Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Webový komunitný systém otázok a odpovedí

Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Ivan Srba
Členovia tímu: Bc. Rastislav Dobšovič, Bc. Marek Grznár, Bc. Jozef Harinek,
Bc. Samuel Molnár, Bc. Peter Páleník, Bc. Dušan Poizl, Bc. Pavol Zbell
Akademický rok: 2013/2014

Table of Contents

1 Úvod

Naším cieľom je vytvoriť webový systém, ktorý bude umožňovať interakciu medzi študentami navzájom, alebo medzi študentami a vyučujúcimi pomocou otázok a odpovedí. Študenti tak budú môcť riešiť svoje problémy so zadaniami alebo nejasnosti z učiva položením otázky v našom systéme. Študent, ktorý otázku položil, určí správnu odpoveď a tak ostatní študenti s rovnakým problémom môžu rýchlo nájsť správne riešenie. Pre prehľadnosť budú mať všetky otázky rôzne *informačné značky*.

Naším cieľom bude v prvom semestri vytvoriť funkčnú webovú službu, ktorá bude umožňovať kladenie otázok, odpovedanie. Ďalej bude k dispozícií hlasovanie pre jednotlivé odpovede alebo komentovanie odpovedí. Nakoniec budeme pomocou informačných značiek umožňovať vyhľadávanie v otázkach.

V druhom semestri sa do systému budú pridávať ďalšie vylepšenia. Bude sa pracovať na prispôsobovaní úvodnej obrazovky pre potreby jednotlivých používateľov.

2 Ciele na zimný semester

1. šprint – Drone:

- zaznamenávanie udalostí,
- lokalizácia,
- prihlásenie,
- profil používateľa.

2. šprint – Roach:

- vloženie novej otázky,
- zobrazenie otázky,
- zobrazenie zoznamu nových otázok.

3. šprint – Hydralisk:

- vloženie novej odpovede
- zobrazenie zoznamu zodpovedaných otázok
- výber najlepšej odpovede

4. šprint – Infestor:

- hlasovanie za odpoveď,
- vyhľadávanie v otázkach,
- vloženie komentáru k odpovedi,
- preferencie a práva.

5. šprint – Ultralisk:

- refaktorizácia.

3 Šprint 1 – Drone

3.1 Autorizácia

3.1.1 Úloha

Navrhnúť rozhranie pre autorizáciu používateľov v systéme na základe ich roli a právomoci. K rozhraniu je potrebné pridať aj príklad použitia.

3.1.2 Návrh

Pre implementáciu rolí a právomoci používateľov sme použili knižnicu *CanCan*[1]. Knižnica implementuje jednoduché rozhranie, pomocou ktorého je možné definovať základné pravidlá pre autorizáciu používateľov. Rozhranie spracováva právomoci primárne na strane jazyka *Ruby.* Na základe toho je architektúra rozšíriteľná aj o právomoci perzistované v rámci databázy. Všetky právomoci používateľa sú definované v súbore `ability.rb`, ktorý sa nachádza v adresári `app/models`. Všetky právomoci sú definované pomocou kľúčového slova `can` alebo `cannot`, ktoré je možné použiť v pohľadoch a pri spracovávaní požiadaviek od aktuálneho používateľa na určenie jeho právomocí.

3.1.3 Implementácia a testovanie

Ako príklad pre autorizáciu používateľa sme zvolili možnosť zmeny mena. Používateľ prihlásený údajmi zo systému *AIS*[2] nemá právomoc na zmenu svojho mena a priezviska. Polia s menom a priezviskom sú pre tohto používateľa v sekcii úpravy profilu vypnuté. Na strane servera sa kontrolujú právomoci používateľa editovať meno a priezvisko a na základe kontroly právomoci sa určujú parametre, ktoré sú povolené v odoslanej požiadavke (angl. request). Právomoc používateľa sme definovali s názvom `change_name`, pričom túto akciu daný používateľ môže vykonať nad modelom `User`. Dané povolenie sme otestovali v jednotkových testoch modelu `User`. Na testovanie sme použili pomocnú metódu `be_able_to` knižnice *CanCan* pre testovací rámec *Rspec*[3].

3.2 Lokalizácia

3.2.1 Úloha

Návrh a implementácia lokalizácie aplikácie. Návrh má brať do úvahy rôzne spôsoby a prístupy k lokalizácii v Rails aplikáciách. Cieľom je nájsť a definovať najvhodnejší spôsob a prístup k lokalizácii aplikácie pre potreby tohto projektu.

