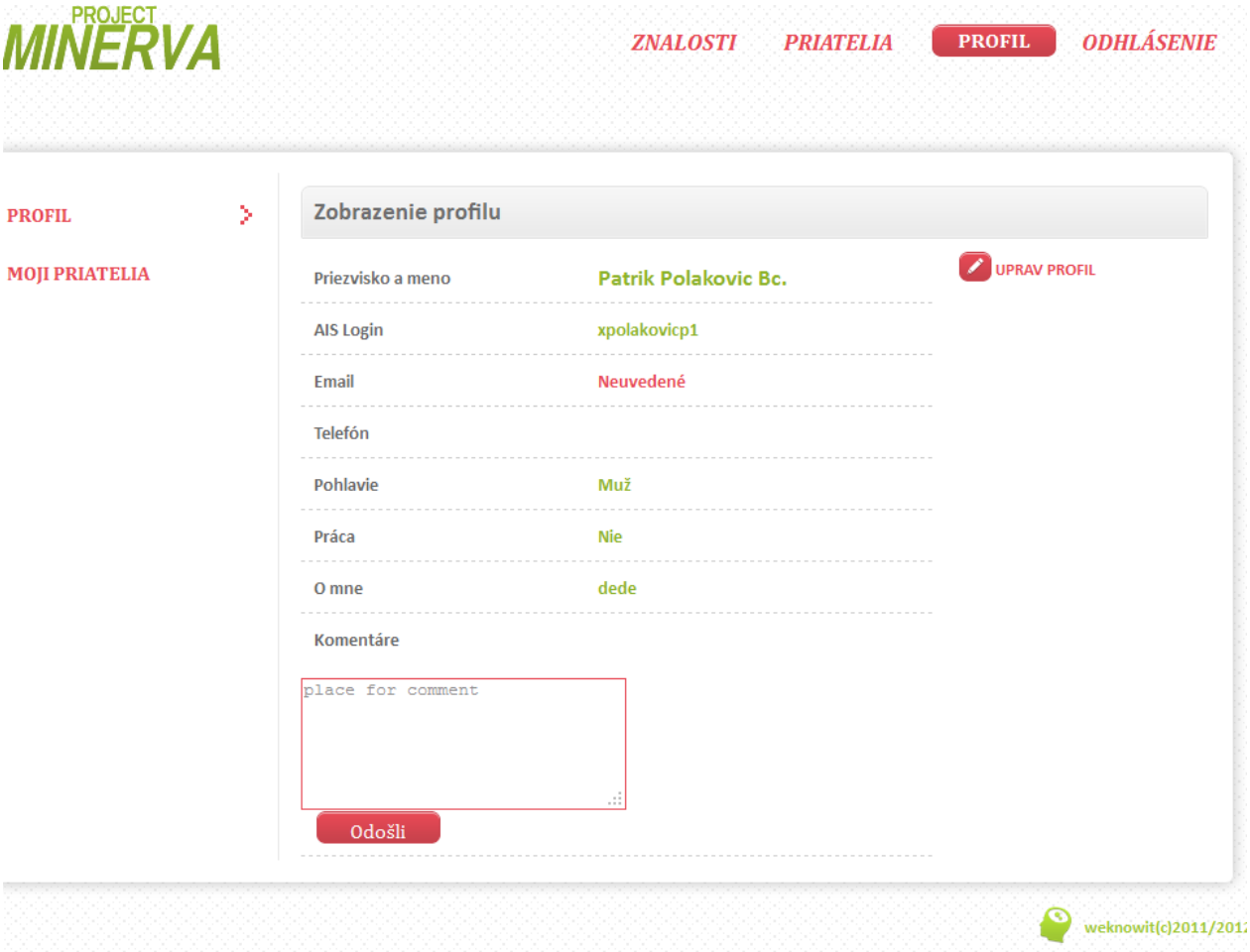
Obr. č. 6.2.1 – Prípady použitia *pridanieKomentaru*

Prípad použitia	**pridanieKomentaru**
ID	6.2
Stručný popis	Používateľ pomocou textového poľa zadá komentár k príslušnému používateľovi.
Primárny aktéri	Prihlásený používateľ, Oprávnený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Prihlásený používateľ
Hlavný scenár	1. Prípad použitia začína pri zobrazení profilu používateľa 2. Používateľ klikne na textové pole, napíše komentár a potvrdí uloženie. 3. Systém uloží komentár k používateľovi a do databázy.

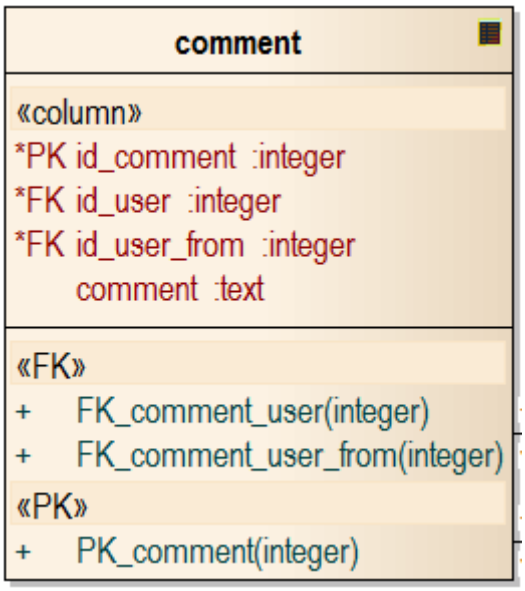
Výstupné podmienky	Prihlásený používateľ.
Alternatívne scénáre	žiadne

6.2.2. Návrh

Ako je znázornené na obrázku 6.2.2, pri zobrazení profilu je k dispozícií textové pole pre zadanie komentáru. Ukladanie do databázy je reprezentované tabuľkou *Comment*, ktorá je zobrazená na obrázku 6.2.3.



Obr. č. 6.2.2 – Obrazovka profilu s komentovaním



Obr. č. 6.2.3 – Tabuľka *comment*

Tabuľka má nasledovné atribúty:

Názov atribútu	Popis
id_comment	PK
id_user	ID používateľa, ktorému bol komentár priradený
id_user_from	ID používateľa, ktorý komentár priradil
comment	text komentáru

6.2.3. Implementácia

Samotnáé políčko pre komentár je riešené pomocou Ajaxu, ktorý zabezpečí spracovanie a uloženie komentáru do databázy bez nutnosti načítať znovu stránku.

```
<tr class='profile_tr'>
<td colspan=2>
      <?php echo CHtml::beginForm(); ?>
      <?php echo CHtml::textArea('comment_content', '', array('class' =>
'profile note')); ?>
      <?php echo CHtml::ajaxSubmitButton(
            "Odošli",
            array('ajax/storeComment', 'id' => $model->id_user),
            array('success'=>'js:function(data){
                  var newnodes = $(data).appendTo("#comm_'.$model-
>id_user.'");

            $("#comm_'.$model->id_user.'").treeview({add:newnodes});
            }'),
            array('class'=>'note_send'));
      ?>
      </div>
</td>
</tr>
<?php echo CHtml::endForm(); ?>
```

6.2.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zapísanie komentáru k používateľovi		**ID Testu**	6.2-01
Rozhranie	Obrazovka profilu používateľa		**ID UC**	6.2
Účel	Okomentovanie seba alebo iného používateľa			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Napísanie a potvrdenie komentáru	Komentár bude spracovaný a uložený	Napísaný a potvrdený komentár je okamžite dostupný na stránke a súčasne aj v databáze.	

6.3. US25 – Zručnosti

Ako používateľ chcem mať k dispozícii zručnosti, aby som s nimi mohol pracovať.

6.3.1. Analýza

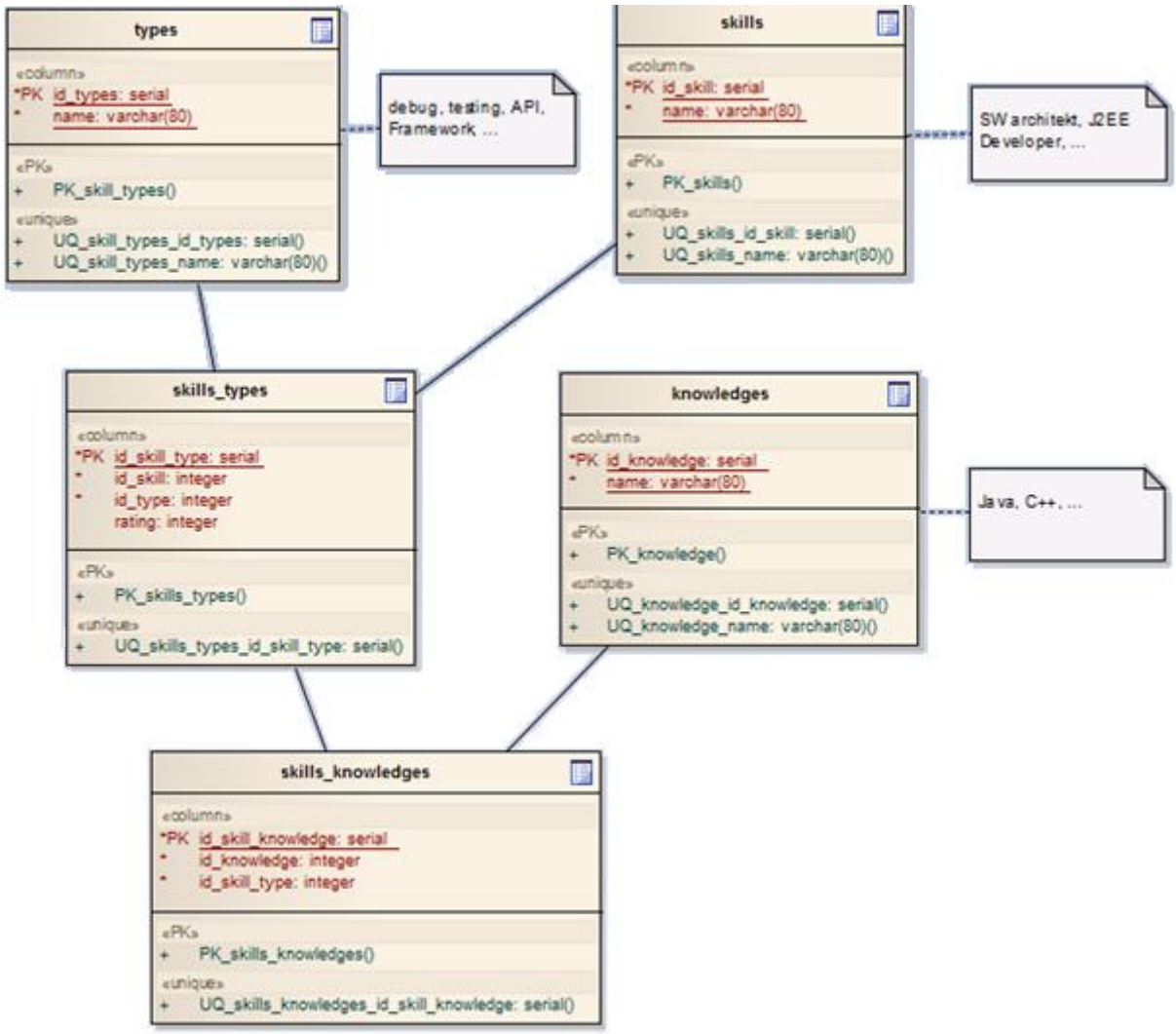
Dôležité je uvedomiť si, čo je *zručnosť* a v čom sa odlišuje oproti *znalosti*.

KSSJ opisuje *znalosť* (angl. knowledge) ako „ovládanie niečoho, vedomosti o niečom na základe štúdia, skúsenosti“. Tejto definície sme sa držali aj pri integrácii znalostí do nášho projektu. Príklad znalosti v našom projekte je Java, PHP, MySQL,...

Zručnosť (angl. skill) vnímame skôr už ako profesie, resp. zamestnania. Nechali sme sa inšpirovať portálom *profesia.sk*, kde sme našli zručnosti ako napr. IT Analytik, PHP Developer, .NET Developer a podobne. Zručnosti popisujeme pomocou množiny znalostí prislúchajúce k danej zručnosti. Teda na to, aby si používateľ mohol zadať zručnosť *IT Analytik*, mal by vedieť znalosti popisujúce zručnosť IT Analytik, a to sú znalosti Enterprise Architect, UML, BPMN, MS Office,...

6.3.2. Návrh

Fyzický model, ktorý sme navrhli pre túto funkcionalitu môžeme vidieť na obrázku číslo 6.3.1.



Obr. č. 6.3.1 – Fyzický model entít Zručnosť a Znalosť

Každá zručnosť je určitého typu, môže mať aj viac typov. Napr. zručnosť *tester* je typu *testing* a *framework*, nakoľko pri testovaní sa používajú rôzne frameworky.

6.3.3. Implementácia

```
CREATE SEQUENCE skill_id_skill_seq INCREMENT 1 START 1
;

CREATE TABLE skill (
      id_skill integer DEFAULT
nextval(('skill_id_skill_seq'::text)::regclass) NOT NULL,
      name varchar(80) NOT NULL
)
WITH (
  OIDS=FALSE
)
;

CREATE SEQUENCE skill_category_id_skill_category_seq INCREMENT 1 START 1
;

CREATE TABLE skill_category (
      id_skill_category integer DEFAULT
```

```
nextval(('skill_category_id_skill_category_seq'::text)::regclass) NOT NULL,
      id_skill integer NOT NULL,
      id_category integer NOT NULL
)
WITH (
  OIDS=FALSE
)
;
```

6.3.4. Testovanie

Keďže sa jedná o automaticky generovaný SQL skript, nie je potrebná explicitná kontrola pre syntax a korektnosť.

Samotný návrh tabuliek a prepojení sme testovali pomocou zápisov a požiadaviek nad tabuľkou databázy pomocou funkcií systému.

7. Siedmy šprint – 30 ročná vojna

V rámci siedmeho šprintu sme riešili nasledovné užívateľské príbehy. Vzhľadom na logické zmeny v systéme sme museli zaradiť do šprintu požiadavky na zmeny vlastností (angl. features) systému.

- F1 – Prerobenie databázy
- F2 – Vytvorenie vzorových funkčných testov
- F3 – Logovanie
- F4 – Refaktoring existujúceho kódu
- US26 – Vstup od používateľa

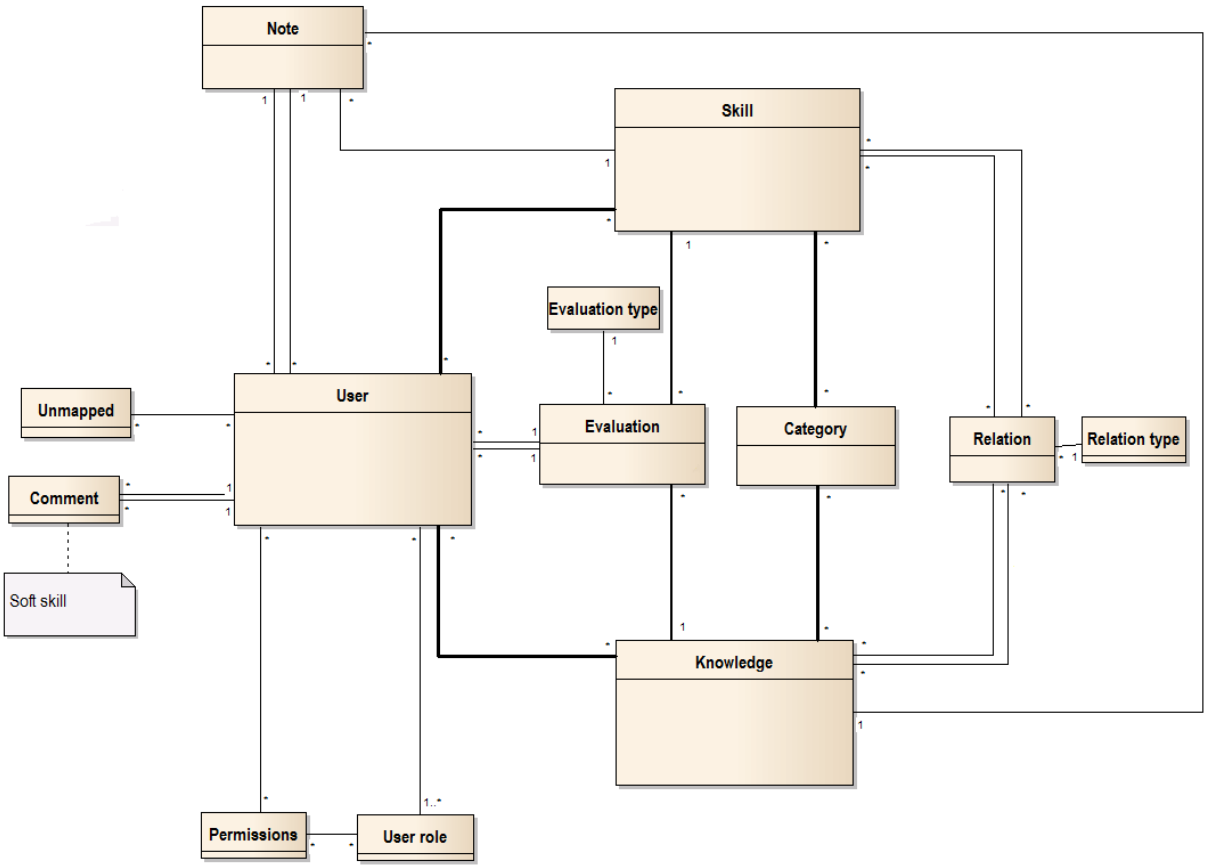
7.1. F1 – Prerobenie databázy

Prvá zmena vlastností systému sa týkala vytvorenia a nasadenia novej databázy. Oddelili sme logicky aj fyzicky zručnosti (angl. skills) od znalostí (angl. knowledge), čo malo za následok odlišnú prácu s týmito objektami.

7.1.1. Analýza

Na spoločnom stretnutí tímu sme navrhovali nový logický dátový model, ktorého konečná podoba je zobrazená na obrázku číslo 7.1.1. Vychádzali sme z existujúceho nasadeného modelu, aby bol počet zmien v logickom modeli, a tým pádom aj fyzickom, minimálny.

Museli sme brať ohľad aj na novú požadovanú funkcionalitu systému, ktorá mimo iného umožňuje pridávať komentáre k používateľom, pridávať poznámky ku znalostiam a zručnostiam, umožňuje nastaviť voliteľnú úroveň znalostí, alebo pridať ľubovoľnú znalosť či zručnosť, ktorá sa v systéme dosiaľ nenachádza. Tieto všetky už implementované, riešené, alebo plánované funkcie museli byť zohľadnené pri tvorbe databázy.



