

Dokumentácia k inžinierskemu dielu po 8. šprinte

S T U . .
.
F I I T .
.

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.....	60
5.1.	US20 – Vymazávanie zručností	60
5.1.1.	Analýza.....	60
5.1.2.	Návrh.....	62
5.1.3.	Implementácia	62
5.1.4.	Testovanie	63
5.2.	US21 – Voliteľná úroveň znalostí	63
5.2.1.	Analýza.....	63
5.2.2.	Návrh.....	64
5.2.3.	Implementácia	65
5.2.4.	Testovanie	66
5.3.	US22 – Pridávanie poznámok	66
5.3.1.	Analýza.....	66
5.3.2.	Návrh.....	67
5.3.3.	Implementácia	68
5.3.4.	Testovanie	69
6.	Šiesty šprint – Husitská vojna	70
6.1.	US23 – Poznámkovanie znalostí iného používateľa	70
6.1.1.	Analýza.....	70
6.1.2.	Návrh.....	71
6.1.3.	Implementácia	71
6.1.4.	Testovanie	72
6.2.	US24 – Komentovanie používateľov	72
6.2.1.	Analýza.....	73
6.2.2.	Návrh.....	74
6.2.3.	Implementácia	75
6.2.4.	Testovanie	76
6.3.	US25 – Zručnosti.....	76
6.3.1.	Analýza.....	76
6.3.2.	Návrh.....	76
6.3.3.	Implementácia	77
6.3.4.	Testovanie	78

7.	Siedmy šprint – 30 ročná vojna.....	79
7.1.	F1 – Prerobenie databázy	79
7.1.1.	Analýza.....	79
7.1.2.	Návrh.....	80
7.1.3.	Implementácia	82
7.1.4.	Testovanie	82
7.2.	F2 – Vytvorenie vzorových funkčných testov.....	83
7.2.1.	Inštalácia PHPUnit	83
7.2.2.	Inštalácia Selenium	83
7.2.3.	Konvencie.....	84
7.2.4.	Vybrané príklady testov	84
7.3.	F3 – Logovanie.....	86
7.3.1.	Analýza.....	86
7.3.2.	Návrh.....	86
7.3.3.	Implementácia	86
7.4.	F4 – Refaktoring.....	88
7.4.1.	Analýza.....	88
7.4.2.	Návrh.....	88
7.4.3.	Implementácia	88
7.4.4.	Testovanie	88
7.5.	US26 – Vstup od používateľa.....	89
7.5.1.	Analýza.....	89
7.5.2.	Návrh.....	89
7.5.3.	Implementácia	90
7.5.4.	Testovanie	92
8.	Ôsmy šprint – Napoleonská vojna	93
8.1.	US27 – Viditeľnosť poznámky.....	93
8.1.1.	Analýza.....	93
8.1.2.	Návrh.....	94
8.1.3.	Implementácia	95
8.1.4.	Testovanie	97
8.2.	US28 – Zobrazenie znalostí.....	97
8.2.1.	Analýza.....	97
8.2.2.	Návrh.....	98

8.2.3.	Implementácia	99
8.2.4.	Testovanie	99
8.3.	US29 – Zobrazenie poznámok	100
8.3.1.	Analýza.....	100
8.3.2.	Návrh.....	101
8.3.3.	Implementácia	102
8.3.4.	Testovanie	103
8.4.	US30 – Zobrazenie komentárov	104
8.4.1.	Analýza.....	104
8.4.2.	Návrh.....	104
8.4.3.	Implementácia	105
8.4.4.	Testovanie	106
9.	Prototyp systému	107
10.	Používateľská príručka.....	111
10.1.	Prihlásenie.....	111
10.2.	Odhlásenie.....	111
10.3.	Zobrazenie svojho profilu	112
10.4.	Editácia údajov profilu.....	113
10.5.	Vyhľadanie používateľa.....	113
10.5.1.	Vyhľadanie používateľa podľa mena	113
10.5.2.	Vyhľadávanie podľa znalostí alebo zručností	114
10.6.	Pridanie komentáru	115

5. Piaty šprint – 40 ročná vojna

V rámci piateho šprintu sme riešili nasledovné používateľské príbehy:

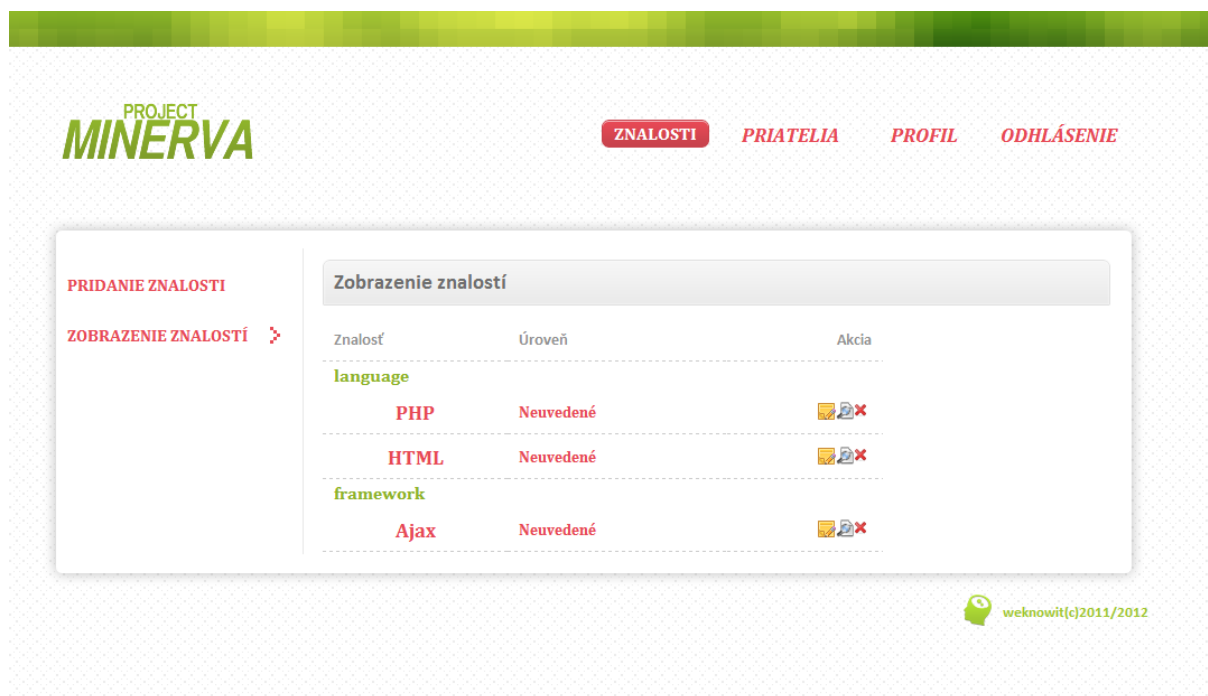
- US20 – Vymazávanie zručností,
- US21 – Voliteľná úroveň znalostí,
- US22 – Pridávanie poznámok.

5.1. US20 – Vymazávanie zručností

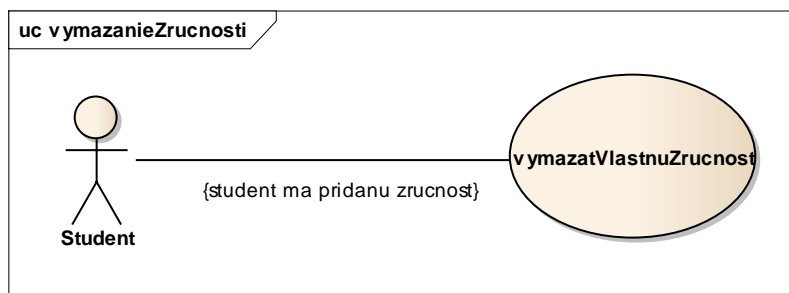
Cieľom používateľského príbehu je umožniť študentovi vymazať si zručnosť, ktorú má už pridanú. Na vymazanie slúži ikonka červeného krížika v zozname vlastných zručností.

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.



