

# **Využitie sociálnych sietí pri vytváraní pracovných tímov**

**Tímový projekt**

**Dokumentácia k inžinierskemu dielu**

Školský rok: 2008/2009

Vedúci tímového projektu : Ing. Michal Barla

Tím č. 6 – Awesome Legends

Bc. Stanislav Jurský

Bc. Martin Valašík

Bc. Lukáš Repka

Bc. Martin Sirota

Bc. František Chvostaľ

Bc. Dušan Zahoranský

# Obsah

Obsah.....	ii
Slovník pojmov .....	iii
1. Zadanie .....	1
2. Úvod .....	2
3. Šprinty .....	3
3.1. Šprint č. 1.....	4
3.1.1. Prihlasovanie do systému cez LDAP .....	4
3.1.2. Vytváranie sociálnej siete študentom .....	5
3.1.3. Prechádzanie profilov študentov .....	6
3.2. Šprint č. 2.....	9
3.2.1. Získavanie informácií o študentovi od študenta .....	10
3.2.2. Vizualizácia prepojení.....	11
3.2.3. Filtrovanie zoznamu študentov .....	13
3.2.4. Vyžiadanie potvrdenia vzťahu druhou stranou .....	14
3.2.5. Rozdelenie aplikácie na zóny s rozdielnou úrovňou ochrany prístupu..	15
3.2.6. Definovanie navigácie v aplikácii a úvodná obrazovka .....	16

# Slovník pojmov

- LDAP** Lightweight Directory Access Protocol, je aplikačný protokol slúžiaci na dotazovanie a modifikovanie adresárových služieb. V tomto projekte slúži na účely overovania prihlasovacích údajov.
- AJAX** Asynchronous JavaScript and XML. Ide o skupinu bežne používaných techník pri programovaní webových stránok na strane klienta. Cieľom je vytváranie stránok schopných dynamicky meniť svoj obsah bez nutnosti znovunačítania.

# 1. Zadanie

Výsledok snaženia študenta v predmete Tímový projekt je logicky veľmi závislý od toho, ako sa na začiatku povytvárajú jednotlivé tímy. Ak by bol tento proces úplne neriadeneý, mohlo by ľahko dôjsť k vzniku extrémnych situácií: vzniklo by niekoľko tímov zložených zo samých vynikajúcich študentov ale rovnako aj niekoľko tímov v ktorých by boli len študenti s podpriemernými výsledkami. Zatiaľ čo v prípade tímu vynikajúcich študentov by zrejme problém počas riešenia projektu nenastal (čo je z hľadiska cieľov predmetu Tímový projekt trochu problém :), tím podpriemerných študentov by bez niekoho, kto tím v správnej chvíli potiahne, pravdepodobne neuspel. Ideálne je, keď sú tímy čo najviac vyvážené, aby si každý tím vyskúšal riešenie rôznych problémov, ktoré vyplývajú z rôznorodosti jednotlivých členov a kde sa slabší môžu niečo naučiť od tých lepších. Taktiež je potrebné myslieť na vyváženosť zručností jednotlivých členov tímu (tím zložený zo samých GUI dizajnérov bez databázistu sa bude trápiť).

Pri navrhovaní zloženia jednotlivých tímov treba zohľadniť:

- predchádzajúce skúsenosti, zručnosti budúcich členov tímu
- predchádzajúce spolupráce budúcich členov tímu
- preferencie, čo by kto chcel robiť v tíme
- preferencie, kto by s kým chcel/nechcel byť v tíme a z akých dôvodov
- povahové vlastnosti budúcich členov tímu
- a rôzne ďalšie atribúty...

Úlohou tímu je navrhnuť a vytvoriť systém, ktorý na základe priamych a nepriamych vstupov (formuláre, iné fakultné systémy a pod.) vytvorí profily jednotlivých študentov a prepojí ich do jednej sociálnej siete na základe rôznych zadaných aj odvodených vzťahov. Vhodnou vizualizáciou takejto sociálnej siete a poskytnutím efektívnych nástrojov na jej ďalšiu analýzu (napr. známe algoritmy na sociálnu analýzu sietí, ktoré určujú populárne a inak významné body siete) systém podporí proces vytvárania tímov študentov v predmete Tímový projekt na našej fakulte (čo však určite nie je jediná aplikácia, na ktorú sa takáto sociálna sieť dá efektívne využiť ;)).

## 2. Úvod

Tento dokument je výsledkom práce tímu č. 6 počas zimného semestra v predmete Tvorba softvérového systému v tíme. Cieľom bolo vytvoriť dynamickú sociálnu sieť tvorenú študentmi fakulty a slúžiacu pedagógom na uľahčenie práce sporej s tvorbou tímov. Konkrétne išlo o zbieranie údajov pomocou formulárov a z rôznych informačných systémov školy, ich následné klasifikovanie, utriedenie, spracovanie a následné vyhodnocovanie.

