Šprint číslo 6

Úloha #813 - Zobrazovanie strán zmlúv

Stanislav Valachovič

Popis úlohy

Úlohou bolo zrýchliť načítavanie stránky zmluvy - nebudú sa načítavať všetky obrázky, ale až po prechode na ne sa načítajú. Pri scrollovaní sa bude meniť url - anchor, ale aby to nespôsobovalo vkladanie nových stavov stránky (čiže keď pôjdem späť aby som sa nepohyboval po anchoroch).

Analýza problému

Pri analýze som sa zameral na url, študoval som spôsoby ako by sa dal vkladať anchor do url bez vytvorenia nového stavu stránky. Pre neskoršie zobrazovanie obrázkov som si len pozrel niekoľko možných variant riešenia.

Návrh riešenia

Pre načítavanie obrázkov budem musieť zisťovať, či je obrázok viditeľný (či som na ňom), ak áno tak sa načíta. V prípade url bude potrebné použiť vlastnosti html5 a túto vlastnosť zmeny linky budú podporovať iba browseri s podporou html5.

Opis implementácie

Pri obrázku sa jeho cesta nastaví na obrázok o rozmere 1x1px a skutočná cesta sa uloží do atribútu data-original. Po prechode na obrázok sa skutočná cesta zamení a dôjde tak k načítaniu obrázku. Zisťovanie či som na obrázku a samotná výmena atribútov sa robí pomocou js. Pre url, ak browser podporuje html5, tak sa používa funkcia replaceState pri vkladaní anchor aktuálneho obrázku, čím sa nevytvorí nový stav, ale nahradí sa existujúci.

Testovanie

Testovanie bolo vykonávané vizuálne - teda či sa obrázok po prechode naň načíta, v prípade url, či sa zmení anchor po prechode na obrázok.

Úloha #778 - Nová branch pre webstránku samotnú

Filip Lörinc

Popis úlohy

Podstatou tejto úlohy bolo, aby bolo možné sledovať jednotlivé aktualizácie na tímovú webstránku tak, že bude pre ňu existovať separátny depozirtár v gitbuse. Jednotlivé úpravy na webstránky sa budú pridávať vo forme commitov. Súčasťou úlohy bude aj vymyslieť skript, ktorý po commite následne skopíruje obsah repozitára (webstránky) na školský server, kde je nominálne tímová webstránka umiestnená.

Analýza problému
Bolo nutné vymyslieť spôsob, ako po commite aktuálnej verzie webstránky v repozitári ju možno skopírovať na školský server. Existuje možnosť spustenia skriptu po commite z gitbusu avšak tento skript by musel byť umiestnený na serveri, kde beží priamo gitbus. Kedže nám toto nebolo umožnené, museli sme vymyslieť iný spôsob. Ten spočíval v klasickom klonovaní obsahu repozitára a následné skopírovanie na vzdialený stroj, kde je uložená webstránka v periodických intervaloch.

Návrh riešenia
Riešenie spočívalo v použitíi školského serveru relax, kde každú hodinu bude bežať skript, ktorý bude klonovať obsah repozitára (sťahovať webstránku) a následne ju celú kopírovať na server labs2 kde je uložená verejná prezentácia nášho tímu. Hodinový interval bol použitý preto, aby server zbytočne nesťahoval a neposielal veľa dát, kedže je to zdieľaný stroj. Hodina a menej je aj dostatočný čas na to aby, tento update webstránky bol považovaný za okamžitý, samozrejme v rámci kompromisu.

Opis implementácie
Kedže sme potrebovali, aby spojenie prebiehalo z jedného serveru na dva ďalsie, ako prvé sme si skopírovali privátne a verejné kľúče na dané zariadenia. Následne sme vytvorili na serveri relax skript, ktorý vytvoril dočasný adresár kam sa pomocou príkazu ‘git clone’ stiahol obsah repozitára. Následne sa ten obsah skopíroval pocou príkazu rsync na server labs2. Následne sa obsah dočasného adresára a adresár samotný vymazal. Periodické opakovanie tohto skriptu bolo zabezpečené pomocou príkazu crontab.