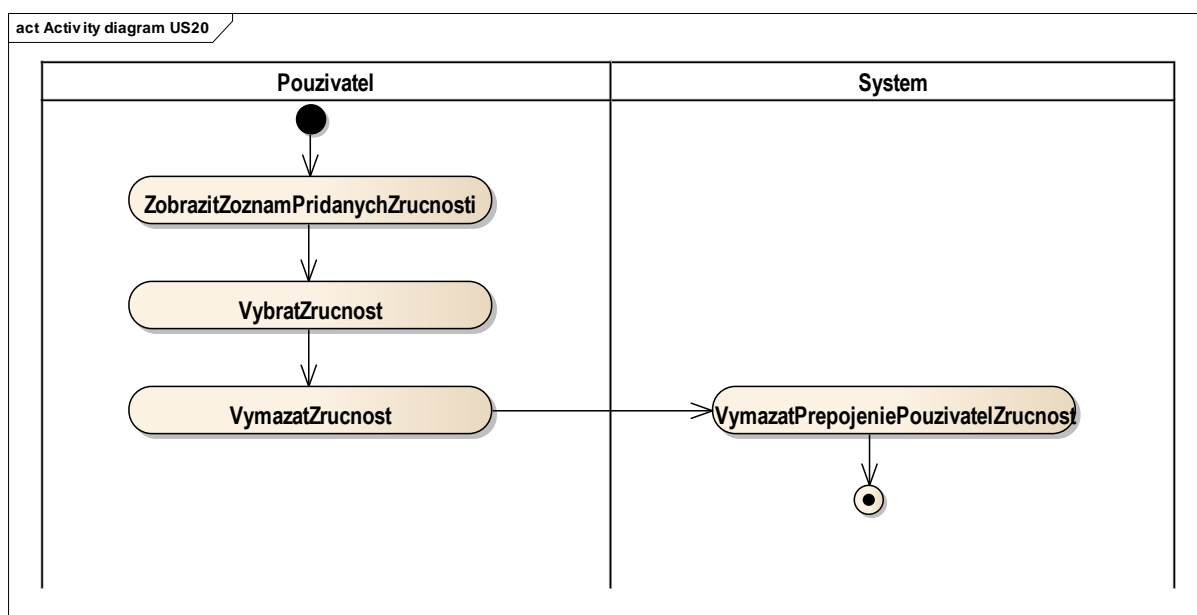
Obr. č. 5.1.1 – Obrazovka zobrazenia zručností



Obr. č. 5.1.2 – Prípád použitia vymazanieZručnosti

Prípád použitia	vymazanieZrucnosti
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	<ol style="list-style-type: none"> 1. Prípád 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

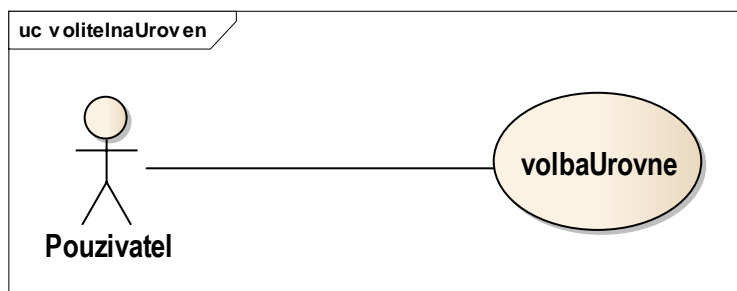
Názov	Úspešné vymazanie vlastnej zručnosti	ID Testu	5.1-01
Rozhranie	Obrazovka Zobrazenie zručností	ID UC	5.1
Účel	Vymazanie vlastnej zručnosti		
Vstupné podmienky	Žiadne		
Výstupné podmienky	Žiadne		
Krok	Akcia	Očakávaná reakcia	Skutočná akcia
1.	Klinutie na ikonku červeného krížika pri zručnosti v 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 – Prípád použitia vol'ba úrovne znalosti