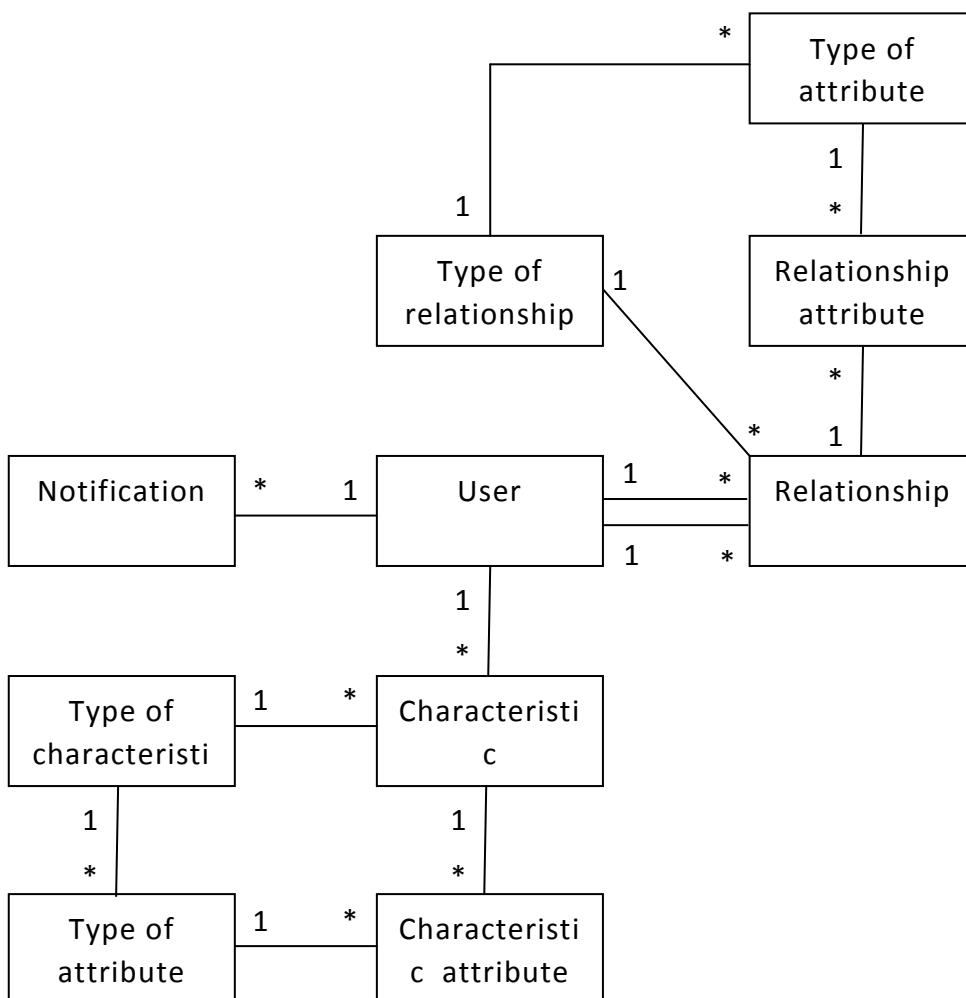
Na odporúčanie tímového vedúceho a aj na základe vlastných skúseností sme sa priklonili k agilnému spôsobu vývoja a to konkrétne k technike Scrum. Ide o iteratívny, inkrementálny proces, ktorý sa v súčasnej dobe hojne využíva pri tvorbe softvérových systémov, ale aj v iných oblastiach. Jeho hlavnou črtou sú tzv. šprinty, ktoré predstavujú jednu iteráciu procesu. Ďalším dôležitým rysom sú tímové stretnutia, z ktorých sa neskôr robia zápisnice. V tomto ohľade sa teda Scrum zhodoval s požiadavkami v tímovom projekte. Samotný proces sme si mierne upravili, šprinty sme skrátili z jedného mesiaca na 2 týždne a interval stretnutí naopak predĺžili, z jedného dňa na jeden týždeň.

Požiadavky zákazníka sú v technike Scrum riešené pomocou tzv. *user stories*, ktoré sa vždy viažu na šprint. Z nich sú následne vytvorené konkrétne úlohy pre členov tímu.

Nakoľko náš proces vývoja sa nezhodoval z bežným procesom vývoja počas tímového projektu, rozhodli sme sa modifikovať aj štruktúru dokumentácie. Namiesto obvyklých veľkých kapitol zodpovedajúcich vodopádovému vývoju budeme projekt opisovať podľa šprintov a samotných úloh v danom šprinte. Každá takáto úloha má svoju vlastnú malú analýzu, návrh a implementáciu a šprint má svoje ciele.

# 3. Šprinty

Ako už bolo spomenuté vyššie, každý šprint predstavuje jednu iteráciu v agilnom vývoji. Počas neho by mali v určitej miere prebehnúť všetky procesy bežné vo vodopádovom vývoji a výsledkom by mala byť do určitej miery funkčná aplikácia. Pred začiatkom každého šprintu sme si definovali už vyššie spomenuté *user stories*, teda požiadavky na systém z hľadiska klienta. Z nich sa vytvorili úlohy, z ktorých každá spadá práve do jednej *user story*. Ešte pred začatím prvého šprintu sme spoločne vytvorili logický dátový model aplikácie, ktorý je možno vidieť na obr. 1.



Obr. 1 – Logický dátový model aplikácie

## 3.1. Šprint č. 1

V prvom šprinte, ktorý trval od 15.10.2008 do 5.11.2008, sme sa sústredili najmä na dobré odrazenie sa a začatie projektu. Kvôli úvodnej chybe pri zadávaní šprintu trval 3 týždne namiesto dvoch. Išlo tu hlavne o vymyslenie a čiastočné implementovanie dátového modelu aplikácie a vytvorenie prvej, veľmi ranej verzie aplikácie. Počas tohto šprintu sme celkovo uzatvorili 3 user stories.

### 3.1.1. Prihlasovanie do systému cez LDAP

Používateľ sa prihlási s prihlasovacími údajmi z AIS aby sa dostal k funkcionalite systému. Na základe údajov LDAP servera a AIS databázy sa mu priradí rola v systéme.

