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

Obsah

1 Úvod

Naším cieľom je vytvoriť webový systém, ktorý bude umožňovať interakciu medzi študentmi navzájom, alebo medzi študentmi 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 dispozícii 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 používateľa a prihlasovacie parametre, ktoré zadal používateľ. Vzdialená služba LDAP

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

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.

3.4.2 Návrh

a) Ako je zobrazené na Obr. 3-1 rozhranie obsahuje tieto základné prvky: foto, email, sociálne siete, o používateľovi, prezývku, meno, presmerovanie na editovanie profilu.

5 Bootstrap: http://getbootstrap.com/

Obr. 3-1: Návrh prostredia pre zobrazenie profilu

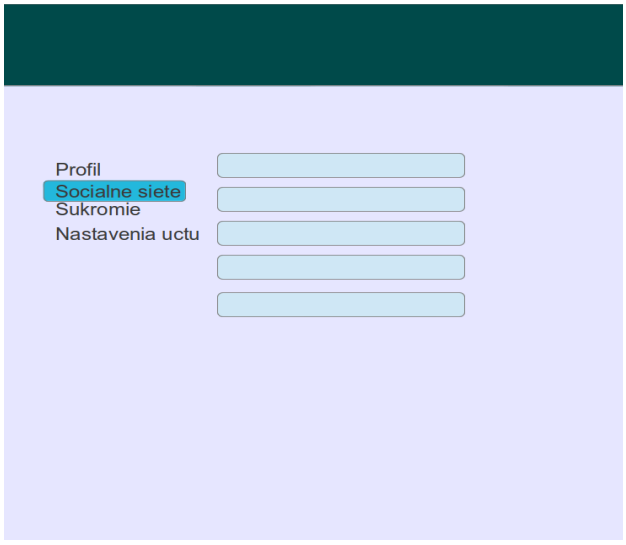
b) Pri návrhu sme podobne ako pre prihlasovanie použili knižnicu *Devise*[6], ktorá poskytuje funkcionalitu pre autentifikáciu používateľa, v tomto prípade sme ju použili pre validáciu údajov vo formulári pre editáciu nastavení účtu (4. časť v grafickom návrhu). Na autorizáciu je použitá knižnica *CanCan*[7], 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. 3-3, editácia profilu je rozdelená do štyroch častí:

- profil,
- sociálne siete,
- súkromie (náčrt je vyobrazený na Obr. 3-2.),
- nastavenia účtu,
- v každej z častí je príslušný formulár, v ktorom je možnosť meniť údaje.

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

Obr. 3-2: Návrh používateľského rozhrania pre nastavenia súkromia



Obr. 3-3: 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 zaznamenávania udalostí. Na databázovej vrstve sú zaznamenané udalosti uchovávané v tabuľke

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

9 Bootstrap: http://getbootstrap.com/

`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`.

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.

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

4.2.2 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. 4-1 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. 4-1: 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 *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.

14 Kaminari: https://github.com/amatsuda/kaminari
15 Bootstrap: http://getbootstrap.com/

5 Šprint 3 – Hydralisk

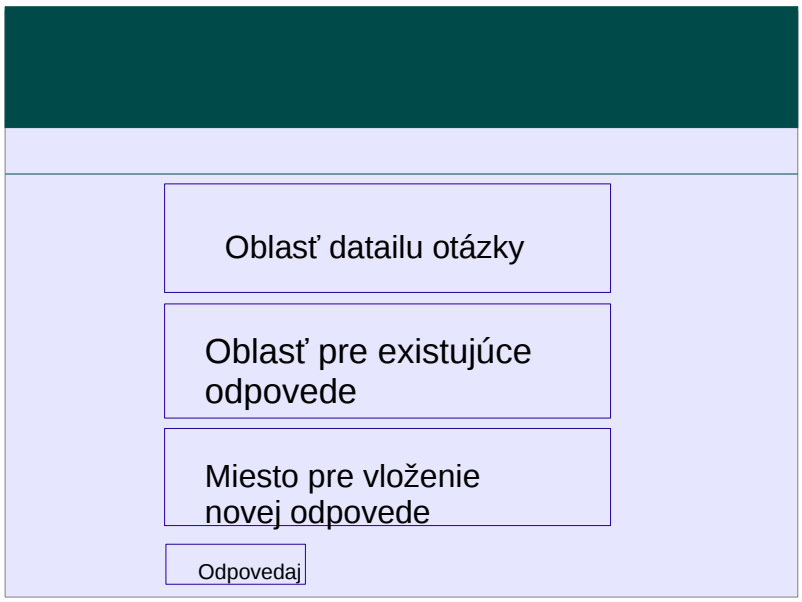
5.1 Vloženie novej odpovede

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. 5-1 je zobrazené rozhranie pre vloženie novej odpovede. Oblasť detailu otázky a oblasť pre existujúce odpovede neboli našou úlohou.



Obr. 5-1: 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 Šprint 4 – Infestor

Štvrtý šprint mal trvanie dva týždne a jeho cieľom bolo vytvoriť roly pre používateľov a rôzne vylepšenia pre odpovede ako napr. pridanie komentárov alebo označovanie odpovedí.

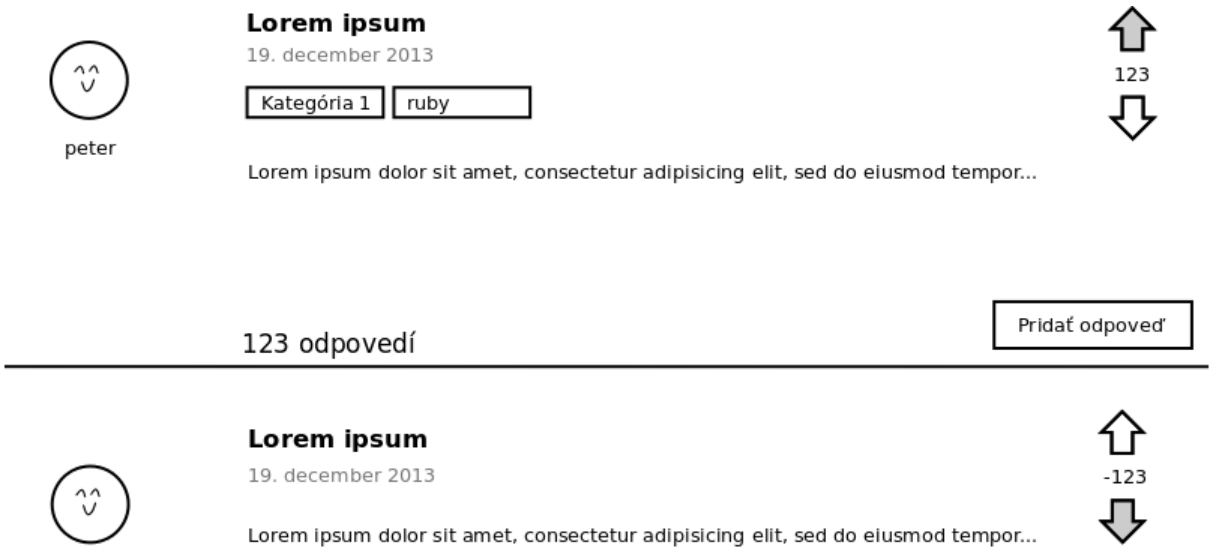
6.1 Hlasovať za otázku alebo odpoveď

Ako používateľ chcem pozitívne alebo negatívne hlasovať za otázku alebo odpoveď.

6.1.1 Analýza

Používateľ bude môcť hlasovať za otázku alebo odpoveď jedným hlasom. Výsledok hlasovania sa bude zobrazovať ako rozdiel medzi počtom kladných a záporných hlasov.

6.1.2 Návrh



Obrázok 6-1: Návrh rozhrania pre hlasovanie

Vedľa otázky alebo odpovede budú dve šípky slúžiace na hlasovanie. Šípka hore bude pridávať kladný hlas a šípka dole zase negatívny. Po kliknutí sa zmení farba šípky ako indikácia pripočítania hlasu. Kliknutie na už zadaný hlas tento hlas zruší.