1 CanCan: https://github.com/ryanb/cancan
2 AIS: http://is.stuba.sk/
3 Rspec: https://github.com/rspec/rspec

3.2.2 Návrh

Pri návrhu sme dôkladne zvážili možnosti implementácie lokalizácie Rails aplikácie. Nerozhodli sme sa pre použitie lokalizovaných pohľadov, t.j. rozdelenia pohľadov napr. pri `app/views/users` do podadresárov `en` a `sk`, kvôli zneprehľadneniu a duplikovania zdrojového kódu samotných pohľadov. Lokalizáciu preto navrhujeme riešiť na úrovni konfiguračných súborov v adresári `config/locales`. V tomto adresári plánujeme vytvoriť prehľadnú štruktúru podadresárov ako napr. `errors`, `helpers`, `models` a `views`. V každom z týchto adresárov budú ďalej podadresáre pre lokalizáciu knižníc tretích strán a podadresár `views` sa bude organizovať podobne ako `app/views`. Podadresáre budú obsahovať konfiguračné súbory `en.yml` a `sk.yml`. Pre potreby lokalizácie sa v zdrojovom kóde budú referencovať preklady pomocou kľúčov, ktoré by mali reflektovať štruktúru podadresárov (najmä v prípade `views`) pomocou metódy `translate(key)`, skrátene `t(key)`.

3.2.3 Implementácia a testovanie

Referenčná implementácia a testovanie je zahrnuté v rámci pohľadu a editovania profilu používateľa. Viď pohľady v adresároch `app/views/{devise,user}` a konfiguračné súbory v adresároch `config/locales/views/{devise,users}`.

3.3 Prihlásenie

3.3.1 Úloha

Návrh a implementácia prihlasovania a registrácie používateľa. Návrh má brať do úvahy možnosť prihlásenia údajmi z Akademického informačného systému (ďalej len AIS) bez potreby manuálnej registrácie používateľa.

3.3.2 Návrh

Pri návrhu sme využili knižnicu *Devise*[4], ktorá obsahuje väčšinu funkcionality pre registráciu a prihlasovanie. Knižnica je rozdelená do viacerých modulov, ktoré obsahujú funkcionalitu pre registráciu, prihlásenie, obnovu hesla, potvrdenie účtu emailom a zamknutie účtu za určitých podmienok. Používateľ je reprezentovaný modelom `User`. Model využíva všetky modulu knižnice *Devise*. Základnými atribútmi používateľa sú login, email a heslo, pričom používateľ sa po registrácii prihlasuje pomocou atribútov login a heslo.

Pre potrebu prihlásenia používateľa údajmi z AIS sme rozšírili modul prihlasovania knižnice *Devise*. Nahradili sme metódu `create` v triede `SessionsController` vlastnou implementáciou prihlasovania. Overovanie údajov používateľa pre AIS je realizované servisným objektom `Users::Authentication`, ktorý ako atribúty uvažuje vzdialenú službu pre autentifikáciu

4 Devise: https://github.com/plataformatec/devise

používateľa a prihlasovacie parametre, ktoré zadal používateľ. Vzdialená služba LDAP Akademického informačného systému je reprezentovaná triedou `Stuba::AIS`, ktorá na základe mena a hesla autentifikuje používateľa. Výsledkom autentifikácie je inštancia objektu `Stuba::User`, ktorá obsahuje atribúty profilu používateľa v akademickom systéme. Pri prvom prihlásení vytvoríme nového používateľa z údajov z AIS profilu. Pre potrebu rozlíšenia AIS používateľa a registrovaného používateľa sme do modelu `User` pridali atribúty

- `ais_login` – login používateľa v systéme AIS,
- `ais_uuid` – identifikačné číslo používateľa v systéme AIS,

pričom oba atribúty sa nastavujú len v prípade prihlásenia údajmi z AIS.

3.3.3 Implementácia a testovanie

Model používateľa je otestovaný len na úrovni validácií a metód, o ktoré sa model rozšíril, keďže väčšina funkcionality pre prihlasovanie je otestovaná v testoch knižnice *Devise*. Autentifikácia pomocou údajov z AIS je izolovaná v rámci vlastnej knižnice. Komponenty tejto knižnice sú otestované v integrácii s knižnicou pre tvorbu LDAP dopytov. Servisný objekt `Users::Authentication`, ktorý implementuje autentifikáciu používateľa pomocou vzdialenej služby, je otestovaný vzorovými údajmi s využitím simulovaného správania vzdialenej služby. Jednotlivé pohľady knižnice *Devise* pre prihlásenie a registráciu sme upravili pre integráciu s rámcom *Bootstrap*[5]. Funkcionalita daných pohľadov je pokrytá akceptačnými testami, pričom akceptačné testy pre prihlasovanie údajmi z AIS opäť simulujú správanie formou metódy `authenticate` triedy `Stuba::AIS` so vzorovými údajmi.