Pre túto *user story* bola vytvorená len jedna úloha.

#### Nasadenie prihlasovania cez LDAP

Cieľom úlohy bola integrácia prihlasovania pomocou metódy LDAP. Študent, ktorý už má konto v školskom informačnom systéme si teda nemusí vytvárať nový účet pre našu aplikáciu, ale môže použiť prihlasovacie údaje z AIS.

#### Analýza problému

Pre správnu prácu väčšiny aplikácii je potrebné vedieť, kto danú aplikáciu používa. Ako riešenie sa bežne používa prihlasovanie, pričom aplikácie môžu používateľa overovať prostredníctvom svojej lokálnej databázy, alebo cez server tretej strany. V našom prípade použijeme kombinovaný spôsob. Používatelia, ktorí majú heslo uložené v lokálnej databáze sa prihlasujú cez ňu a v prípade, že používateľ nie je v lokálnej databáze, alebo tam nemá heslo, overujú sa údaje cez LDAP sever ldap.stuba.sk .

#### Návrh

Na prihlasovanie je potrebné v databáze uchovávať pre jednotlivých používateľov ich prihlasovanie meno a heslo. Heslo je uložené ako SH1 hash hesla kvôli bezpečnosti. V prípade, že používateľ nemá uložené heslo v lokálnej databáze, overia sa jeho prihlasovacie údaje cez ldap server ldap.stuba.sk . V prípade, keď sa používateľ, overený cez ldap server, nenachádza v lokálnej databáze vytvorí sa pre neho položka v databáze a z ldap serveru sa zistia informácie o používatelovi, konkrétne meno, priezvisko a rola v akej vystupuje.

#### Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Models: *user.rb*
- V zložke Controllers: *login\_controller.rb*

Na prihlásenie používateľa slúži metóda *login* v *login\_controller.rb* ktorá zavolá metódu *authenticate* z *user.rb*, ktorá vykoná logiku prihlasovania.

## Testovanie

V zložke *Functional Tests* v súbore *login\_controller\_test.rb* je vytvorený testovací algoritmus pre overenie funkčnosti *login\_controller.rb*. V zložke *Unit Tests* v súbore *user\_test.rb* je vytvorený testovací algoritmus pre overenie funkčnosti *user.rb*

## 3.1.2. Vytváranie sociálnej siete študentom

Úloha požadovala vytvorenie formuláru, kde študent vidí ostatných študentov aktuálneho ročníka a môže vyjadrovať svoj vzťah k nim. Pri pridaní vzťahu sa podľa typu vzťahu môže, alebo nemusí čakať na potvrdenie vzťahu druhou stranou. Pri inverzných vzťahoch sa automaticky doplní aj opačný smer vzťahu.

Iba jedna úloha bola priradená k tejto *user story*.

## Formulár a model na vytvorenie a upravovanie vzťahu zo zoznamu používateľov

Táto úloha vyžadovala vytvorenie kompletnej aplikačnej logiky nad databázou z hľadiska vzťahov medzi používateľmi.

### Analýza problému

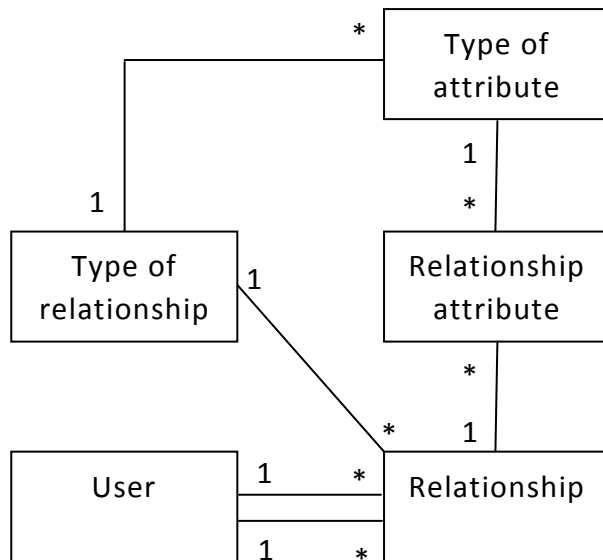
Niekedy sa dá vytvoriť vzťah medzi používateľmi na základe ich charakteristík. Napríklad vzťah *spolužiaci* možno vytvoriť podľa navštevovaného ročníka. Niekedy je však potrebné, aby bol vzťah k inému používateľovi definovaný explicitne používateľom. Niektoré vzťahy je možné definovať ako inverzné (*priatelia*, *kamaráti*) či dokonca tranzitívne (*spolužiaci*).

### Návrh

Na obr. 2 je vidieť časť databázového modelu použitú na reláciu vzťahu medzi používateľmi. Kvôli existencii inverzných resp. obojstranných vzťahov sa v modeli nachádza dvojité hrana medzi používateľom a vzťahom. Relácia vzťahu je totiž navrhovaná ako orientovaná hrana od koho – ku komu.

Formulár vzťahov daného používateľa pribudne do formuláru pre profil jedného študenta. Bude obsahovať informáciu od koho a ku komu vzťah smeruje i typ vzťahu.





Obr. 2 – Model vzťahov

## Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Models: *relationship.rb*, *user.rb*, *type\_of\_relationship.rb*
- V zložke Controllers: *relationships\_controller.rb*, *users\_controller.rb*, *type\_of\_relationships\_controller*
- V zložke Views: *layouts/users.html.erb*, *users/\_show\_relationship\_detail.html.erb*, *users/\_show\_relationships.html.erb*

O zabezpečenie zobrazenia jedného vzťahu sa stará *\_show\_relationship\_detail.html.erb*, ktorý je renderovaný z *\_show\_relationships.html.erb*. Ten je volaný zo zobrazenia profilu používateľa *users.html.erb*.