Obr. č. 7.1.1 – Nový logický model systému

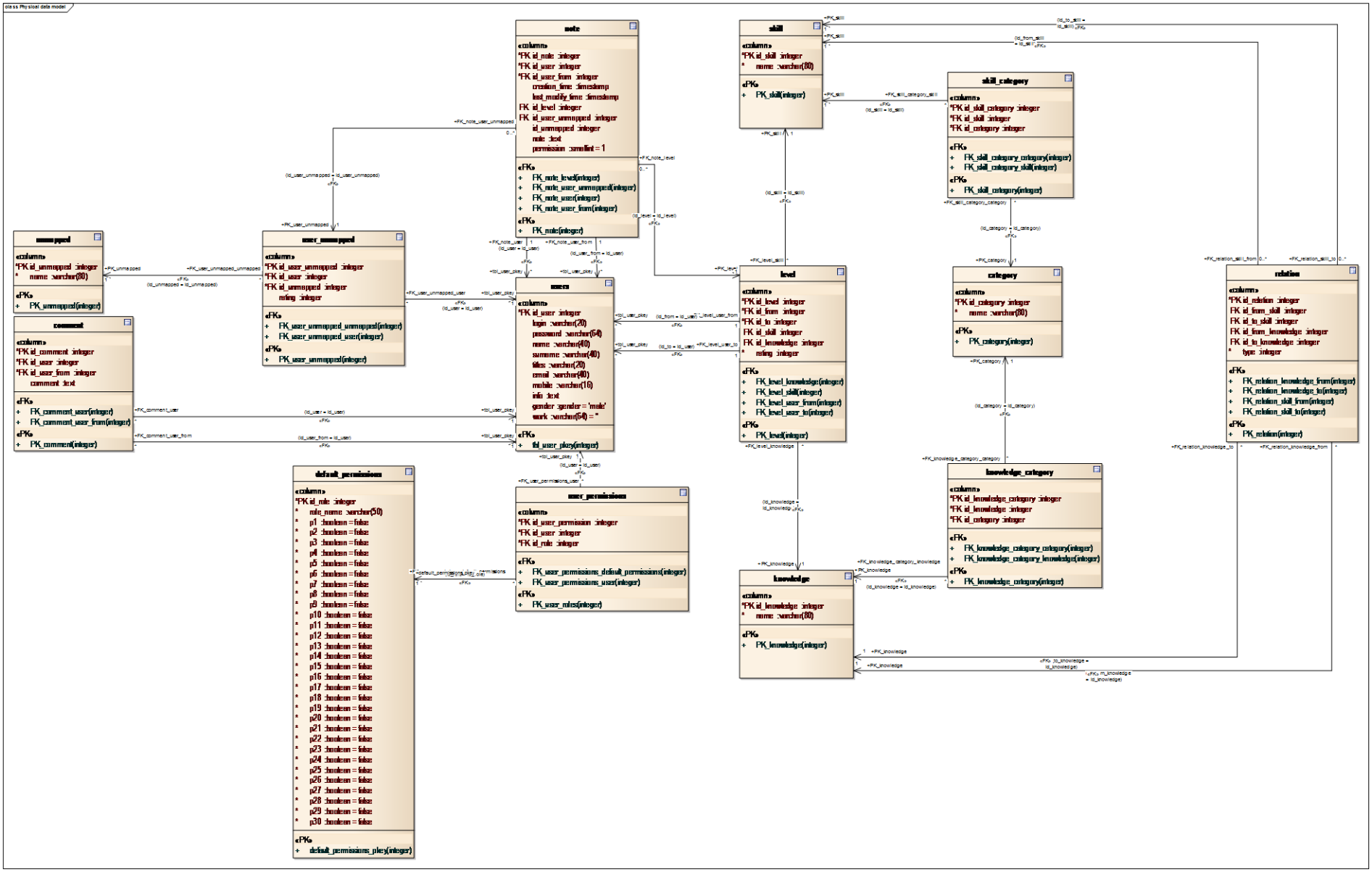
7.1.2. Návrh

Na základe Logického dátového modelu zobrazeného na obrázku číslo 7.1.1 bol vytvorený fyzický model v modelovacom nástroji Enterprise Architect[1]. Ako je znázornené na obrázku číslo 7.1.2, boli namodelované všetky potrebné tabuľky, atribúty, sekvencie, relácie, až po primárne a cudzie kľúče. V nastaveniach tabuliek bol zvolený typ databázy PostgreSQL, ktorý využívame, čo nám umožnilo pridávať k atribútom korektné dátové typy.

Novým významným objektom tejto databázy je tabuľka *Level*, ktorá spája používateľa so znalosťou, alebo zručnosťou, pričom umožňuje evidovať úroveň tohto vzťahu číselnou stupnicou. Taktiež pomocou tabuľky *Note* je možné tento vzťah okomentovať ľubovoľným textom.

Tabuľka *Level* obsahuje dva atribúty s reláciami na dva cudzie kľúče – *id_skill* a *id_knowledge*, ktoré určujú konkrétnu znalosť alebo zručnosť. Pri vkladaní to tabuľky sa nastaví buď jeden, alebo druhý atribút na konkrétnu hodnotu, pričom zvyšný atribút má hodnotu *Null*. Podľa tohto vieme určiť či si používateľ pridal znalosť alebo zručnosť.

[1] http://www.sparxsystems.com.au/



Obr. č. 7.1.2 – Nový fyzický model systému

7.1.3. Implementácia

Fáza implementácie netrvala dlho, pretože návrh bol vykonaný precízne a dôkladne. Spomenutý nástroj Enterprise Architect umožňuje generovať z fyzického dátového modelu priamo SQL skript vo forme DDL balíka na konštrukciu kompletnej hotovej databázy.

Vygenerovaný SQL skript je možné ihneď vykonať nad prázdnou databázou bez nutnosti ďalšieho zásahu alebo interakcie. Nasleduje ukážka SQL skriptu pre vytvorenie tabuľky *Level* spolu s definovaním primárneho kľúča a cudzích kľúčov.

```
CREATE TABLE level (
      id_level integer DEFAULT
      nextval(('level_id_level_seq'::text)::regclass) NOT NULL,
      id_from integer NOT NULL,
      id_to integer NOT NULL,
      id_skill integer,
      id_knowledge integer,
      rating integer NOT NULL
)
;
ALTER TABLE level ADD CONSTRAINT PK_level
      PRIMARY KEY (id_level)
;
ALTER TABLE level ADD CONSTRAINT FK_level_knowledge
      FOREIGN KEY (id_knowledge) REFERENCES knowledge (id_knowledge)
;

ALTER TABLE level ADD CONSTRAINT FK_level_skill
      FOREIGN KEY (id_skill) REFERENCES skill (id_skill)
;

ALTER TABLE level ADD CONSTRAINT FK_level_user_from
      FOREIGN KEY (id_from) REFERENCES users (id_user)
;

ALTER TABLE level ADD CONSTRAINT FK_level_user_to
      FOREIGN KEY (id_to) REFERENCES users (id_user)
;
```

7.1.4. Testovanie

Keďže SQL skript bol generovaný strojovo, nie je nutné testovať a kontrolovať jeho syntax a funkčnosť. Databáza však bola testovaná vývojármi pri refaktorovaní existujúceho kódu na nový databázu. Bola teda ešte zrevidovaná drobnými úpravami, aby sa vyriešili problémy ktoré vznikli pri jej používaní.

7.2. F2 – Vytvorenie vzorových funkčných testov

7.2.1. Inštalácia PHPUnit

PHPUnit je framework slúžiaci na unit testovanie pre programovací jazyk PHP. Inštaluje sa nasledovným spôsobom pomocou príkazového riadku:

```
pear channel-update pear.php.net
pear upgrade-all
pear channel-discover pear.phpunit.de
pear channel-discover components.ez.no
pear channel-discover pear.symfony-project.com
pear update-channels
pear install -a -f phpunit/PHPUnit
```

Je možné, že vypíše nasledovné upozornenie:

```
warning: phpunit/PHPUnit requires PEAR Installer (version >= 1.9.4),
installed version is 1.9.2
```

V tomto prípade nainštaluje najnovšiu verziu PEARu:

```
pear install PEAR
```

V konfiguračnom súbore php.ini zmeníme riadok:

```
include_path = "..\BitNami WAPPStack\php\PEAR"
```

Je potrebné reštartovať Apache server.

Na overenie, či máme správne nainsťalovaný PHPUnit napíšeme do príkazového riadku:

```
phpunit
```

7.2.2. Inštalácia Selenium

mal by byť prítomný na ceste

```
BitNami WAPPStack\php\PEAR\PHPUnit\Extensions\SeleniumTestCase.php
```

Ak nie je, naištalujeme ho z príkazového riadku príkazom

```
pear install phpunit\PHPUnit_Selenium
```

7.2.3. Konvencie

- Testy ukladáme do priečinka *tests\functional\XYTest.php*.
- testy pre triedy *Class* sa nachádzajú v triede *ClassTest*
- trieda *ClassTest* dedí z *PHPUnit_Framework_Testcase*
- jednotlivé testy sú verejné metódy, pomenované *test**
- v metódach používame výrokové metódy ako napr. *assertEquals()*, ktoré sa používajú na potvrdenie, či aktuálna hodnota sa zhoduje s požadovanou

7.2.4. Vybrané príklady testov

Testovanie správneho získavania zručností:

```
public function testGetSkillDataList()
    {
        //$this->$sc = new SkillController; - nanestastie takto to nejde
        //takze musime prekopirovat metodu zo SkillController.php
        $connection = Yii::app()->db;

        /*grupujeme a distancujeme aby sme predisli duplicitam*/
        $sql = "SELECT DISTINCT ON (skill_name) id, skill_name FROM
unique_skills GROUP BY id, skill_name ORDER BY skill_name";
        $skills = $connection->createCommand($sql)->query();

        $source = array();
        foreach ($skills as $row) {
            $source[] = array(
                'id' => $row['id'],
                'value' => $row['skill_name'],
            );
        }

    //ocakavany vystup
    $req = array(
      0 => array("id" => 3, "value" => "BLBOST"),
      1 => array("id" => 1, "value" => "java"),
      2 => array("id" => 2, "value" => "php")
    );
    //print_r($source);

    //porovnavame, ci aktualna hodnota $source sa zhoduje s pozadovanou
    $this->assertEquals($req, $source);
    }
```

Použitie data providera, ktorý vracia pole polí:

```
<?php
class DataTest extends PHPUnit_Framework_TestCase
{
    /**
     * @dataProvider provider
     */
    public function testAdd($a, $b, $c)
    {
        $this->assertEquals($c, $a + $b);
    }
```

```
    public function provider()
    {
        return array(
          array(0, 0, 0),
          array(0, 1, 1),
          array(1, 0, 1),
          array(1, 1, 3)
        );
    }
}
?>
```

Testovanie výnimiek:

```
<?php
class ExceptionTest extends PHPUnit_Framework_TestCase
{
    /**
     * @expectedException InvalidArgumentException
     */
    public function testException()
    {
    }
}
?>
```

Zistenie, či pole obsahuje prvok pomocou *assertContains()*:

```
<?php
class ContainsTest extends PHPUnit_Framework_TestCase
{
    public function testFailure()
    {
        $this->assertContains(4, array(1, 2, 3));
    }
}
?>
```

Očakávaný výstup z konzolového riadku (vypíše chybu, pretože 4 sa nenachádza v poli):

```
phpunit ContainsTest
PHPUnit 3.6.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsTest::testFailure
Failed asserting that an array contains 4.

/home/sb/ContainsTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

7.3. F3 – Logovanie

Ako vývojár chcem mať k dispozícii zaznamenávanie udalostí našej webovej aplikácie.

7.3.1. Analýza

Logovanie udalostí je dôležité, pretože nám uľahčuje a sprehľadňuje údržbu aplikácie. Pri debugovaní je podstatné, aby sme zaznamenali všetky udalosti, ktoré môžu nastať, a to hlavne chyby, upozornenia a trace, teda postupnosť volania modulov.

7.3.2. Návrh

Všetky udalosti sa budú zaznamenávať do súboru "*/protected/runtime/minerva_log*". Chceme zaznamenávať všetko, čo sa deje v aplikácii, všetky chyby, varovania a volania modulov.

7.3.3. Implementácia

Použili sme triedu CPSLiveLogRoute, ktorú sme upravili pre naše potreby. V nasledujúcom kóde, je uvedené, ako sa realizuje zápis logovania.

```
protected function formatLogMessage( $message, $level = 'I', $category =
null, $time = null )
    {
        if ( null === $time )
            $time = time();

        $level = strtoupper( $level[0] );
        //Yii::app()->user->id;
        //$this->getComponent('user');
        //Yii::app()->user->getId() or Yii::app()->user->id
        //Yii::app()->user->getId();
        //User::model()-
>find('username=:username',array(':username'=>$Yii::app()->user->id))->id;
        //$usr = User::model()->findAllByAttributes(array("login" => $this-
>username, "password" => hash("sha256", $this->password)));

      $session=new CHttpSession;
      $session->open();

       $name = '';
       $id = '';

       //echo $session['name']
       $pole = $session->getKeys();

       // ak nie sme lognuty
       if (empty($pole))
       {
         $logged_out_str = @date( 'M d H:i:s', $time ) . ' UNLOGGED USER' .
' [' . sprintf( '%-30s', $category ) . '] ' . ': <' . $level . '> ' .
$message . PHP_EOL;
         return $logged_out_str;
       }
       // ak sme lognuty
       else
       {
         $name = ($session[$pole[1]]);
```

```
        $id =  ($session[$pole[0]]);
        $logged in str = @date( 'M d H:i:s', $time ) . ' NAME: ' . $name .
' ID: ' . $id . ' [' . sprintf( '%-30s', $category ) . '] ' . ': <' .
$level . '> ' . $message . PHP_EOL;
        return $logged_in_str;
       }

     //return  @date( 'M d H:i:s', $time ) . ' NAME: ' . $name . ' ID: ' .
$id . ' [' . sprintf( '%-30s', $category ) . '] ' . ': <' . $level . '> ' .
$message . PHP_EOL;
    }
 }
```

Zaznamenávame aj ktorí používatelia sú prihlásení, ak nie je nikto, tak pri jednotlivom zázname v logu sa vypíše reťazer „UNLOGGED USER“.

Bolo potrebné zmeniť súbor /*config/main.php*, kde sme pridali tieto riadky:

```
'log' => array(
            'class' => 'CLogRouter',
            'routes' => array(
                array(
                    'class' => 'CPSLiveLogRoute',
                    'levels' => 'error, warning, info, trace',
                    'maxFileSize' => '10240',
                    'logFile' => 'minerva_log',
                ),
            ),
),
```

Rozlišujeme viacero úrovní, error, warning, info a trace. Trace používame len pre potreby debugovania, ale z dôvodu, že by logovací súbor bol enormne veľký je táto úroveň vypnutá.

Príklad z výpisu logovania:

```
Apr 10 06:31:46 NAME: xszorada ID: 10 [system.db.CDbCommand          ] :
<T> Querying SQL:
                  SELECT
                        id,
                        type
                  FROM
                        skill_types
in D:\Tono\BitNami
WAPPStack\apache2\htdocs\protected\controllers\SkillController.php (316)
in D:\Tono\BitNami WAPPStack\apache2\htdocs\protected\views\skill\form.php
(245)
in D:\Tono\BitNami
WAPPStack\apache2\htdocs\protected\controllers\SkillController.php (157)
Apr 10 06:31:48 NAME: xszorada ID: 10 [system.CModule                ] :
<T> Loading "request" application component
in D:\Tono\BitNami WAPPStack\apache2\htdocs\index.php (13)
Apr 10 06:31:48 NAME: xszorada ID: 10 [system.CModule                ] :
<T> Loading "urlManager" application component
in D:\Tono\BitNami WAPPStack\apache2\htdocs\index.php (13)
Apr 10 06:31:48 NAME: xszorada ID: 10 [system.web.filters.CFilterChain] :
<T> Running filter SiteController.filteraccessControl()
in D:\Tono\BitNami WAPPStack\apache2\htdocs\index.php (13)
```

7.4. F4 – Refaktoring

Vzhľadom na novú databázu a logické aj fyzické zmeny v štruktúre celého systému je potrebné používať názvy zručnosť (angl. skill) a znalosť (angl. knowledge) správnym spôsobom. Taktiež funkcie obsahujúce v názve *skill* by nemali pracovať so znalosťami a naopak.

7.4.1. Analýza

Doteraz sme sa v našom systéme nezaoberali zručnosťami, ale iba znalosťami. Používali sme však všade vo funkciách názov *skill*, čo predstavuje zručnosť, nie znalosť.

Teraz keď pracujeme s oboma pojmami, pôvodný kód ostal veľmi neprehľadný a nie je z neho jasné čo ktorá funkcia robí a čo daná metóda vlastne vracia.

7.4.2. Návrh

Bolo potrebné manuálne kontrolovanie funkcií, premenných, controllerov a modelov. Zvolili sme postup rozdelenia po controlleroch. Prechádzať postupne kód, opravovať názvy funkcií, premenných, pri použití modelu alebo zobrazenia (angl. view) skontrolovať aj tieto súbory. Týmto postupom je možné prejsť celý kód systému bez vynechania určitých funkcií alebo metód.

7.4.3. Implementácia

Popri nahradzovaní nesprávne pomenovaných metód, modelov, funkcií či premenných sme opravovali aj nesprávne získavanie údajov z databázy. Miesto siahodlhých SQL príkazov sme tento kód nahradili elegantnou formou získavania údajov pomocou *CactiveRecord modelu*.

Táto ukážka kódu znázorňuje upravenú funkciu, ktorá vracia pole dvojíc ID znalosti a názov znalosti získavaním údajov pomocou modelu.

```
public function getKnowledgeDataList()
{
      $data = Knowledge::model()->findAll();
      $knowledge = array();
      foreach($data as $row)
      {
          $knowledge[] = array(
             'id' => $row['id_knowledge'],
             'value' => $row['name']
          );
       }
       return $knowledge;
}
```

7.4.4. Testovanie

Základné testovanie opraveného kódu sme vykonávali pomocou príkazového riadku operačného systému Windows. Testovali sme syntaktickú správnosť kódu príkazom

```
php -l <filename>
```

7.5. US26 – Vstup od používateľa

Ako používateľ chcem zadať ľubovoľnú znalosť, teda aj takú, ktorá nie je v systéme.

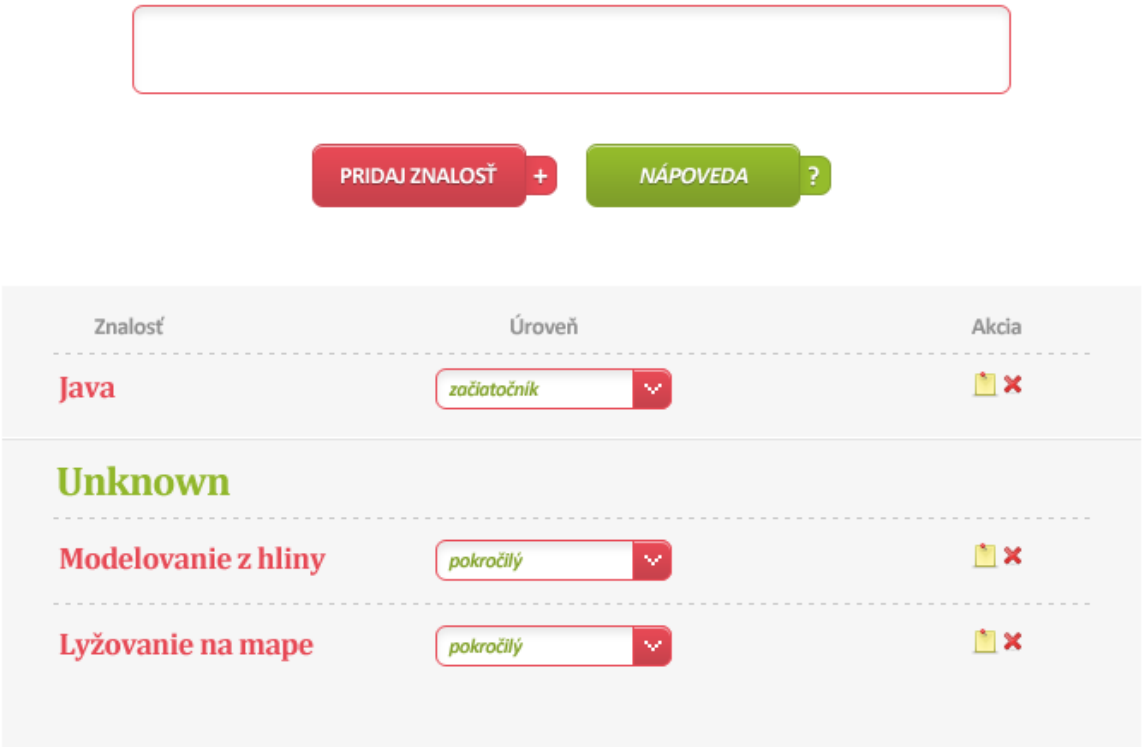
7.5.1. Analýza

Systém nemôže hneď v začiatku obsiahnuť všetky znalosti, a teda musí umožniť používateľovi pridať si do svojho zoznamu akúkoľvek znalosť. Táto znalosť sa mu následne priradí do kategórie *Unknown* a zotrvá tam dovtedy, pokiaľ nebude zaradená do príslušnej kategórie.

Prípad použitia	**VstupOdPouzivatela**
ID	7.5
Stručný popis	Používateľ si môže zadať akúkoľvek znalosť do svojho zoznamu.
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky na pridanie znalostí. 2. Používateľ do textového poľa zadá akúkoľvek znalosť a klikne na tlačidlo pridaj, znalosť sa mu objaví v zozname pod textovým poľom. 3. Používateľ klikne na tlačidlo Odoslať. 4. Systém zapíše údaje do systému.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

7.5.2. Návrh

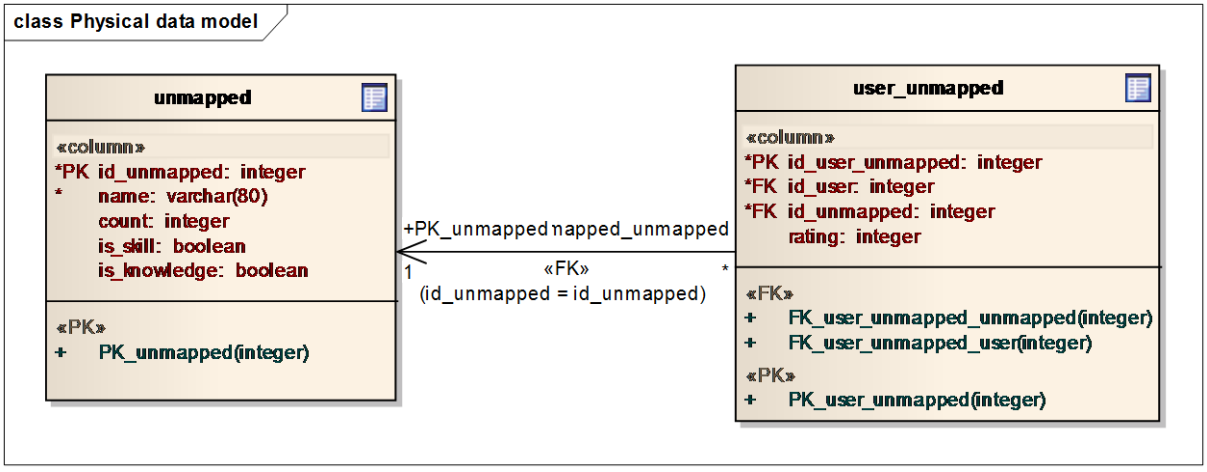
Na obrázku 7.5.1 je zobrazená obrazovka po pridaní znalosti, ktorá nie je v systéme. Tieto znalosti sú opticky oddelené, aby používateľ vedel, že tieto znalosti či zručnosti si pridal sám.



Obr. č. 7.5.1 – Obrazovka pridávania neznámych znalostí a zručností

7.5.3. Implementácia

Pre uskutočnenie možnosti ukladať neznáme veci sme museli vytvoriť novú entitu *unmapped*, ktorá uchováva neznáme veci. Aby bolo možné tieto položky mapovať na konkrétneho používateľa, tak musela byť pridaná aj nová tabuľka *user_unamapped*.



Obr. č. 7.5.2 – Relácia tabuliek *unmapped* a *user_unmapped*

Popis atribútov entity *unmapped*:

Atribút	Popis
name	názov znalosti/zručnosti
count	počet výskytov
is_skill	indikátor, či položka je zručnosť
is_knowledge	indikátor, či položka je znalosť

Model

Pre vytvorenie modelu bol použitý Gii generátor, cez ktorý sme vygenerovali modely *Unmapped* a *UserUnmapped*.

Controller

Získané znalosti z viewu bolo treba spracovať v príslušnom ovládači. Najprv bolo treba zistiť, či už sa daná znalosť/zručnosť nachádza v databáze, aby sme nevytvárali duplicity. Ak ešte neexistuje v databáze, tak sa uloží s početnosťou 1 a potom naviaže na používateľa. Ak už existuje, tak sa inkrementuje početnosť výskytov:

```
foreach($itemAray as $index => $item) {
      if((int) $item == 0) {
            $unmp = Unmapped::model()->find('(translate("name",
\'ľščťžýáíéúäô\', \'lsctzyaieuao\') ilike translate(:name,
\'ľščťžýáíéúäô\', \'lsctzyaieuao\'))', array(':name' =>
$_POST[$unp_name][$index]));
            if(!$unmp) {
                  $item_unmapped = new Unmapped;
                  $item_unmapped->name = $_POST[$unp_name][$index];
                  $item_unmapped->count = 1;
                  $item_unmapped->is_skill = $is_skill;
                  $item_unmapped->is_knowledge = $is_knowledge;
                  if(!$item_unmapped->save()) {
                        $this->render('error', array( 'code' => '500',
'message' => 'Internal server error.'));
                        Yii::app()->end();
                  }
                  $user_unmapped = new UserUnmapped;
                  $user_unmapped->id_unmapped = $item_unmapped-
>id_unmapped;
                  $user_unmapped->id_user = $id;
                  ($is_knowledge == true) ? $user_unmapped->rating =
                        $_POST['knowledge_level'][$index] : $user_unmapped-
>rating = $_POST['skill_level'][$index];
                  if(!$user_unmapped->save()) {
                        $this->render('error', array( 'code' => '500',
'message' => 'Internal server error.'));
                        Yii::app()->end();
                  }
            }
            else {
                  $unmp->count = $unmp->count + 1;
                  $unmp->update();
                  $user_unmapped = new UserUnmapped;
                  $user unmapped->id unmapped = $unmp->id unmapped;
                  $user_unmapped->id_user = $id;
                  if(!$user_unmapped->save()) {
                        $this->render('error', array( 'code' => '500',
'message' => 'Internal server error.'));
```

```
                    Yii::app()->end();
                }
            }
```

7.5.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Pridanie znalosti, ktorá nie je v systéme	**ID Testu**	7.5-01
Rozhranie	Obrazovka pridania znalostí	**ID UC**	7.5
Účel	Zistenie, či sa znalosť zapíše do systému		
Vstupné podmienky	Znalosť, ktorá nie je v systéme		
Výstupné podmienky	Žiadne		

Krok	**Akcia**	**Očakávanáreakcia**	**Skutočnáakcia**
1.	Napísanie znalosti, ktorá nie je v systéme.	Funkcia automatického doplnenia znalostí nebude reagovať.	Autocomplete nereaguje.
2.	Kliknutie na tlačidlo “Pridaj”.	Znalosť sa zobrazí do tabuľky pod textovým poľom.	Znalosť zobrazená v tabuľke.
3.	Kliknutie na tlačidlo “Odošli”	Znalosť sa zapíše do systému.	Znalosť zapísaná v systéme.

8. Ôsmy šprint – Napoleonská vojna

V rámci ôsmeho šprintu sme riešili nasledovné užívateľské príbehy:

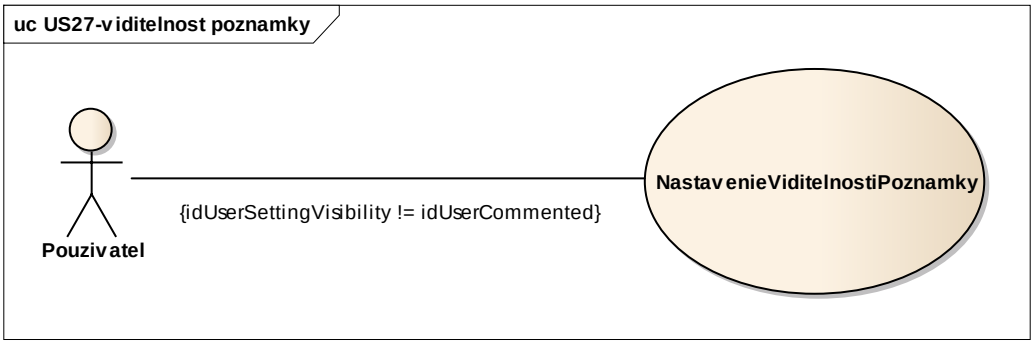
- US27 – Viditeľnosť poznámky
- US28 – Zobrazenie znalostí
- US29 – Zobrazenie poznámok
- US30 – Zobrazenie komentárov

8.1. US27 – Viditeľnosť poznámky

Ako komentátor chcem mať možnosť nastavenia zobrazenia komentu aspoň do nasledovných kategórií: default (vidia všetci), študenti (iba študenti), učitelia, študent (iba komentovaný študent).

8.1.1. Analýza

Pri pridávaní poznámky niekedy chceme aby danú poznámku videla iba určitá skupina používateľov alebo iba používateľ sám. Takáto situácia môže nastať napríklad pri negatívnm hodnotení študenta a poznámku vtedy nemusia vidieť všetci používatelia ale napríklad iba učitelia.



Obr. č. 8.1.1 – Prípad použitia Viditeľnosť poznámky

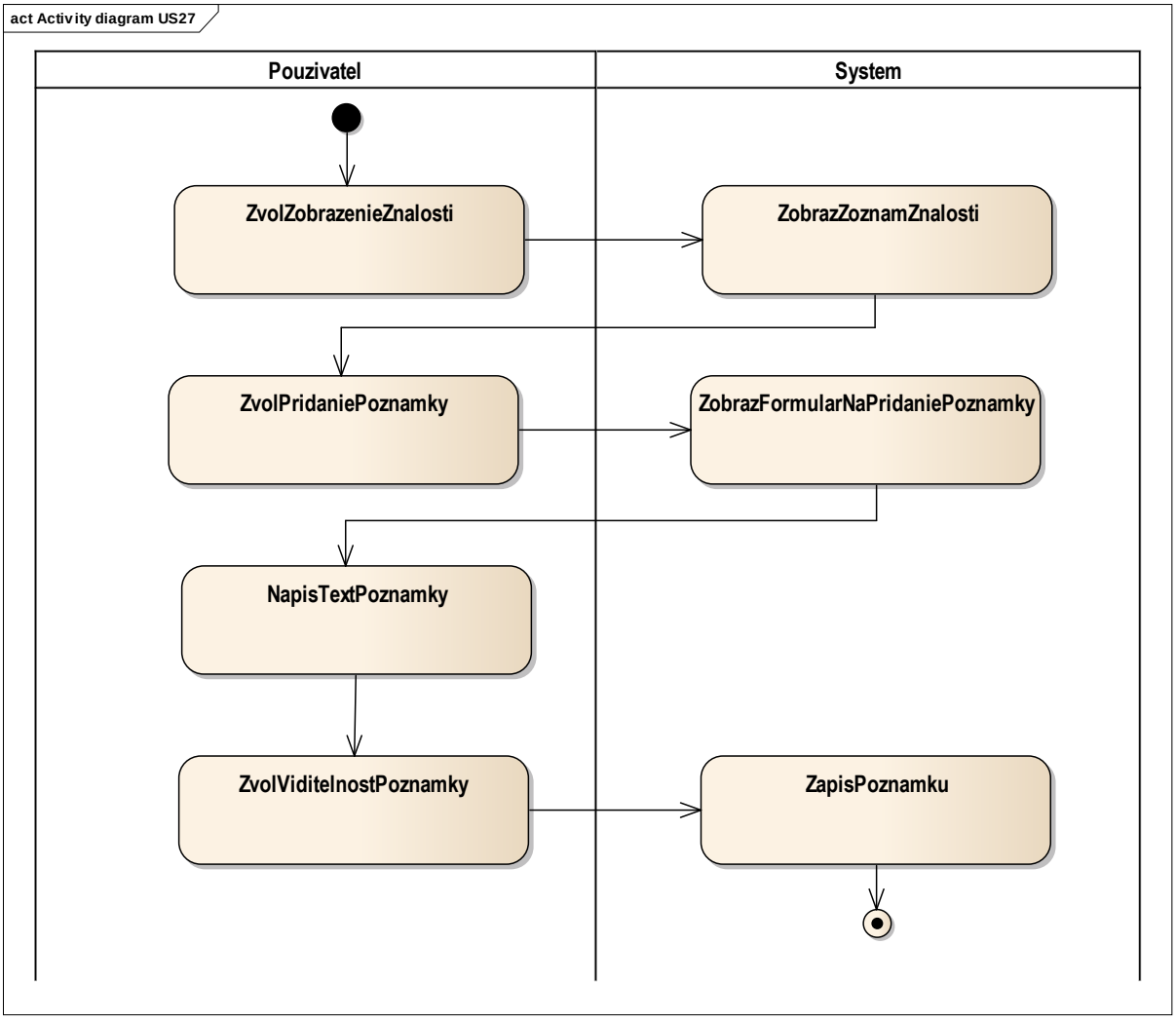
Prípad použitia	**ViditelnostPoznamky**
ID	8.1
Stručný popis	Používateľ nastavuje poznámke viditeľnosť.
Primárny aktéri	Prihlásený používateľ

Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky znalostí iného používateľa. 2. Používateľ si klikne na ikonku poznámky a následne sa mu zobrazí textové pole, kam môže písať poznámku k znalosti a taktiež sa mu zobrazí „dropdown menu“, kde môže vybrať viditeľnosť poznámky. 3. Používateľ vyplní poznámku a nastaví jej viditeľnosť. 4. Používateľ odošle údaje. 5. Systém spracuje formulár a uloží údaje do databázy.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

8.1.2. Návrh

Na obrázku číslo 8.1.2 je zobrazená obrazovka nastavenia viditeľnosti poznámky pri pridáaní poznámky k znalosti iného používateľa.

Obr. č. 8.1.2 – Obrazovka pridania poznámky ku znalosti



Obr. č. 8.1.3 – Diagram aktivít

8.1.3. Implementácia

Pri implementácii sme využili všeky súčasti MVC modelu a na zapísanie údajov do databázy sme použili *CActiveRecord*.

Model

Pre entitu poznámky sme vytvorili model *Note*, ktorý danú entitu vyjadruje v systéme. K atributom sme pridali v modeli pravidlá a popisy.

```
public function rules()
	{
		// NOTE: you should only define rules for those attributes that
		// will receive user inputs.
		return array(
			array('id_user, id_user_from', 'required'),
			array('id_user, id_user_from, id_level,
id_user_unmapped', 'numerical', 'integerOnly'=>true),
			array('creation_time, last_modify_time, note', 'safe'),
			// The following rule is used by search().
```

```
                  // Please remove those attributes that should not be
searched.
                  array('id_note, id_user, id_user_from, creation_time,
last_modify_time, id_level, id_user_unmapped, note, permission', 'safe',
'on'=>'search'),
            );
      }
public function attributeLabels()
      {
            return array(
                  'id_note' => 'Id Note',
                  'id_user' => 'Id User',
                  'id_user_from' => 'Id User From',
                  'creation_time' => 'Creation Time',
                  'last_modify_time' => 'Last Modify Time',
                  'id_level' => 'Id Level',
                  'id_user_unmapped' => 'Id User Unmapped',
                  'note' => 'Note',
                  'permission' => 'Permission',
            );
      }
```

Databáza

Aby US fungoval, bola potrebná aj zmena v databáze v entite *Note*. Pridali sme atribút *permission*, do ktorého sa číselne ukladá hodnota, ktorá určuje viditeľnosť poznámky.

Ovládač

V controlleri *AjaxController* sme pridali do premennej hodnotu z "dropdown menu", ktorú sme získali z formulára pomocou $_POST.

```
$note->permission = $_POST['note_perm'];
```

View

Vo vieweri sme pridali iba "dropdown menu" pomocou *CHtml::dropDownList* a nastavili mu atribút *name*, aby sme mohli následne pomocou $_POST v controlleri získať vybranú hodnotu.

```
<?php
   echo "<div class='note_perm'><br>Viditeľnosť poznámky<br>";
      echo CHtml::dropDownList('note_perm', 'Včetci', array('1' =>
'Včetci', '2' => 'Iba študenti', '3' => 'Iba učitelia', '4' => 'Iba tento
študent')); ?>
      <input name="CommentLevelId" type="hidden" value="<?php echo
$subdata['id_level']; ?>" />
      <br />
   </div>
```

8.1.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zapísanie poznámky k znalosti s určením viditeľnosti		**ID Testu**	8.1-01
Rozhranie	Zobraz znalosti		**ID UC**	8.1
Účel	Zistenie, či sa poznámka zapíše k znalosti			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Kliknutie na ikonku poznámky.	Zobrazenie textového poľa na vkladanie poznámky.	Textové pole na pridanie poznámky zobrazené.	
2.	Zadanie poznámky do textového poľa.	Zobrazenie textu poznámky v textovom poli.	Text poznámky je zobrazený v textovom poli.	
3.	Určenie úrovne viditeľnosti z výberu	Vo výbere sa zobrazí vybratá položka.	Vo výbere sa zobrazila vybratá položka.	
4.	Kliknutie na tlačidlo “Pridaj”	Zobrazenie oznamu že znalosť bola úspešne zapísaná do systému.	Oznam o zapísaní do system zobrazený.	

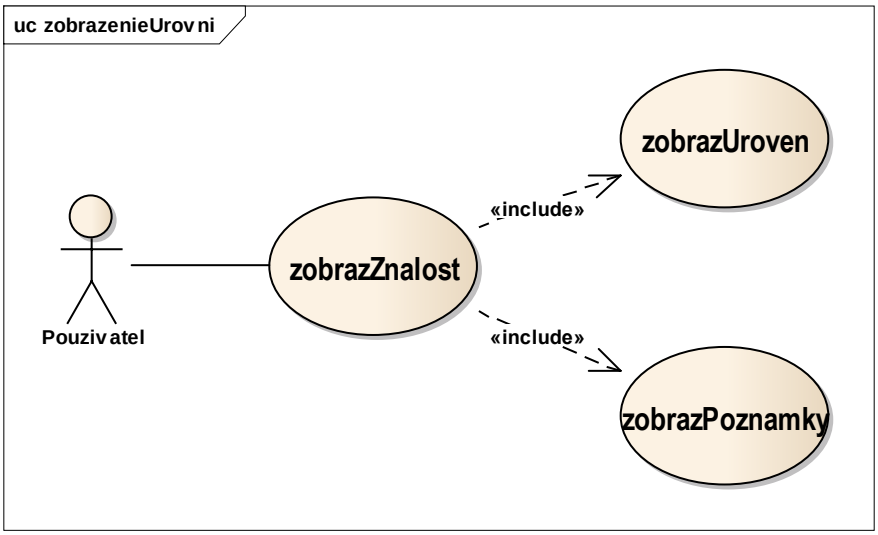
8.2. US28 – Zobrazenie znalostí

Ako používateľ chcem, aby sa zobrazovali znalosti aj s príslušnými úrovňami.

8.2.1. Analýza

S novou nasadenou databázou sú spojené možnosti nastavovať úroveň ovládania konkrétnych znalostí. Tie sú uchovávané v databázove tabuľke *Level*, spolu so znalosťou a používateľom, ktorý má znalosť aj úroveň priradené.

Pre spolupracovanie s novou databázou je potrebné zmeniť taktiež zobrazovanie znalostí a zručností, aby boli viditeľné nové pridané atribúty.

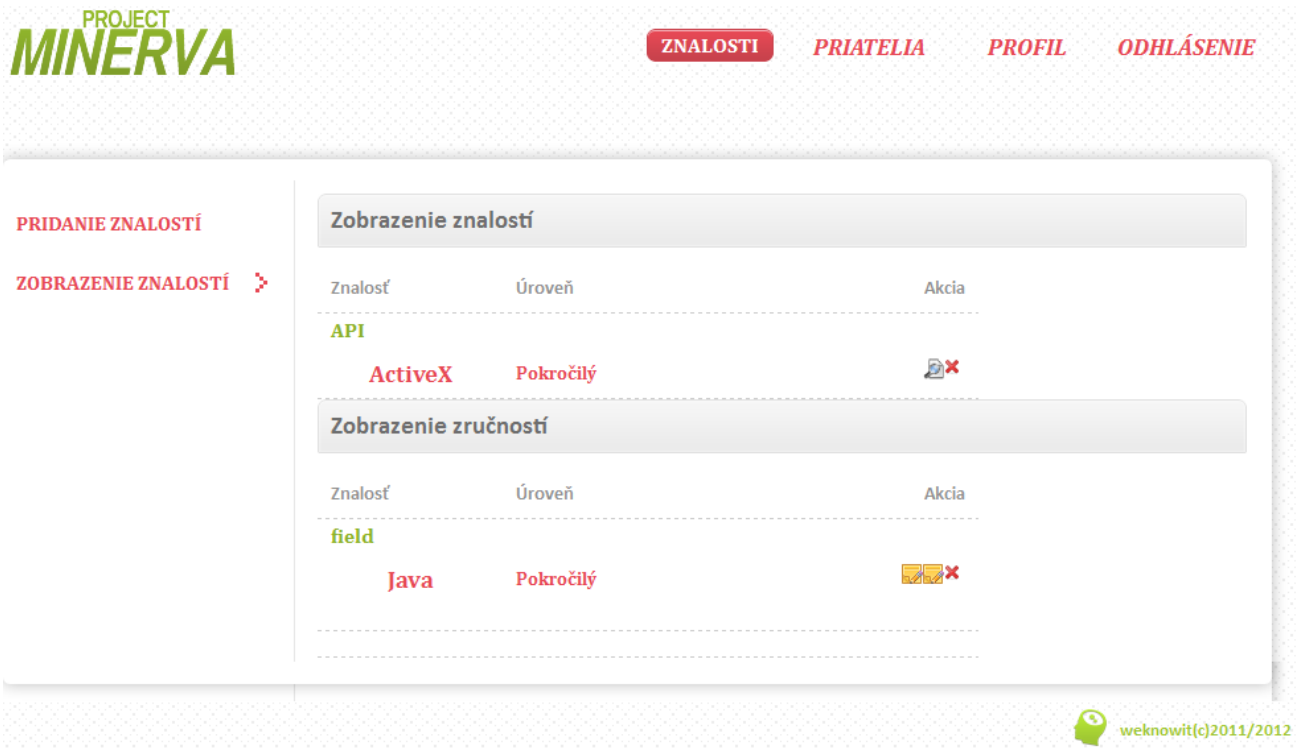


Obr. č. 8.2.1 – Diagram prípadov použitia *zobrazenieZnalosti*

Prípad použitia	**zobrazenieZnalosti**
ID	8.2
Stručný popis	Používateľ prehliada priradené znalosti
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený, Používateľ je oprávnený prezerať znalosti.
Hlavný scenár	1. Používateľ si zvolí zobraziť svoje znalosti alebo znalosti iného používateľa. 2. Systém overí oprávnenia používateľa a zobrazí znalosti hľadaného používateľa spolu s úrovňou a poznámkami k jednotlivým znalostiam.
Výstupné podmienky	Žiadne
Alternatívne scenáre	2a. Systém overí oprávnenia používateľa a zobrazí chybovú správu o nedostatočnom oprávnení.

8.2.2. Návrh

Dosiaľ bola možnosť prezerať iba zoznam pridaných znalostí pre jednotlivých používateľov. Nový návrh na obrázku číslo 8.2.1 zobrazuje znalosti aj zručnosti na jednej obrazovke, pričom sú logicky oddelené. Pri každej znalosti je zobrazená aj úroveň jej ovládania používateľom.



Obr. č. 8.2.2 – Obrazovka zobrazovania znalostí a zručností

8.2.3. Implementácia

Stĺpce jednotlivých atribútov sú napísané jednoduchým spôsobom,

```
<tr class='profile_tr'>
      <td align='left' class='skill_td_label'>Znalosť</td>
      <td align='left' class='skill_td_label'>Úroveň</td>
      <td align='right' class='skill_td_label'>Akcia</td>
</tr>
```

Pričom znalosti sa doplnia pri vykonani php kódu ešte na strane servera. Znalosti a zručnosti majú doplnenú úroveň pomocou priradených hodnôt z controllera, preto si pri zobrazení zavolajú dostupnú premennú, pole *$rating*.