3.4 Profil používateľa

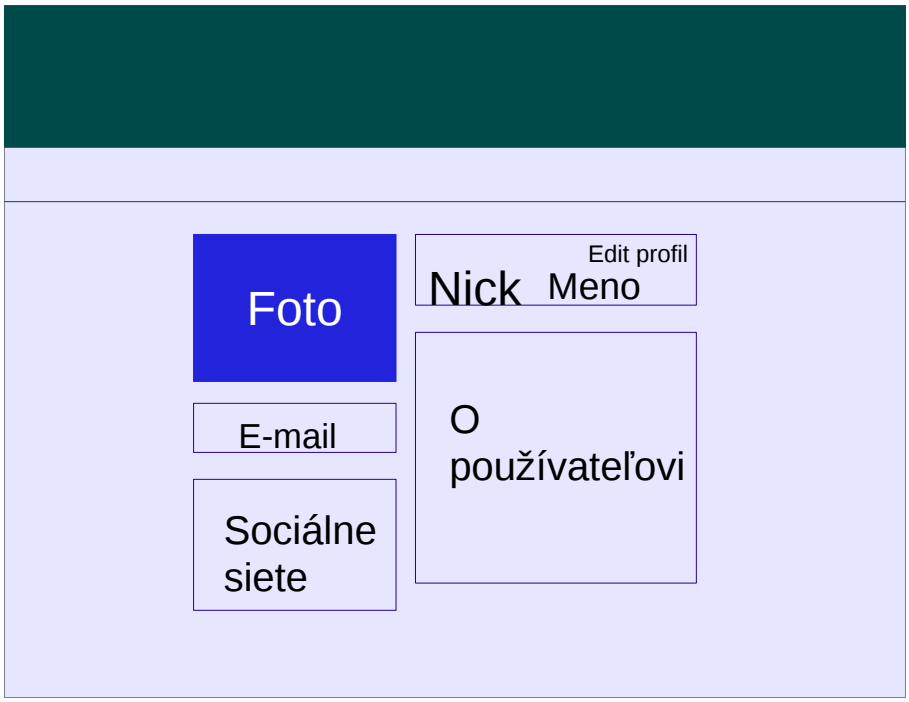
3.4.1 Úloha

a) Navrhnúť a implementovať rozhranie pre zobrazenie profilu používateľa. Samotný návrh musí obsahovať autorizáciu používateľa. Ak sa používateľ rozhodne nezverejniť svoje meno a email iným používateľom, tak tieto údaje sú viditeľné iba pre neho. Pred inými používateľmi sú tieto informácie skryté.

b) Návrh a implementácia formulárov pre zmenu profilu a účtu používateľa. Návrh má brať do úvahy autorizáciu používateľa, podľa práv v systéme (používateľ prihlásený pomocou AIS konta nemôže editovať svoje meno). Návrh má takisto brať do úvahy rozdelenie menu podľa špecifikácie na logické celky.

5 Bootstrap: http://getbootstrap.com/

3.4.2 Návrh



Obr. 1: Návrh prostredia pre zobrazenie profilu

a) Ako je zobrazené na Obr. 1 rozhranie obsahuje tieto základné prvky: foto, email, sociálne siete, o používateľovi, prezývku, meno, presmerovanie na editovanie profilu.

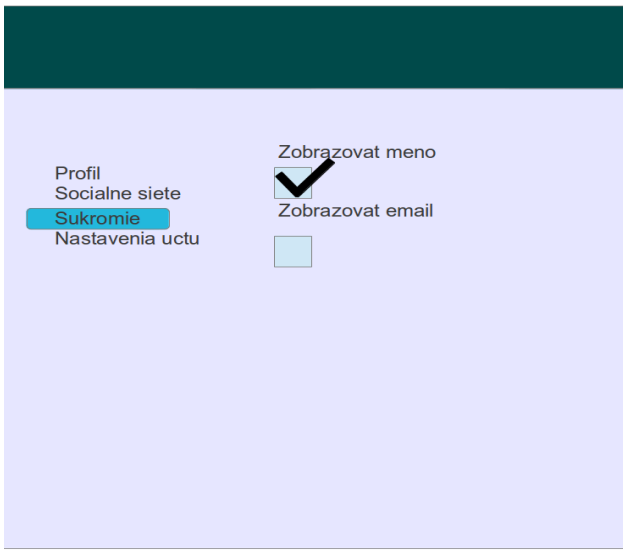
b) Pri návrhu sme podobne ako pre prihlasovanie použili knižnicu *Devise*[6], ktorá poskytuje funkcionalitu pre autentifikáciu používateľa, v tomto prípade sme ju použili pre validáciu údajov vo formulári pre editáciu nastavení účtu (4. časť v grafickom návrhu). Na autorizáciu je použitá knižnica *CanCan*[7], pomocou ktorej je spravované nastavenie práv v systéme. Podľa toho či má na to používateľ práva, má možnosť zmeniť si meno (AIS používateľ si meno zmeniť nesmie).