## Testovanie

Testovanie prebieha nad databázou *2008team06isp2*, kde sa nachádzajú testovacie dáta pre vzťahy. V zložke *Functional Tests* v súbore *relationships\_controller\_test.rb* je vytvorený testovací algoritmus na overenie funkčnosti implementácie vzťahov nachádzajúcej sa v zložke Controllers.

### 3.1.3. Prechádzanie profilov študentov

Pedagóg vidí zoznam študentov zvoleného ročníka spolu s informáciou o vyplnenom profile a možnosti prezerania daného profilu.

Pre túto *user story* boli vytvorené dve úlohy.

# Vytvorenie databázového modelu pre profil

V tejto úlohe išlo o vytvorenie modelu profilu, ktorý sa neskôr použije pri práci s formulárom.

## Analýza problému

Sociálna sieť bude uchovávať množstvo údajov o jednotlivých používateľoch. Či už sa jedná o dáta získané od samotného používateľa, alebo ide o dáta získané z externých informačných zdrojov, tieto dáta musia byť uložené spôsobom, ktorý bude umožňovať ich uchovávanie bez ohľadu na typ informácie a bude taktiež obsahovať základné metadáta a danej informácii.

## Návrh

Model pre profil používateľa, ktorý je na obr. 3, obsahuje 5 entít:

Characteristics – Prepojovacia tabuľka. Bude pridelovať používateľovi atribúty jednej charakteristiky určitého typu.

Type of characteristic – Obsahuje základné údaje o type charakteristiky

Type of attribute – Obsahuje základné údaje o type atribútu. Každý typ charakteristiky môže mať jeden a viac typov atribútov.

Characteristic attribute – Atribút charakteristiky určitého typu.

User – samotný používateľ

Entitám modelu profilu používateľa prislúchajú nasledujúce Views:

1. Zobrazenie všetkých typov charakteristík
2. Vytvorenie nového typu charakteristiky
3. Zobrazenie jedného typu charakteristiky. Obsahuje taktiež formulár na vytvorenie typu atribútu a výpis typov atribútov prislúchajúcich zobrazenému typu charakteristiky.
4. Editovanie typu charakteristiky
5. Editovanie typu atribútu.
6. Vytvorenie charakteristiky určitého typu
7. Zobrazenie charakteristiky spolu s formulárom na vytvorenie atribútu a s výpisom všetkých atribútov danej charakteristiky.
8. Editovanie atribútu charakteristiky

## Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Models: *type\_of\_char.rb*, *type\_of\_attribute.rb*, *characteristic.rb*, *char\_attribute.rb*
- V zložke Views/char\_attributes: *edit.html.erb*
- V zložke Views/characteristics: *new.html.erb*, *show.html.erb*
- V zložke Views/type\_of\_attributes: *edit.html.erb*

- V zložke Views/chars: *edit.html.erb*, *index.html.erb*, *new.html.erb*, *show.html.erb*
- V zložke Controllers: *char\_attributes\_controller.rb*, *characteristics\_controller.rb*, *type\_of\_attributes\_controller.rb*, *type\_of\_chars\_controller.rb*

## Zobrazenie zoznamu študentov a ich profilov

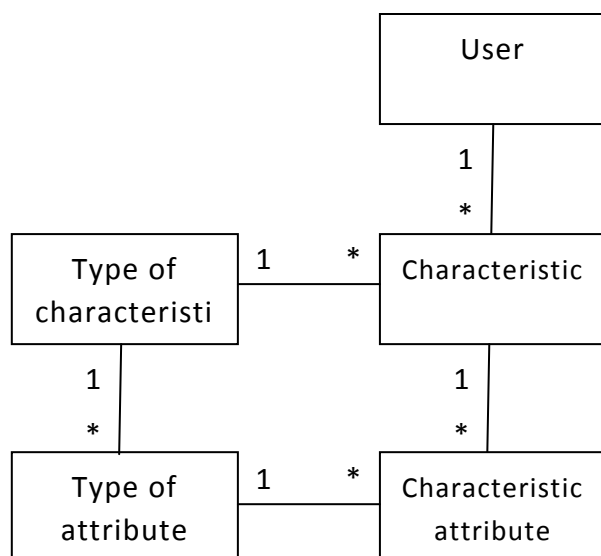
Táto úloha vyžadovala zobrazenie študentov pomocou modelu vytvoreného v predchádzajúcej úlohe. V zobrazenom zozname mala aplikácia automaticky identifikovať študentov, ktorí majú vytvorení profil a umožniť zobraziť tieto hodnoty.

### Analýza problému

Pri práci z veľkým množstvom dát je pre používateľa prioritou ich prehľadné usporiadanie a zobrazenie kľúčových údajov. Sociálna sieť vo vyvíjanom systéme má potenciál uchovávanie informácií o množstve študentov a podrobných charakteristík každého z nich. Z tohto dôvodu je potrebné používateľovi poskytnúť možnosti ako si tieto dáta prezerať, zobraziť podrobný profil každého z nich a umožniť mu tieto dáta filtrovať podľa potreby.

### Návrh

Zobrazenie profilov využíva časť databázového modelu pre uchovávanie charakteristík jednotlivých študentov. Úlohou je zobrazenie dát, nie ich rozmiestnenie a formátovanie na formulári, tieto aspekty sú riešené v neskorších úlohách projektu. Obr. 2 zachytáva časť databázového modelu, ktorý je k tejto úlohe relevantný.