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

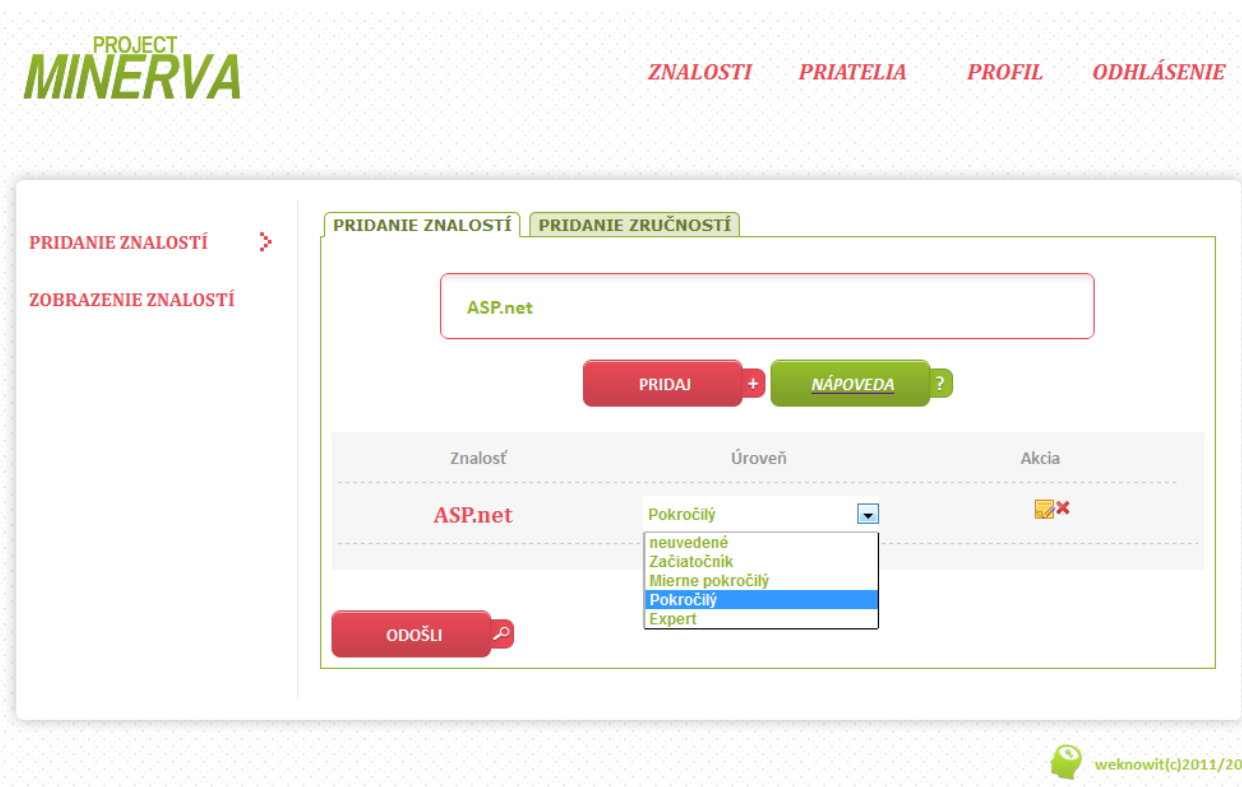
Vstupné podmienky	Používateľ je prihlásený, Používateľ má zvolenú znalosť, ktorú chce pridať
Hlavný scenár	<ol style="list-style-type: none"> 1. Prípád použitia začína po zobrazení obrazovky Pridanie znalostí. 2. Používateľ napíše časť alebo celý názov znalostí do formuláru. 3. Systém zobrazí ponuku 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	<ol style="list-style-type: none"> 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 - *neuveďené*, 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 funkcionálnosť je riešená pomocou javascriptu. Nasledujúci kód znázorňuje akým spôsobom sú zobrazené úrovne pri daných znalostiach.

```

var opt = document.createElement('option');
opt.innerHTML = 'neuveденé';
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 úrovni pre znalosť.	Po kliknutí sa nám zobrazia úrovne pre danú znalosť.	Po kliknutí na combobox sa zobrazil zoznam úrovni 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.

Prípád 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	<ol style="list-style-type: none"> 1. Prípád 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

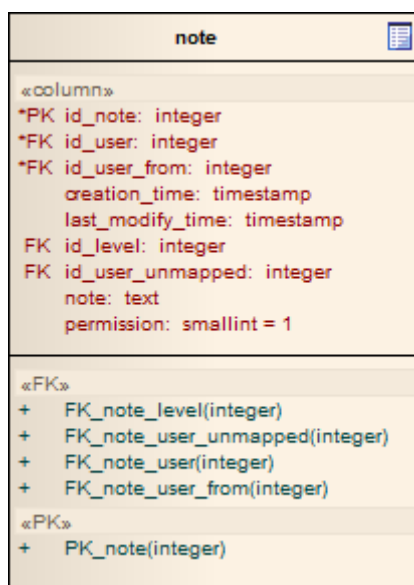
5.3.2. Návrh

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
<i>id_note</i>	PK
<i>id_user</i>	ID používateľa, ktorému bola poznámka priradená
<i>id_user_from</i>	ID používateľa, ktorý poznámku priradil
<i>creation_time</i>	čas vytvorenia poznámky
<i>id_level</i>	ID vybranej znalosti/zručnosti, ku ktorej sa poznámka vzťahuje
<i>id_user_unmapped</i>	ID neznámej znalosti/zručnosti, ku ktorej sa poznámka vzťahuje
<i>note</i>	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.

Prípád 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	<ol style="list-style-type: none"> 1. Prípád použitia začína pri zobrazení zoznamu znalostí používateľa. 2. Používateľ klikne na ikonu poznámky, čím sa mu zobrazí textové 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.

Zobrazenie znalostí

Znalosť	Akcia
IDE	
Adobe Illustrator	
<div style="border: 1px solid #ccc; height: 80px; width: 100%;"></div> <div style="text-align: center; margin-top: 5px;"> <input type="button" value="Odošli"/> </div>	
language	
Java	
other	
CAD/CAM	

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.

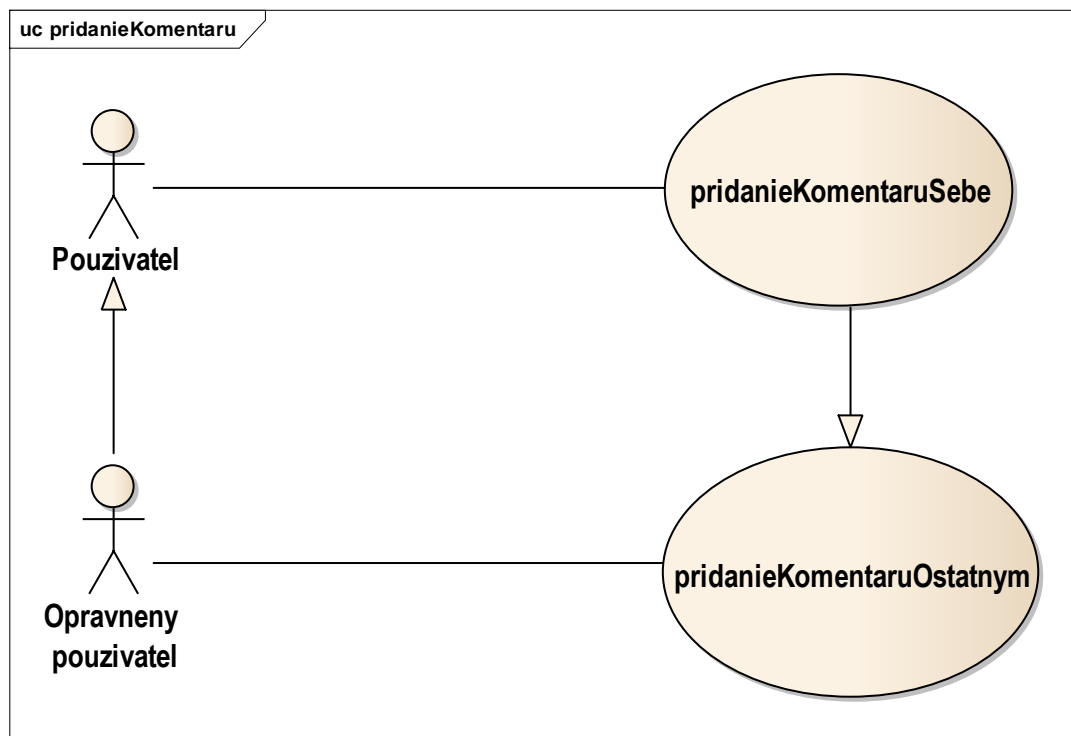
Názov	Zapísanie poznámky ku znalosti iného používateľa	ID Testu	6.1-01
Rozhranie	Obrazovka znalostí iného používateľa	ID UC	6.1