Ako je znázornené na Obr. 3, editácia profilu je rozdelená do štyroch častí:

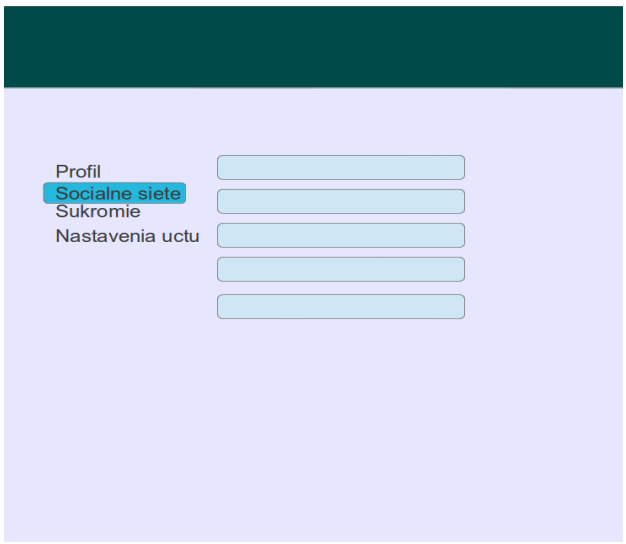
- profil,
- sociálne siete,

6 Devise: https://github.com/plataformatec/devise
7 CanCan: https://github.com/ryanb/cancan

- súkromie (náčrt je vyobrazený na Obr. 2.),
- nastavenia účtu,
- v každej z častí je príslušný formulár, v ktorom je možnosť meniť údaje.



Obr. 2: Návrh používateľského rozhrania pre nastavenia súkromia



Obr. 3: Návrh používateľského rozhrania pre úpravu profilu

3.4.3 Implementácia a testovanie

a) Na zobrazenie *Gravatar*[8] fotografie sme implementovali `helper`, ktorý pomocou `gravatar_email` vracia používateľovu fotku z *Gravatar*. Keďže mail na G*ravatar* nemusí byť vyplnený zobrazí sa základný obrázok. Na implementáciu dizajnu grafického rozhrania pre zobrazenie profilu sme využili rámec *Bootstrap*[9].

Rozhranie pre zobrazenie profilu sme otestovali akceptačnými testami. Na spravovanie práv pre zobrazenie voliteľných prvkov je použitá knižnica *CanCan*.

b) Formuláre pre úpravu profilu používateľa sme otestovali akceptačnými testami. V modeli používateľa boli otestované doplnené metódy a validácie. Do modelu bola pridaná metóda vracajúca mail na *Gravatar* – `gravatar_email` (Globally recognized avatar, služba spájajúca mailovú adresu používateľa s jeho globálnym profilom), keďže mail pre službu *Gravatar* nemusí byť rozdielny od mailu účtu – vtedy je toto pole v databáze prázdne a má sa zobraziť mail účtu. Formuláre sú naštýlované pomocou rámca Bootstrap.

Pohľad pre editáciu profilu používateľa je rozdelený na štyri časti, ktoré tvoria menu. V každej časti je samostatný formulár. Okrem formulára, ktorý spracúva knižnica *Devise* (zmena nastavení účtu) sú formuláre spracované kontrolórom pre používateľa v metóde `update_profile`.

3.5 Zaznamenávanie udalostí

3.5.1 Úloha

Návrh a implementácia zaznamenávania udalostí, ktoré nastanú v systéme. Návrh má brať do úvahy možnosť automatického aj manuálneho zaznamenávania udalostí vzhľadom na interakciu používateľa so systémom.

3.5.2 Návrh

Pre potreby projektu sme navrhli jednoduché automatické aj manuálne zaznamenávanie udalostí, ktoré vznikajú pri interakcii používateľa so systémom priamo v `Controller` triedach Rails Aplikácie. Pri manuálnom zaznamenávaní udalostí je cieľom zaznamenať udalosť (dáta) v čo najmenšom počte riadkov zdrojového kódu a prehľadnou formou (kvôli zachovaniu prehľadnosti samotnej logiky v moduloch `Controller`).