Obr. 3 – Model profilu používateľa

Zobrazenie profilov sa skladá z dvoch formulárov:

1. Formulár, ktorý zobrazí zoznam všetkých používateľov z tabuľky *User*. Tento formulár obsahuje aj komponenty na filtrovanie zobrazených používateľov, úloha však nepokrýva samotný algoritmus filtrovania, je to príprava formuláru pre použitie v neskorších fázach projektu. Časť formuláru pre filtráciu sa skladá z týchto komponentov:
  - Combobox na výber typu charakteristiky, dáta z tabuľky *Type of characteristic*
  - 3 krát combobox na výber typov atribútov, dáta z tabuľky *Type of attribute*
  - Tlačidlo pre spustenie filtrácie
2. Formulár profilu jedného študenta. Tento formulár zobrazuje jeden záznam z tabuľky *User* spolu so všetkými jeho charakteristikami, čo sú záznamy z tabuľky *Characteristic*, ktoré sa k nemu vzťahujú. Ku každej charakteristike sa zobrazia všetky vzťahujúce záznamy z *Characteristic attribute* spolu s ich typom, ktorý sa vyhľadá v tabuľke *Type of attribute*.

## Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Models: *user.rb*
- V zložke Views/users: *index.html.erb*, *show.html.erb*
- V zložke Controllers: *user\_controller.rb*

Zobrazenie zoznamu študentov zabezpečuje metóda *index* v *user\_controller.rb* ktorá prečíta všetkých používateľov z databázovej tabuľky *User* a zobrazí ich na formulár. Zobrazenie detailného profilu je implementované v metóde *show* v *user\_controller.rb*.

## Testovanie

K testovaniu funkcionality je vytvorená štruktúra tabuliek z obr. 2 naplnená testovacími dátami v databáze *2008team06isp2*. V zložke *Unit Tests* je vytvorený testovací algoritmus pre zobrazenie študentov a ich profilov v súboroch *user\_test* a *user\_controller\_test*.

## 3.2. Šprint č. 2

Druhý šprint sme začali 5.11.2008 a ukončili 12.11.2008. Kvôli kompenzácii prvého trojtýždňového šprintu sme tento skrátili na jeden týždeň. Cieľom bolo ďalšie vylepšovanie aplikácie a dopĺňanie funkčnosti. Celkovo sme definovali 6 rôznych *user stories*.

## 3.2.1. Získavanie informácií o študentovi od študenta

Vytvorí formulár, kde študent môže zadať, skontrolovať, alebo zmeniť svoj profil a ďalší formulár, pomocou ktorého študent absolvuje psycho-test.

Táto *user story* zatiaľ nebola realizovaná.

### Prepracovať formulár na zobrazenie charakteristiky

Úloha požadovala, aby pomocou modelu vytvoreného v úlohe „Vytvorenie databázového modelu pre profil“ bolo možné zobraziť zoznam atribútov a charakteristík priradených používateľovi. Ďalej zahrňovala vytvorenie rozhrania pre správu týchto informácií, ktoré bude umožňovať vytvorenie, zobrazenie, editáciu a vymazanie charakteristík a ich atribútov predstavujúcich profil študenta.

#### Analýza problému

Údaje o študentoch, ktoré budú tvoriť dôležitú súčasť sociálnej siete je potrebné do systému zadať. Niektoré údaje sa budú importovať automaticky z externých informačných systémov, iné je potrebné zadať priamo prostredníctvom formulára. Tieto údaje je potrebné zobraziť a v prípade potreby editovať alebo vymazať. Preto je nutné vytvoriť prehľadné rozhranie na správu týchto údajov.

#### Návrh

##### **Vytvorenie novej charakteristiky:**

Na vytvorenie novej charakteristiky je potrebné zadať iba typ charakteristiky. Po vytvorení, je možné začať s priradovaním atribútov k tejto charakteristike, pričom je možné pridať len atribúty, ktorých typ je priradený danému typu charakteristiky.

##### **Vytvorenie nového atribútu charakteristiky:**

Pridá do zoznamu atribútov charakteristiky novovytvorený atribút. Na vytvorenie atribútu je potrebné zadať jeho názov a typ.

##### **Zobrazenie atribútov zvolenej charakteristiky:**

Zobrazí typ charakteristiky a používateľa, ktorému bola daná charakteristika priradená. Nasleduje výpis zoznamu atribútov zvolenej charakteristiky, ktorý obsahuje typ atribútu, názov, čas kedy bol vytvorený, a odkazy na jeho editáciu, resp. odstránenie.

##### **Editácia atribútu:**

Umožní používateľovi zmeniť názov atribútu.

**Vymazanie charakteristiky:**

Zmaže charakteristiku používateľa obsahujúcu atribúty.

**Vymazanie atribútu:**

Odstráni zvolený atribút zo zoznamu atribútov danej charakteristiky používateľa.

## Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Views/characteristics: *new.html.erb*, *show.html.erb*
- V zložke Views/char\_attributes: *edit.html.erb*
- V zložke Controllers: *characteristics\_controller.rb*,  
*char\_attributes\_controller.rb*

## 3.2.2. Vizualizácia prepojení

Pedagóg má možnosť prezerať vizualizovanú sociálnu sieť študentov. Po kliknutí na uzol sa mu zobrazí profil študenta, pričom má možnosť aplikovať na danú sieť rôzne filtre, analyzátory, alebo zhlukovanie.