Testovanie
Testovanie prebehlo pokusným commitom, ktorý sa následne prejavil na aktuálnosťou zavesenej webstránky.

Úloha #819 - Aktualizácia zmlúv

Martin Molnár

Popis úlohy
Zistiť možnosti aktualizácie zmlúv na crz.gov. Zistiť ako sa pridávajú dodatky, ako sa vzniká doplnená zmluva. Navrhnúť možnú implementáciu párovania zmlúv na dodatky.

Analýza problému
Na stránke Crz.gov sa zmluvy pridávajú v pravidelných intervaloch(cca každých 20 min). Najskôr sa pridávajú dokumenty(zmluvy a dodatky) sekvenčne. Následke sa v nepravidelných a nedeterministických intervaloch pridá doplnená zmluva, ktorá napáruje zmluvy na dodatky. Táto doplnená zmluva má rovnaké id ako pôvodná napárovaná zmluva. Dodatok má tiež rovnaké id

ako pôvodný dodatok.
Je potrebné rozlišovať medzi normálnym dokumentom a doplnenou zmluvou.

Návrh riešenia

Pri parsovaní centrálneho registra sa normálne dokumenty budú ukladať sekvenčne. V prípade ak príde doplnená zmluva, tak sa pozrú jej prílohy a pre každú prílohu sa nájde dokument s príslučným id. Tento dokument sa následne vymaže z db a tým sa zabezpečí konzistencia.

Opis implementácie

Algoritmus aktualizácie je doplnený do metódy create_from_metadata. Pre rozlíšenie doplnenej zmluvy je nutné dorobiť a dopĺňať atribút doplnena. Na základe tohto atribútu sa vykoná akcia pre aktualizáciu zmluvy.

Testovanie

Testovanie bolo vykonané empiricky. Našla sa stránka na centrálnom registri, kde je zmluva v stave doplnená. Následne sa spustí stahovanie. Po sťahovaní by sa mali napárovať dodatky na zmluvy a predošlé samostatné dokumenty by sa mali vymazať.

Úloha #818 - Nový parser na centrálny register zmlúv

Michal Kundrát

Popis úlohy

Úlohou bolo vytvoriť parser, ktorý bude vedieť získať potrebné údaje o zmluvách zo stránok nového centrálneho registra zmlúv, na ktorom sa majú uskladňovať všetky nové zmluvy.

Analýza problému

Bolo potrebné preskúmať stránku centrálneho registra zmlúv a navrhnúť vhodný spôsob, akým sa bude táto stránka parsovať, keďže stránka má pomerne zložitú štruktúru. Hlavná stránka je rozdelená stránkovaním, na každej stránke je zoznam zmlúv, z ktorých každá má vlastnú podstránku, pričom aj jednotlivé dodatky k zmluvám majú tiež vlastnú podstránku.

Návrh riešenia

Pre každú úroveň stránky centrálneho registra zmlúv bude vytvorená samostatná metóda, ktorá ju dokáže zpasrovať a následné vráti objekt s potrebnými údajmi.

Opis implementácie

Ako knižnicu pre prácu s xml dokumentom som si zvolil knižnicu Nokogiri. Parser obsahuje metódy pre parsovanie každej jednej z jednotlivých úrovní stránky Crz.sk a sú to: parsePage, parseSubpage, parseDodatokSubpage. Každá z nich vráti objekt naplnený príslušnými datami.

Testovanie

Testy boli robené pomocou unit testu RSpec. Test porovnáva, či vrátený objekt naozaj obsahuje také isté údaje, ako sa nachádzajú na vstupnom dokumente.