Do dátového modelu sa pridá tabuľka `Votes` ktorá bude obsahovať hlasy používateľov. Bude obsahovať stĺpec `upvote` typu `boolean` určujúce či ide o kladný alebo záporný hlas.

6.1.3 Implementácia

Kedže hlasovanie je spoločné pre otázky aj odpovede tak je funkcionalita implementovaná v spoločnom `concern`. Hlasovanie na úrovni modelu je implementované v `Votable` a pre kontroler vo `Voting`. Samotné hlasovanie je implementované v metóde `toggle_vote_by!`. Ak daný používateľ ešte nehlasoval vytvorí nový záznam. Pokiaľ sa mení hlas z negatívneho na pozitívny alebo opačne len sa prehodí logická hodnota v poli `upvote`. Nakoniec ak používateľ odznačil svoje hlasovanie tak sa záznam odstráni.

6.1.4 Testovanie

Testujú sa štyri scenáre použitia:

- hlasovanie pri otázke ktorá nemá hlas
- zmena hlasu
- zrušenie hlasu
- pridanie hlasu k už existujúcemu hlasu

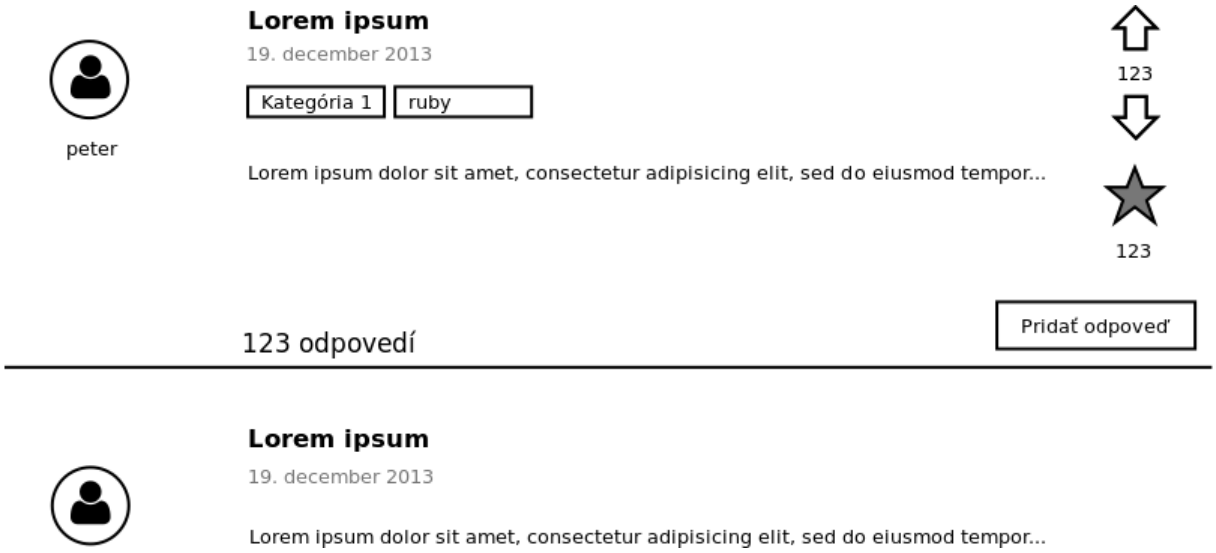
V každom scenári sa kontroluje či sa správne zaráta vykonaná akcia do celkového výsledku hlasovania.

6.2 Označiť otázku ako obľúbenú

Ako používateľ chcem označiť otázku ako obľúbenú, aby som sa k nej mohol neskôr vrátiť.

6.2.1 Analýza

Existujúce prístupy využívajú hviezdu ako grafický prvok pre obľúbené prvky systému. Rovnakú sémantiku sme sa rozhodli využiť aj my, pričom okrem symbolu hviezdy ponúkneme používateľom aj možnosť vidieť obľúbenosť otázky podľa počtu označení.



Obr. 6-2: Návrh rozhrania pre označenie otázky ako obľúbenej

6.2.2 Návrh

Tlačidlo so symbolom hviezdy pre označenie sme sa rozhodli umiestniť pod hlasovanie spolu s počtom označení.

6.2.3 Implementácia

Pre reprezentáciu obľúbených otázok v databáze sme vytvorili model `Favorite`. Tento model obsahuje referenciu na používateľa, ktorý otázku označil ako obľúbenú a referenciu na samotnú obľúbenú otázku.

Pohľad na otázku sme rozšírili o akciu `favor`. Akcia označenia otázky ako obľúbenej sa realizuje asynchrónne pomocou atribútu `data-remote`. Po dokončení akcie sa spustí spätné JavaScript volanie, ktoré nahradí element pre označenie otázky novým obsahom. Preto nie je potrebné, aby sa celá stránka generovala pri jednoduchej akcii ako označenie obľúbenej otázky. Pri každej otázke sa zobrazuje aj počet používateľov, ktorí ju označili ako obľúbenú.

6.2.4 Testovanie

Pre označenie otázky sme v modeli implementovali dve metódy – `toggle_favoring_by!` a `favored_by?`. Pre obe metódy sme vytvorili jednotkové testy pre model `Question`. Funkcionalitu označenia otázky sme realizovali pomocou akceptačného testu s využitím rozhrania Selenium[17].

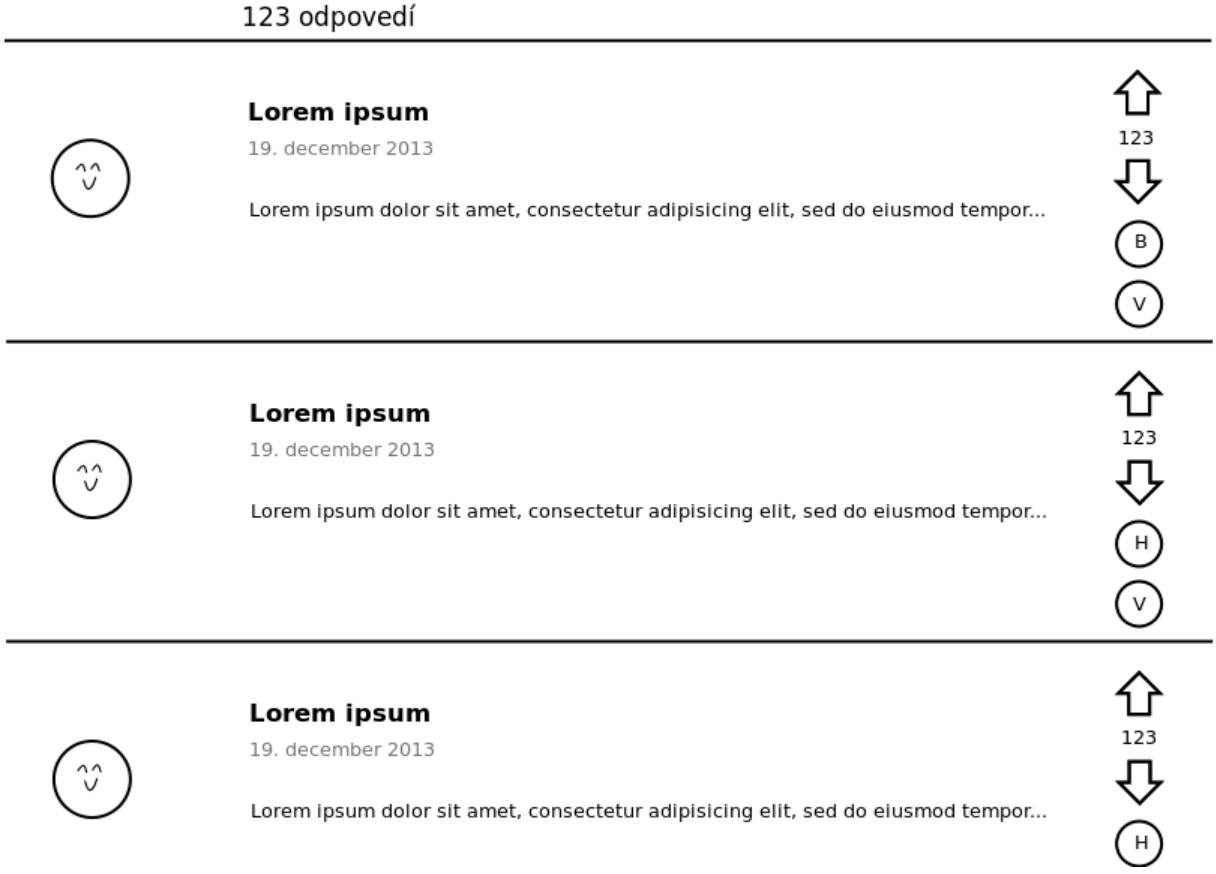
17 Selenium: https://github.com/jnicklas/capybara