3.5.3 Implementácia a testovanie

Pri implementácii sme použili vzor *Service Object*, ktorý výrazne uľahčuje zaznamenávanie udalostí aj mimo modulov `Controller` a okrem toho umožňuje aj lepšie testovanie samotnej funkcionality

8 Gravatar: http://www.gravatar.com/

9 Bootstrap: http://getbootstrap.com/

zaznamenávania udalostí. Na databázovej vrstve sú zaznamenané udalosti uchovávané v tabuľke `Events`, ktorá obsahuje čas uloženia udalosti a dáta udalosti vo formáte JSON. Udalosť môže obsahovať (takmer) ľubovolné JSON dáta, pričom tie sú tesne pred uložením vždy automaticky obohatené o rôzne podporné dáta ako napr. identifikátor sedenia, URL, identifikátor akcie triedy `Controller` a jej parametre (aj z formulárov – heslá a iné citlivé informácie sú bezpečne automaticky odstránené a do databázy sa neukladajú), dáta o používateľovi ako login alebo e-mail. Pri automatickom ukladaní udalostí sa mechanizmus ukladania spúšťa v rámci `before_action` každej akcie ľubovoľnej triedy, ktorá priamo dedí od triedy `ApplicationController`. Na manuálne ukladanie udalostí priamo v triede `Controller` (napríklad pre potrebu uloženia rôznych dát udalosti podľa vetvenia implementovanej akcie) slúži metóda `log`, ktorej jediný parameter sú JSON dáta udalosti (dátový typ `Hash` v Ruby), ktoré musia mať na v koreni kľúč `action`, ktorého hodnota stručne popisuje udalosť. Okrem kľúča `action` je možné v JSON dátach špecifikovať aj iné (takmer) ľubovoľné kľúče, ktoré budú uložené pre danú udalosť.

4 Šprint 2 – Roach

4.1 Vloženie novej otázky

4.1.1 Úloha

Navrhnúť a implementovať pridávanie nových otázok do systému. Návrh má brať do úvahy asociovanie kategórií a značiek (angl. tag) k vytváranej otázke.

4.1.2 Návrh

Navrhli sme jednoduchý model otázok – `Question`, ktorého základnými atribútmi sú názov a text otázky. K tejto entite sme pridali asociáciu na model kategórie – `Category`, ktorý dopĺňa sémantiku otázky a jej zaradenie. Pre rozšírenie sémantiky otázky sme si zvolili značky, ktoré budú používatelia môcť pridávať k otázkam. Na implementáciu značiek sme si zvolili knižnicu *acts-as-taggable-on*[10]. Knižnica implementuje dva modely – `Tag` a `Tagging`. Model `Tagging` slúži ako prepojovacia tabuľka medzi inými entitami a modelom `Tag`. Knižnica je rozšíriteľná pre všetky scenáre pridávania značiek alebo iných označení pre entity, pretože využíva polymorfické asociácie.

Model značiek sme rozšírili o normalizáciu, pričom všetky značky sú normalizované na malé písmená a všetky medzery sú nahradené pomlčkou. Pri implementácii rozhrania sme si zvolili knižnicu *select2*[11], ktorý unifikuje vzhľad pre pole s možnosťou výberu (angl. select box). Nad knižnicou sme vytvorili jednoduché API pre definovanie vlastností pre polia so značkami. Vzhľad elementov *select2* sme prispôsobili rámcu *Bootstrap*. Pre lokalizáciu textov v JavaScript kóde sme použili knižnicu *i18n-js*[12].

4.1.3 Implementácia a testovanie

Pridanie novej otázky sme otestovali na úrovni akceptačných testov. Pre potreby otestovania funkcionality knižnice *select2* sme vytvorili pomocné metódy pre vloženie vstupu do polí *select2*. Vytvorili sme *FactoryGirl*[13] definície pre model `Category`, `Question` a `Tag`.

10 acts-as-taggable-on: https://github.com/mbleigh/acts-as-taggable-on
11 select2: http://ivaynberg.github.io/select2/
12 i18n-js: https://github.com/fnando/i18n-js
13 FactoryGirl: https://github.com/thoughtbot/factory_girl

4.2 Zobrazenie otázky

4.2.1 Úloha

Zobraziť vybranú otázku. Okrem samotnej otázky a textu otázky je treba zobraziť aj základné informácie o používateľovi ktorý otázku položil.

4.2.2 Návrh

Zobrazenie otázky zabezpečí samostatný pohľad. Otázka bude zobrazená v troch stĺpcoch. V prvom stĺpci bude zobrazené informácie o autorovi ako sú meno, jeho fotka. Meno. V strednom stĺpci sa bude zobrazovať nadpis otázky, dátum položenia otázky, značky a samotný text otázky. Pravý stĺpec bude obsahovať hlasovanie k otázke a či je otázka zodpovedaná.