```
echo '<td class="skill_td_rating">'.$rating[$subdata['rating']].'</td>';
```

8.2.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

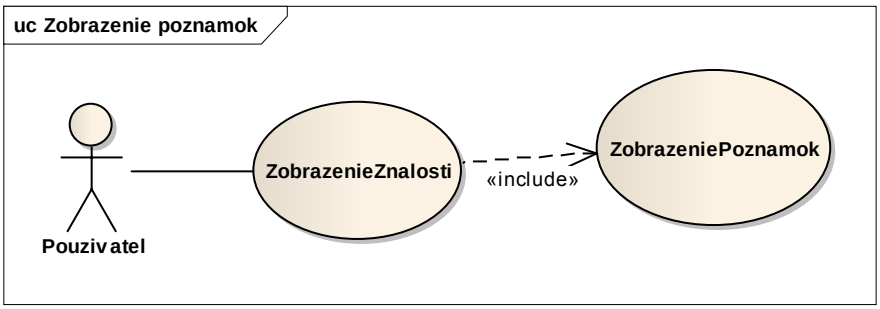
Názov	Zobrazenie znalostí používateľa spolu s úrovňami		**ID Testu**	8.2-01
Rozhranie	Zobrazenie znalostí		**ID UC**	8.2
Účel	Zistenie, či sa zobrazia uložené úrovne znalostí			
Vstupné podmienky	Priradené znalosti a úrovne k hľadanému používateľovi			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Zobrazenie znalostí s úrovňami.	Systém zobrazí znalosti hľadaného používateľa spolu s úrovňami.	Úrovne sú zobrazené pri jednotlivých znalostiach tak, ako sú uložené v databáze.	

8.3. US29 – Zobrazenie poznámok

Chcem aby sa zobrazovali poznámky pre jednotlivé znalosti/zručnosti (vlastné aj cudzie).

8.3.1. Analýza

Po pridaní poznámky musí mať používateľ možnosť si ju pozrieť. Pri svojom zozname znalostí bude po kliknutí na ikonku vidieť svoje vlastné poznámky ale aj poznámky od ďalších používateľov. Pre prehľadnosť budú poznámky v „zrolovanej“ forme, teda bude vidieť iba dátum, čas a meno používateľa, ktorý poznámku pridal. Po kliknutí na ikonu sa poznámka „rozroluje“ a objaví sa text poznámky.



Obr. č. 8.3.1 – Diagram prípadu použitia *ZobrazeniePoznamok*

Prípad použitia	**ZobrazeniePoznamok**
ID	8.3
Stručný popis	Používateľ si môže prezrieť poznámky k znalosti/zručnosti
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny

Vstupné podmienky	Používateľ je prihlásený
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky znalostí používateľa. 2. Používateľ si klikne na ikonku lupy a následne sa mu zobrazia poznámky od používateľov, ktoré su zatiaľ „zrolované“. 3. Používateľ klikne na plusko pri poznámke a zobrazí sa mu text danej poznámky.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

8.3.2. Návrh

Na obrázku číslo 8.3.2 je zobrazená obrazovka zobrazených poznámok. Poznámky sú buď „zrolované“ alebo „rozrolované“. Pri „zrolovanej“ poznámke je viditeľný len dátum, čas a meno používateľa, ktorý poznámku zadal. Ak si ju používateľ zadá sám, ako meno je uvedené „ja“.

Obr. č. 8.3.2 – Zobrazenie poznámok pri znalostiach

8.3.3. Implementácia

Pri implementácii sme využili všeky súčasti MVC modelu a na zapísanie údajov do databázy sme použili CActiveRecord.

Model

Pre entitu *poznámky* sme vytvorili model *Note*, ktorý danú entitu vyjadruje v systéme. K atribútom sme pridali v modeli pravidlá a popisy.

```
public function rules()
	{
		// NOTE: you should only define rules for those attributes that
		// will receive user inputs.
		return array(
			array('id_user, id_user_from', 'required'),
			array('id_user, id_user_from, id_level,
id_user_unmapped', 'numerical', 'integerOnly'=>true),
			array('creation_time, last_modify_time, note', 'safe'),
			// The following rule is used by search().
			// Please remove those attributes that should not be
searched.
			array('id_note, id_user, id_user_from, creation_time,
last_modify_time, id_level, id_user_unmapped, note, permission', 'safe',
'on'=>'search'),
		);
	}
public function attributeLabels()
	{
		return array(
			'id_note' => 'Id Note',
			'id_user' => 'Id User',
			'id_user_from' => 'Id User From',
			'creation_time' => 'Creation Time',
			'last_modify_time' => 'Last Modify Time',
			'id_level' => 'Id Level',
			'id_user_unmapped' => 'Id User Unmapped',
			'note' => 'Note',
			'permission' => 'Permission',
		);
	}
```

Controller

Pre naplnenie stromu s poznámkami sa využíva nasledovná funkcia, ktorá pre každú ajax požiadavku vyberie príslušné poznámky z databázy a odošle k používateľovi:

```
public function actionAjaxFillNoteTree($id, $unmp) {
	if(isset($unmp) && $unmp == 'true') {
		$data = Note::model()->findAll('id_user_unmapped =
:id_user_unmapped',
			array(':id_user_unmapped' => $id));
	}
	else {
		$data = Note::model()->findAll('id_level = :id_level', array(
			':id_level' => $id));
	}
```

```
	$notes = array();
	$expanded = false;
	foreach($data as $row) {
		$user = Users::model()->findByPk($row->id_user_from);
		$date = strtotime($row->creation_time);
		$notes[] = array(
			'text' => date("d.m.Y (H:i:s)", $date)." Pridal: <b>".
			(($user->id_user != Yii::app()->user->getId()) ? $user-
>login : "ja")."</b>:",
			'expanded' => $expanded,
			'children' => array(
				array(
					'text' => CHtml::encode($row->note),
				),
			),
		);
		$expanded = false;
	}
	echo CTreeView::saveDataAsJson($notes);
}
```

Viewer

Do viewu bol pridaný widget *CTreeView* , ktorý načítal poznámky zo serveru a zobrazil.

```
<?php
	$this->widget('CTreeView',array(
		'url' => array(
			'/ajax/ajaxFillNoteTree',
			'id' => $data['id_unmapped'], 'unmp' => 'true'),
		'animated'=>'slow',
		'collapsed'=>'false',
		'htmlOptions'=>array(
			'class'=>'treeview-famfamfam',
			'id' => 'unmptr_'.$data['id_unmapped']
			),
		));
?>
```

8.3.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	“Rozrolovanie” poznámky		**ID Testu**	8.3-01
Rozhranie	Obrazovka zoznamu znalostí		**ID UC**	8.3
Účel	Zistenie, či sa poznámka “rozroluje”			
Vstupné podmienky	Pridaná aspoň jedna poznámka k znalosti			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Kliknutie na ikonku lupy.	Zobrazenie “zrolovaných” poznámok k znalosti.	Zobrazené poznámky bez textu iba s dátumom, časom a menom.	

2.	Kliknutie na ikonu plus pri ľubovoľnej poznámke.	Zobrazenie textu poznámky.	Text poznámky zobrazený.

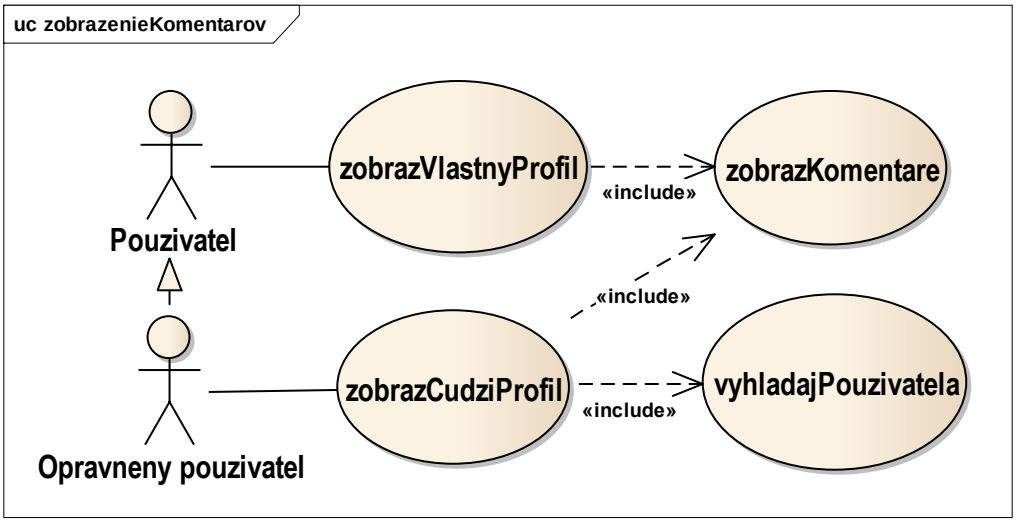
8.4. US30 – Zobrazenie komentárov

Tento US zahŕňa umožnenie zobrazovania pridaných komentárov pri jednotlivých používateľoch.

8.4.1. Analýza

Na základe funkcionality implementovanej v používateľskom príbehu US24 je vhodné zobrazovať komentáre, ktoré k používateľovi pridá niekto cudzí, alebo on sám.

Na obrázku číslo 8.4.1 je znázornený diagram prípadov použitia, ktorý znázorňuje akým spôsobom sú previazané funkcie komentovania s používateľským profilom a akú úlohu tu hrajú používateľské oprávnenia.

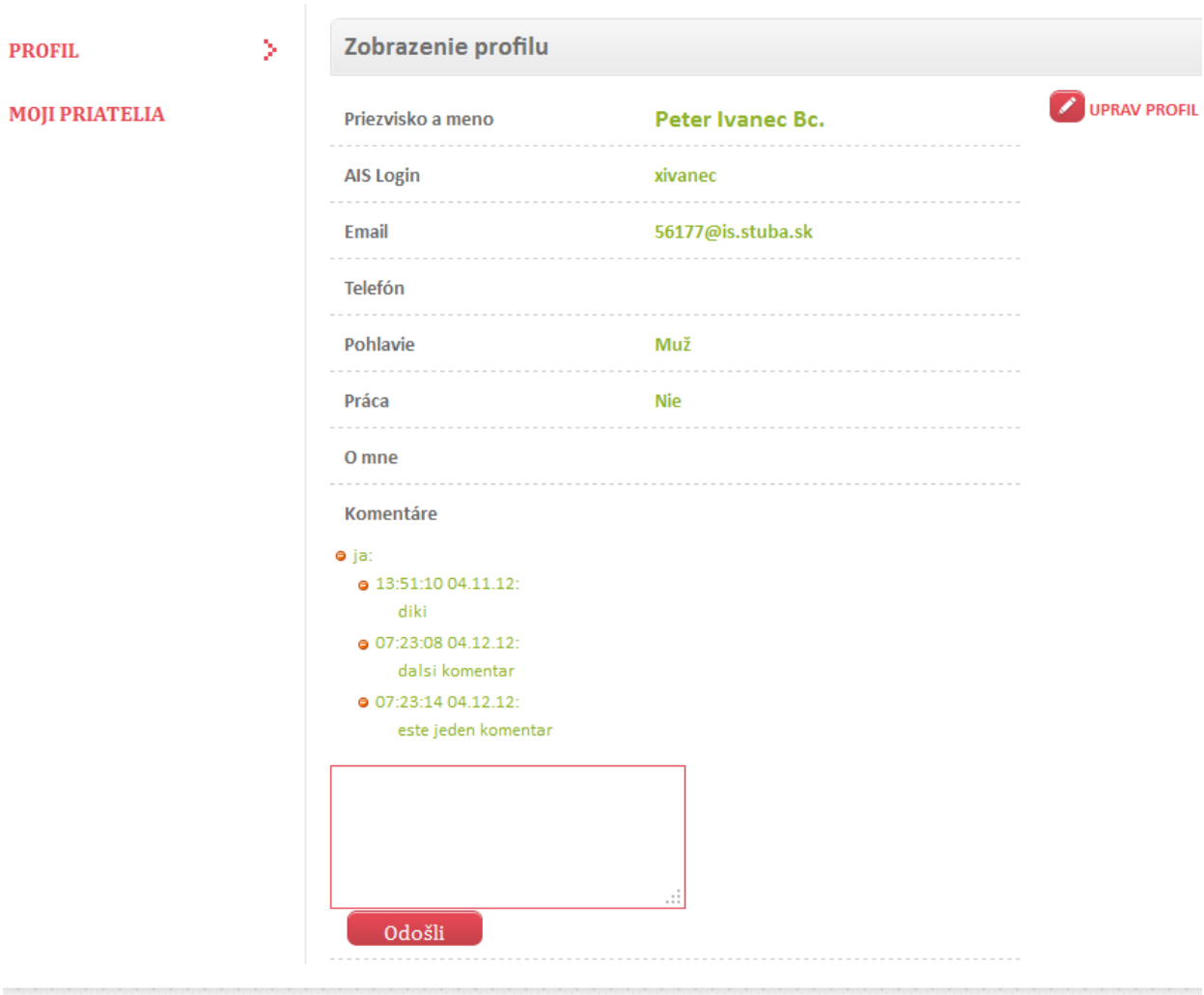


Obr. č. 8.4.1 – Diagram prípadov použitia *zobrazenieKomentarov*

8.4.2. Návrh

Ako je znázornené na obrázku číslo 8.4.2, komentáre sú zobrazované pomocou stromovej štruktúry na profilovej stránke používateľa. Kliknutím na príslušné ikonky pri komentároch sa dajú jednotlivé uzly skryť alebo zobraziť.

Komentáre sú kategorizované podľa používateľa, ktorý komentár pridal. Rozlíšené sú v rámci tejto kategórie pomocou časov kedy boli pridané.



Obr. č. 8.4.2 – Obrazovka profilu so zobrazenými komentármi

8.4.3. Implementácia

Táto funkcionalita je takisto ako pridávanie komentárov v US24 implementovaná cez Ajax, preto sa komentáre zobrazia dynamicky na stránke ihneď po odoslaní, nemusí nastať znovunačítanie zobrazenia.

Komentáre sú kategorizované podľa *id_user_from* atribútu, ktorý ako cudzí kľúč v tabuľke *Comment* identifikuje používateľa, ktorý daný komentár pridal. V rámci tejto kategorizácie sa pomocou dátového typu *timestamp* (bez časového pásma) pridá časový identifikátor pridania komentáru, podľa ktorého komentáre zoraďujeme. Nakoniec sa vypíše samotný atribút *comment*, v ktorom je uložený text komentára.

V nasledujúcom kóde je znázornené zobrazovanie stromovej štruktúry komentárov pomocou *CTreeView*.

```
<?php
$this->widget('CTreeView',array(
      'url' => array('/ajax/ajaxFillCommentTree', 'id' => $model->id_user),
      'animated'=>'slow',
      'collapsed'=>'false',
```

```
        'htmlOptions'=>array(
                'class'=>'treeview-famfamfam',
                'id' => 'comm_'.$model->id_user
        ),
));
?>
```

8.4.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zobrazenie pridaných komentárov		**ID Testu**	8.4-01
Rozhranie	Zobrazenie profilu používateľa		**ID UC**	8.4
Účel	Zistenie, či sa zobrazia uložené komentáre			
Vstupné podmienky	Priradené komentáre k hľadanému používateľovi			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Zobrazenie profilu používateľa spolu s komentármi	Systém zobrazí profil používateľa, v ktorom bude stromová štruktúra zadaných komentárov.	Komentáre sú korektne zobrazené na používateľskom profile.	

9. Deviaty šprint – Prvá svetová vojna

V rámci deviateho šprintu sme riešili nasledovné používateľské príbehy:

- F5 – Znormalizovanie SQL súborov
- F6 – Naplnenie DB dátami
- F7 – Zjednotenie pridávania a vyhľadávania
- F8 – Redizajn + plagát
- US31 – Úprava vyhľadávania podľa mena
- US32 – Úprava vyhľadávania podľa znalostí a zručností
- US33 – Výber dát podľa vzťahov
- US34 – Zobrazenie hierarchie pri pridaní a vyhľadaní znalostí

9.1. F5 – Znormalizovanie SQL súborov

V rámci sprehľadnenia pomocných súborov obsiahnutých v projektovom adresári sme sa rozhodli kategorizovať a rozdeliť databázové SQL príkazy uložené v projektovom adresári *protected/data*.

9.1.1. Analýza

Bolo potrebné analyzovať obsah priečinka, kde sa nachádzalo väčšie množstvo .sql súborov, ktoré obsahovali rôzne verzie databáz a rôzne verzie vložených dát. Našim cieľom ich bolo vhodne štrukturovať a sprehľadniť, aby boli zrozumiteľné a čitateľné aj pre používateľov mimo nášho tímu.

9.1.2. Návrh

Na základe analýzy sme sa rozhodli rozdeliť dáta nasledujúcimi kategóriami:

- SQL príkazy pre vytvorenie prázdnej databázy
- SQL príkazy pre vytvorenie používateľov a ich práva
- SQL príkazy pre naplnenie tabuliek potrebnými dátami pre používanie systému
- SQL príkazy s vlastnými používateľskými dátami (pridané znalosti a zručnosti konkrétnym používateľom, poznámky, komentáre...)

9.1.3. Implementácia

Ako bolo navrhnuté v časti návrhu, priečinok bol zbavený prebytočných starých súborov a dát, pričom dáta, ktoré sme sa rozhodli zachovať sme premiestnili do príslušných súborov:

- *create_only.sql* – Obsahuje príkazy pre vytvorenie prázdnych tabuliek a atribútov, definovanie primárnych a cudzích kľúčov a multiplicitu či obmedzenia jednotlivých vzťahov

- *inserts_users.sql* – Súbor obsahujúci príkazy pre vloženie používateľov systému do databázy.
- *inserts_legit.sql* – Súbor obsahujúci všetky relevantné dáta k používaniu systému ako preddefinované znalosti a zručnosti, kategórie a hierarchiu.
- *inserts_custom* – Súbor obsahujúci všetky vlastné pridané dáta ako pridané znalosti konkrétneho používateľa, pridané zručnosti konkrétneho používateľa, poznámky, komentáre či definovanie vlastných zručností a znalostí.