6.3 Označiť odpoveď ako najlepšiu, pomocnú a overenú

Ako autor otázky chcem označiť najlepšiu odpoveď a rôzne pomocné odpovede. Ako učiteľ chcem označiť ľubovoľnú odpoveď ako mnou overenú. Ako používateľ chcem pri odpovediach k prehliadanej otázke vidieť najlepšiu, pomocné a overené odpovede, ak ich už niekto označil.

6.3.1 Analýza

Existujúce systémy poskytujú rôzne možnosti pre označenie odpovedí. Spravidla však ide o označenie najlepšej alebo pomocných odpovedí. V prípade edukačných systémov alebo systémov v ktorých vystupuje určitá autorita (učiteľ) má zmysel označovať odpovede ako overené (v zmysle správnosti) touto autoritou. Známe systémy používajú rôzne prvky pre označenie odpovedí. Pre označenie najlepšej odpovede je to zvyčajne zelená zaškrtávacia značka, pre pomocné napr. žltá žiarovka a pre overené biela zaškrtávacia značka na modrom podklade. Podobnú sémantiku sme sa rozhodli využiť aj my. Návrh rozhrania v pohľade na odpovede otázky je uvedený na Obr. 6-3



Obr. 6- 3: Návrh rozhrania pre označenie odpovedí.

6.3.2 Návrh

Tlačidlá pre označenie odpovede (najlepšej *B*, pomocnej *H* a overenej *V*) sme sa rozhodli umiestniť

pod hlasovanie v stĺpci pod seba, pričom tie sa zobrazia ako vykonateľné akcie v prípade, že má používateľ dostatočné práva na označenie odpovede, inak sa zobrazia iba ako ikony. Súčasne sa napríklad nepovolí (a ani nezobrazí) označenie odpovede ako pomocnej ak už bola označená ako najlepšia.

6.3.3 Implementácia

Pre reprezentáciu obľúbených odpovedí v databáze sme vytvorili modely `Labeling` a `Label`. `Label` reprezentuje samotnú značku a `Labeling` označenie, t.j. väzbu medzi odpoveďou, značkou a používateľom (autorom značky).

Riadenie odpovedí sme rozšírili o akciu `label`. Akcia označenia odpovede požaduje na vstupe jediný parameter `value`, ktorý obsahuje hodnotu značky. Samotná akcia sa realizuje asynchrónne pomocou atribútu `data-remote`. Po dokončení akcie sa spustí spätné JavaScript volanie, ktoré nahradí ovplyvnené elementy (časti odpovedí) novým obsahom. Preto nie je potrebné, aby sa celá stránka generovala pri tak jednoduchej akcii ako označenie odpovede. Pri každom označení odpovede sa vykoná aj príslušná autorizácia a preverí sa či má používateľ dostatok práv na to aby mohol akciu vykonať.

6.3.4 Testovanie

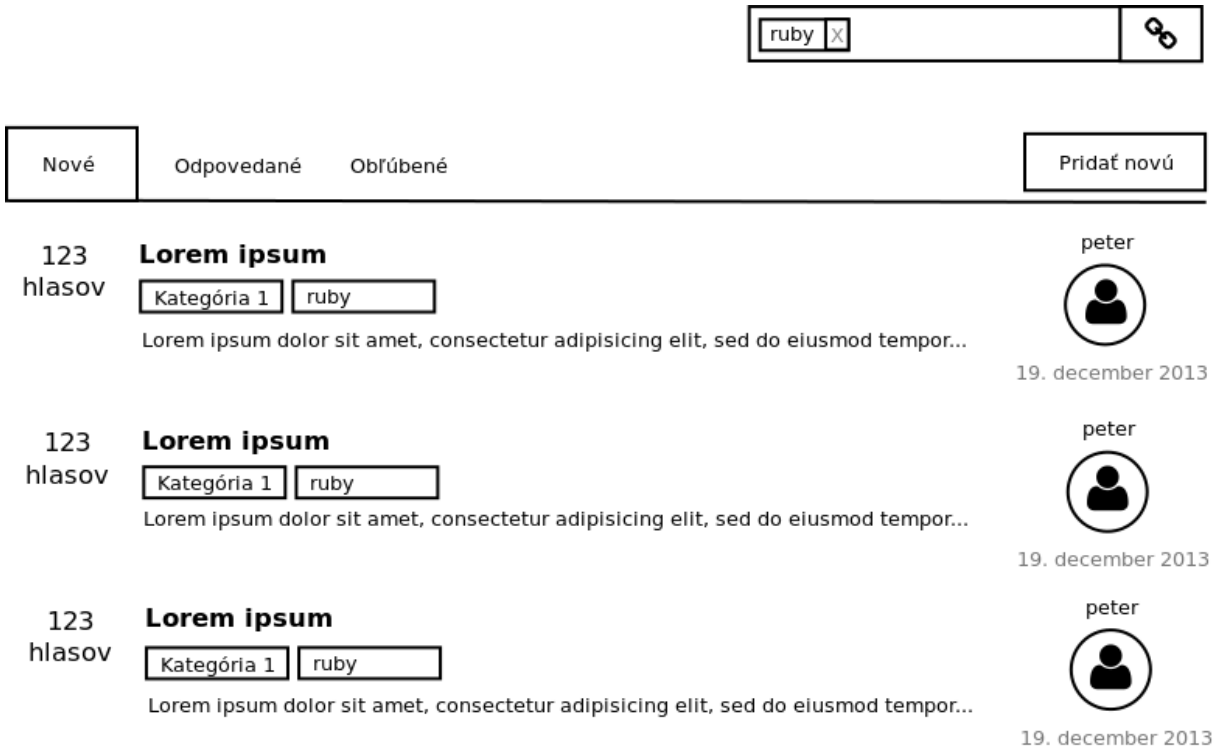
Pre označenie odpovede sme v modeli implementovali viaceré metódy, z ktorých spomenieme hlavne `toggle_labeling_by!`, `labeled_with` a `labeled_by?`. Pre spomenuté metódy sme vytvorili jednotkové testy v modeli `Answer`. Funkcionalitu označenia odpovede sme otestovali pomocou akceptačného testu, v ktorom sme pokryli všetky najdôležitejšie prípady aj s ohľadom na autorizáciu používateľa v danom kontexte.

6.4 Filtrovanie otázok podľa značiek

Ako používateľ, chcem filtrovať otázky podľa značiek (angl. tagov), aby som našiel otázky, ktoré ma zaujímajú.

6.4.1 Analýza

Pre podporu tejto funkcionality rozšírime pohľad na zoznam otázok o možnosť pridávania značiek ako filtra pre zoznam otázok. Filtru otázok vyhovujú len také otázky, ktoré obsahujú všetky značky špecifikované používateľom ako filter. Filtrovanie sa prispôsobuje aktuálne zvolenej záložke otázok.