4.2.3 Implementácia a testovanie

Na zobrazenie otázky slúži štandardná metóda `show` v triede `QuestionsController`. V tejto metóde sa nájde otázka s *id* predaným ako parameter. Nájdenie otázky zabezpečuje trieda `Question` implementujúca dátový model. Triedy implementujúce dátový model dedia od `Base::ActiveRecord`. Pohľad je implementovaný pomocou *Bootstrap*.

Prezývka a fotka slúžia ako odkaz na profil používateľa. Dátum a čas položenia otázky zobrazuje špeciálna knižnica ktorá automaticky zobrazuje čas vo forme času ktorý uplynul od položenia otázky. Napríklad pred 15 minútami, dvoma dňami a podobne. Od určitej doby ale zobrazuje už len čistý dátum. Informácie o používateľov sú vyčlené do samostatného *partial view* v súbore `views/users/_square.html.erb`. Hlasovanie k otázke je vyčlené taktiež do samostatného súboru `views/questions/_voting.html.erb`.

4.3 Zobrazenie nových otázok

4.3.1 Úloha

Návrh a implementácia zobrazenia nových otázok. Zobrazovať sa budú len nové otázky, ktoré budú zoradené podľa ich dátumu vzniku. Spolu s otázkami sa budú zobrazovať aj značky, kategórie, meno používateľa, ktorý otázku položil a doba, kedy bola otázka položená.

4.3.2 Návrh

V rozhraní pre zobrazenie nových otázok sú podľa Obr. 4 uvedené prvky:

- tlačidlo na pridanie novej otázky,

- názov otázky s kategóriami a značkami,
- doba kedy bola otázka pridaná,
- autor otázky,
- A – počet hlasov,
- B – počet odpovedí,
- C – počet videní.

Obr. 4: Zobrazenie zoznamu nových otázok

Do návrhu sa pridalo stránkovanie pomocou knižnice *kaminari*[14], ktorá sprehľadňuje prehľadávanie aj vo väčšom množstve otázok.

4.3.3 Implementácia a testovanie

Na implementáciu dizajnu grafického rozhrania pre zobrazenie profilu sme využili rámec

14 Kaminari: https://github.com/amatsuda/kaminari

Bootstrap[15] doplnené o *custom CSS*. V zložke `views/kaminari` za pomoci `_paginator` a ďalších sme vytvorili stránkovanie, pričom sme v `questions_controller` pomocou `Question.order(:created_at).page(params[:page]).per(10)` zadefinovali, aby sa radili otázky podľa dátumu pridania a nastavili sme desať otázok na stránku. Pohľad pre zobrazenie otázok vypisuje otázky, kategórie spolu so značkami, ktoré sú farebne odlíšené. Informácie o otázke ako jej názov, autor, dátum vytvorenia a jej štatistiky sa spracovávajú z databázy.

Pre overenie funkcionality sme vytvorili akceptačný test. Testujeme korektnosť vypisovaných údajov na stránke.

15 Bootstrap: http://getbootstrap.com/

5 Šprint 3 – Hydralisk

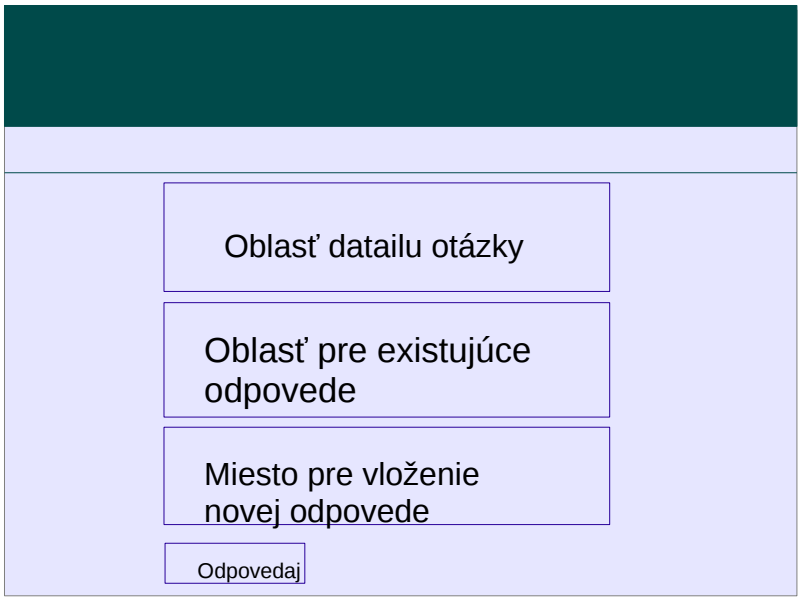
5.1 Vloženie novej odpovede

5.1.1 Úloha

Navrhnúť a implementovať rozhranie pre vloženie novej odpovede. Používateľ bude môcť odpovedať na otázku, ktorej detail si zobrazí. Návrh má umožniť odpovedi priradiť rôzne „stavy“.