Účel	Zistenie, či sa poznámka zapíše k znalosti iného používateľa		
Vstupné podmienky	Napísaná poznámka v textovom poli		
Výstupné podmienky	Žiadne		
Krok	Akcia	Očakávaná reakcia	Skutočná akcia
1.	Kliknutie na ikonku poznámky.	Zobrazenie textového poľa na vkladanie poznámky.	Textové 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 koničky, záľuby, výhody, v podstate všetky *soft skills*.



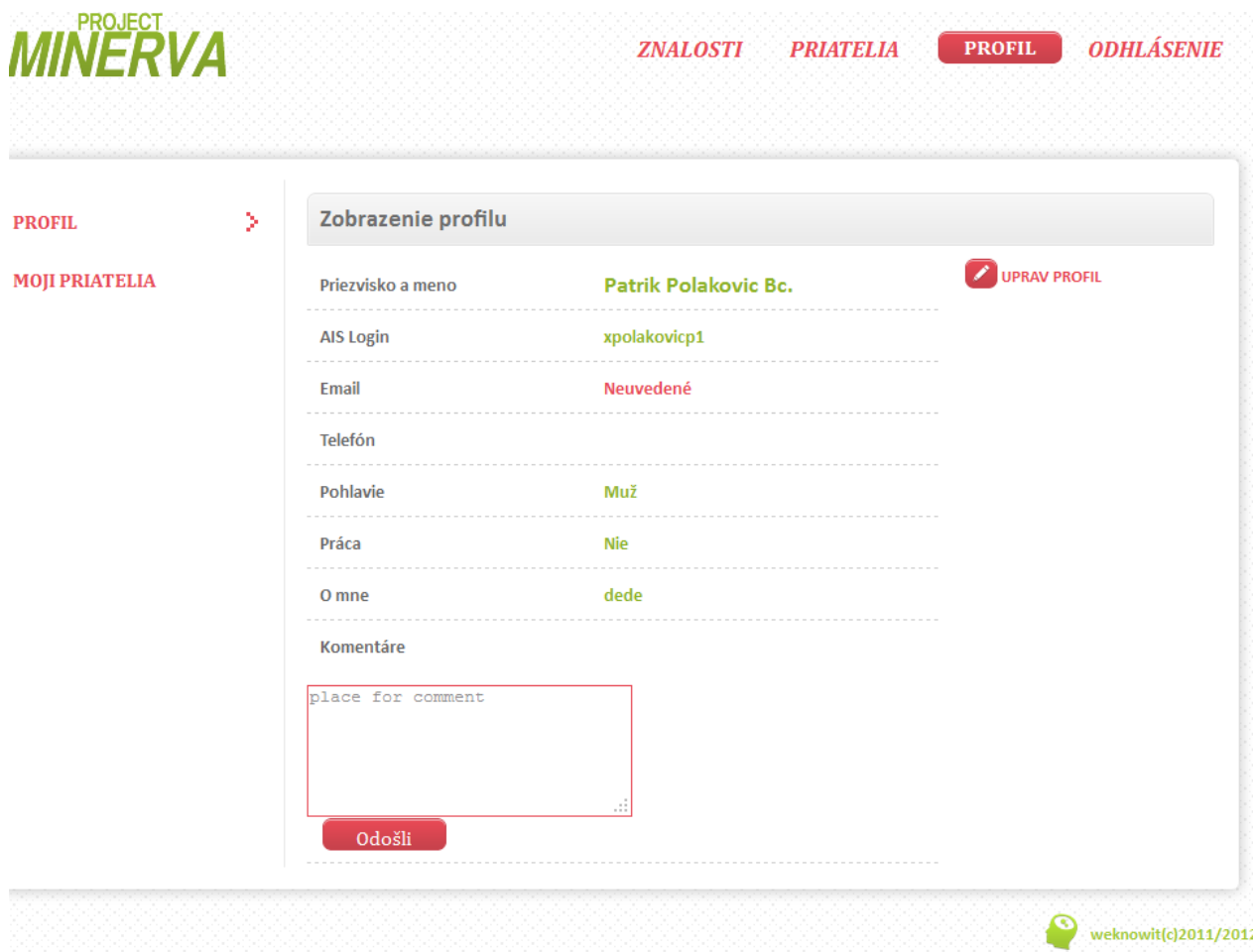
Obr. č. 6.2.1 – Prípady použitia *pridanieKomentaru*

Prípád 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	<ol style="list-style-type: none"> 1. Prípád 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.

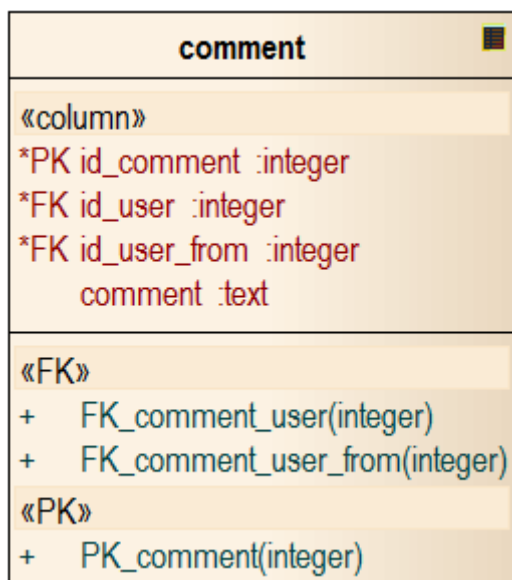
Výstupné podmienky	Prihlásený používateľ.
Alternatívne scenáre	žiadne

6.2.2. Návrh

Ako je znázornené na obrázku 6.2.2, pri zobrazení profilu je k dispozícii 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
<i>id_comment</i>	PK
<i>id_user</i>	ID používateľa, ktorému bol komentár priradený
<i>id_user_from</i>	ID používateľa, ktorý komentár priradil
<i>comment</i>	text komentáru

6.2.3. Implementácia

Samotné 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

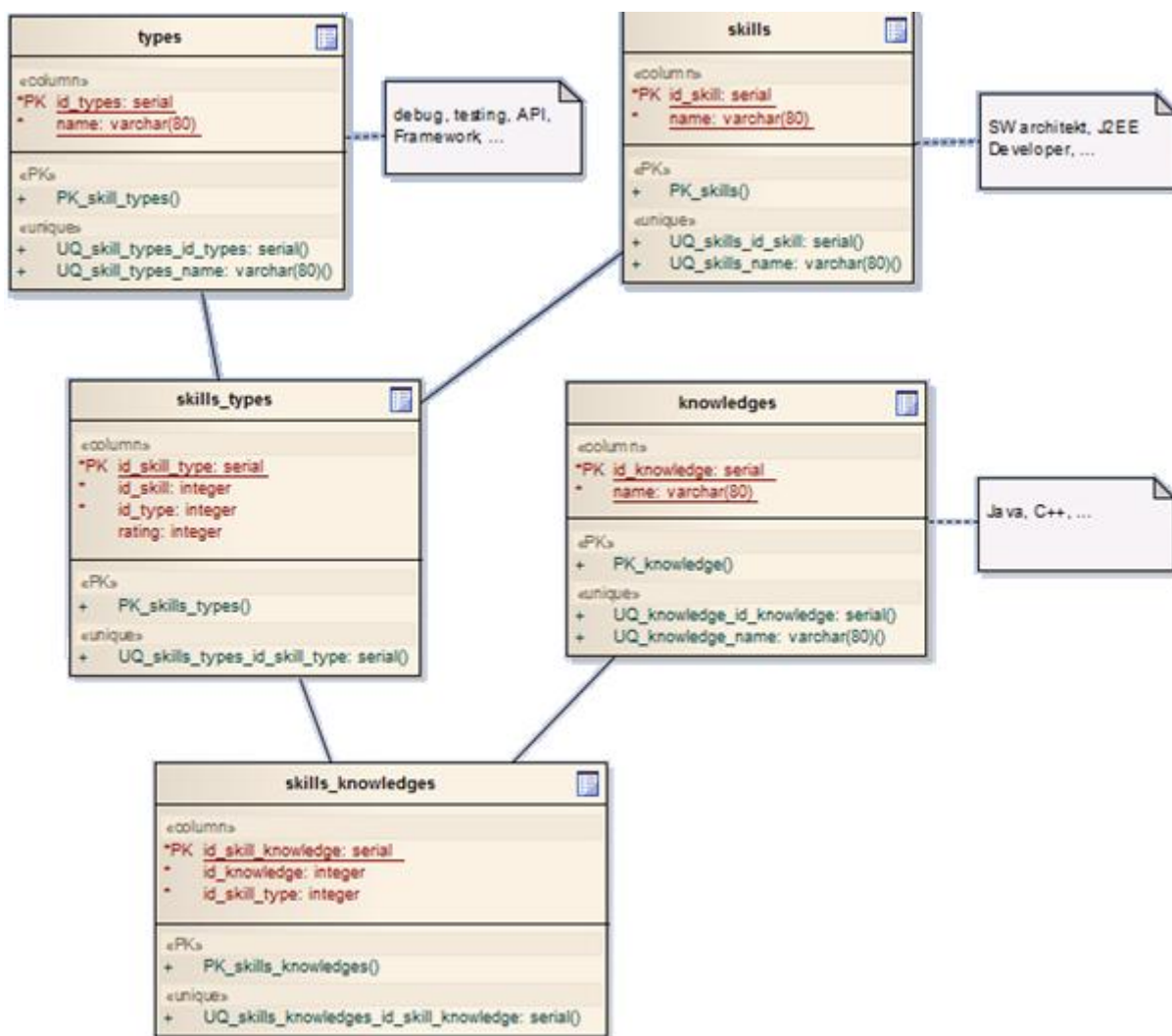
Dôležité je uvedomiť si, čo je *zručnosť* a v čom sa odlišuje oproti *znalosti*.

KSSJ opisuje *znalosť* (angl. knowledge) ako „ovládanie niečoho, vedomosti o niečom na základe štúdia, skúsenosti“. Tejto definícii 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 funkcionálnosť 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 funkcionálnosť 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.

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 prázdnu databázu 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 novú 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 nainštalovaný 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, nainštalujeme 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 prekopiropovat 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 reťazec „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 siahodlých 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 získaním 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.

Prípád 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	<ol style="list-style-type: none"> 1. Prípád 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.

PRIDAJ ZNALOSŤ
+

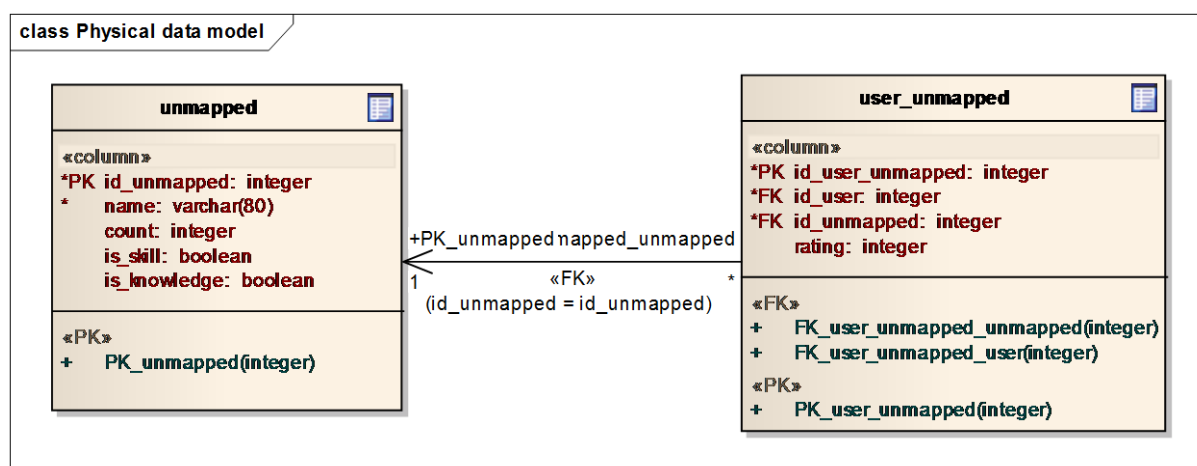
NÁPOVEDA
?

Znalosť	Úroveň	Akcia
Java	zočiatočník	
Unknown		
Modelovanie z hlíny	pokročilý	