Obr. 6-4: Návrh rozhrania pre filtrovanie otázok

6.4.2 Návrh

Identifikovali sme, že veľká časť funkcionality pre výber značky do filtra je podobná s pohľadom pre vytvorenie novej odpovede. Preto sme sa rozhodli pridať túto rovnaké pole do rozhrania zoznamu otázok. Filter pre rozhranie bude brať do úvahy len otázky pre zvolenú záložku. Po kliknutí na značku uvedenú v zozname otázok sa značka pridá do filtrovacieho poľa. Vedľa vyhľadávacieho poľa je tlačidlo pre prekopírovanie aktuálneho dopytu do schránky (angl. clipboard).

6.4.3 Implementácia

Funkcionalitu filtrovania sme podporili pomocou asynchrónnych AJAX volaní na server. Pre perzistenciu parametrov ako záložka, strana a aktuálne zvolené značky v rámci URL adresy sme použili funkcionalitu HTML5 pushState[18] pre prácu s históriou webového prehliadača. Pre automatické dopĺňanie značiek do filtrovacieho poľa sme použili knižnicu *select2*[19], ktorú sme použili a prispôsobili už v predchádzajúcom šprinte na implementáciu pridávania značiek pri vytváraní otázky. Funkcionalitu pre podporu pridania značky do filtra zo zoznamu otázok sme implementovali s využitím rozhrania knižnice *select2* pre dynamické pridávanie značiek.

18 HTML5 pushState: http://www.w3.org/TR/2012/WD-html5-20121025/single-page.html#dom-history-pushstate
19 Select2: http://ivaynberg.github.io/select2/

6.4.4 Testovanie

Filtrovanie sme otestovali pomocou rozhrania Selenium[20] pre interakciu s webovým prehliadačom. Vytvorili sme akceptačné testy pre viaceré prípady interakcie s filtrom pre značky. Prvý prípad berie do úvahy len filtrovanie podľa jednej značky. V rámci testu sme overili, či korektne fungujú aj asynchrónne stránkovanie otázok. Ostatné testy sa zamerali na filtrovanie podľa viacerých značiek a značiek zo zoznamu otázok, pričom sme kontrolovali aj korektné zobrazenie v prípade nenájdených výsledkov.

20 Selenium: https://github.com/jnicklas/capybara

7 Šprint 5 – Ultralisk

Posledný, piaty šprint zimného semestra, trval jeden týždeň a jeho náplňou bolo doladiť vzniknuté chyby, pridať možnosť filtrovania otázok, upraviť a doplniť niektoré funkcionality podľa požiadaviek, ktoré sme dostali od budúcich používateľov nášho systému.

7.1 Vloženie komentáru pre otázku a odpoveď

Ako používateľ chcem mať možnosť pridať komentár k otázke aj odpovedi, aby som bol schopný spýtať sa na detail, upozorniť na nedostatok alebo inak explicitne reagovať.

7.1.1 Analýza

Pre podporu tejto funkcionality dávame možnosť napísať komentár priamo pod otázku alebo odpoveď, ku ktorej sa má viazať. Na tomto mieste sa potom komentár aj zobrazuje, resp. sa tu nachádza zoznam komentárov zoradený chronologicky.

Obr. 7-1: Zobrazenie a vloženie komentáru

7.1.2 Návrh

Identifikovali sme, že model komentáru pre otázku aj odpoveď je identický, preto je preň možné použiť jednu tabuľku v databázovom modeli. Pre správne naviazanie konkrétneho komentáru k otázke alebo odpovedi sa preto použije polymorfická asociácia.

7.1.3 Implementácia

Pre vloženie komentáru sme vytvorili model `Comment`, ktorý obsahuje polymorfickú asociáciu `commentable`. Používateľ môže komentovať otázku samotnú, ale aj jednotlivé odpovede. Preto sme sa rozhodli na úrovni `routes` implementovať `concern`, ktorý nám umožní polymorfickú asociáciu pre otázky a odpovede. Na vytvorenie komentáru nám slúži `comment_controller`, v ktorom metóda `create` slúži na samotné vytvorenie a uloženie komentáru do databázy. Pomocou metódy `find_commentable` v `controller` zistíme pre aký typ objektu vytvárame komentár.

7.1.4 Testovanie

Pre vytvorenie komentáru sme vytvorili testy, v ktorých sme testovali úspešné a neúspešné vytvorenie komentára. Zobrazenie komentára sme otestovali na vlastných dátach. Sledovali sme zobrazenie jednotlivých komentárov, ich polohu a zarovnanie.

7.2 Zobrazenie počtu hlasov, odpovedí a zobrazení pri otázke

Ako používateľ chcem vedieť koľko má otázka hlasov, odpovedí a zobrazení.

7.2.1 Analýza

Počet hlasov je vyriešený v hlasovaní. Z neho sa použije už existujúca metóda. Taktiež počet odpovedí nám poskytuje dátový model. Jediná nová funkcionalita bude počet zobrazení. Zaznamenávať sa bude každé zobrazenie otázky. Zobrazovať ale bude len počet používateľov ktorý danú otázku videli.

7.2.2 Návrh

Zobrazenia sa bude zaznamenávať nová dátová entita `View`. Bude obsahovať len čas svojho vytvorenia a reláciu na používateľa a otázku. Kedže v budúcnosti sa môže vyskytnúť viacero entít ktoré budú zobraziteľné tak sa funkcionalita implementuje v samostatnom *concerne* `Viewable`. Tak bude možné v budúcnosti ľahko implementovať pre ďalšie entity.

7.2.3 Implementácia

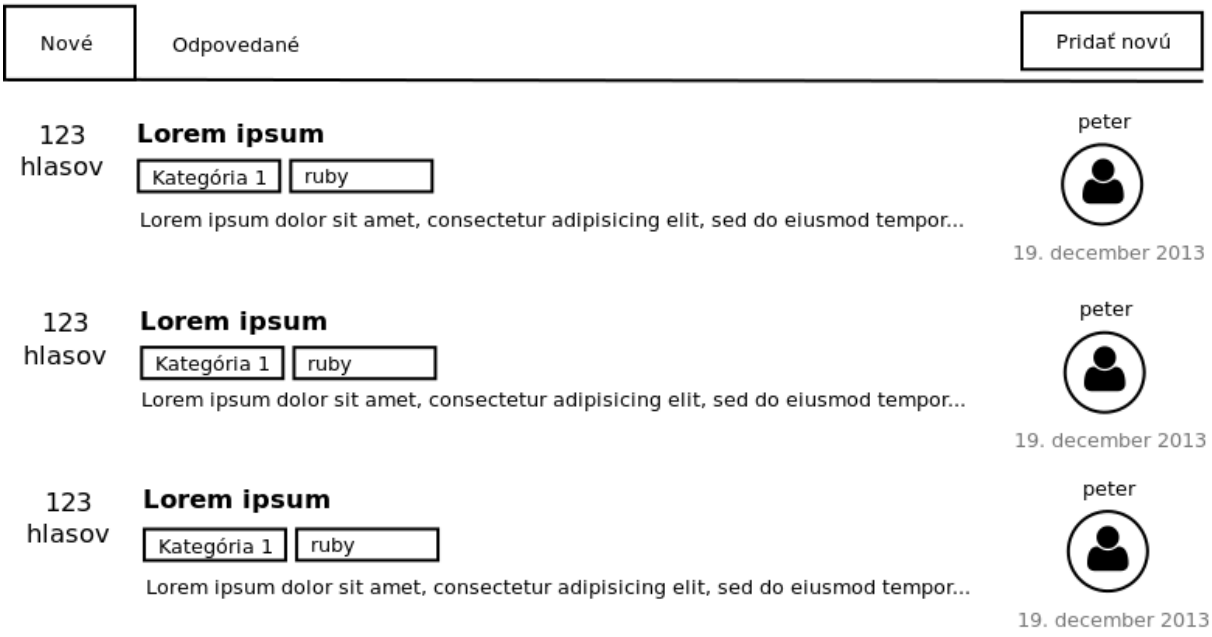
Zaznamenávanie zobrazenia otázky je vložené do metódy `show` v triede `QuestionController`. Počet zobrazení vracia metóda `total_views` z triedy `Viewable`.