5.1.2 Návrh

Navrhli sme model odpovedí. Hlavnými atribútmi sú text odpovede a identifikátor otázky, ku ktorej sa viaže a identifikátor používateľa, ktorý odpoveď vytvoril. Na Obr. 5 je zobrazené rozhranie pre vloženie novej odpovede. Oblasť detailu otázky a oblasť pre existujúce odpovede neboli našou úlohou.



Obr. 5: Návrh grafického rozhrania pre vloženie novej odpovede

5.1.3 Implementácia a testovanie

Grafický návrh rozhrania pre vloženie novej otázky sme sa rozhodli implementovať ako *partial view* `_answer_form.html.erb`, ktorý sa pomocou príkazu `<%= render 'answer_form' %>` zavolá vo view pre zobrazenie otázky (kapitola 3.2. Zobrazenie otázky). Implementácia pomocou *partial view* je hlavne z dôvodu prehľadnosti zdrojového kódu. Na implementovanie „stavov“ sme použili gem *acts-as-taggable-on*[16], ktorý nám umožní pohodlné značkovanie (udelenie „stavov“).

Vytvorenie otázky sme otestovali akceptačnými testami.

5.2 Zobrazenie odpovedí pri otázke

5.2.1 Úloha

Zobraziť odpovede na predtým zodpovedanú otázku

5.2.2 Návrh

Otázky sa budú zobrazovať pod otázkou. Doplní sa teda pohľad zobrazenia otázky. Každá otázka bude mať viacero odpovedí. Odpoveď sa bude zobrazovať podobne ako otázka. Odpoveď bude mať text odpovede, čas, autora. Podobne ako otázka je aj odpoveď rozdelená do troch stĺpcov. Prvý bude obsahovať informácie o používateľovi, druhý dátum vyplnenia odpovede, a tretí hlasovanie k odpovedi.

5.2.3 Implementácia a testovanie

Zobrazenie odpovedí pod otázkou je implementované v samostatnom *partial view* `_answers.html.erb`. V tomto pohľade je zobrazený počet odpovedí. Zobrazenie jednotlivých odpovedí je implementované v `/views/questions/_answer.html.erb`. Hlasovanie k odpovedi je v samostatnom *partial view* `views/answers/_voting.erb.html`.

16 Acts-as-taggable-on: https://github.com/mbleigh/acts-as-taggable-on

6 Celkový pohľad po prvý kontrolný bod

A prvé tri šprinty sa nám podarilo vytvoriť prototyp komunitného systému otázok a odpovedí, ktorý obsahuje nasledovné funkcionality:

- Autorizácia – V systéme existuje viac druhov používateľov, preto boli vytvorené rôzne právomoci pre rôzne skupiny.

- Lokalizácia – Náš systém bude obsahovať dve jazykové verzie. Implementované sú funkcie pre preklady do iného jazyku. Zatiaľ sú vytvorené len slovenské preklady.

- Prihlásenie – Do nášho systému je možné sa registrovať priamo ako nový používateľ alebo použiť prihlasovacie údaje z IS.

- Profil používateľa – Po úspešnom prihlásení sa do nášho systému si môže používateľ zobraziť svoj profil. Okrem zobrazenie môže upraviť aj jednotlivé položky ako email alebo prezývku. Niektoré zmeny nie sú povolené za základe skupiny do akej používateľ patrí. Môže si nastavovať úrovne súkromia alebo pridávať odkazy na svoj profil na sociálnych sieťach.

- Vloženie novej otázky – Implementované je aj pridávanie otázok spolu s kategóriami a značkami.

- Zaznamenávanie udalostí – Nami vytvorený systém dokáže zaznamenávať aktivity používateľa, na základe ktorých bude možné vytvárať štatistiky alebo odporúčanie.

- Zobrazenie otázky – Otázky sa zobrazujú v zozname zoradené podľa dátumu ich pridania. Spolu s nimi sa zobrazia aj značky, kategórie, dátum vytvorenia a autor otázky. Po kliknutí na konkrétu otázku sa zobrazí celá otázka aj s odpoveďami.