Lyžovanie na mape	pokročilý	

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_unmapped*.



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

Popis atribútov entity *unmapped*:

Atribút	Popis
<i>name</i>	názov znalosti/zručnosti
<i>count</i>	počet výskytov
<i>is_skill</i>	indikátor, či položka je zručnosť
<i>is_knowledge</i>	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($itemArray as $index => $item) {
    if((int) $item == 0) {
        $unmp = Unmapped::model()->find('(translate("name",
        \'Iščťžýáíéúäö\', \'lscťzyaieuo\') ilike translate(:name,
        \'Iščťžýáíéúäö\', \'lscťzyaieuo\'))', 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	Očakávaná reakcia	Skutočná akcia	
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:

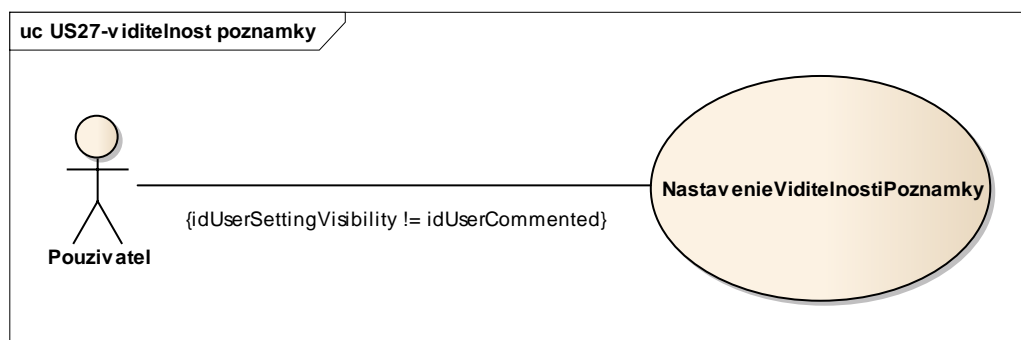
- US27 – Viditeľnosť poznámky
- US28 – Zobrazenie znalostí
- US29 – Zobrazenie poznámok
- US30 – Zobrazenie komentárov

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 negatívnom hodnotení študenta a poznámku vtedy nemusia vidieť všetci používatelia ale napríklad iba učitelia.



Obr. č. 8.1.1 – Prípád použitia Viditeľnosť poznámky

Prípád použitia	ViditeľnosťPoznámky
ID	8.1
Stručný popis	Používateľ nastavuje poznámke viditeľnosť.
Primárny aktéri	Prihlásený používateľ

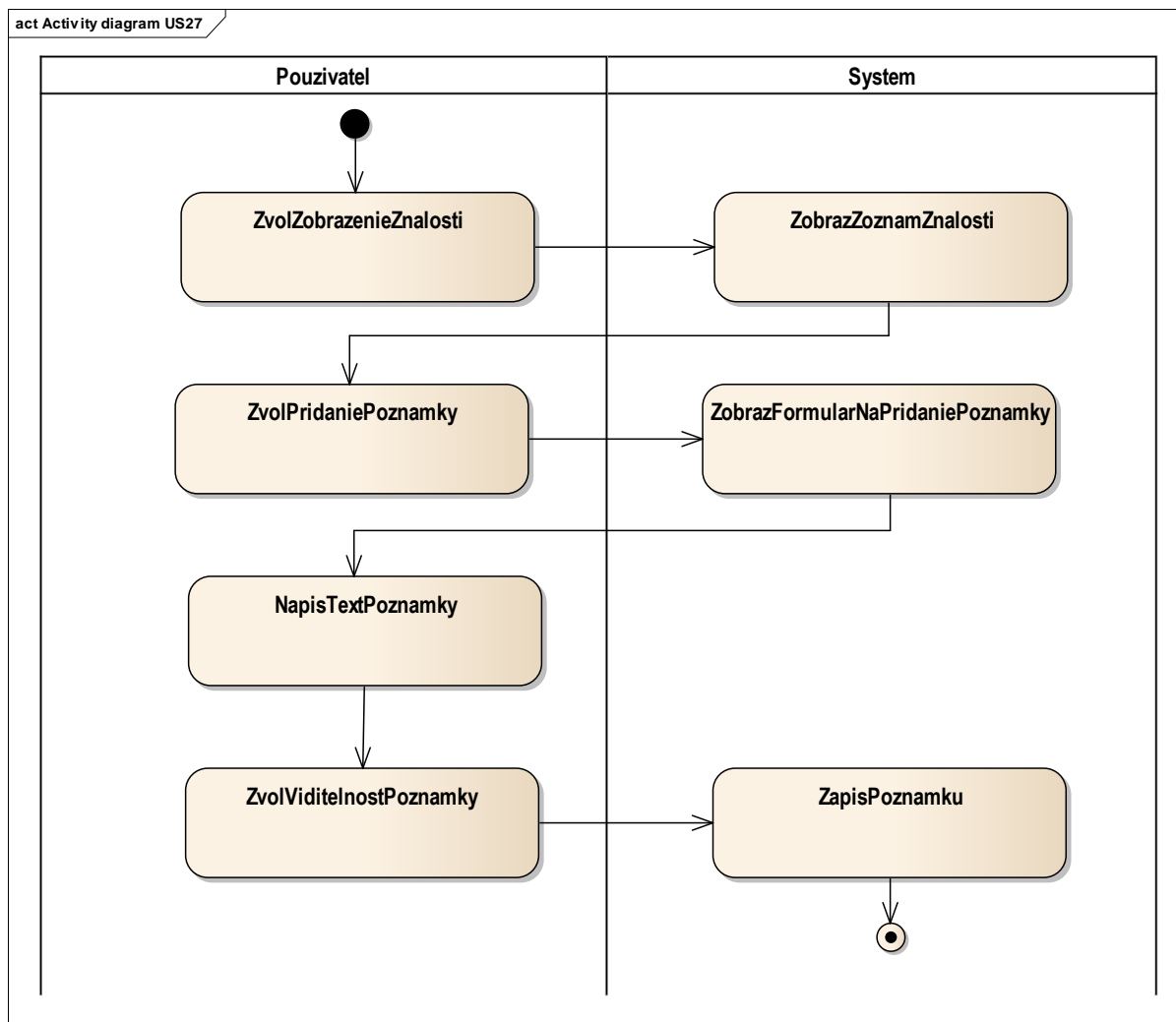
Vedľajší aktéri	Žiadny
Vstupné podmienky	Používateľ je prihlásený
Hlavný scenár	<ol style="list-style-type: none"> 1. Prípád použitia začína po zobrazení obrazovky znalostí iného používateľa. 2. Používateľ si klikne na ikonku poznámky a následne sa mu zobrazí textové pole, kam môže písať poznámku k znalosti 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 všetky 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 pridalí 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', 'Všetci', array('1' =>
'Všetci', '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.

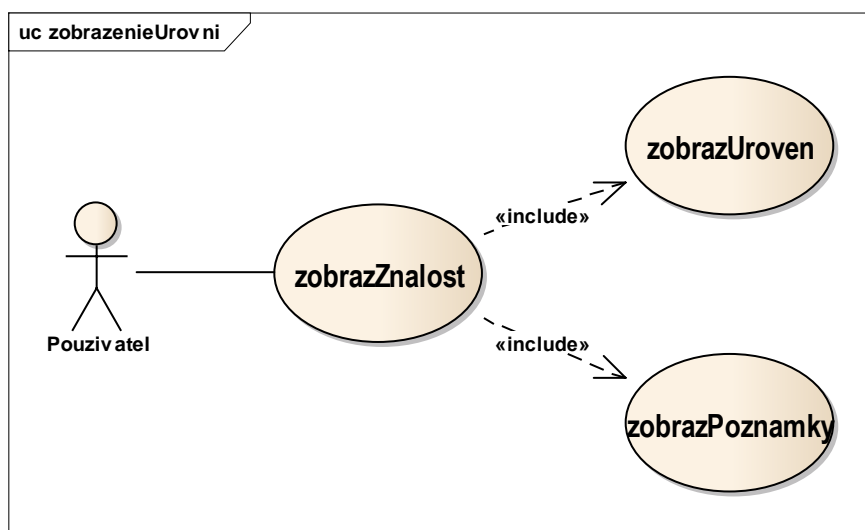
8.2. US28 – Zobrazenie znalostí

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 databázovej 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	<ol style="list-style-type: none"> 1. Používateľ si zvolí zobrazit' 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.

The screenshot shows the 'PROJECT MINERVA' web application. At the top, there are navigation links: ZNALOSTI (highlighted in red), PRIATELIA, PROFIL, and ODHLÁSENIE. On the left sidebar, there are two menu items: PRIDANIE ZNALOSTÍ and ZOBRAZENIE ZNALOSTÍ (with a right-pointing arrow). The main content area is divided into two sections:

- Zobrazenie znalostí**: A table with columns 'Znalosť', 'Úroveň', and 'Akcia'. It lists 'API' with 'ActiveX' and 'Pokročilý' levels, and an action icon.
- Zobrazenie zručností**: A table with columns 'Znalosť', 'Úroveň', and 'Akcia'. It lists 'field' with 'Java' and 'Pokročilý' levels, and an action icon.

At the bottom right of the screenshot, there is a small logo and the text 'weknowit(c)2011/2012'.

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 vykonaní 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.

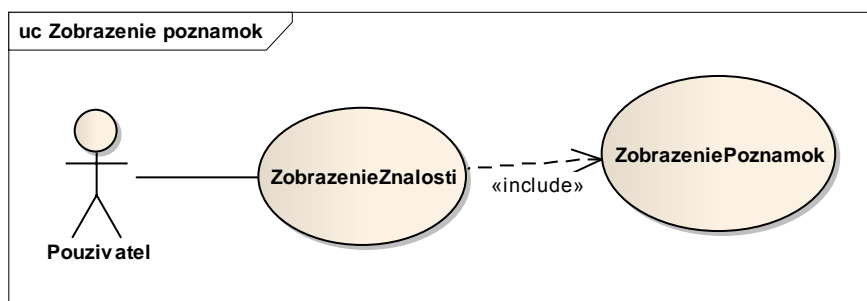
Názov	Zobrazenie znalostí používateľa spolu s úrovňami	ID Testu	8.2-01
Rozhranie	Zobrazenie znalostí	ID UC	8.2
Účel	Zistenie, či sa zobrazia uložené úrovne znalostí		
Vstupné podmienky	Priradené znalosti a úrovne k hľadanému používateľovi		
Výstupné podmienky	Žiadne		
Krok	Akcia	Očakávaná reakcia	Skutočná 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	<ol style="list-style-type: none"> 1. Prípád 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“.

Znalosť	Úroveň	Akcia
framework		
ASP.net	Začiatočník	
<ul style="list-style-type: none"> 10.04.2012 (20:17:55) Pridal: ja: 10.04.2012 (21:52:43) Pridal: ja: ahoj 10.04.2012 (22:42:18) Pridal: ja: 11.04.2012 (00:01:22) Pridal: ja: 11.04.2012 (00:47:35) Pridal: ja: 		
language		
Java	Začiatočník	

Obr. č. 8.3.2 – Zobrazenie poznámok pri znalostiach

8.3.3. Implementácia

Pri implementácii sme využili všetky 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.
----	--	----------------------------	--------------------------

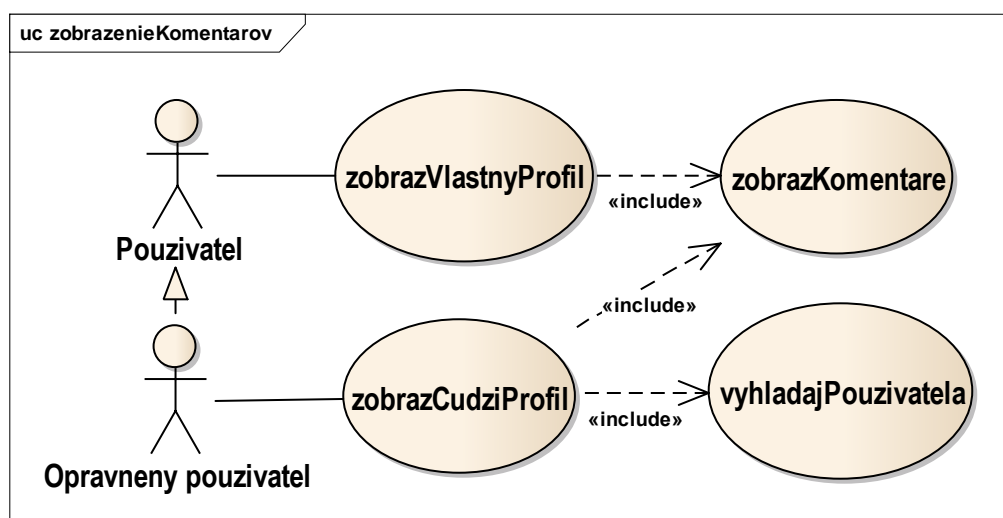
8.4. US30 – Zobrazenie komentárov

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.

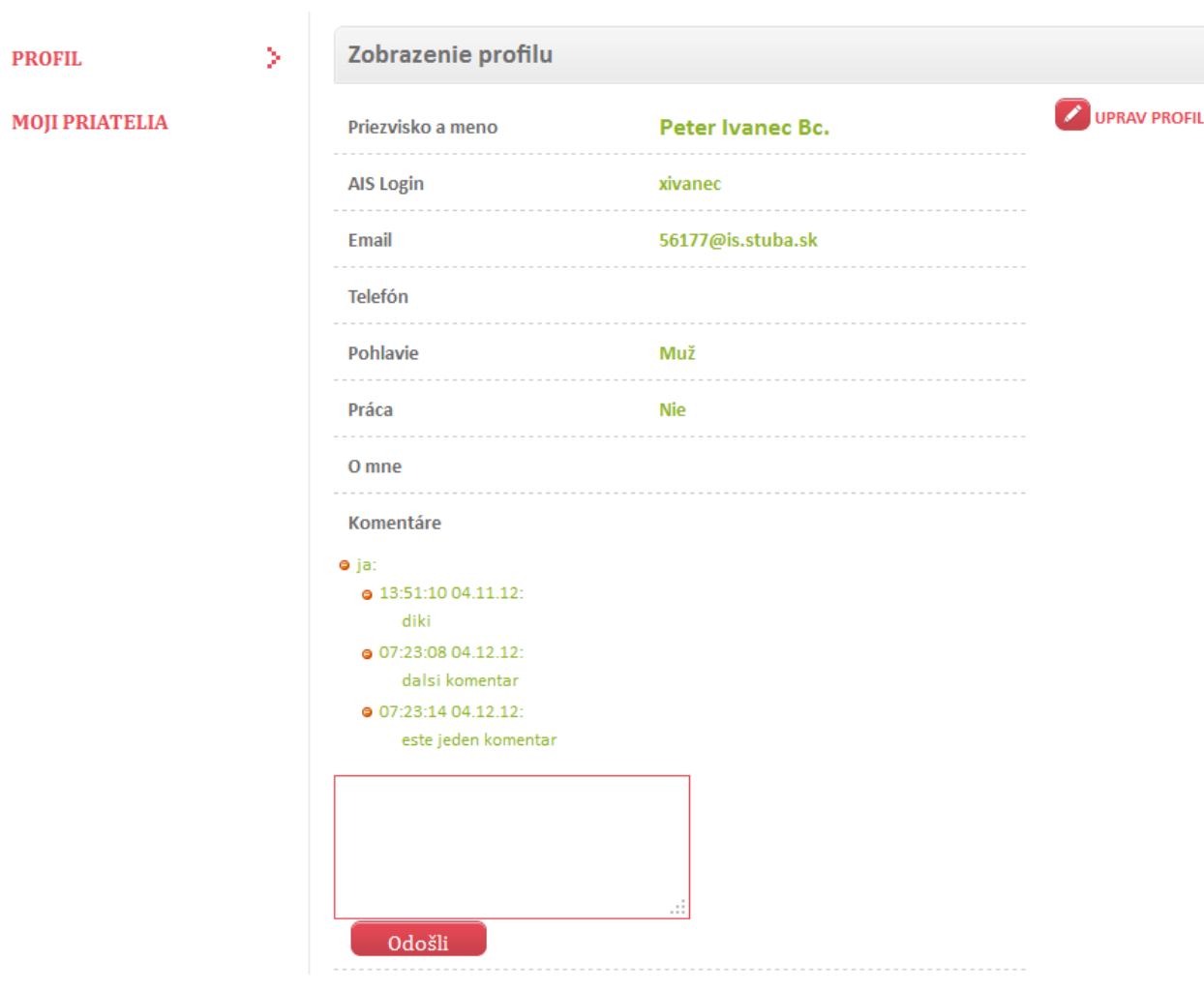


Obr. č. 8.4.1 – Diagram prípadov použitia *zobrazenieKomentarov*

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 zobrazit'.

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



PROFIL >

MOJI PRIATELIA

Zobrazenie profilu

Príezvisko a meno **Peter Ivanec Bc.** [UPRAV PROFIL](#)

AIS Login **xivanec**

Email **56177@is.stuba.sk**

Telefón

Pohlavie **Muž**

Práca **Nie**

O mne

Komentáre

ja:

- 13:51:10 04.11.12: diki
- 07:23:08 04.12.12: dalsi komentar
- 07:23:14 04.12.12: este jeden komentar

Odošli

Obr. č. 8.4.2 – Obrazovka profilu so zobrazenými komentármi

8.4.3. Implementácia

Táto funkcionálnosť 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 zoradíme. 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
),
));
?>
```