```
/*
 * Notes
 */

INSERT INTO note (id_note, id_user, id_user_from, creation_time,
last_modify_time, id_level, id_user_unmapped, note, permission) VALUES
(59, 7, 7, '2012-04-11 11:52:39.5659', '2012-04-11 11:52:39.5659', 42,
NULL, 'getget', 1);

INSERT INTO note (id_note, id_user, id_user_from, creation_time,
last_modify_time, id_level, id_user_unmapped, note, permission) VALUES
(60, 7, 7, '2012-04-11 12:39:00.030584', '2012-04-11 12:39:00.030584',
NULL, 6, 'faef', 1);

INSERT INTO note (id_note, id_user, id_user_from, creation_time,
last_modify_time, id_level, id_user_unmapped, note, permission) VALUES
(61, 7, 7, '2012-04-11 12:39:08.693353', '2012-04-11 12:39:08.693353',
NULL, 6, 'faefaef', 1);

/*
 * Knowledge & Skills of TestUsers
 */

 /* Test User ID = 2 */
INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (131, 2, 2, NULL, 22, 0);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (132, 2, 2, NULL, 42, 2);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (133, 10, 2, NULL, 13, 3);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (134, 10, 2, NULL, 6, 4);
```

9.2. F6 – Naplnenie DB dátami

Používateľ bude mať k dispozícii databázu naplnenú dátami.

9.2.1. Analýza

Keďže sa pracovalo väčšinou na testovacích dátach, bolo potrebné naplniť databázu reálnymi dátami. Z veľkej časti sa vychádzalo z dokumentu, v ktorom sú spísané požadované vlastnosti na pracovné pozície. Tento dokument vznikol na základe analýzy vyše 400 pracovných pozícií v IT oblasti z webu *profesia.sk*. Boli identifikované základné pracovné pozície (v

našej doméne ekvivalent so zručnosťami) a požadované znalosti potrebné pre tieto pozície. Následne sa doplnila databáza týmito dátami. Rovnako sa pre ne vytvorila hierarchia.

9.2.2. Implementácia

Pre každú tabuľku v našej databáze sa vytvorilo pravidlo, ktoré zabezpečovalo, aby sa nepridávali jednotlivé záznamy do databázy, ak už nejaký existuje (insert ignore).

```
/*
 * Category
 */
CREATE RULE "my_table_on_duplicate_ignore" AS ON INSERT TO category
  WHERE EXISTS(SELECT 1 FROM category
               WHERE (id_category)=(NEW.id_category))
  DO INSTEAD NOTHING;
INSERT INTO category (id_category, name) VALUES (1, 'other');
INSERT INTO category (id_category, name) VALUES (2, 'debug');
INSERT INTO category (id_category, name) VALUES (3, 'language');
INSERT INTO category (id_category, name) VALUES (4, 'version');
INSERT INTO category (id_category, name) VALUES (5, 'test');
INSERT INTO category (id_category, name) VALUES (6, 'server');
INSERT INTO category (id_category, name) VALUES (7, 'API');
INSERT INTO category (id_category, name) VALUES (8, 'framework');
INSERT INTO category (id_category, name) VALUES (9, 'database');
INSERT INTO category (id_category, name) VALUES (11, 'protocol');
INSERT INTO category (id_category, name) VALUES (12, 'software');
INSERT INTO category (id_category, name) VALUES (13, 'platform');
INSERT INTO category (id_category, name) VALUES (14, 'OS');
INSERT INTO category (id_category, name) VALUES (15, 'hardware');
INSERT INTO category (id_category, name) VALUES (10, 'IDE');
DROP RULE "my_table_on_duplicate_ignore" ON category;
```

9.3. F7 – Zjednotenie pridávania a vyhľadávania

Pre jednoduchosť používania sme sa rozhodli pre zjednotenie vyhľadávania aj pridávania oboch: znalostí a zručností. To znamená, že z tých istých prvkov rozhrania bude možné vyberať ako znalosti, tak i zručnosti.

9.3.1. Analýza

Z analytického hľadiska je potrebné vedieť rozlíšiť znalosť a zručnosť pri dátach preposielaných do grafických prvkov.

9.3.2. Návrh

Pre rozlíšenie či sa jedná o znalosť alebo zručnosť môžeme použiť *id* atribút, ktorý sa prenáša vždy spolu s názvom znalosti/zručnosti. Modifikujeme ho pridaním prefixu „*k*_“ alebo „*s*_“ pre samotné *id*.

Teda pre znalosti sme pridali prefix „*k*_“ a zručnosti „s_“.

9.3.3. Implementácia

Pri implementácií bolo nutné zmeniť kód na nasledovných miestach:

- výber dát pri vyhľadávaní/pridávaní do autocomplete dialógu
- samotná časť pridávania znalostí/zručností
- časť prijatia parametrov pri vyhľadávaní znalostí zručností

Výber dát sa zmenil tak, že sa vyberajú ako znalosti, tak i zručnosti s tým že sa pridávajú spomínané prefixy. Vybrané dáta spájajú do jedného poľa, ktoré je predané ďalej autocomplete prvku:

```
public function getSkillDataList() {
	$skills = Skill::model()->datalist()->findAll();
	$source = array();
	foreach ($skills as $row) {
		$source[] = array(
		'id' => 's_'.$row->attributes['id_skill'],
		'value' => $row->attributes['name'],
		);
	}
	return $source;
}



public function getKnowledgeDataList() {
	$skills = Knowledge::model()->datalist()->findAll();
	$source = array();
	foreach ($skills as $row) {
		$source[] = array(
		'id' => 'k_'.$row->attributes['id_knowledge'],
		'value' => $row->attributes['name'],
		);
	}
	return $source;
}

public function getDataList() {
	return array_merge($this->getKnowledgeDataList(), $this-
>getSkillDataList());
}
```

Do časti pre vyhľadávanie bolo dorobené zostavenie podmienky pre vyhľadávanie podľa prefixov obdržaných parametrov:

```
$cond = split("_", $selected, 2);
$column = '';
switch($cond[0]) {
	case 's' :
		$column = 'id skill';
		break;
	case 'k' :
		$column = 'id_knowledge';
		break;
}
/* zostav CDbCriteria*/
$criteria->addCondition('t.'.$column.' = :'.$selected, 'OR');
```

Obdobný trik bol použitý I pri pridávaní, kedy podľa prefix bolo potrebné zistiť typ parametra a priradiť príslušný cudzí kľúč:

```
($item[0] == "k") ? $user_level->id_knowledge = (int) $item[1] :
$user_level->id_skill = (int) $item[1];
```

9.4. F8 – Redizajn + plagát

Potrebný redizajn produktu a vytvorenie plagátu na konferenciu IIT.SRC.

9.4.1. Analýza

Redizajn – Počas práce na projekte sa vyskytli požiadavky na niekoľko dizajnových zmien produktu. Jednalo sa hlavne o veľkosť textových polí a tlačidiel. Tlačidlá by teda mali byť menšie ako aj textové polia. Ďalej je potrebné mať vycentrované a správne zarovnané tabuľky pre ich lepšiu prehľadnosť.

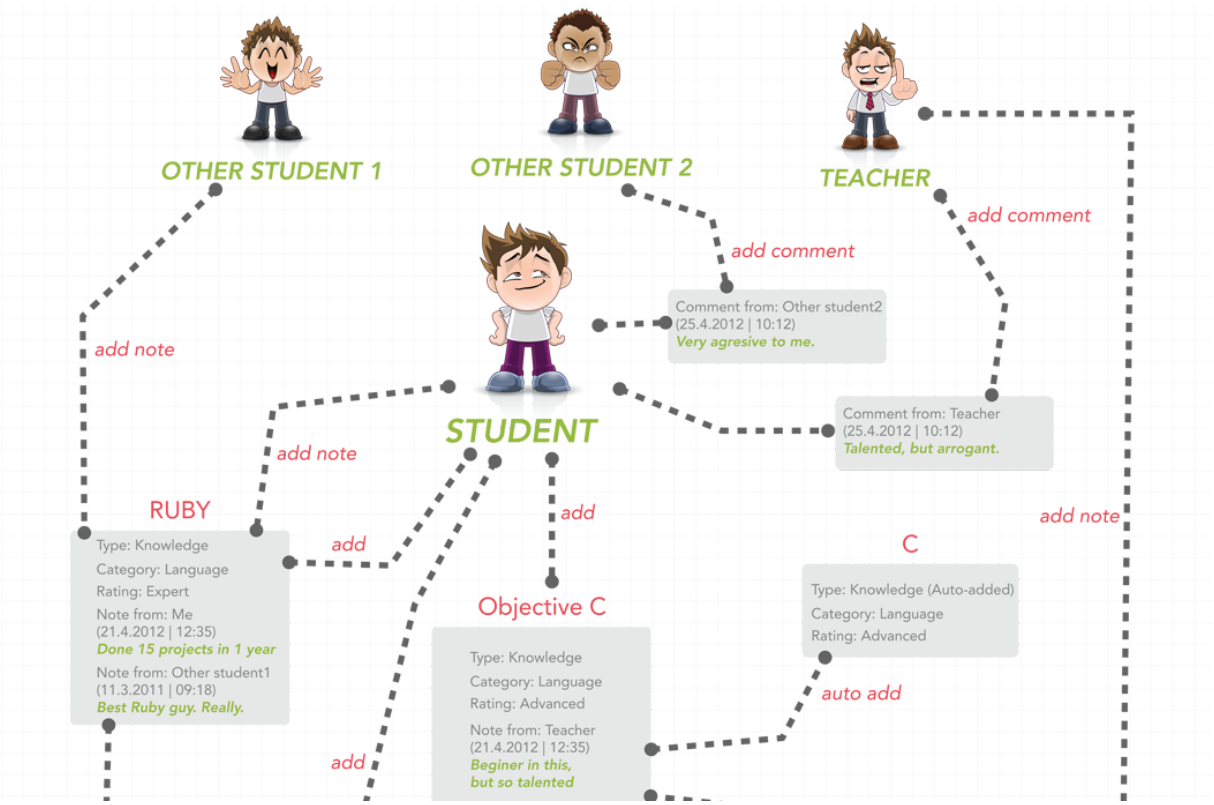
Plagát – Plagát na IIT.SRC by mal obsahovať všetky dôležité aspekty nášho projektu. Hlavným prvkom by mala byť mapou znázornená naša hierarchia znalostí a zručností. Ďalšími dôležitými prvkami sú pridané funkcie nášho systému ako aj technológie, ktoré sme použili na implementáciu nášho projektu.

9.4.2. Návrh

Redizajn – Navrhnuté boli zmeny na rôznych obrazovkách a hlavne tam, kde sa vyžadovala priama interakcia používateľa zo systémom. Prerobené teda budú tlačidlá, tabuľky a textové polia. Na obrázku 9.4.1 je zobrazený návrh zmeny zadávania resp. vyberania znalosti alebo zručnosti.

Obr. č. 9.4.1 - Zmena dizajnu pridávania znalostí a zručností

Plagát – Na plagáte teda bude mapa, vyjadrujúca hierarchiu znalostí a zručností, technológie, ktoré sme použili a výhody nášho systému. Na obrázku 9.4.2 je znázornená časť mapy znázorňujúcej našu hierarchiu.



Obr. č. 9.4.2 - Návrh mapy na plagát

9.4.3. Implementácia

Na implementáciu sme nepotrebovali vytvorenie modelov alebo nových controllerov. Upravovali sa iba views a prislúchajúci css súbor.

9.5. US31 – Úprava vyhľadávania podľa mena

Ako priemerný študent chcem, aby sa po vyhľadaní podľa mena zobrazili ihneď znalosti študenta s odkazom späť na vyhľadanie so zachovaným "query".

9.5.1. Analýza

Náš systém po vyhľadaní študenta presmeruje používateľa na obrazovku s jeho znalosťami a zručnosťami. Problémom však ostáva, keď odkazy naľavo nekorešpondujú s prezeraným používateľom. Tieto odkazy by bolo dobré zmeniť na také, aby čo najviac korešpondovali s vyhľadaným študentom. Pri odkazoch by mala byť možnosť vrátenia sa na výsledky vyhľadávania.

9.5.2. Návrh

V ľavom menu budú vystupovať 3 položky (odkazy) :

- Odkaz na profil prezeraného používateľa,
- Zobrazenie jeho znalostí a zručností,
- Odkaz späť na výsledok vyhľadávania.

Samostatnou položkou budú komentáre k používateľovi, ktoré pridávajú iní používatelia. Na tejto obrazovke sa budú dať prezerať, ale nie pridávať.

Na obrázku 9.5.1 je znázornené menu, ktoré sa objaví po vyhľadaní.

Obr. č. 9.5.1 - Ľavé menu po vyhľadaní používateľa

9.5.3. Implementácia

Pri implementácii sa pracovalo s controllermi a s viewmi na zobrazenie znalostí a vyhľadávania.

View

Vo viewe *showSkill* sme museli upraviť menu tak, aby sa inak zobrazovalo pri rôznych používateľoch a zobrazeniach.

```
<?php
      if($_GET['id'] == Yii::app()->user->id) {
            echo "<li><a href='?r=/skill/addSkill&id=".Yii::app()->user-
>getId()."'>PRIDANIE ZNALOSTÍ</a></li>";
            echo "<li class='active_menu'><a
href='?r=skill/showUserSkills&id=";
```

```
            echo Yii::app()->user->getId();
            echo "'>ZOBRAZENIE ZNALOSTÍ</a></li>";
       }
       else {
            echo "<li><a
href='?r=user/view&id=".$_GET['id']."'>PROFIL</a></li>";
            echo "<li class='active_menu'><a
href='?r=user/searchUsers'>ZOBRAZENIE ZNALOSTÍ</a></li>";
       }
      if(isset($_GET['s'])) {
            echo "<li><a href='?r=user/searchUsers'>SPÄŤ</a></li>";
      }
      if(isset(Yii::app()->session['zdroj'])) {
            echo "<li><a
href='?r=user/search&sessionVar=true'>SPÄŤ</a></li>";
      }
      echo "<li class='comments_h'>KOMENTÁRE K POUŽÍVATEĽOVI</li>";
      if(isset($userComments) && !empty($userComments)) {
            $this->widget('zii.widgets.jui.CJuiAccordion', array(
                        'panels'=> $userComments,
                        'options'=>array(
                                    'collapsible'=>true,
                                    'active'=>1,
                        ),
                        'htmlOptions'=>array(
                                    'style'=>'width:200px;'

                        ),
            ));
      }
      else {
            echo '<p class="comments_empt">(prázdne)';
      }
?>
```

Controller

V controlleroch *skillController* a *userController* bolo potrebné nastaviť *session* pre ukladanie vyhľadávania, aby sa mohlo spätne použiť.

9.5.4. Testovanie

Na testovanie sme použili dva akceptačné testy

Názov	Zobrazenie profilu vyhľadaného používateľa		**ID Testu**	9.5-01
Rozhrani	Obrazovka zoznamu znalostí		**ID UC**	9.5
Účel	Zistenie, či odkaz v menu odkazuje na profil vyhľadaného používateľa			
Vstupné podmienky	Zobrazenie zoznamu znalostí a zručností po vyhľadaní používateľa			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Kliknutie na odkaz profilu	Zobrazenie profilu vyhľadaného používateľa	Zobrazený profil vyhľadaného používateľa	

Názov	Vrátenie sa späť na výsledky vyhľadávania		**ID Testu**	9.5-02
Rozhrani	Obrazovka zoznamu znalostí		**ID UC**	9.5
Účel	Zistenie, či po kliknutí na odkaz späť sa nám zachovajú predošlé výsledky hľadania			
Vstupné podmienky	Zobrazenie zoznamu znalostí a zručností po vyhľadaní používateľa			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Kliknutie na odkaz späť	Zobrazenie predošlých výsledkov hľadania	Zobrazené predošlé výsledky vyhľadávania	

9.6. US32 – Úprava vyhľadávania podľa znalostí a zručností

Ako používateľ chcem mať pri vyhľadávaní študentov podľa znalostí a po kliknutí na konkrétneho študenta možnosť vrátenia sa späť na výsledky vyhľadávania (zoznam Študentov).

9.6.1. Analýza

Pre používateľa môže byť často krát užitočné sa vrátiť na výsledky vyhľadávania. Pri prezeraní znalostí a zručností konkrétneho študenta bude zobrazená linka s označením SPÄŤ, ktorá umožní vrátiť sa na zoznam študentov vyhovujúcich predošlému vyhľadaniu.

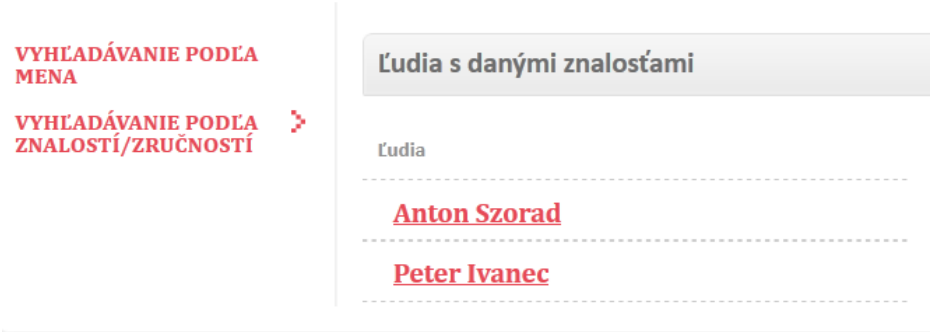
Zoznam študentov budeme ukladať do session a odkaz zobrazovať len v prípade, ak sa tento zoznam študentov bude nachádzať v session.

Prípad použitia	**Editácia Profilu**
ID	9.6
Stručný popis	Pri prezeraní profilu konkrétneho študenta bude zobrazený odkaz s označením SPÄŤ, ktorý umožní vrátiť sa na zoznam študentov vyhovujúcich predošlému vyhľadaniu.
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený a vyhľadá si študentov podľa určitých znalostí
Hlavný scenár	1. Prípad použitia začína po zobrazení formuláru na vyhľadanie podľa znalosti/zručnosti. 2. Používateľ si prezrie znalosti a zručnosti určitého študenta

	vyhovujúce danému vyhľadávaniu. 3. Používateľ sa bude chcieť vrátiť na zoznam vyhľadaných študentov a klikne na linku SPÄŤ. 4. Systém zobrazí zoznam vyhľadaných študentov.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

9.6.2. Návrh

Na obrázku 9.6.1 je zobrazená obrazovka znázorňujúca zoznam študentov s tými istými hľadanými znalosťami resp. zručnosťami.



Obr. č. 9.6.1 - Obrazovka znázorňujúca zoznam študentov s tými istými hľadanými znalosťami resp. zručnosťami.

Ďalší obrázok znázorňuje obrazovku so znalosťami a zručnosťami daného študenta. Tu je podstatný odkaz SPÄŤ, ktorý umožní vrátiť sa na zoznam študentov.

Obr. č. 9.6.3 - Obrazovka so znalosťami a zručnosťami daného študenta

9.6.3. Implementácia

Upravil sa súbor *showskill.php*, kde ak je v session uložený zoznam študentov, tak sa zobrazí odkaz SPÄŤ.