7.3 Zobrazenie zodpovedaných otázok

Ako používateľ chcem vidieť všetky otázky, ktoré už majú odpoveď.

7.3.1 Analýza

Pohľad na zoznam otázok doplníme o navigáciu pomocou záložiek, ktoré budú obsahovať zvlášť nové a zvlášť odpovedané otázky, čo umožní používateľovi prehľadným spôsobom prechádzať cez otázky. Návrh môžeme vidieť na nasledovnom obrázku.



Obr. 7-2: Návrh zobrazenia pre odpovedané otázky

7.3.2 Návrh

Táto funkcionalita je doplnenie už existujúcej, ktorá vytvára zoznam otázok avšak zobrazuje zatiaľ len nové otázky. Implementovanie záložiek je pomocou *JavaScript.* Nasledovné vypisovanie zodpovedaných otázok prebieha pomocou sledovania v databáze, či je konkrétnej otázke priradená odpoveď označená ako najlepšia.

7.3.3 Implementácia

Implementácia je veľmi podobná ako pri nových otázkach. Využívajú sa rovnaké postupy. Doplnila sa len o prepojenie tabuliek a na základe existencie výskytu v tabuľke s odpoveďami sa otázka zaradí do zodpovedaných.

7.3.4 Testovanie

Testovanie rovnaké ako pri používateľskom príbehu zobrazenie nových otázok pomocou akceptačných testov. Kontroluje sa výpis zodpovedaných kde sa hľadá nadpis a kategória otázky.

7.4 Kategória ako pomenovaná množina značiek

Model `Category` bol doplnený o atribút `tags`. Tento atribút obsahuje pole textových reťazcov určujúcich značky pre kategóriu, ktoré sú pri vytváraní otázky s touto kategóriou pridané k ostatným značkám otázky. Túto funkcionalitu zabezpečuje metóda `add_tags` doplnená do modelu `Question`, ktorá je zavolaná pred vytvorením otázky.

Testy pre model `Question` boli doplnené o testy pre metódu `add_tags`.

8 Celkový pohľad po prvý kontrolný bod

Za 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.

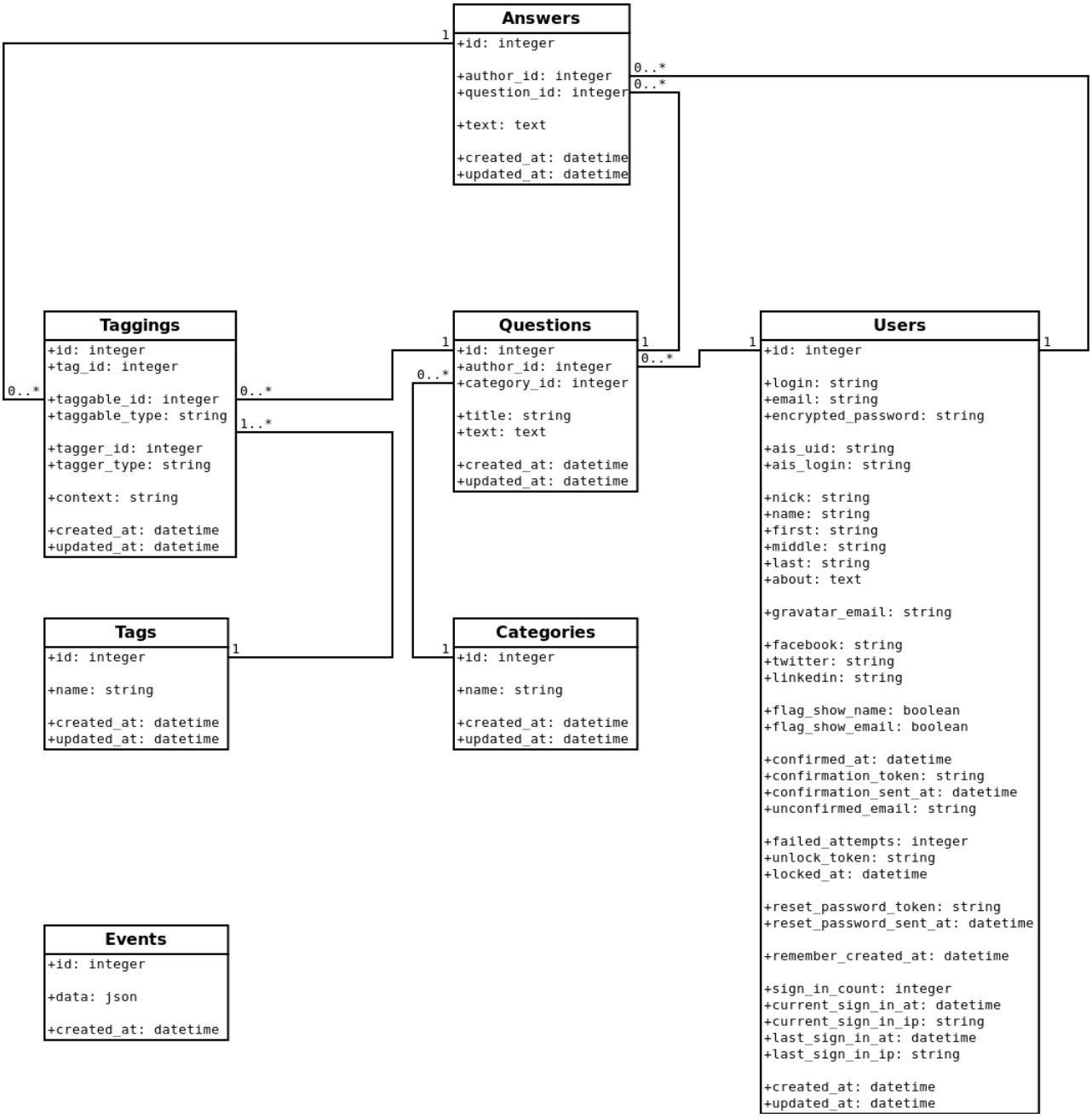
8.1 Architektúra

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.

8.2 Dátový model

Na Obr. 8-1 je znázornený vytvorený dátový model, ktorý obsahuje zatiaľ všetky použité entity.



Obr. 8-1: Dátový model

9 Celkový pohľad po druhý kontrolný bod

Od prvého kontrolného bodu sme do webového komunitného systému otázok a odpovedí doplnili nasledovnú funkcionalitu:

- Vkladanie komentárov pre otázku – Používateľ môže okrem odpovede na otázku pridávať aj komentár k odpovedi.

- Hlasovanie – Pre každú otázku aj odpoveď je možnosť hlasovať pre každého používateľa a tým vyzdvihnúť odpovede alebo otázky.

- Značenia odpovedí – Odpovede môžu byť označené viacerými symbolmi. Prvý je označenie odpovede ako najlepšia. Toto značenie pridáva používateľ, ktorý otázku vytvoril. Takisto pridáva označenie pomocnej odpovede.

- Rola používateľa – V našom systéme sa predpokladá viac druhov používateľov. Zatiaľ identifikovaní sú študent a učiteľ. Každý z nich má iné práva.

- Obľúbené otázky – V systéme je možné pre každého používateľa vytvoriť si vlastný zoznam obľúbených otázok, ku ktorým sa vie jednoducho a rýchlo dostať.

- Štatistiky otázok – V zozname otázok sa pri každej otázke zobrazuje aj počet videní otázky, počet odpovedí a počet komentárov.