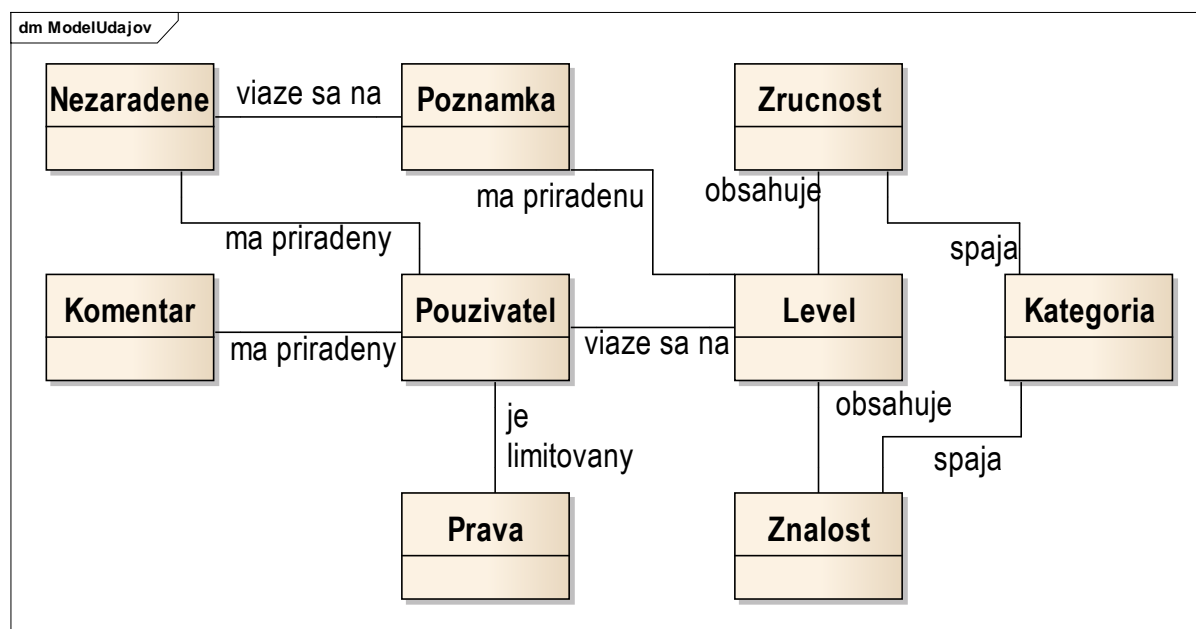
8.4.4. Testovanie

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. Prototyp systému

Súčasná verzia prototypu systému Minerva obsahuje všetku spomenutú funkcionálnosť 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.

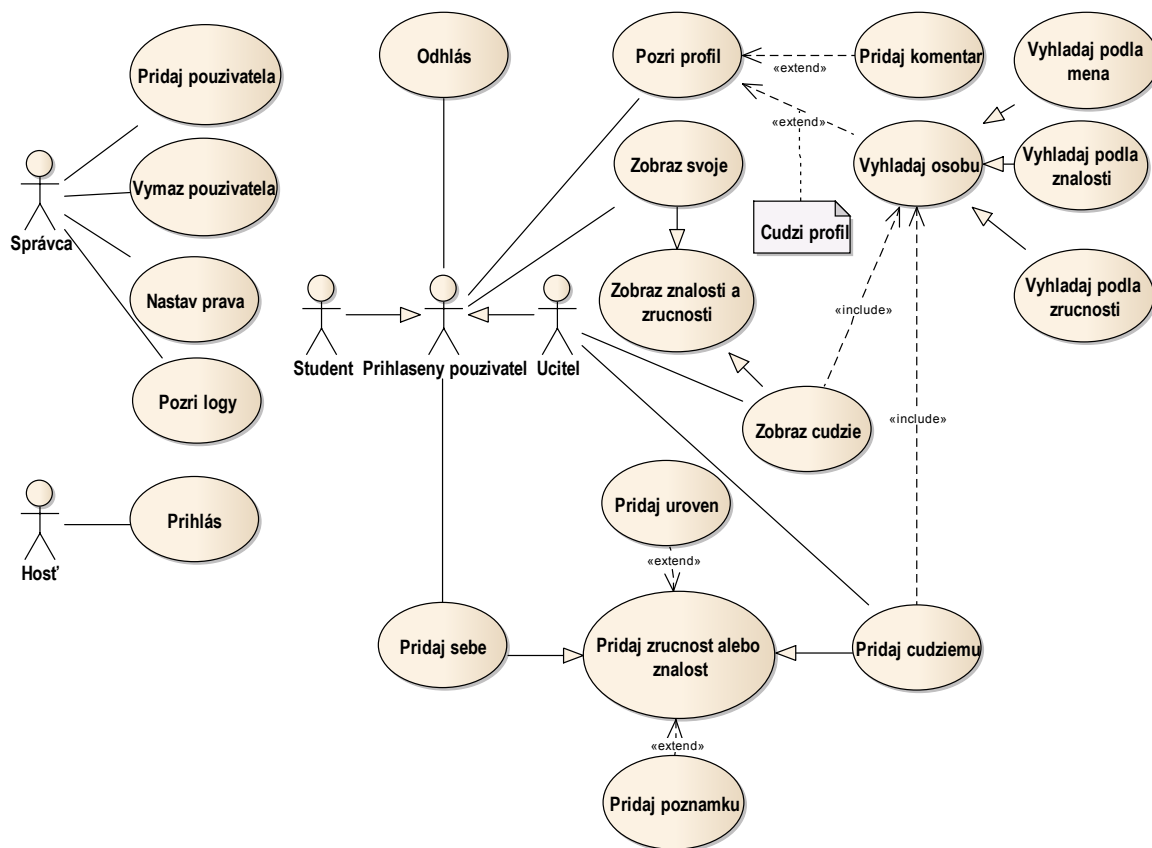


Obr. č. 9.1.1 – Model údajov systému

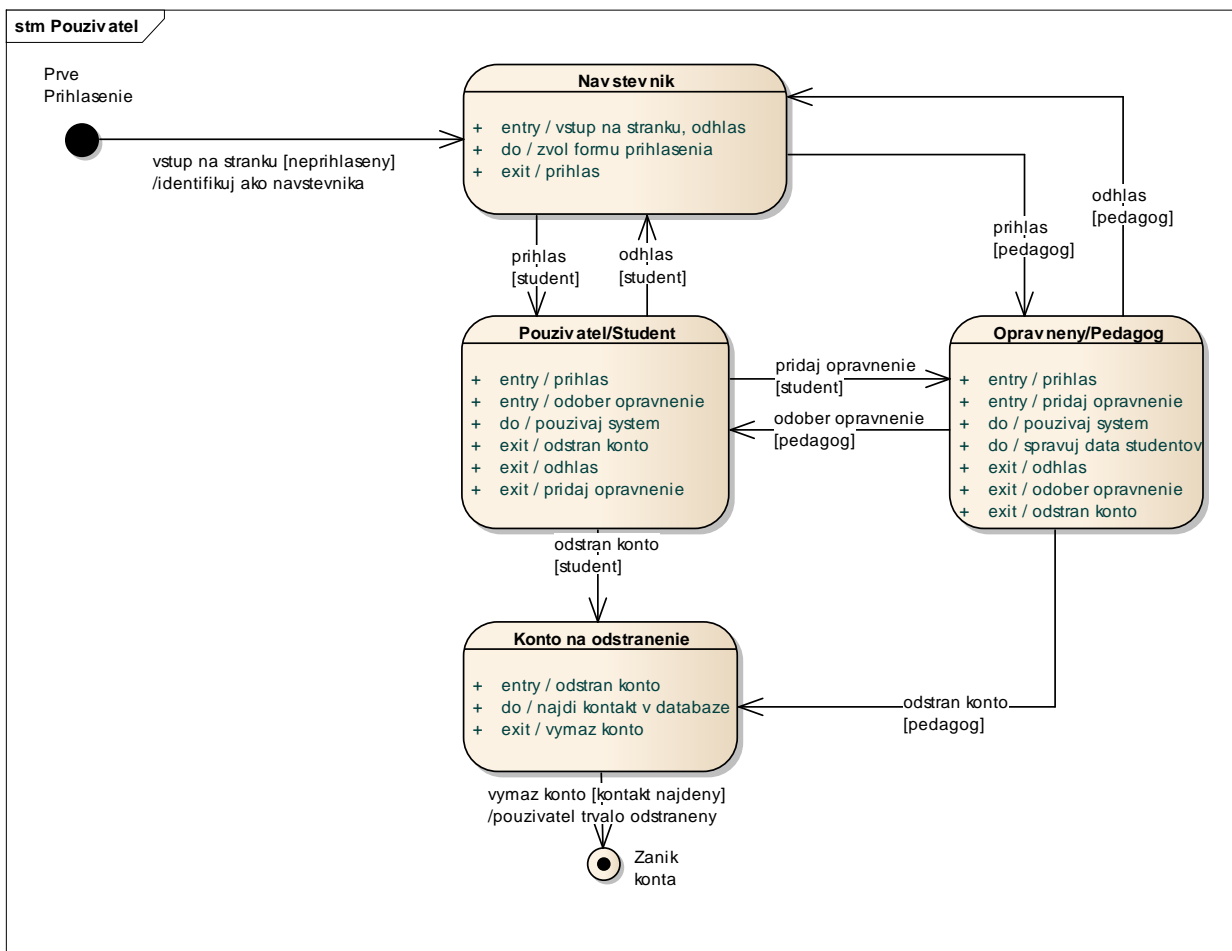
Funkcionálnosť ponúkanú prototypom znázorňuje globálny diagram prípadov použitia na obrázku číslo 9.1.2. Na základe používateľských oprávnení identifikujeme celkovo štyri roly – *Študent*, *Učiteľ*, *Správca*, *Host*. V diagrame je jasne identifikované, ktoré možnosti systému sú pre koho určené, ktoré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