```
if(isset($_GET['s'])) {
            echo "<li><a href='?r=user/searchUsers'>SPÄŤ</a></li>";
      }
```

Ďalej sa upravil súbor *showSkillUserView.php* kde namiesto

```
<?php echo $data['name'];?>
```

sa pridal kód

```
<a class='skill_td_data' href='?r=/skill/showUserSkills&id=<?php echo $data['id'];
?>'>
  <?php echo $data['name'];?>
</a>
```

Najdôležitejšie bolo upraviť *UserController.php*, kde sa upravila funkcia *actionSearch()*:

```
public function actionSearch($sessionVar=Null) {
            $model = new KnowledgeAutocomplete();

            if(isset($sessionVar)) {
                  if(isset(Yii::app()->session['zdroj'])){
                        $this->render('/knowledgeuser/showKnowledgeUser',
array('zdroj_ludi' => Yii::app()->session['zdroj']));
                        Yii::app()->end();
                  }
            }

            unset(Yii::app()->session['zdroj']);
```

9.6.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Možnosť vrátenia sa späť na zoznam študentov s danými znalosťami resp. zručnosťami		**ID Testu**	9.6-01
Rozhrani	Obrazovka so znalosťami a zručnosťami daného študenta		**ID UC**	9.6
Účel	Vrátenie sa späť na zoznam študentov s danými			
Vstupné podmienky	Viacero nájdených študentov s rovnakými znalosťami a zručnosťami			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Vyhľadanie študentov s danými znalosťami resp. zručnosťami	Zobrazenie zoznamu študentov	Zobrazený zoznam študentov	
2.	Kliknutie na položku zo zoznamu študentov	Zobrazenie znalostí a zručností daného študenta	Zobrazené znalosti a zručnosti daného študenta, pričom sa vpravo nachádza odkaz SPÄŤ	
3.	Kliknutie na odkaz SPÄŤ	Vrátenie sa na obrazovku zobrazujúcu zoznamu študentov vyhovujúcich podmienkam vyhľadávania	Znovu zobrazený zoznam študentov	

9.7. US33 – Výber dát podľa vzťahov

9.7.1. Analýza

Cieľom tohto US bolo rozšírenie funkcionality pre prácu s dátami, ktorá bola použitá v inej funkcionalite. Takto majú programátori možnosť použiť funkcie pre prácu s dátami.

9.7.2. Návrh

Funkcionalita berie informácie z databázy. Cieľom je spraviť výber hierarchie podľa typu ich vzťahu, podľa toho, či sa jedná o vzťah znalosť - znalosť, znalosť - zručnosť alebo zručnosť - zručnosť. Dôležitou funkcionalitou je aj rekurzívna funkcia, ktorej vstupom je znalosť alebo zručnosť a výstupom jej hierarchia smerom hore od vstupu. Napr. ak sa zadá znalosť Java ME, vráti sa Java, Java Programátor a Programátor.

Príklad výstupu funkcie *getKnowledgeKnowledgeByRelation* s parametrom 2. Parameter 2 je typ vzťahu *partOf*:

```
Array (
  [0] => Array (
    [id_relation] => 1
    [id_relation_type] => 2
    [id_from] => 9
    [id_to] => 56
    [name_from] => ASP.net
    [name_to] => .NET
    [relation_type] => k2k
```

```
  )
  [1] => Array (
    [id_relation] => 2
    [id_relation_type] => 2
    [id_from] => 11
    [id_to] => 56
    [name_from] => C#
    [name_to] => .NET
    [relation_type] => k2k
  )
  [2] => Array (
    [id_relation] => 3
    [id_relation_type] => 2
    [id_from] => 38
    [id_to] => 28
    [name_from] => JDBC
    [name_to] => J2EE
    [relation_type] => k2k
  )
  [3] => Array (
    [id_relation] => 5
    [id_relation_type] => 2
    [id_from] => 40
    [id_to] => 28
    [name_from] => JNDI
    [name_to] => J2EE
    [relation_type] => k2k
  )
  [4] => Array (
    [id_relation] => 4
    [id_relation_type] => 2
    [id_from] => 39
    [id_to] => 28
    [name_from] => JMS
    [name_to] => J2EE
    [relation_type] => k2k
  )
)
```

9.7.3. Implementácia

Menil sa len jeden súbor a to *SkillController.php*:
Vrátenie všetkých vzťahov znalosť - znalosť určitého typu:

```
public function getKnowledgeKnowledgeByRelation($relationType) {
            $data = Relation::model()->with('idFromKnowledge',
'idToKnowledge')->findAll(
            't.id_relation_type = :relation AND t.id_from_knowledge IS NOT
null AND t.id_to_knowledge IS NOT null', array(
                  ':relation' => $relationType));

            $result = array();
            foreach($data as $row) {
                  $result[] = array(
                        'id_relation' => $row->id_relation,
                        'id_relation_type' => $row->id_relation_type,
                        'id_from' => $row->id_from_knowledge,
                        'id_to' => $row->id_to_knowledge,
                        'name_from' => $row->idFromKnowledge->name,
                        'name_to' => $row->idToKnowledge->name,
                        'relation_type' => 'k2k',
```

```
                );
            }
            return $result;
      }
```

Obdobne boli vytvorené aj funkcie pre vzťahy typu „znalosť – zručnosť“ a „zručnosť – zručnosť“.
Funkcia *getDataByPrecondition* vráti všetky vzťahy typu precondition:

```
public function getDataByPrecondition() {
            return array_merge(
                  $this->getKnowledgeKnowledgeByRelation(1),
                  $this->getSkillKnowledgeByRelation(1),
                  $this->getSkillSkillByRelation(1)
            );
      }
```

Rovnako boli vytvorené aj *getDataByPartOf()* a *getDataByMandatory()*.
Rekurzívna funkcia vracajúca hierarchiu:

```
public function getRelationHierarchy($id,$level, $returnarray=Array()) {
    if(($this->checkSkillOrKnowledgeInDataByAllTypes($id))===true){
      $childs = $this->getDataWhereIsIdTo($id);
      foreach($childs as $child){
         $child["level"]=$level;
         if(!(in_array($child, $returnarray))){
           array_push($returnarray, $child);
           $returnarray = array_merge($this-
>getRelationHierarchy($child['id_from'],$level+1, $returnarray));
         }
      }
    }
   return($returnarray);
  }
```

Pomocné funkcie:

```
public function getDataWhereIsIdTo($id_to) {
        $dataByAllTypes = $this->getDataByAllTypes();
    $result = array();
    foreach ($dataByAllTypes as $row){
      if($row['id_to'] === $id_to){
        $result[] = $row;
      }
    }
    return $result;
 }

  public function getDataByAllTypes() {
            return array_merge(
                  $this->getDataByPrecondition(),
                  $this->getDataByPartOf(),
                  $this->getDataByMandatory()
            );
      }
```

9.7.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zobrazenie hierarchie danej znalosti		**ID Testu**	9.7-01
Rozhranie	--		**ID UC**	9.7
Účel	Zistenie, či sa zobrazia správne údaje			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Použitie rekurzívnej funkcie *getRelationHierarchy()*	Výstup funkcie	Skontrolovaný výstup funkcie porovnaním s dátami v DB	

9.8. US34 – Zobrazenie hierarchie pri pridaní a vyhľadaní znalostí

Chcem, aby sa zobrazila hierarchia pri pridávaní aj zobrazení znalostí.

9.8.1. Analýza

Keďže náš systém obsahuje podrobnú hierarchiu niektorých znalostí a zručností, je potrebné zobraziť znalosti a zručnosti v tejto hierarchii aj používateľovi. Je potrebné mu ju zobraziť prehľadne, aby sa v nej čo najlepšie vyznal a pochopil ju.

Prípad použitia	**Zobrazenie Hierarchie**
ID	1
Stručný popis	Používateľ si môže prezrieť hierarchiu danej znalosti/zručnosti
Primárny aktéri	Prihlásený používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky znalostí a zručností používateľa. 2. Používateľ si klikne na ikonku hierarchickej štruktúry a následne sa mu zobrazí ako daná znalosť a zručnosť vystupuje v hierarchii.
Výstupné	Žiadne

podmienky	
Alternatívne scenáre	Žiadne

9.8.2. Návrh

Na zobrazenie hierarchie budeme používať textové vysvetlenie s tým, že znalosti budú postupne odsadené podľa ich pozícii v hierarchii. Znalosti, ktoré sú na rovnakej úrovni budú odsadené rovnakým spôsobom. Návrh takejto hierarchie ku konkrétnej znalosti je na obrázku 9.8.1 (dáta sú len ilustračné).

Obr. č. 9.8.1 - Zobrazenie hierarchie pre konkrétnu znalosť

9.8.3. Implementácia

Pre zobrazovanie hierarchie sme potrebovali upraviť view súbor *showSkill*. Pridané bolo tlačítko pre zobrazenie a skrytie hierarchie,

```
<?php
    echo '<input type="button" class="skill_button_hierarchy_view"
onClick="showComments(\'hier_'.$subdata['id_level'].'\');"/>';
?>
```

ktoré využíva JavaScript na zobrazovanie hierarchie.

```
<tr>
  <td class="hierarchy_text" colspan="3">
    <div id="hier_<?php echo $subdata['id_level']; ?>"
style="display:none;">
      <?php
        $level_order = array();
        foreach ($hierarchy as $index => $variable) {
          if ($index==$subdata['id_knowledge']) {
            foreach ($variable as $value) {
              echo '<div style="margin-left:' . 2*$value['level'] .
'em;">';
              echo $value['name_from'];
              echo ' zahŕňa ';
```

```
                echo '<b>'.$value['name_to'] . '</b> ';
                echo '</div>';
                echo '<br />';
              }
            }
          }
        ?>
      </div>
    </td>
</tr>
```

9.8.4. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zobrazenie hierarchie		**ID Testu**	9.8-01
Rozhrani	Obrazovka zoznamu znalostí		**ID UC**	9.8
Účel	Zistenie, či po kliknutí na ikonku hierarchie sa hierarchia zobrazí			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Kliknutie na ikonku hierarchie	Zobrazenie hierarchie k danie zručnosti alebo znalosti	Zobrazená hierarchia k danej zručnosti alebo znalosti	

10. Desiaty šprint – Druhá svetová vojna

V rámci desiateho šprintu sme riešili nasledovné používateľské príbehy:

- US35 – Naplnenie dát
- US36 – Administrátorské rozhranie
- US37 & US38 – Nové API funkcie

10.1. US35 – Naplnenie dát

Toto US bolo zamerané na prípravu prototypu pre potreby prezentovania verejnosti na konferencii IIT.SRC. Bolo potrebné pridať testovacích používateľov a taktiež vyplniť v databáze ich vzťahy s tabuľkami a atribútmi.

10.1.1. Analýza

Pre potreby prezentovania bolo potrebné pridať konkrétnych testovacích používateľov do systému a pridať hodnoty do ich profilov. Taktiež bolo potrebné určiť, akými znalosťami a zručnosťami ten-ktorý používateľ disponuje, mať určenú ich úroveň, prípadne mať priradenú poznámku či komentár. Tieto údaje bolo potrebné pridať k správnym existujúcim SQL súborom.

10.1.2. Návrh

V rámci deviateho šprintu bola vykonaná úprava SQL súborov, preto bola vyriešená otázka kam konkrétne SQL príkazy umiestniť. Každý používateľ dostal priradený profil a ľubovoľný počet znalostí a zručností na rôznych úrovniach.

10.1.3. Implementácia

Príkazy pre pridanie samotných používateľov boli umiestnené do súboru *inserts_users.sql*.

```
/*test useri*/

INSERT INTO users (id_user, login, password, name, surname, titles, email,
mobile, info, gender, work)
VALUES (2,'test1','098f6bcd4621d373cade4e832627b4f6','User1','Test',
'Bc.','xtest1@stuba.sk','88083582','Jedna sa o prveho testovacieho
pouzivatela.','male', 'nezamestnany student');

INSERT INTO users (id_user, login, password, name, surname, titles, email,
mobile, info, gender, work)
VALUES (18,'test2','098f6bcd4621d373cade4e832627b4f6','User2','Test',
'Ing.','xtest2@stuba.sk','0914558291','Jedna sa o druheho testovacieho
pouzivatela.','male', 'zamestnany');

INSERT INTO users (id_user, login, password, name, surname, titles, email,
mobile, info, gender, work)
VALUES (10,'test3','098f6bcd4621d373cade4e832627b4f6','User3','Test',
'Mgr.','xtest3@stuba.sk','','Jedna sa o tretieho testovacieho
```

```
pouzivatela.','female', 'nezamestnany student');
```

Príkazy pre priradenie znalostí a zručností boli umiestnené v súbore *inserts_custom.sql*.

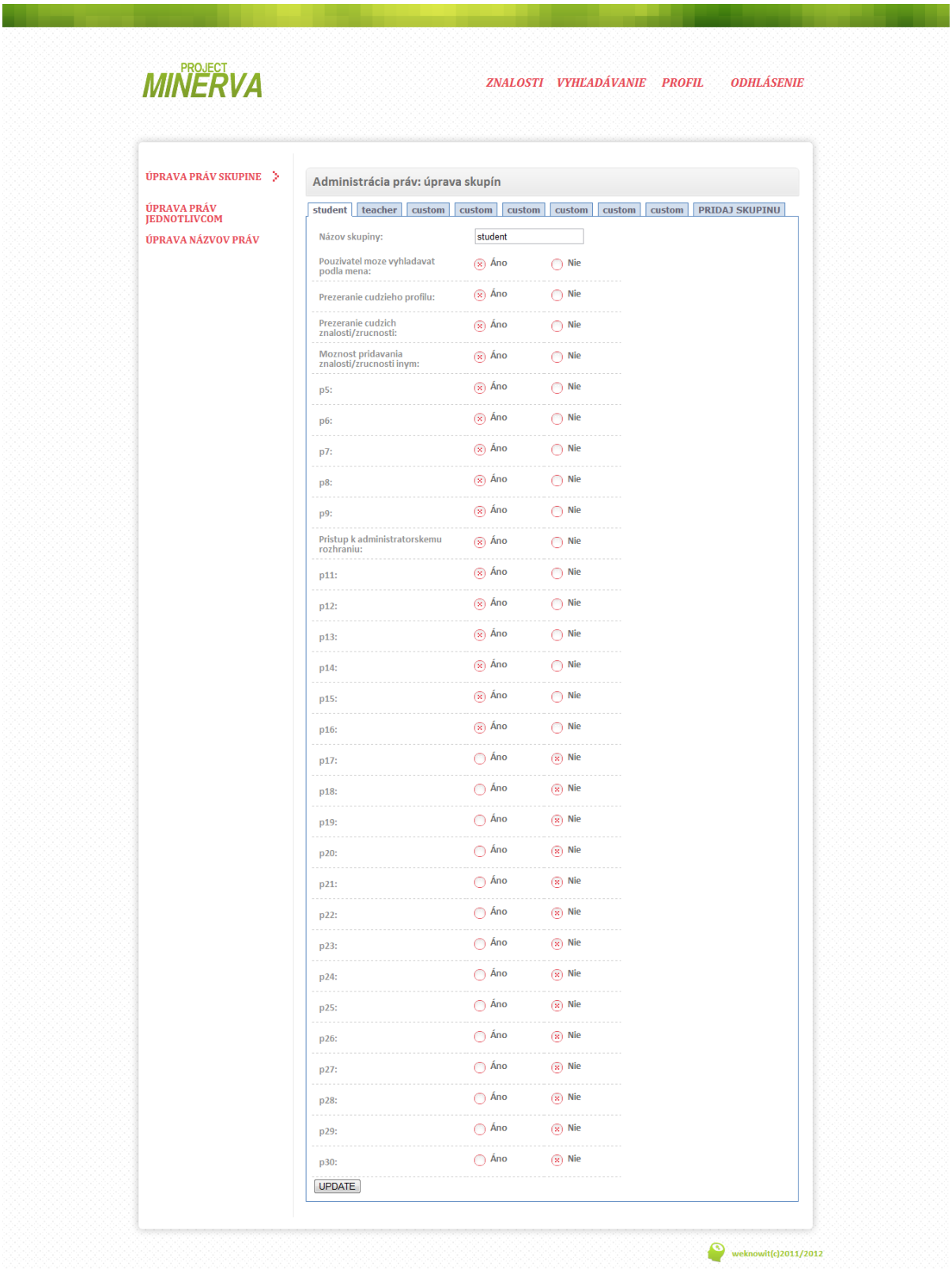
```
/* Test User ID = 12 */

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (169, 12, 12, NULL, 17, 2);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (170, 12, 12, NULL, 12, 2);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (171, 12, 12, NULL, 49, 0);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (172, 3, 12, NULL, 38, 2);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (173, 17, 12, 5, NULL, 3);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (174, 17, 12, 7, NULL, 1);

/* Test User ID = 13 */

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (175, 2, 13, NULL, 13, 0);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (176, 15, 13, NULL, 11, 1);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (177, 15, 13, 7, NULL, 3);