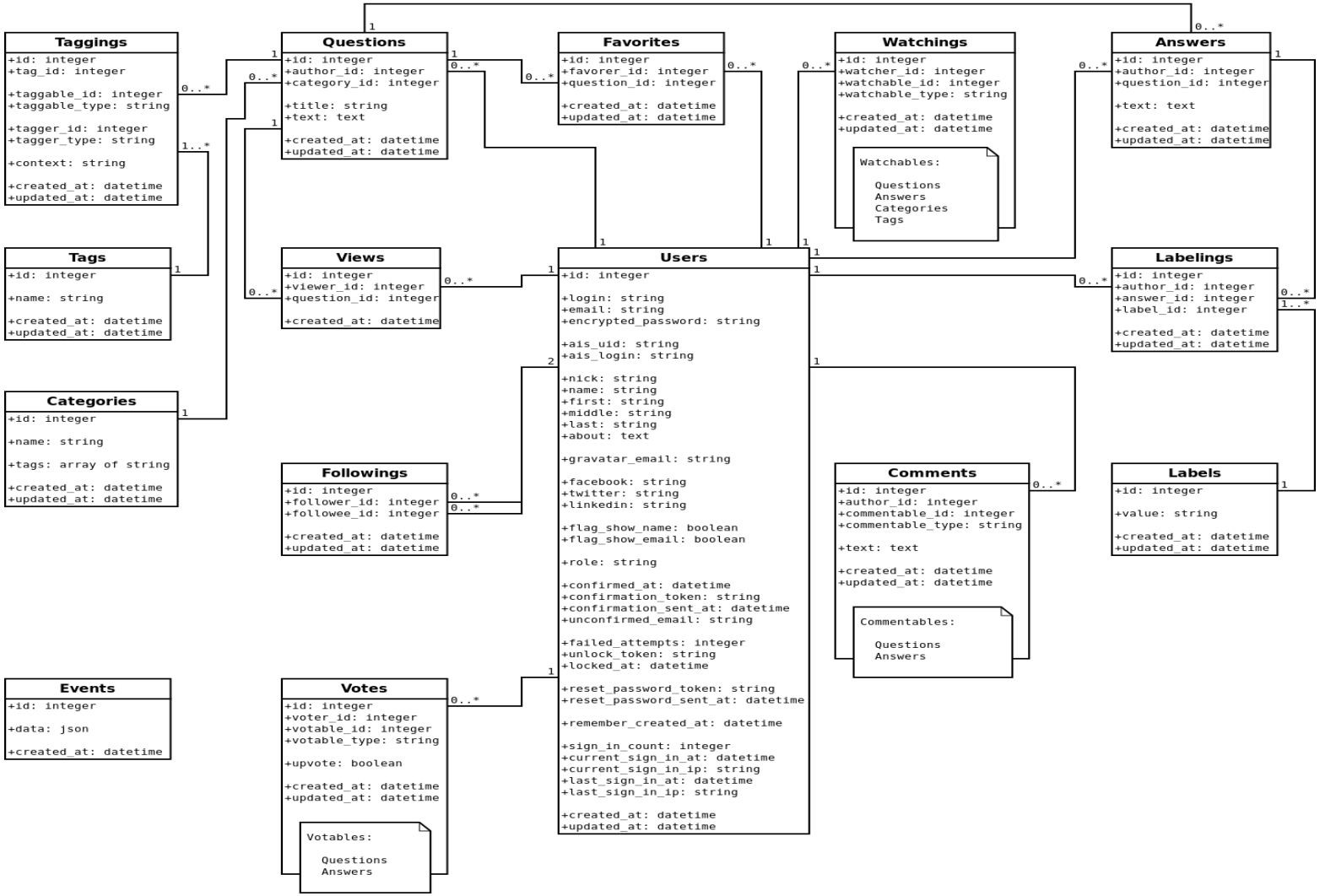
- Filtre – Pre rýchle vyhľadávanie otázok slúži filter, ktorý sa nachádza v zozname otázok. Kliknutím na značku alebo dopísaním značky priamo do poľa sa automaticky zobrazia len otázky vyhovujúce kritériám.

9.1 Architektúra

Architektúra systému nezaznamenala žiadne výrazné zmeny oproti prvému kontrolnému bodu. Na druhej strane však došlo k výraznému rozšíreniu dátového modelu dôsledkom implementácie novej funkcionality systému.

Z pohľadu modularizácie samotnej aplikácie na úrovni zdrojového kódu došlo k značnému refaktorovaniu najmä tried kontrolérov a modelov, resp. pomocníkov a pohľadov. Spoločná funkcionalita bola vytiahnutá do tzv. *concerns* a *scopes*, resp. pomocných metód a *partial views*. Tento architektonický zásah významne prispel k aplikovaniu DRY princípu (angl. Do not Repeat Yourself, t.j. odstránení duplicitného alebo príliš podobného zdrojového kódu), znovupoužiteľnosti a lepšej prehľadnosti zdrojového kódu.

9.2 Dátový model



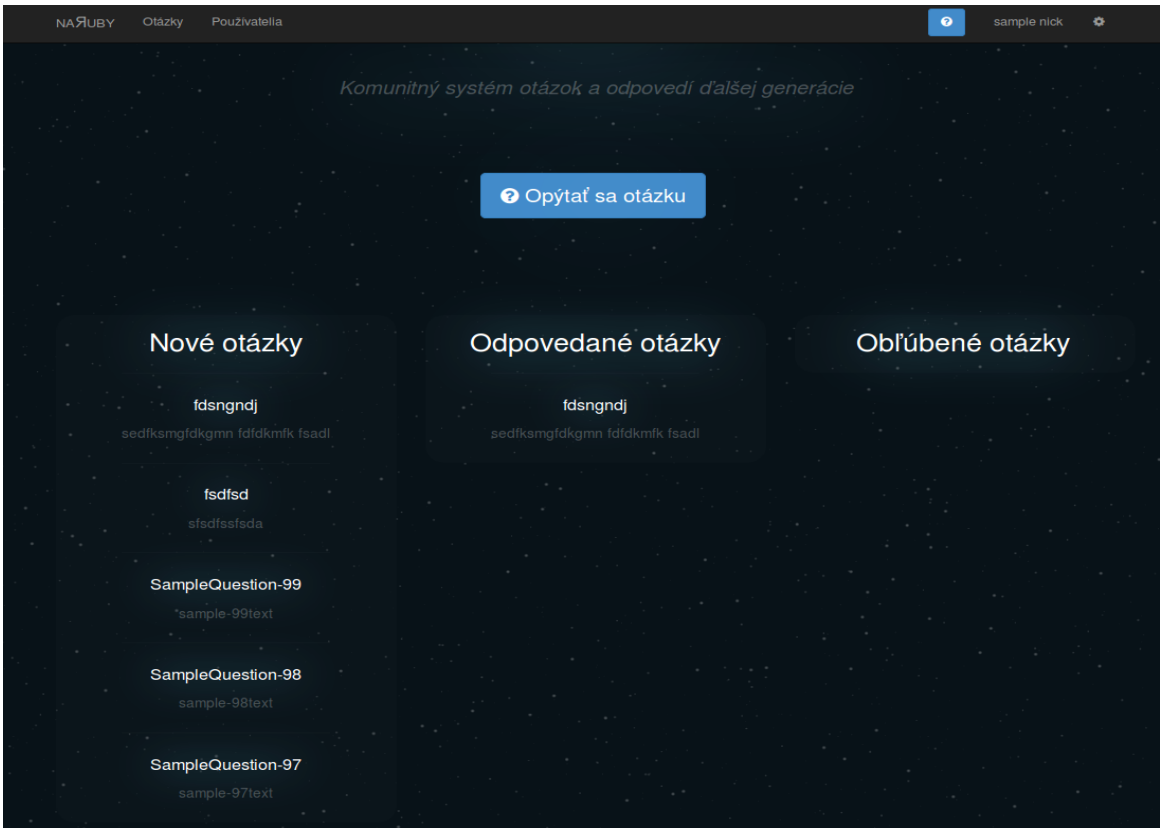
Obr. 9-1: Dátový model

A Používateľská príručka

Náš projekt má za úlohu uľahčiť komunikáciu študentov medzi sebou a medzi učiteľmi a žiakmi. Systém je vytvorený ako komunitný systém otázok a odpovedí, čiže nie je to fórum ale systém fungujúci na spôsobe otázok a odpovedí. Systém je webová aplikácia, takže jeho spustenie je zadaním stránky vo webovom prehliadači. Následne je nutné sa registrovať, alebo sa prihlásiť pomocou AIS prihlasovacích údajov.