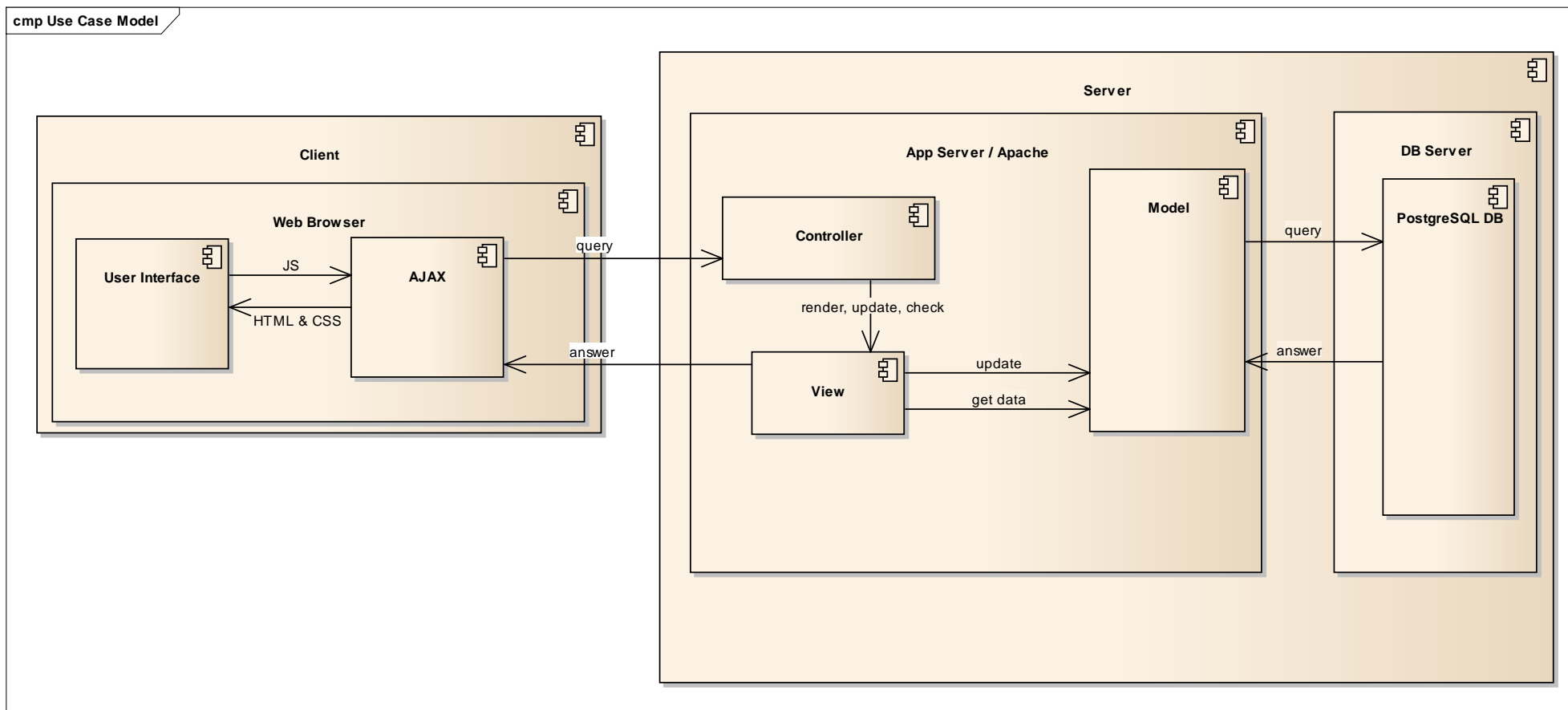
Čo sa týka nasadenia a hardvérovej podpory systému, architektúra systému je znázornená na obrázku číslo 9.1.4.



Obr. č. 9.1.2 – Celkový diagram prípadov použitia



Obr. č. 9.1.3 – Stavový diagram *Použivateľ*



Obr. č. 9.1.4 – Architektúra systému

10. Používateľská príručka

10.1. Prihlásenie

Pri prvej návšteve systému nie je možné používať jeho funkcionality, 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

10.2. 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

10.3. Zobrazenie svojho profilu

Svoj vlastný profil je možné zobraziť z ľubovoľnej obrazovky, pretože sa tiež nachádza v hornom menu, ktoré nás správa 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

10.4. 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

10.5. Vyhľadanie používateľa

Používateľov v systéme môžeme hľadať tromi 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.



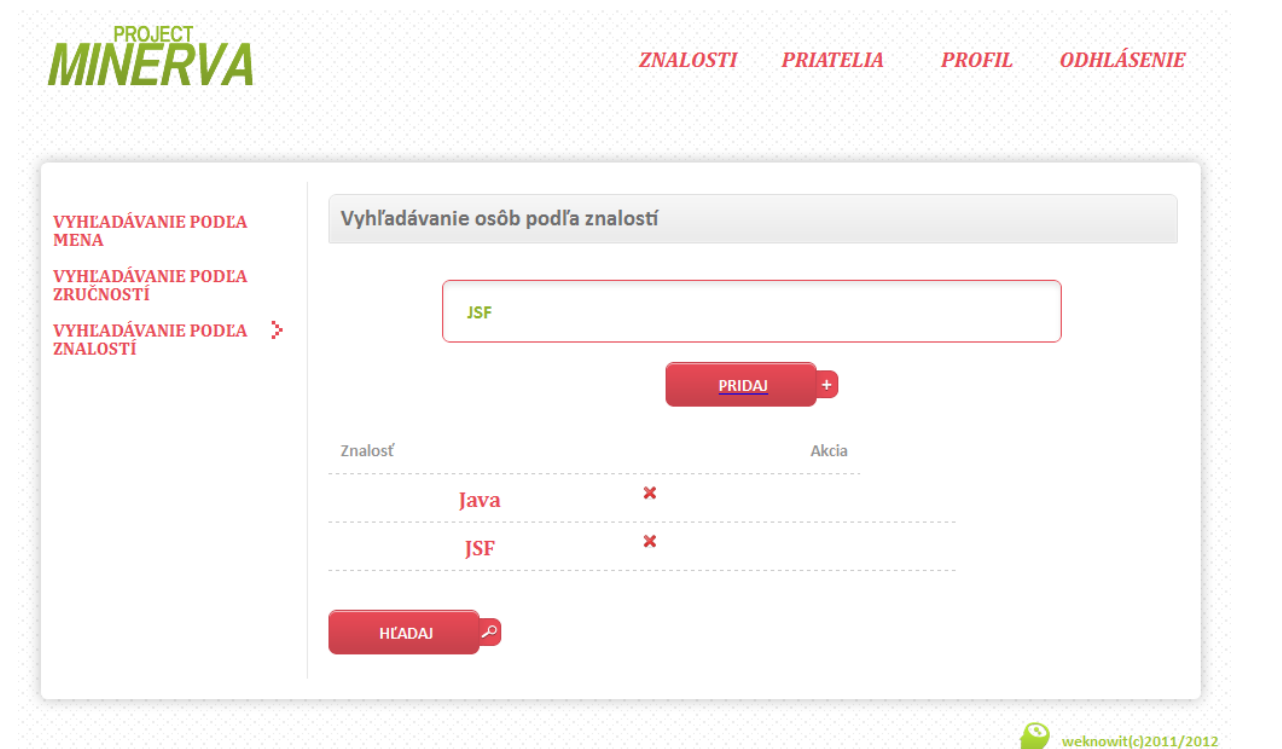
Obr. č. 10.5.1 – Vyhľadávanie podľa mena

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 *HLADAJ* spustíme vyhľadávanie.



Obr. č. 10.5.2 – Vyhľadávanie podľa znalostí

10.6. 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.