INSERT INTO level (id_level, id_from, id_to, id_skill, id_knowledge,
rating) VALUES (178, 14, 13, 8, NULL, 4);
```

10.2. US36 – Administrátorské rozhranie

Cieľom US bolo upraviť administrátorské rozhranie, aby fungovalo s novým databázovým modelom a pridať úpravu práv používateľov.

10.2.1. Analýza

Úprava práv sa skladá z troch hlavných častí, každá má vlastnú obrazovku. Na prvej obrazovke sa upravujú práva skupinám, ktoré sa prideľujú jednotlivým používateľom. Cez túto obrazovku sa dajú vytvárať aj nové skupiny. Na druhej obrazovke sa dajú upravovať práva konkrétnym používateľom. Používateľ sa vyberie prostredníctvom auto-complete vstupného poľa, následne sa mu priradia práva jednej z vytvorených skupín. Tretia obrazovka slúži na úpravu mien konkrétnych práv.



Obr. č. 10.2.1 - Obrazovka Úprava práv skupine

Obr. 10.2.2 - Obrazovka Úprava práv jednotlivcom

Obr. 10.2.3 - Obrazovka Úprava názvov práv

Prípad použitia	**Úprava Práv Skupine**
ID	1
Stručný popis	Používateľ s právami administrátora má možnosť upraviť práva skupine.
Primárny aktéri	Používateľ s právami administrátora
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený, Používateľ má práva administrátora
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky administrácie, Úprava práv skupine.

	2. Používateľ vyberie záložku s názvom skupiny, ktorú chce upraviť. 3. Používateľ vyberie jednu z možností áno/nie pri všetkých právach, ktoré chce upraviť. 4. Používateľ potvrdí úpravu práv. 5. Systém upraví práva skupiny.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

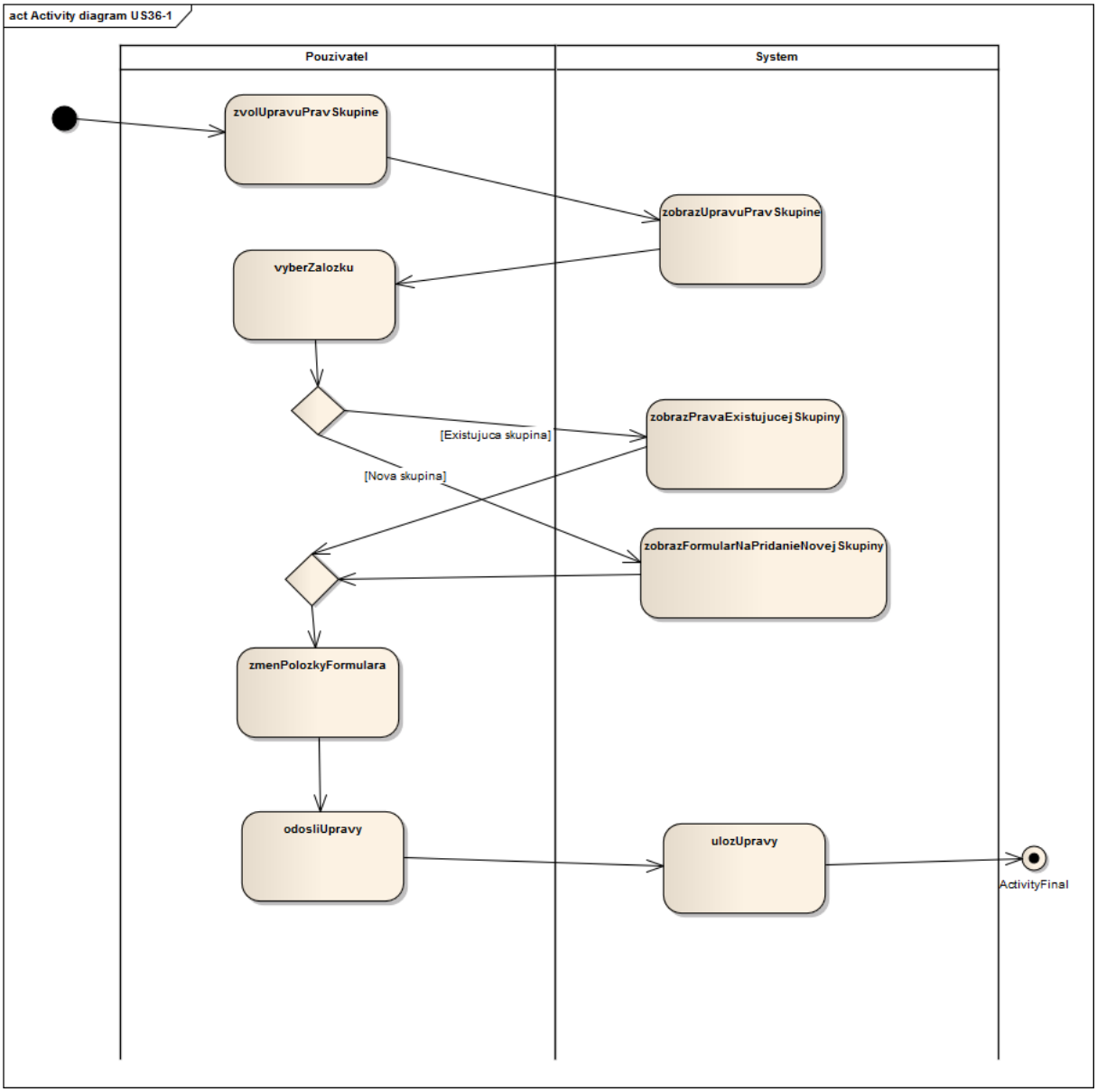
Prípad použitia	**Vytvorenie novej skupiny práv**
ID	2
Stručný popis	Používateľ s právami administrátora má možnosť vytvoriť novú skupinu práv.
Primárny aktéri	Používateľ s právami administrátora
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený, Používateľ má práva administrátora
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky administrácie, Úprava práv skupine. 2. Používateľ vyberie záložku s názvom Pridaj skupinu. 3. Používateľ napíše názov skupiny. 4. Používateľ vyberie jednu z možností áno/nie pri všetkých právach. 5. Používateľ potvrdí vytvorenie skupiny. 6. Systém vytvorí skupinu.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

Prípad použitia	**Úprava práv jednotlivca**
ID	3
Stručný popis	Používateľ s právami administrátora má možnosť upraviť práva jednotlivca.
Primárny aktéri	Používateľ s právami administrátora
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený, Používateľ má práva administrátora
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky administrácie, Úprava práv jednotlivcom. 2. Používateľ vpíše do vstupného poľa názov používateľa, ktorému chce upraviť práva. 3. Používateľ vyberie jednu zo skupín práv, ktorú chce priradiť používateľovi. 4. Používateľ potvrdí úpravu práv. 5. Systém upraví priradí jednotlivcovi práva vybratej skupiny.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

Prípad použitia	**Úprava názvov práv**
ID	4
Stručný popis	Používateľ s právami administrátora má možnosť upraviť názvy práv.
Primárny aktéri	Používateľ s právami administrátora
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený, Používateľ má práva administrátora
Hlavný scenár	1. Prípad použitia začína po zobrazení obrazovky administrácie,

	Úprava názvov práv. 2. Používateľ vyberie riadok práva, ktorý chce upraviť, alebo pridanie nového práva. V tomto riadku má možnosť zmeniť názov práva a jeho popis. 3. Používateľ potvrdí úpravu práv kliknutím na UPDATE v príslušnom riadku. 4. Systém upraví názov a popis práva.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

10.2.2. Návrh



Obr. 10.2.4 : Diagram aktivít – úprav práv skupine

10.2.3. Implementácia

Implementácia bola vykonaná v rámci controlerra *AdminController* a 6 súborov, ktoré plnia funkciu view. Nachádzajú sa v adresári *protected/views/admin*.

10.2.4. Testovanie

Názov	Zmena práv skupiny		**ID Testu**	10.2-01
Rozhrani	Obrazovka Úprava práv skupine		**ID UC**	10.2
Účel	Zmena práv skupiny			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	

1.	Kliknutie na záložku s názvom skupiny práv, ktorú chcem zobraziť.	Zobrazenie skupiny práv so zoznamom a hodnotami všetkých práv.	Zobrazila sa skupina práv so všetkými právami a ich hodnotami.
2.	Zmena hodnôt niektorých práv.	Hodnoty sa po potvrdení zmien zmenia a v rámci skupiny sa objavia pri upravených právach nové hodnoty.	Upravené práva v rámci skupiny sa zmenili a zobrazujú sa aktuálne hodnoty.

10.3. US37 – Nové API funkcie – AISID + AIS meno

Nakoľko US37 a US38 zdieľajú tú istú funkcionalitu, sú zdokumentované v jednej kapitole. Pre US37 chcem funkcionalitu, aby keď zadám AIS ID, alebo AIS meno, tak mi vráti vzájomne rozlíšené znalosti a zručnosti danej osoby vrátane úrovne tohto vzťahu.

Pre US38 chcem funkcionalitu, aby keď zadám

- údaje konkrétneho používateľa spolu so znalosťou alebo so zručnosťou , vráti mi poznámky daného vzťahu
- konkrétnu znalosť alebo zručnosť, vráti mi v akých kategóriách sa nachádza
- údaje konkrétneho používateľa, vráti mi všetky komentáre k jeho osobe

10.3.1. Analýza

Keďže projekt, ak bude použitý na fakulte, bude po naplnení študentmi obsahovať informácie o ich znalostiach a zručnostiach, tak by mali byť tieto informácie dostupné aj mimo systému pomocou API. API by malo poskytovať údaje pomocou štruktúrovaných dát (XML, JSON) a pre najlepšie použitie by malo byť dostupné cez odkaz priamo zo systému. Údaje by sa mali poskytovať priamo z databázy pre čo najaktuálnejšie údaje.

Prípad použitia	**Zobrazenie údajov cez API**
ID	1
Stručný popis	Ľubovoľný používateľ môže pomocou API získať znalosti/zručnosti študenta pomocou jeho AIS mena z nášho systému.
Primárny aktéri	Ľubovoľný používateľ
Vedľajší aktéri	Žiadny
Vstupné podmienky	Žiadne
Hlavný scenár	1. Prípad použitia začína po zadaní linku s vyplneným parametrom *uid* do prehliadača alebo v samotnom kóde

	aplikácie využívajúcej API systému 2. Pri zadaní linku do prehliadača sa zobrazí pole znalostí a zručností. V kóde sa toto pole uloží do premenne resp. sa využije podľa programátora.
Výstupné podmienky	Žiadne
Alternatívne scenáre	Žiadne

10.3.2. Návrh

Ako výstup nášho API sme zvolili štruktúrované dáta pomocou JSON. Po zadaní špecifického linku vráti systém používateľovi podľa zadaného AIS mena pole, ktoré oddeľuje znalosti od zručností a od ostatných používateľom zadaných zručností a znalostí. Ku každej znalosti/zručnosti je taktiež priradená úroveň, ktorú používateľ zadal.

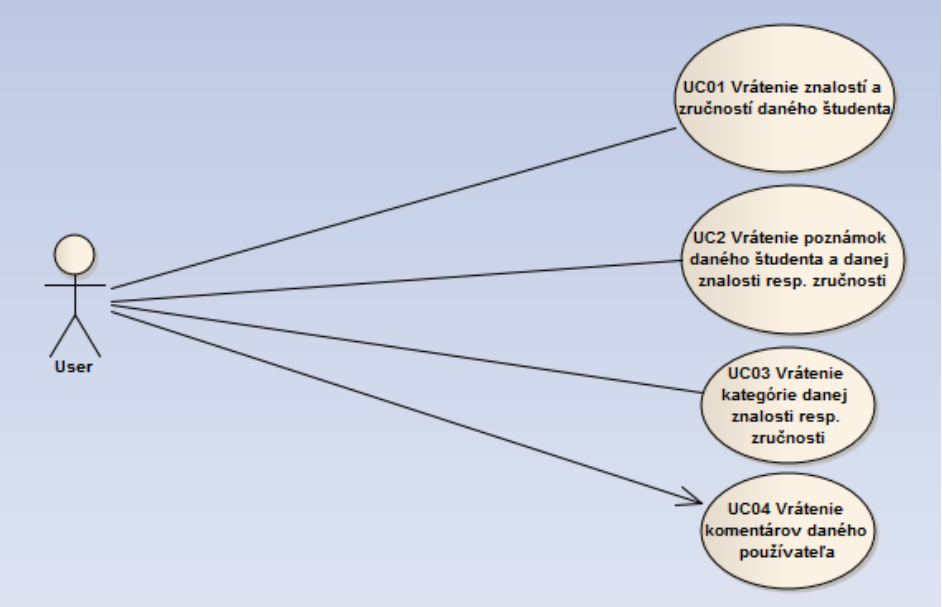
Ak zadáme zlé AIS meno tak systém vráti prázdne polia. Je to z dôvodu bezpečnosti aby niekto z vonku neskúšal existenciu používateľov.

Výstupom teda bude dvojrozmerné pole (key-value) s nasledujúcou štruktúrou:

Kľúčom v tomto poli je vždy označenie či sa jedná o znalosti, zručnosti alebo znalosti/zručnosti zadané študentom a hodnotou je pole znalostí resp. zručností aj s úrovňou. Štruktúra teda vyzerá nasledovne:

```
{"knowledges":[{"name":"znalost1","rating":2},{"name":"znalost2","rating":0
}, … {"name":"znalostN","rating":0}],
"skills":[{"name":"zrucnost1","rating":0}],
"unmapped":[{"name":"uzivatelomPridanaZnalostZrucnost1","rating":0},{"name"
:" uzivatelomPridanaZnalostZrucnost2","rating":0 }]}
```

Na nasledujúcom diagrame sú znázornené prípady použitia.



Obr. č. 10.3.1 - Diagram prípadov použitia pre US37 a US38

10.3.3. Implementácia

Vzhľadom na povahu tejto úlohy sme nepotrebovali vytvárať model ani view. Zmeny v databáze tiež neboli potrebné.

Controller

Vytvorili sme controller *ServiceController.php*, kde sme vytvorili funkcie na vyberanie údajov z databázy. Následne sme potom cez funkciu *actionRequestUserSkillsByID* tieto údaje poskytli ako JSON.

```
public function actionRequestUserSkillsByID($uid) {
            $data = array(
                  'knowledges' => array(),
                  'skills' => array(),
                  'unmapped' => array(),
            );
            $criteria = new CDbCriteria;
            if($uid != "") {
                  $criteria->condition = "login = :login";
                  $criteria->params = array(':login' => $uid);
            }
            else {
                  echo CJSON::encode($data);
                  Yii::app()->end();
            }
            $user = Users::model()->find($criteria);
            if(!$user) {
                  echo CJSON::encode($data);
                  Yii::app()->end();
            }
            $data['knowledges'] = $this->getKnowledgesOfUsers($user-
>id_user);
            $data['skills'] = $this->getSkillsOfUsers($user->id_user);
            $data['unmapped'] = $this->getUnmappedOfUser($user->id_user);
            echo CJSON::encode($data);
            Yii::app()->end();
```

```
	}
```

Podobne sa postupovalo aj pre US38, kde sa pridali funkcie *actionRequestCommentsOfUser*, *actionRequestCategoryofSkillOrKnowledge* a *actionRequestNoteofUserAndSkillOrKnowledge*.

```
public function actionRequestCommentsOfUser($uid) {
		$data = array(
			'comments' => array(),
		);
		$criteria = new CDbCriteria;
		if($uid != "") {
			$criteria->condition = "login = :login";
			$criteria->params = array(':login' => $uid);
		}
		else {
			echo CJSON::encode($data);
			Yii::app()->end();
		}
		$user = Users::model()->find($criteria);
		if(!$user) {
			echo CJSON::encode($data);
			Yii::app()->end();
		}
		$data['comments'] = $this->getCommentsOfUser($user->id_user);
		echo CJSON::encode($data);
		Yii::app()->end();
	}

	public function actionRequestCategoryofSkillOrKnowledge($uid) {
		$data = array(
			'categories skill' => array(),
			'categories_knowledge' => array(),
		);
		$criteria = new CDbCriteria;
		if($uid != "") {
			$criteria->condition = "name = :name";
			$criteria->params = array(':name' => $uid);
		}
		else {
			echo CJSON::encode($data);
			Yii::app()->end();
		}
		$skill = Skill::model()->find($criteria);
		$knowledge = Knowledge::model()->find($criteria);
		if(!($skill || $knowledge)) {
			echo CJSON::encode($data);
			Yii::app()->end();
		}
		if(isset($skill)) $data['categories_skill'] = $this-
>getCategoryOfSkill($skill->id_skill);
		if(isset($knowledge)) $data['categories_knowledge'] = $this-
>getCategoryOfKnowledge($knowledge->id_knowledge);
		echo CJSON::encode($data);
		Yii::app()->end();
	}

	public function actionRequestNoteofUserAndSkillOrKnowledge($uid,
$skid) {
```

```
            $data = array(
                  'notes' => array(),
            );
            $criteria = new CDbCriteria;
            $criteria2 = new CDbCriteria;
            if($uid != "" && $skid != "") {
                  $criteria->condition = "name = :name";
                  $criteria->params = array(':name' => $skid);
                  $criteria2->condition = "login = :login";
                  $criteria2->params = array(':login' => $uid);
            }
            else {
                  echo CJSON::encode($data);
                  Yii::app()->end();
            }
            $skill = Skill::model()->find($criteria);
            $knowledge = Knowledge::model()->find($criteria);
            $user = Users::model()->find($criteria2);
            if(!($skill || $knowledge) || !$user) {
                  echo CJSON::encode($data);
                  Yii::app()->end();
            }
            if(isset($skill)){
                  $id_sk = $skill->id_skill;
                  $which = 'skill';
            } else {
                  $id_sk = $knowledge->id_knowledge;
                  $which = 'knowledge';
            }

            $data['notes'] = $this->getNotesOfUserAndSK($user->id_user,
$id_sk, $which);

            echo CJSON::encode($data);
            Yii::app()->end();
      }
```

10.3.4. Použitie API

Prvým krokom pri použití API je znalosť AIS mena, cez ktoré sa používateľ k údajom dostane. Toto AIS meno sa vyplní ako parameter uid v nasledujúcom odkaze:

```
http://team16-
11.ucebne.fiit.stuba.sk/minerva/index.php?r=/service/requestUserSkillsByID&
uid=AISmeno
```

Po správnom zadaní AIS mena systém vráti štruktúrované údaje vo formáte JSON. Praktický príklad použitia aj s výstupom je nasledovný:

Zadaný odkaz:

```
http://team16-
11.ucebne.fiit.stuba.sk/minerva/index.php?r=/service/requestUserSkillsByID&
uid=xpolakovicp1
```

Výstup:

```
{"knowledges":[{"name":"Google Web Toolkit
```

```
(GWT)","rating":2},{"name":"Visual Basic
.NET","rating":0},{"name":"Linux","rating":0},{"name":"Linux","rating":0},{
"name":"Linux","rating":0},{"name":"Windows
Vista","rating":0},{"name":"ASP.net","rating":0},{"name":"Adobe
Flex","rating":0},{"name":"Adobe Flex","rating":0},{"name":"Adobe
Flex","rating":0},{"name":"Adobe Flex","rating":0},{"name":"Microsoft
Virtual Server","rating":0}],"skills":[{"name":"Android
Developer","rating":0}],"unmapped":[{"name":"dqwefqwef","rating":0},{"name"
:"d","rating":0},{"name":"a","rating":0},{"name":"s","rating":0},{"name":"s
","rating":null},{"name":"a","rating":null},{"name":"a","rating":null},{"na
me":"a","rating":null}]}
```


Teda názov znalosti vždy nájdeme pomocou indexu ***name*** a jeho úroveň pomocou indexu ***rating***. Ako je možné vidieť v tomto poli sú oddelené znalosti, zručnosti aj nemapované znalosti/zručnosti teda tie, ktoré si používateľ zadal sám. Výhodou je, že štruktúrované JSON dáta je možné spracovať či už cez AJAX alebo samotné PHP. Jednoduchý príklad pri použití cez AJAX (s použitím Yii konštrukcií):

```
<?php

    echo CHtml::ajaxSubmitButton(
        "Get data",
        array('service/requestUserSkillsByID', 'uid' => 'xpolakovicp1'),
        array('success'=>'js:function(html){
            alert(html);
            }')
    );