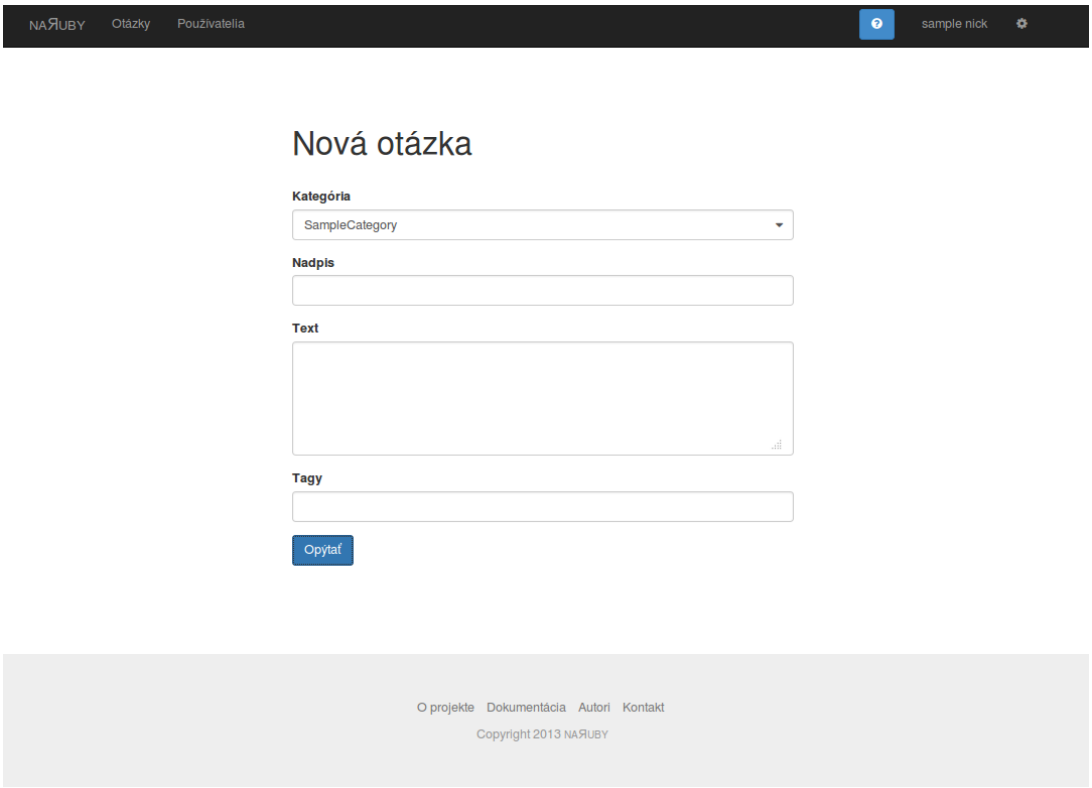
2 Pridanie novej otázky

Pred každým úkonom ktorý chce používateľ vykonať v systém je potrebné aby bol prihlásený. Keď sa prihlási do systému, tak sa mu zobrazí úvodná strana, ktorá je vyobrazená na Obr. A-1



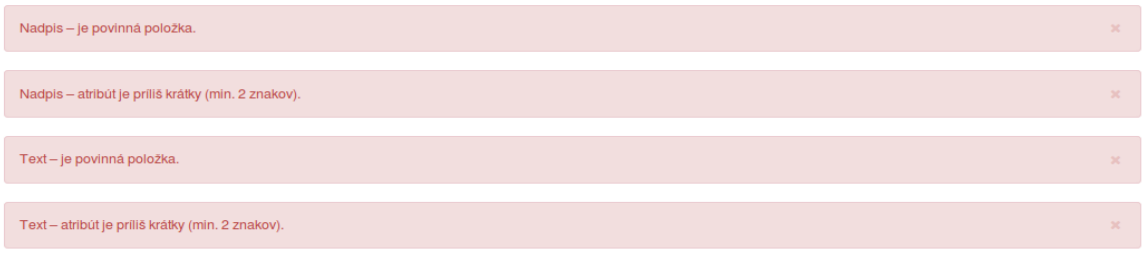
Obr. A-1: Úvodná obrazovka

Na tejto obrazovke vidno prehľad niektorých vybraných otázok. Pre pridanie otázky používateľ klikne na tlačidlo opýtať sa otázku. Zobrazí sa mu obrazovka pre pridanie novej otázky, ktorá je zobrazená na Obr. A-2.



Obr. A-2: Zobrazenie pre položenie novej otázky

Aby otázka bola vložená je nutné aby mala vyplnené všetky textové polia. To sú kategória, nadpis otázky, text otázky a značky, ktorých môže byť viac. Značky môže používateľ písať vlastné alebo použiť už existujúce. Ak nebudú vyplnené vypíšu sa niektoré z hlášok z Obr. A-3.

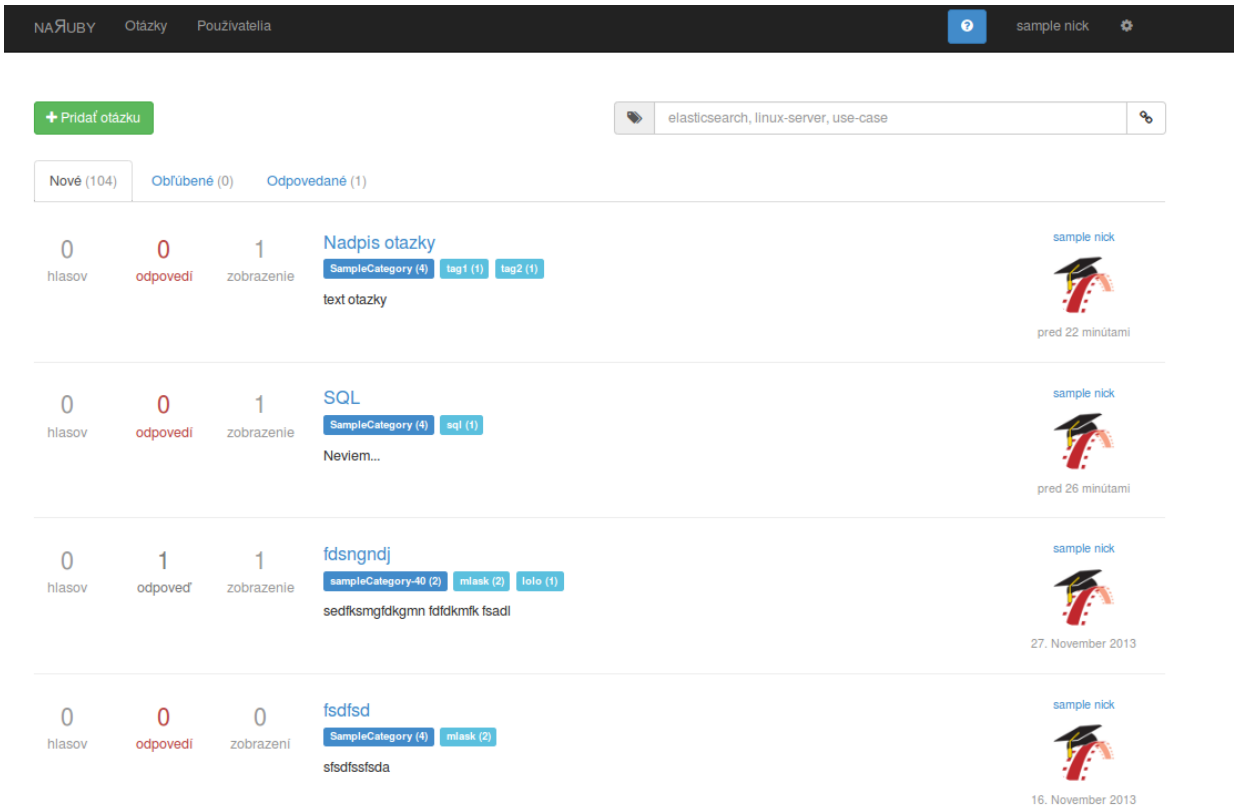


Obr. A-3: Možné chybové hlášky

Po správnom vyplnení textových polí a odoslaní otázky sa presmeruje stránka na zobrazenie otázky, ktorá obsahuje všetky informácie ktoré sme vyplnili. Obsahuje aj čas vzniku otázky a autora otázky.