Úloha #815 - Server: vypýtať, dostať, nainštalovať + rozbehať aplikáciu

Matej Maruš

Popis úlohy

Úloha obsahuje celý proces získania respektíve vybavenia virtuálneho stroja , inštalácia operačného systému, ktorý by spĺňal naše požiadavky a spojazdniť tímový projekt so všetkými potrebnými aplikáciami ktoré využíva.

Analýza problému

Na základe analýzy tímového projektu sme sa rozhodli o využití nasledujúcich hardvérových prostriedkov. Konkrétne aplikácia vyžaduje veľký výpočtový výkon pre vytváranie obrázkov z pdf súborov. No najväčšia záťaž predstavuje OCR, ktorý vytvára čistý text z pdf súborov. Preto je potrebné požiadať o aspoň 3 CPU jadra, ktoré by mali dostatočne spĺňať naše potreby. Tímový projekt využíva asynchrónne spracovanie, ktoré je potrebné na využitie všetkých jadier. Pevný disk je najlepšie rozdeliť na 2 fyzické partície, na ktorých sa nachádzajú sťahované dáta, ten by mal obsahovať dostatočnú kapacitu. Ďalej je potrebne na ďalšom fyzickom disku vytvoriť logické disky pre usporiadanie jednotlivých adresárov do štruktúry. Na základe internetových odporúčaní sme sa rozhodli pre Ext4 file system. Je dostatočne vyladený a bezproblémový. Veľkosť RAM palete sme zvolili na 2 GB.
Operačný systém bol vybraný Debian v najvyššej verzii. Jeho repozitáre obsahujú všetky potrebné externé aplikácie, ktoré využívame.

Webový server je najlepšie zvoliť apache. Keďže je široko používaný a bezproblémovo funguje s Ruby On Rails aplikáciami.

Návrh riešenia

Na základe analýzy splnenia úlohy je potrebne vykonať nasledujúce kroky:

- Spísať požiadavky, ktoré sme zanalyzovali a poslať kompetentným osobám
- Získať prístup k virtuálnemu stroju.
- Prejsť celkovou inštaláciu s konfiguráciou, ktorú sme zanalyzovali.
- Nainštalovať nevyhnutné externé aplikácie pre správny chod tímového projektu
- Zabezpečiť základnú bezpečnosť proti prienikom cudzích osôb.
- Deploynuť aplikáciu na server a sprístupniť ho verejnosti.

Opis implementácie

Komunikácia s kompetentnými osobami bola dosť problémová a preto sa realizácia úlohy presunula do častí ďalšieho šprintu. Inštalácia OS Debian prebehla bez problémov, verzia bola Zvolena Debian 6. Niektoré aplikácie ale neboli v najvyššej verziu, ktoré potrebujeme pre našu aplikácie preto boli využite aj ďalšie repozitáre, nie len oficiálne.

Pre bezpečnosť sme nastavili Firewall, ktorý je otvorený len pre port 80 a 22. Následne bezpečnosť bola inštalácia Ruby on Rails inštalovaná pod bežným používateľom, ktorý bol inštalovaný cez RVM systém.

Webový server bol nakoniec zvolený Apache a nakonfigurovaný pre Ruby On Rails.

Úloha #814 - Registrácia používateľov + Facebook login

Matej Maruš

Popis úlohy

Registrácia používateľov, ktorí budú chcieť komentovať alebo pridávať tagy na našu stránku. Umožnenie prihlásenie za pomoci Facebook Loginu bez vypĺňania informácii o používateľovi.

Analýza problému

Pre celý proces prihlasovania a spravovania používateľov sme na základe informácii z príručiek k Ruby On Rails sme zvolili gem Devise (https://github.com/plataformatec/devise). Ten v najvyššej verzii podporuje aj modul pre prihlasovanie sa s facebook. Pre potreby prihlasovanie sa s facebook loginom je potrebné si vytvoriť aplikácie na developers.facebook.com. Veľké množstvo informácii o používateľoch nepotrebuje uchovávať a preto sa snažíme o minimum údajov.
Je potrebné získať aj smtp server pre korektne posielanie emailovej autentifikácie používateľa. To je spojené s úlohou #822.