?>
```

Veľmi podobne sa používa API aj pre ostatné funkcie.

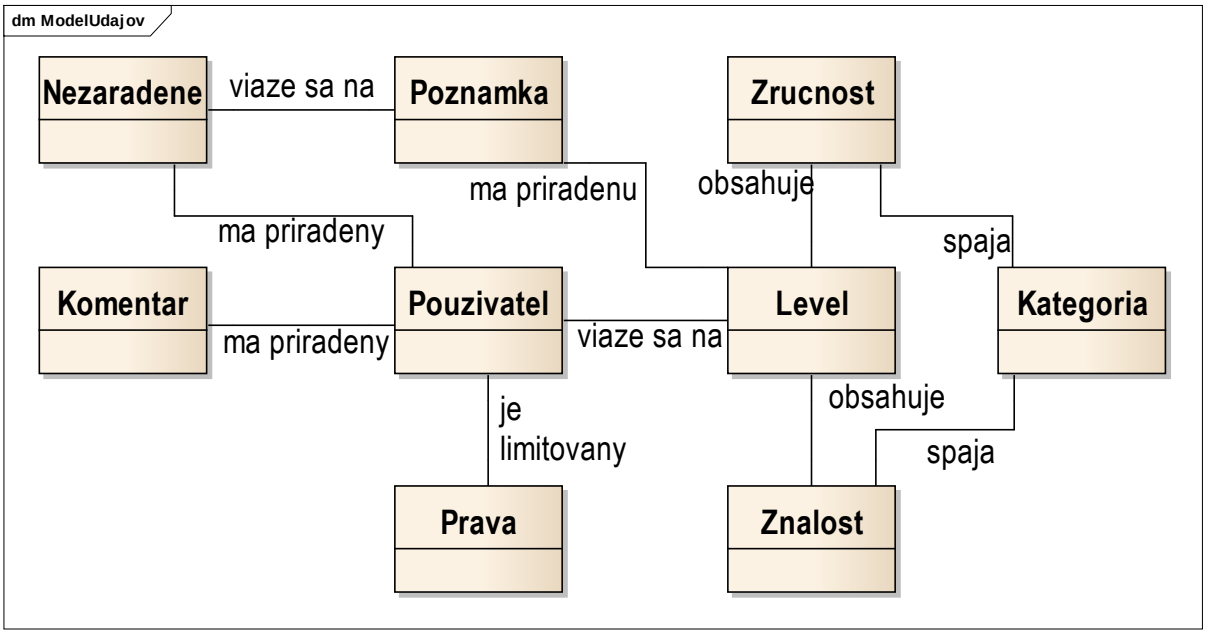
10.3.5. Testovanie

Na testovanie sme použili jeden akceptačný test.

Názov	Zobrazenie JSON dát v prehliadači		**ID Testu**	10.3-01
Rozhrani	--		**ID UC**	10.3
Účel	Zistenie, či sa zobrazia údaje po zadaní linku			
Vstupné podmienky	Znalosť AIS mena študenta			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná akcia**	
1.	Zadanie korektného linku do prehliadača	Zobrazenie znalostí a zručností vo formáte JSON.	Znalosti a zručnosti študenta sa zobrazili vo formáte JSON	

9. Prototyp systému

Súčasná verzia prototypu systému Minerva obsahuje všetku spomenutú funkcionalitu a funguje ako komplexný celok. Nová verzia prototypu, ktorá je nasadená na novú verziu databázy sa skladá z deviatich základných entít. Ich vzájomné vzťahy charakterizuje obrázok číslo 9.1.1.

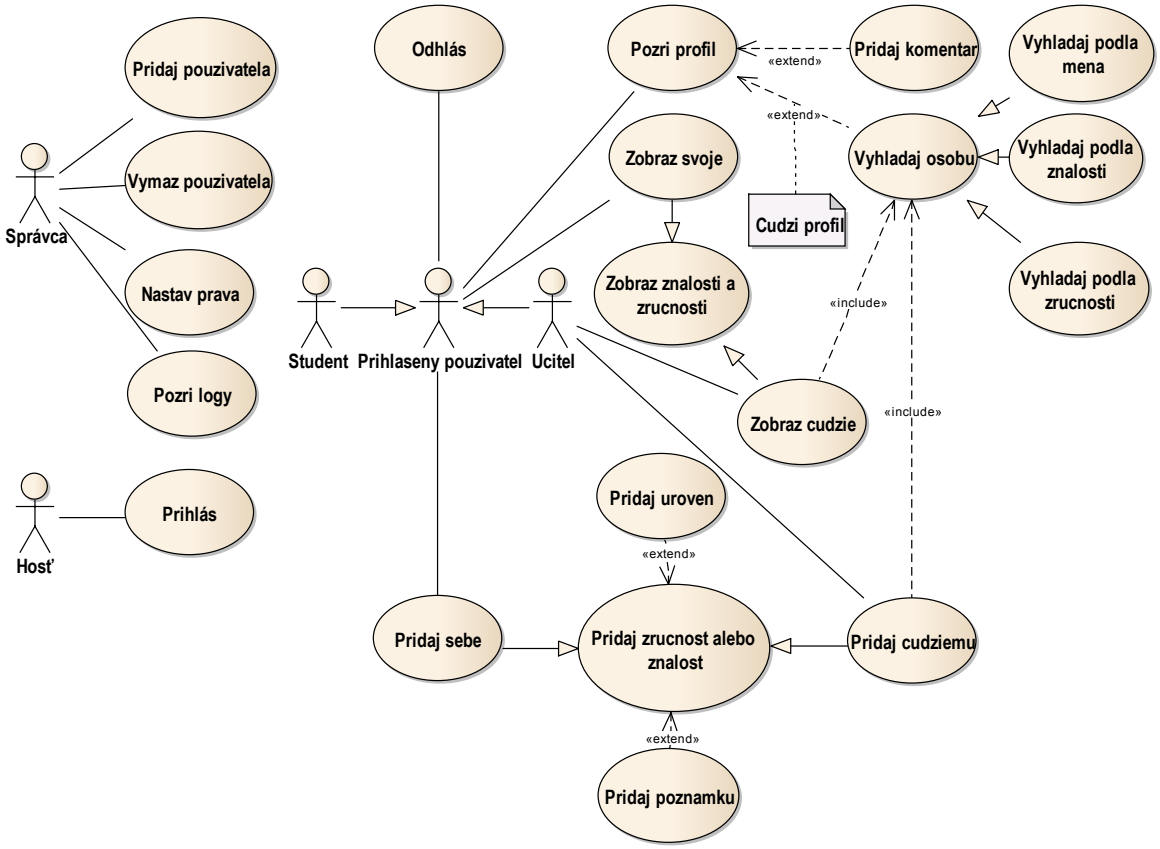


Obr. č. 9.1.1 – Model údajov systému

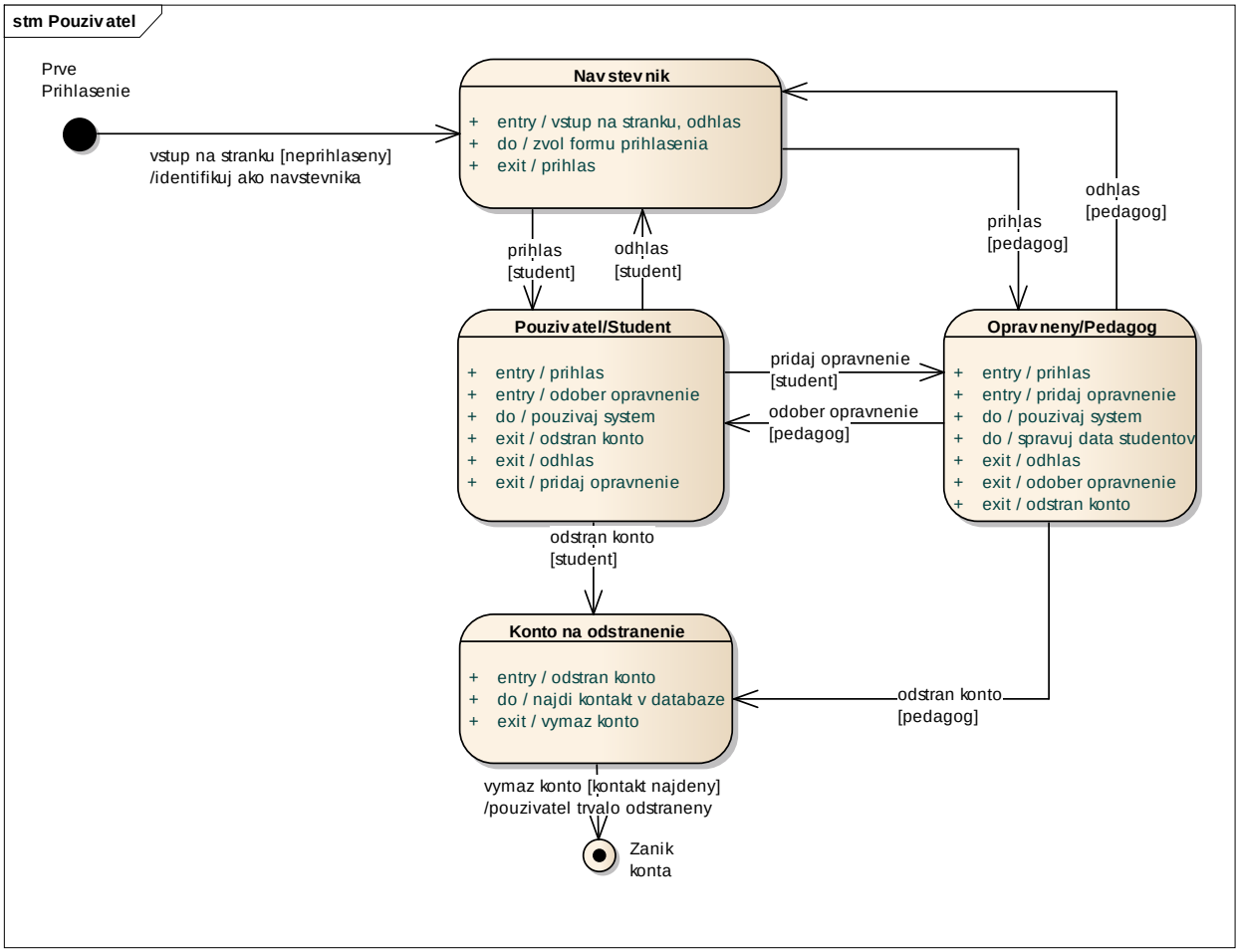
Funkcionalitu ponúkanú prototypom znázorňuje globálny diagram prípadov použitia na obrázku číslo 9.1.2. Na základe používateľských oprávnení identifikujeme celkovo štyri roly – *Študent, Učiteľ, Správca, Hosť*.V diagrame je jasne identifikované, ktoré možnosti systému sú pre koho určené, ktoré prípady použitia sú navzájom závislé a ktoré prepojené.

Lepší pohľad na roly je na obrázku 9.1.3, kde sú na stavovom diagrame znázornené tri základné roly systému – *Hosť*, *Študent* a *Učiteľ*.

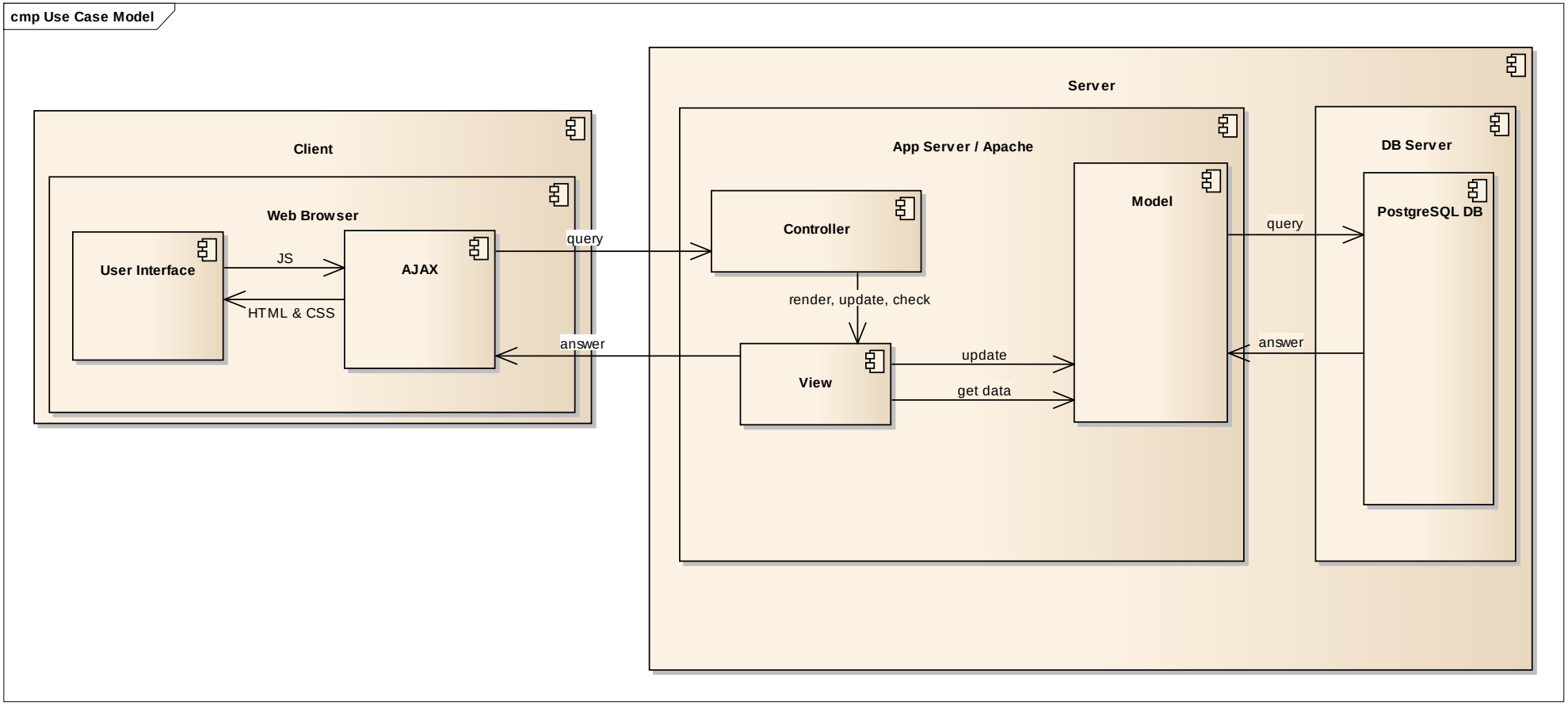
Čo sa týka nasadenia a hardvérovej podpory systému, achitektúra systému je znázornená na obrázku číslo 9.1.4.



Obr. č. 9.1.2 – Celkový diagram prípadov použitia



Obr. č. 9.1.3 – Stavový diagram *Používateľ*



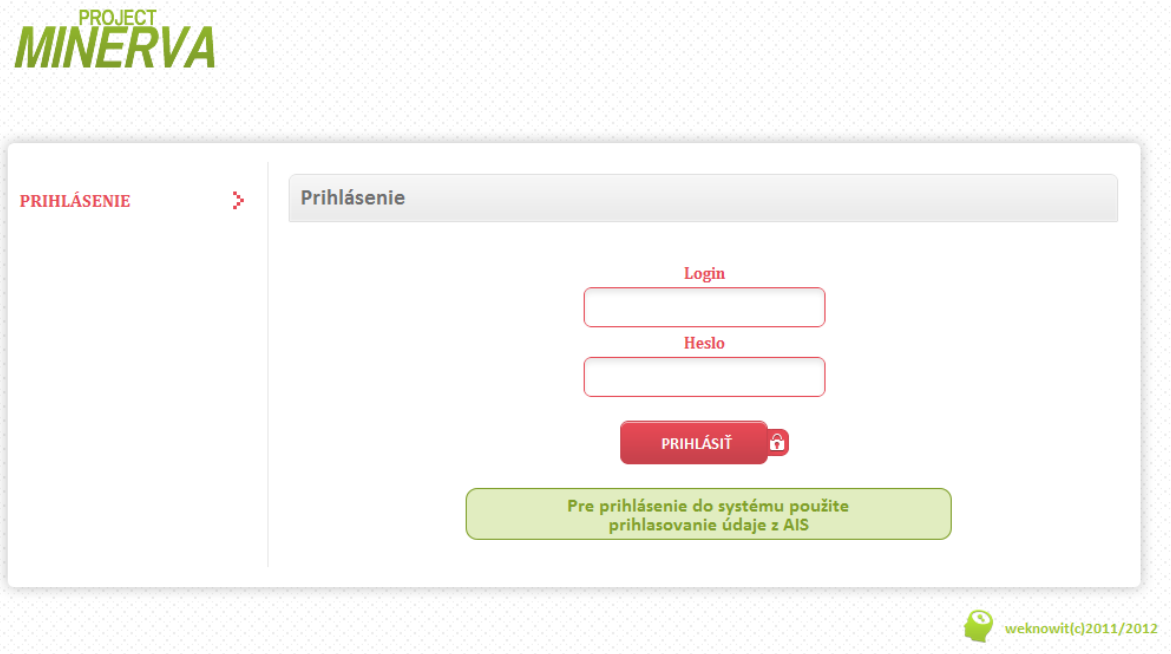
Obr. č. 9.1.4 – Architektúra systému

10. Používateľská príručka

12.1. Prihlásenie

Pri prvej návšteve systému nie je možné používať jeho funkcionalitu, pokiaľ sa neprihlásime.

Pre prihlásenie do systému je potrebné kliknúť na linku *PRIHLÁSENIE* v ľavom paneli. Nasleduje prihlasovacia obrazovka znázornená na obrázku číslo 10.1.1. Je potrebné sa prihlásiť svojimi prihlasovacími údajmi z akademického informačného systému STU.



Obr. č. 10.1.1 – Obrazovka prihlásenia

12.2. Odhlásenie

Pre možnosť odhlásenia musí byť používateľ úspešne prihlásený. Na obrázku číslo 10.1.2 je znázornená úvodná obrazovka systému po prihlásení.

Používateľ má možnosť odhlásiť sa pomocou tlačidla *ODHLÁSENIE*, ktoré sa nachádza v hornom menu systému úplne napravo. Takisto na obrázku vidieť možnosť odhlásiť sa pomocou odkazu v ľavom menu.

Obr. č. 10.2.1 – Úvodná obrazovka systému

12.3. Zobrazenie svojho profilu

Svoj vlastný profil je možné zobraziť z ľubovoľnej obrazovky, pretože sa tiež nachádza v hornom menu, ktoré nás sprevádza na každom skoku systémom.

Ako je znázornené na obrázku číslo 10.1.3, po kliknutí na *PROFIL* sa zobrazí obrazovka profilu aktuálne prihláseného používateľa.

Obr. č. 10.3.1 – Obrazovka používateľského profilu

12.4. Editácia údajov profilu

Na obrázku číslo 10.3.1 máme zobrazenú obrazovku profilu. Nachádza sa tam aj tlačidlo *UPRAV PROFIL*, ktoré umožní upraviť údaje, ktoré sú tam zobrazované. Na obrázku číslo 10.4.1 vidíme obrazovku editácie profilu, ktorá sa objaví po kliknutí na tlačidlo *UPRAV PROFIL*.

Obr. č. 10.4.1 – Úprava profilových údajov

12.5. Vyhľadanie používateľa

Používateľov v systéme môžeme hľadať troma spôsobmi. Podľa ich mena, znalostí, alebo podľa zručností. V hornom menu aplikácie máme tlačidlo *PRIATELIA*. Po kliknutí na toto tlačidlo sa nám zobrazí vyhľadávanie používateľa podľa mena, z odkazov v ľavom menu si môžeme zvoliť aj vyhľadávanie podľa znalostí alebo zručností.

10.5.1. Vyhľadanie používateľa podľa mena

Na úvod sa nám zobrazí obrazovka znázornená na obrázku číslo 10.5.1, ktorá obsahuje vyhľadávací formulár, ktorý podľa zadávaných znakov automaticky filtruje možných kandidátov.

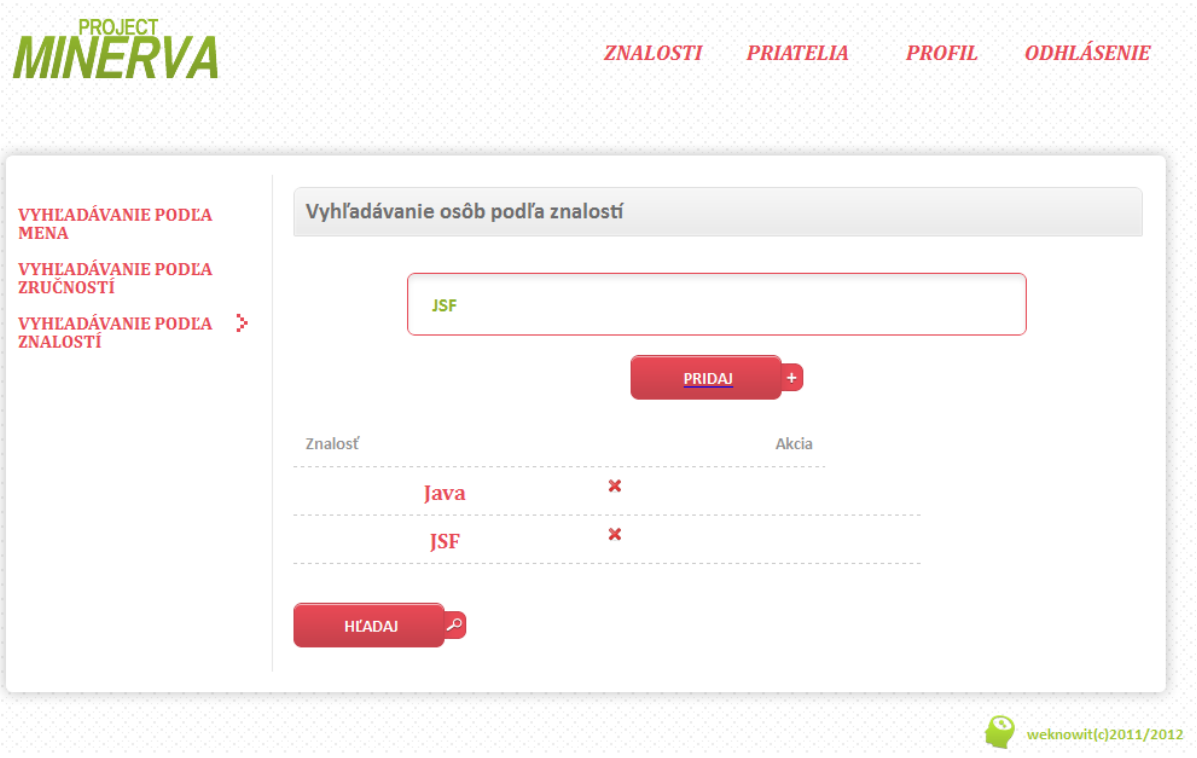
Obr. č. 10.5.1 – Vyhľadávanie podľa mena

10.5.2. Vyhľadávanie podľa znalostí alebo zručností

Vyhľadávanie podľa znalostí a zručností je trochu odlišné. Na tieto vyhľadávania sa dostaneme pomocou odkazov v ľavom menu v sekcii *PRIATELIA*.

Pri vyhľadávaní podľa znalostí alebo zručností máme možnosť najprv vybrať buď množinu znalostí, alebo množinu zručností a následne vyhľadať používateľov, ktorí majú pridanú hľadanú množinu prvkov.

Vyhľadávanie je znázornené na obrázku číslo 10.5.2. Tlačidlom *PRIDAJ* zaraďujeme vybrané znalosti do množiny a tlačidlom *HĽADAJ* spustíme vyhľadávanie.



Obr. č. 10.5.2 – Vyhľadávanie podľa znalostí

12.6. Pridanie komentáru

Pomocou postupu v príručke 10.3 sa dostaneme na svoj profil. Ako je znázornené na obrázku číslo 10.3.1, naspodu je textové pole, do ktorého môžeme ihneď písať. Po napísaní textu klikneme na tlačidlo *ODOŠLI* a komentár je pridaný na profilovú obrazovku ihneď, takisto aj uložené do databázy.