K tejto *user story* neboli pridelené úlohy, ale bola vypracovaná predbežná analýza. Jej implementácia sa presúva do ďalšieho šprintu.

## Analýza nástrojov na vizualizáciu grafu

V analýze sú priblížené vybrané nástroje na vizualizáciu grafov. Všetky tieto nástroje sú voľne dostupné.

### JUNG

Vykresľovanie grafu je možné manažovať a vykresľovať pomocou rôznych implementovaných rozložení. K dispozícii sú rozloženia pre hierarchické vizualizácie stromov a nehierarchické vizualizácie grafov so staticky vypočítanými pozíciami objektov a následným vykreslením alebo aj dynamicky počítanými pozíciami a priebežným vykresľovaním.

Pre hierarchické a nehierarchické vizualizácie je ale potrebné použiť inú reprezentáciu grafu. Tá sa vytvára pridávaním objektov (*Vertex*) a hrán (*Edge*). Vo verzii *JUNG 2.0* sa ako hrany a objekty pridávajú ľubovoľné objekty s tým, že sa každému objektu priradí aj jedinečný identifikátor. Ten môže byť opäť ľubovoľným objektom.

*JUNG* poskytuje možnosť vlastnej alebo úpravu už existujúcej vizualizácie objektov a hrán. Poskytuje aj dobrú podporu sledovania akcií na grafe a tak ponúka vytvorenie vlastnej interakcie s grafom. Podpora *GraphML* je žiaľ pre svoju zlú

implementáciu nepoužiteľná, ale k dispozícii je taktiež už použiteľná podpora Pajek formátu.

*JUNG* má API dokumentáciu k verzii 1.7.6 a aj pre verziu 2.0 . Manuál je žiaľ k dispozícii iba k verziám 1.0 a z externého zdroja k 2.0. Jednotlivé verzie však nie sú navzájom kompatibilné.

## Prefuse

*Prefuse* využíva na spracovávanie a interakciu s grafom akcie. Pomocou tejto koncepcie ponúka mnohé vstavané akcie ako animácia vizualizácie, filter objektov a hrán grafu, atď.. Na vizualizáciu ponúka rozloženia ako so staticky vypočítanými pozíciami, tak aj s dynamicky prepočítavanými pozíciami počas vykresľovania. Umožňuje zobrazovať aj zoskupenia objektov grafu. Všetky vizualizácie pôsobia prívetivo.

*Prefuse* poskytuje nielen vizualizáciu grafov ale aj iné možnosti ako *FishEye*, *TreeMap* a iné.

Dokumentácia *Prefuse* je v rozpracovanom stave. K dispozícii je čiastočná prvá kapitola manuálu a API dokumentácia v JavaDoc.

Tento nástroj poskytuje plnú a kvalitnú podporu GraphML, čím dáva k dispozícii možnosť presunúť generovanie grafu inej aplikácii.

## JavaScript Information Visualization Toolkit (JIT)

*JIT* je vizualizačný nástroj napísaný v jazyku JavaScript. To prináša výhody aj nevýhody. Poskytuje tak možnosť priameho prepojenia webovej stránky s vizualizáciou a tiež *AJAX* volania pre ďalšie potrebné súčasti. Definovanie grafu nad ktorým má pracovať je vo formáte *JSON*. *JIT* poskytuje aj možnosť upravenia a kontroly interakcie s grafom pomocou *ControllInterface*. K dispozícii je aj bočná lišta so záložkami, ktorej obsah je možné upraviť a prispôbiť pre potreby vizualizácie.

Medzi hlavné nevýhody patrí fakt, že tento vizualizátor pracuje na úrovni webovej stránky, kde sa neočakáva potreba veľkého výkonu a teda zložitejšie grafy je nemožné vykresliť. V prípade, keď je požiadavka vykresliť grafy, ktoré je možné rozdeliť na jednoduché podgrafy s malým počtom hrán a objektov, je možné na takéto vykresľovanie použiť *AJAX* volania a vykresľovať tak len jeden podgraf.

## Návrh na použitie v rámci projektu

Pre potreby vizualizácie grafov s počtom objektov rovnajúcemu sa počtu študentov, ktorí majú zapísaný predmet Tvorba softvérového systému v tíme (cca. 100) a počtom hrán rovnajúcemu sa počtu všetkých vzťahov študentov medzi sebou odporúčam použiť nástroj *Prefuse*. Implementovať daný nástroj ako *Java Applet* a využiť vstavanú podporu *GraphML* pre generovanie grafu.

### 3.2.3. Filtrovanie zoznamu študentov

Pedagóg má možnosť filtrovať zoznam študentov na základe ich charakteristík, pričom programovo nie je obmedzený ich počet a kombinácie (AND/OR). Doposiaľ bola vytvorená jedna úloha.

#### Filtrovanie používateľov podľa charakteristiky

Úlohou je doplniť filtrovanie pomocou techniky AJAX pri zobrazení všetkých používateľov.

#### Analýza problému

Pri zobrazovaní zoznamu študentov musí byť poskytnutá možnosť nastaviť si zobrazenie určitého okruhu študentov na základe atribútov. Toto filtrovanie má umožniť tvorcovi sociálnej siete zobraziť študentov, ktorí sú si podobní na základe požadovaného atribútu. Aby zobrazovanie nového zoznamu študentov netrvalo dlho a nemusel byť zakaždým načítavaný celý obsah stránky, je potrebné pri filtrovaní použiť AJAX.