Návrh riešenia

- Na základe analýzy navrhujeme vykonať nasledujúce kroky pre splnenie požiadaviek úlohy:
- Pridať gem Devise do projektu a následne pomocou bundle ho nainštalovať.
- Vygenerovať súbory podľa návodu ktoré sa nachádza na https://github.com/plataformatec/devise
- Vytvoriť aplikáciu na https://developers.facebook.com/
- Nastaviť SMTP pre posielanie potvrdzovacích emailov, pre korektne prihlásenie sa do našej webovej aplikácie.
- Ukladať len email a používateľské meno a priezvisko. Pre potreby identifikácie pridaného komentáru.

Opis implementácie

Podľa návrhu sme inštalovali gem Devise a premigrovali databázu, keďže boli vytvorené ďalšie tabuľky pre ukladanie informácii o používateľoch.
Taktiež konfigurácia triedy User, ktorá je jednoduchá. Pridáva sa len symboly, ktoré funkcionality chceme pridať. To môžete vidieť na nasledujúcej časti kódu

```
class User < ActiveRecord::Base
       devise :database_authenticatable, :confirmable, :recoverable, :rememberable, :tr
```

```
ackable, :validatable
  end
```

Taktiež konfigurácia pre použitie s facebook loginom si vyžadovala vyplnenie tajného kódu a mena z facebook aplikácie, ktorú sme vložili do */config/initializers/devise.rb*

config.omniauth :facebook, "163636693691511ß", "15ae8707b23f2270c8013f9e0ea5f23d"

Úloha #820 - Vymyslieť názov domény

Tomáš Háber

Popis úlohy

Vymyslieť názov domény.

Analýza problému

Návrh riešenia

Opis implementácie

Na základe voľných domén a atraktívnych som vybral niekoľko kandidátov. Hlasovaním bola zvolená doména nasedane.sk. Vhodnosť môjho návrhu potvrdzuje aj vznik projektu Aliancie Fair Play s veľmi podobným názvom a to “Z našich daní”.

Testovanie

Úloha #817 - Úvodna tabuľka so zoznamom zmlúv

Tomáš Háber

Popis úlohy

Zlepšiť čitateľnosť hlavnej stránky.

Analýza problému

Rozhodol som sa inšpirovať centrálnym registrom zmlúv a napodobnil usporadianie tabuľky so zmluvami.

Návrh riešenia

Opis implementácie

Upravil som HTML kód tak, aby bola stránka lepšie čitateľná.

Testovanie

Úloha #823 - Upraviť widget

Miroslav Polgár

Popis úlohy
Upraviť widget tak, aby bol vytvorený fork na repozitár originál widgetu v githube. Potom aplikovať zmeny, ktoré sú v našom widgete a vytvoriť submodule.

Analýza problému
Pri analýze som zistil, že bude potrebné tiež zmeniť view page, kde sa widget nachádza, pretože boli zmenené cesty (k submodulu).

Opis implementácie
Vytvoril som fork, aplikoval som zmeny, push do nového repozitára. Vymazal som starý adresár s widgetom z nášho gitu. Potom som vytvoril submodule a inicializoval ho. Upravil som stránku s widgetom a tiež json, ktorý je používaný widgetom.

Testovanie
Zobrazenie widgetu so zmluvou.

Šprint číslo 7

Úloha #825 - Štatistika / aktivita používateľov

Stanislav Valachovič

Popis úlohy
Pre používateľov sa vytvorí stránka, kde sa bude zobrazovať ich štatistika (dátum registrácie, posledné prihlásenie, najviac komentované zmluvy, 3 najlepšie a najhoršie komentáre a všetky jeho komentáre).

Analýza problému
Pri analýze som sa zameral na nájdenie/naštudovanie spôsobu ako stránkovať komentáre, aby sa nezobrazovali všetky, ale napr. po 10 na stránku.