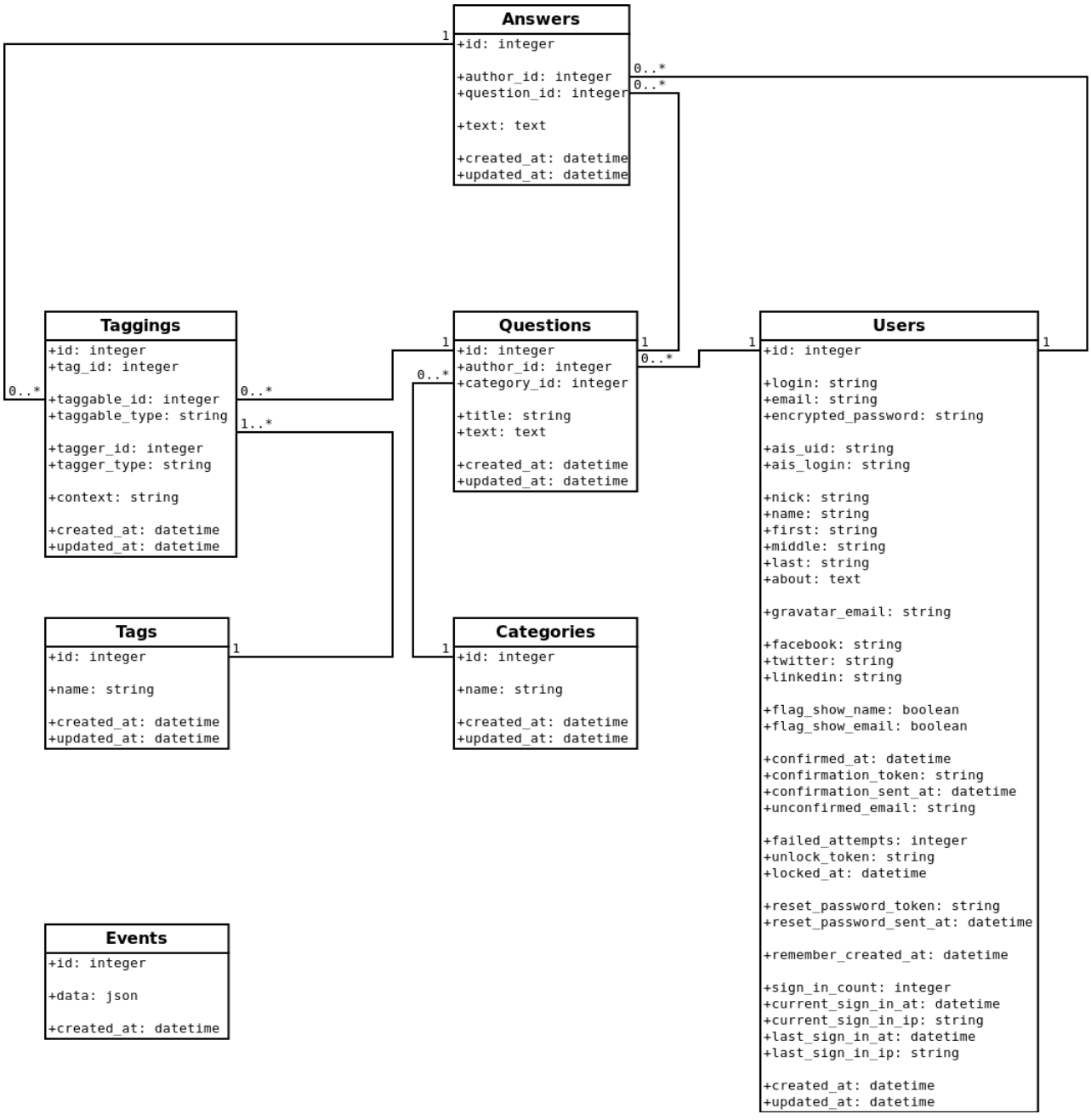
6.1 Architektúra

Z architektonického hľadiska ide zatiaľ o klasickú webovú aplikáciu založenú na návrhovom vzore MVC – modelu, pohľadu a kontrolóra (angl. model-view-controller). Na databázovej vrstve aplikácia používa relačnú databázu PostgreSQL 9, na aplikačnej vrstve sú to rámce Rails 4 (webový rámec MVC) a rámec Bootstrap 3 (rámec pre tvorbu používateľského rozhrania).

Okrem spomenutých rámcov je v projekte použité aj značné množstvo knižníc na rôznych úrovniach architektúry, napr. pri testovaní, asociovaní entít so značkami alebo autentifikácii používateľa. V tejto fáze prototypovania aplikácie je cieľom znovupoužiť čo najviac existujúcich riešení (knižníc), v neskorších fázach naopak podľa potreby implementovať vlastné riešenia.

6.2 Dátový model

Na Obr. 6 je znázornený vytvorený dátový model, ktorý obsahuje zatiaľ všetky použité entity.



Obr. 6: Dátový model