#### Návrh

Filtrovanie ma ponúknuť možnosť výpisu zoznamu študentov na základe konkrétneho atribútu. Časť s filtrovaním obsahuje jeden combobox na výber typu charakteristiky, jeden combobox na výber typu atribútu priradeného zvolenému typu charakteristiky, ktorý sa zobrazí po zvolení typu charakteristiky a textové políčko na zadanie názvu zvoleného atribútu. Po odoslaní formulára sa zobrazí výpis zoznamu študentov, ktorým je pridelená charakteristika zvoleného typu obsahujúca atribút zvoleného typu so zadaným obsahom.

#### Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Views/user: *index.html.erb*, *\_user.html.erb*
- V zložke Controllers: *users\_controller.rb*

View *\_user.html.erb* obsahuje zdrojový kód na výpis jedného študenta, ktorý sa vloží do view *index.html.erb* po odoslaní požiadavky na výpis zoznamu filtrovaných používateľov. Filtrovanie a výpis zabezpečuje metóda *filter* v súbore *user\_controller.rb*. V tom istom súbore sa nachádza aj metóda *show\_attributes\_combobox*, ktorá zobrazí, resp. aktualizuje combobox s typmi atribútov na základe zvoleného typu charakteristiky.



## 3.2.4. Vyžiadanie potvrdenia vzťahu druhou stranou

Typ vzťahu má pri sebe informáciu o tom, či vyžaduje potvrdenie druhou stranou. Vzťah má následne atribút, či je alebo nie je potvrdený. V prípade, že A zdefiniuje vzťah V k používateľovi B a pritom typ V vyžaduje potvrdenie, tak sa pri najbližšom prihlásení používateľa B tomuto zobrazí výzva na potvrdenie vzťahu. Ak ju príjme, tak sa relácia označí ako potvrdená. V opačnom prípade sa vymaže.

Pre túto *user story* bola zatiaľ vytvorená jedna úloha.

### Rozšíriť vzťahy o potvrdenie

Úloha pozostáva z nasledujúcich vecí:

- Doplnenie do typu vzťahu informáciu o tom, či vyžaduje potvrdenie
- Doplniť do vzťahu informáciu o tom či je potvrdený
- Zapracovať potvrdzovaciu logiku a zobrazenie pre vzťahy, ktoré treba potvrdiť.

### Analýza problému

Pri niektorých typoch vzťahu môže nastať situácia, že sa ocitneme vo vzťahu k niekomu, s kým si neželáme byť. Takémuto vytvoreniu je možné zabrániť, ak má druhá strana možnosť tento vzťah potvrdiť alebo zamietnuť.

### Návrh

Pri vytvorení vzťahu aplikácia overí, či sa jedná o vzťah, ktorý treba potvrdiť. Ak je tomu tak, potom sa druhej strane ponúkne možnosť tento vzťah potvrdiť alebo zamietnuť. V prípade, že bol vzťah potvrdený, tak sa jeho vytvorenie riadi rovnakými pravidlami ako doteraz (napr. pri inverznom vzťahu sa automaticky vytvorí obrátený vzťah). V prípade, že nebol vzťah prijatý druhou stranou, tak sa tento vymaže z databázy.

### Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Controllers: *relationships\_controller.rb*,  
*type\_of\_relationships\_controller*
- V zložke Views: *users/\_show\_relationship\_detail.html.erb*

Do tabuľky *type\_of\_relationships* je doplnený atribút *acknowledge*, ktorý určuje, či konkrétny typ vzťahu vyžaduje potvrdenie druhou stranou. Do tabuľky *relationships* je pridaný atribút *status* označujúci, či bol daný vzťah potvrdený druhou stranou, ak je to potrebné.

## Testovanie

V zložke *Functional Tests* sa v súbore *relationships\_controller\_test.rb* nachádzajú testy pre overenie správania sa implementácie v zložke *Controllers*.

## 3.2.5. Rozdelenie aplikácie na zóny s rozdielnou úrovňou ochrany prístupu

Niektoré časti aplikácie sú prístupné bez prihlásenia, niektoré sú prístupné len prihláseným používateľom, niektoré iba používateľom v roli pedagóga a niektoré časti len pre administrátora.

Zatiaľ boli pre túto *user story* vytvorené dve úlohy.

### Rozdelenie prístupu na metódy kontrolerov podľa role prihláseného používateľa

Úloha zahŕňa dve veci:

- Vytvorenie role administrátora
- Úprava možnosti volania metód podľa role prihláseného používateľa

### Analýza problému

Po prihlásení používateľa do systému je potrebné rozlišovať medzi rolami používateľov a poskytnúť im podľa ich rolí aj rôzne možnosti, ktoré prostredie poskytuje. Používatelia sú rozdelení na 3 skupiny a to skupinu administrátorov, učiteľov a študentov. Najväčšie práva má skupina administrátorov a najmenšie skupina študentov. Každá vyššia skupina zároveň obsahuje aj práva nižších skupín.

### Návrh

Na obmedzenie prístupu používateľa na metódy kontroléru sa vytvorí *before\_filter* metóda, ktorá funguje tak, že sa vykoná vždy pred spustením samotnej metódy, na ktorú sa viaže. V tejto *before\_filter* metóde sa vyhodnotí akú rolu má prihlásený používateľ a keď má dostatočné práva, povolí mu vykonanie metódy. V opačnom prípade ho presmeruje naspäť.

### Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke *Controllers*: *application.rb*, *char\_attributes\_controller.rb*, *characteristics\_controller.rb*, *login\_controller.rb*, *relationships\_controller.rb*,

*type\_of\_attributes\_controller.rb, type\_of\_chars\_controller.rb,  
type\_of\_relationships\_controller.rb, users\_controller.rb*

V *application.rb* sa nachádzajú *before\_filter* metódy *staff\_filter* a *admin\_filter*, ktoré overujú rolu prihlasného používateľa a podľa nej povolia alebo zamietnu prístup k metóde. Vo všetkých controller triedach sa nachádza nastavenie, na ktoré metódy sa ma použiť *staff\_filter* alebo *admin\_filter*.

## Testovanie

V zložke *Functional Tests* sa v súboroch nachádzajú Unit testy pre jednotlivé kontrolery.

## Zahrnúť všetky časti aplikácie (okrem prihlasovania) do chránenej zóny

Úloha zatiaľ nebola realizovaná.

## 3.2.6. Definovanie navigácie v aplikácii a úvodná obrazovka

Prihlásený používateľ musí mať možnosť jednoduchšej navigácie, bez potreby intenzívneho používania tlačidiel späť a vpred. Už tu musí byť oddelená navigácia pre študenta, pedagóga a administrátora. Po prihlásení používateľ vidí úvodnú obrazovku s informáciami a notifikáciami od posledného prihlásenia.

Pre túto *user story* boli zatiaľ vytvorené tri úlohy.

## Kompletná navigačná logika

Úloha vyžadovala zapracovanie navigačnej logiky podľa role používateľa.

## Analýza problému

Z každého formulára, ktorý má používateľ zobrazený musí mať možnosť navigácie do iných častí systému.

## Návrh

Každý formulár musí obsahovať navigáciu na úvodnú obrazovku používateľa. Okrem toho formulár zobrazí dynamicky tie navigačné odkazy, na ktoré má používateľ právo v závislosti od role v akej je zaradený.

## Implementácia

Funkcionalita úlohy implementačne zasahuje do všetkých views v systéme. Každý view obsahuje odkaz na úvodnú obrazovku a explicitne vytvorenú množinu odkazov na views so súvisiacim obsahom. Každý odkaz má priradené role, ktorým sa tento odkaz zobrazí a majú právo ho použiť.

## Úvodná obrazovka

Úloha spočívala v navrhnutí a implementovaní úvodnej obrazovky podľa role používateľa a zahrnúť do nej vykreslenie informácií od posledného prihlásenia.

### Analýza problému

Po prihlásení používateľa do systému je potrebné poskytnúť mu úvodnú obrazovku, kde bude mať zobrazené možnosti práce, ktoré mu systém ponúka. Používateľ musí mať na úvodnej obrazovke prehľad o zmenách vo vzťahoch v sociálnej sieti, ktorých je súčasťou.

### Návrh

Štruktúra úvodnej obrazovky je závislá od role pod akou sa používateľ prihlási do systému. Používateľovi v roli študent sa zobrazia možnosti súvisiace s prezeraním a editovaním svojho profilu. Používateľ v roli pedagóg bude mať navyše možnosť vytvárať skupiny študentov na základe vzťahov v sociálnej sieti a filtrovať študentov. Administrátor bude mať okrem toho zobrazené možnosti pre editáciu sociálnej siete a jej dát v databáze.

Každému používateľovi nezávisle od role sa zobrazia na úvodnej obrazovke zmeny v sociálnej sieti, ktoré sa k nemu vzťahujú. Tieto zmeny sa vyhľadajú v tabuľke *notification*.

### Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Models: *login.rb*
- V zložke Views/login: *index.html.erb*
- V zložke Controllers: *login\_controller.rb*

Vykreslenie úvodnej obrazovky zabezpečuje metóda *index* v *login\_controller.rb*, táto metóda je implicitne zavolaná z metódy *login* po úspešnom prihlásení používateľa.

### Testovanie

V zložke *Unit Tests* v súboroch *user\_test* a *user\_controller\_test* sa nachádzajú metódy pre otestovanie zobrazenia úvodnej obrazovky pre jednotlivé typy rolí (študent, pedagóg a administrátor).

## Vytvorenie notifikačnej tabuľky

Úloha spočíva vo vytvorení tabuľky, ktorá bude obsahovať notifikáciu pre používateľa. Notifikácie sa zobrazia na úvodnej obrazovke a budú obsahovať odkaz na vzťahy.

## Analýza problému

Používateľ by mal po prihlásení byť oboznámený o nových udalostiach, ktoré sa ho týkajú. Konkrétne sa jedná o nové vzťahy.

## Návrh

Používateľ po prihlásení uvidí úvodnú obrazovku, na ktorej budú vo forme viet s odkazmi zobrazené nové vzťahy. Tieto notifikácie zatiaľ nebudú obsahovať len informácie od posledného prihlásenia ale informácie o všetkých udianých udalostiach od registrácie používateľa. Funkcionalita schovávania a mazania notifikácií môže byť implementovaná v neskorších fázach projektu, rovnako ako aj definovanie notifikačnej správy žiadajúcou stranou.

## Implementácia

Funkcionalita úlohy je implementovaná v týchto zdrojových súboroch:

- V zložke Models: *notification.rb*
- V zložke Controllers: *notifications\_controller.rb*
- V zložke Views: *login/index.rhtml*, *users/\_notification\_detail.html.erb*, *users/\_notifications.html.erb*

Zobrazenie *\_notifications.html.erb* sa renderuje v *index.rhtml*, ktorý obsahuje úvodnú obrazovku po prihlásení.

## Testovanie

V zložke *Functional Tests* sa v súboroch nachádzajú testy pre jednotlivé implementácie zo zložky Controllers.