Návrh riešenia
Statické údaje o používateľovi ako dátum registrácie a čas posledného prihlásenia sa jednoducho budú iba vypisovať z db, pri ostatných ako najviac komentované zmluvy a pod budú potrebné jednoduché selecty. Pre zobrazovanie týchto informácii bude potrebné vytvoriť nový kontroller.

Opis implementácie
Pre používateľov bol vytvorený UsersController a metóda show, pomocou ktorej sa

zobrazujú potrebné informácie o používateľovi. Pre stránkovanie sa použil už používaný v projekte ‘paginate’.

Testovanie

Testovanie bolo vykonávané pomocou kontroly, či sa skutočne zobrazujú korektné informácie o danom používateľovi.

Úloha #827 - Upraviť model databázy

Martin Molnár

Popis úlohy

Je potrebné zmeniť model databázy a refaktoring celého projektu, nakoľko sa zmenil zanikol model Contract a model Document. Namiesto toho sa vytvoril model

Analýza problému

Na stránke Crz.gov sa zmluvy pridávajú v pravidelných intervaloch(cca každých 20 min). Najskôr sa pridávajú dokumenty(zmluvy a dodatky) sekvenčne. Následke sa v nepravidelných a nedeterministických intervaloch pridá doplnená zmluva, ktorá napáruje zmluvy na dodatky. Táto doplnená zmluva má rovnaké id ako pôvodná napárovaná zmluva. Dodatok má tiež rovnaké id ako pôvodný dodatok.
Je potrebné rozlišovať medzi normálnym dokumentom a doplnenou zmluvou.

Návrh riešenia

Pri parsovaní centrálneho registra sa normálne dokumenty budú ukladať sekvenčne. V prípade ak príde doplnená zmluva, tak sa pozrú jej prílohy a pre každú prílohu sa nájde dokument s príslučným id. Tento dokument sa následne vymaže z db a tým sa zabezpečí konzistencia.

Opis implementácie

Algoritmus aktualizácie je doplnený do metódy create_from_metadata. Pre rozlíšenie doplnenej zmluvy je nutné dorobiť a dopĺňať atribút doplnena. Na základe tohto atribútu sa vykoná akcia pre aktualizáciu zmluvy.

Testovanie

Testovanie bolo vykonané empiricky. Našla sa stránka na centrálnom registri, kde je zmluva v stave doplnená. Následne sa spustí stahovanie. Po sťahovaní by sa mali napárovať dodatky na zmluvy a predošlé samostatné dokumenty by sa mali vymazať.

Úloha #856 -Testy pre jednotlivé parsre

Michal Kundrát

Popis úlohy

Úlohou bolo pokryť testami vytvorené parsre.

Analýza problému

Návrh riešenia

Opis implementácie

Pre každý z parserov bol vytvorený samostatný RSpec test, v ktorom sa kontroluje, či parsre naozaj vracajú zo vstupného html dokumentu príslušné údaje, pričom tieto testy fungujú aj offline keďže vstupné dokumenty sú lokálne kópie.

Testovanie

Úloha #826 - Spraviť API pre externistov

Tomáš Háber

Popis úlohy

Navhrnúť rozhranie pre externých používateľov, ktorý by chceli používať náš projekt a údaje z neho.

Analýza problému

Hľadal som vhodné riešenie API a to konkrétne REST a SOAP, prípadne vlastné na HTTP založené rozhranie.

Návrh riešenia

Navrhol som API využívajúce formát JSON a dodržujúce štandardy REST architektúry.

Opis implementácie

Výsledné API vyzerá nasledovne :
JSON API vyzera takto :

/zmluvy/
/zmluvy/:id
/zmluvy/search/:q

/supplier/
/supplier/:id

/ministry/
/ministry/:id

Testovanie

Testoval som pomocou webového prehliadača a aplikačných testov.

Úloha #829 - Úprava hlavnej stránky

Tomáš Háber

Popis úlohy
Upraviť hlavnú stránku do peknej podoby.