3 Pridanie odpovede k otázke

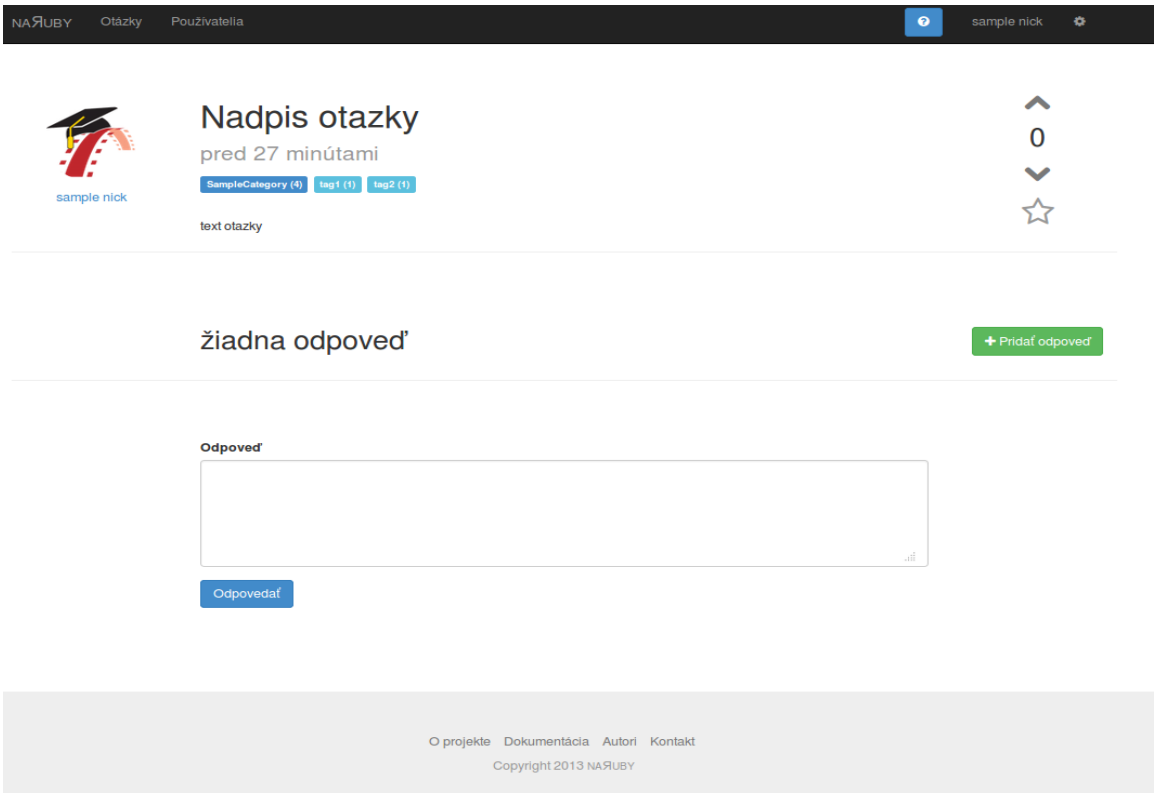
Pre pridanie odpovede k otázke si musí používateľ ako prvé vybrať otázku na ktorú chce odpovedať. Tú si vyberie v zozname otázok, ktorý je na Obr. A-4.



Obr. A-4: Zobrazenie zoznamu otázok

V zozname otázok má na výber aké otázky si chce nechať zobraziť. Môže si vybrať medzi novými, zodpovedanými alebo obľúbenými, ktoré si konkrétny používateľ už vybral.

Pri každej otázke je zobrazený počet hlasov, odpovedí a počet zobrazení otázky. Pre pridanie odpovede si používateľ zvolí otázku a klikne na ňu. Následne sa zobrazí zobrazenie pre pridanie odpovede ako vidíme na Obr. A-5.

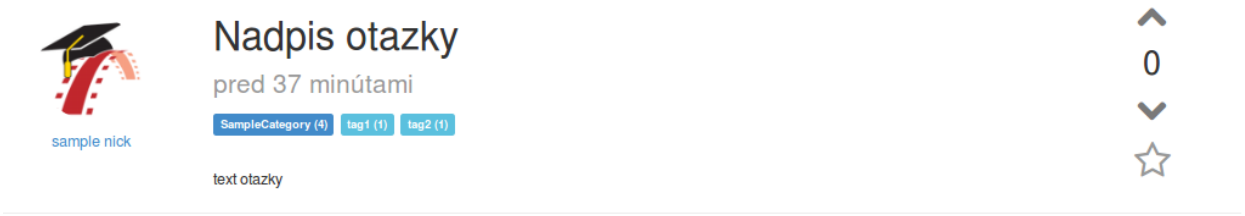


Obr. A-5: Zobrazenie otázky a odpovedí

Pri zobrazení otázky sú zobrazené aj odpovede už pridané. Novú odpoveď pridá používateľ vpísaním odpovede do textového a následným kliknutím na tlačidlo odpovedať. Po pridaní odpovede sa stránka zaktualizuje a zobrazí našu pridanú.

4 Hlasovanie a zvolenie najlepšej odpovede

Pre hlasovanie je nutné najprv vybrať otázku podobne ako pri odpovedaní. Pri zobrazení otázky s odpoveďami je zobrazené hlasovanie reprezentované šipkami hore a dole. Šípka hore reprezentuje pozitívny hlas a šípka dole negatívny hlas. Svoje rozhodnutie je možné zmeniť. Hlasovať má možnosť používateľ za otázky aj za odpovede. Zobrazenie hlasovania je ukázané na Obr. A-6.



Obr. A-6: Zobrazenie hlasovania

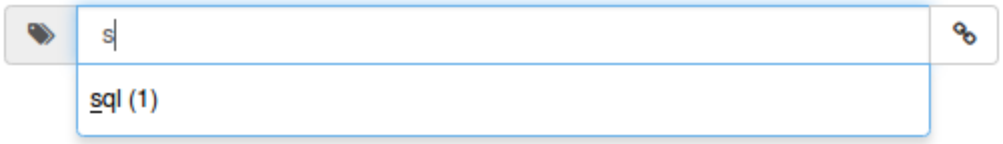
Ak sa na svoju otázku pozerá autor tak ma možnosť pridať k odpovediam označenia ako najlepšia odpoveď a pomocná odpoveď. Najlepšia odpoveď je znázornená zelenou „fajkou“ a pomocna pomocou symbolu žiarovky, pričom pomocných odpovedí môže byť viac. Odpoveď, ktorá je označená ako najlepšia už nemôže byť označená ako pomocná. Zobrazenie je ukázané na Obr. A-7.

Obr. A-7: Označenie odpovede

Po kliknutí na niektoré z označení sa daná voľba farebne zvýrazní a zobrazí sa všetkým používateľom.

5 Vyhľadávanie v otázkach

Vyhľadávanie prebieha pomocou značiek, ktoré sa písali pre jednotlivé otázky. Používateľ zadá do textového poľa značku, ktorú mu systém automaticky doplní ako je zobrazené na Obr. A-8.



Obr. A-8: Filter podla tagov

Pri doplnení názvu značky sa zobrazí aj počet výskytov zadanej značky. Po vybraní značiek sa používateľovi zobrazia otázky poľa zadaných značiek.