Analýza problému
Hľadal som vhodné riešenie API a to konkrétne REST a SOAP, prípadne vlastné na HTTP založené rozhranie.

Návrh riešenia
Navrhol som API využívajúce formát JSON a dodržujúce štandardy REST architektúry.

Opis implementácie
Výsledné API vyzerá nasledovne :
JSON API vyzera takto :

/zmluvy/
/zmluvy/:id
/zmluvy/search/:q

/supplier/
/supplier/:id

/ministry/
/ministry/:id

Testovanie
Testoval som pomocou webového prehliadača a aplikačných testov.

Úloha #828 - Synchronizácia Redmine a nového úložiska

Filip Lörinc

Popis úlohy
Pôvodné úložisko bolo uložené na severi relax na ktorý bol aj napojený redmine, ktorý tam beží tiež. Úlohou bolo synchronizovať obsah starého a nového úložiska tak, aby boli všetky zmeny viditeľné aj priamo v systéme redmine.

Analýza problému
Bolo treba zistiť ako skopírovať obsah nového úložiska do starého. Riešením je periodické spúšťanie skriptu, ktorý tento obsah bude medzi strojmi kopírovať.

Opis implementácie

Vytvorili sme skript na severi relax, ktorý zresetuje pomocou príkazu git (s danými prepínačmi reset --hard HEAD a clean -f) lokálny repozitár a následne stiahne repozitár z gitbusu. Periodické vykonávanie skptu každú hodinu sa zabezpečilo príkazom crontab.

Testovanie
Testovanie sa robilo kontrolou, či skript naozaj priebežne čistí starý depozitár a napĺňa obsah novým. Výsledok sa prejavil videním commitov v systéme redmine.

Úloha #830 - Zaregistrovať doménu

Filip Lörinc

Popis úlohy
Vymyslenú doménu bolo nutné zaregistrovať u registrátora domén za poplatok.

Opis implementácie
Doména bola zaregistrovaná u registrátora websupport a bol nastavený dns záznam smerujúci na IP adresu nám prideleného školského servera.

Úloha #822- Rozbehať SMTP na serveri

Matej Maruš

Popis úlohy
Na serveri ktoré sme získali v úlohe #815 nainštalovať smtp server pre posielanie vlastných emailov z našej domény. Napr. no-reply@nasedane.sk

Analýza problému

Debian obsahuje v seba pomocný virtuálny balík pre nainštalovanie SMTP servera. PO konzultácii s vedúcim tímového projektu a taktiež externou osobou ktorá nám prideľovala kapacity na virtuálnej stroji mi navrhli použiť interný školský SMTP server, ktorý sa nachádza na adrese http://dcs.fiit.stuba.sk
Keďže tento server je možné využívať iba interne z počítačov, ktoré sa nachádzajú v rovnakej lokálnej sieti. Tak na základe toho sme vytvorili aj ďalšie konto na gmail.com, ktorý obsahuje SMTP server pre posielanie emailov. Tento využívame na development.

Návrh riešenia

- Získať od kompetentnej osoby konfiguračné parametre na využitie školského SMTP servera.
- Registrácia na Gmail.com a získanie parametrov pre použitie ich SMTP servera.

Opis implementácie

Spočiatku sme nevedeli správne nakonfigurovať server, keďže sme nemali všetky potrebné

údaje a komunikácie bola trochu problémová nakoniec za pomoci vedúceho projektu sme dostali správne údaje, ktoré sme pridali do /config/enviroments/production.rb pre produkčné nasadenie. Taktiež sme nastavili aj development verziu pre interné potreby vývoja a testovania.

Úloha #824 - Umožňiť komentovanie len registrovaným používateľom

Matej Maruš

Popis úlohy

Návštevníci stránky, ktorí nebudú prihlásení budú mať obmedzené možnosti pracovania so stránkou. Nebudú môcť pridávať komentáre a taktiež budú mať obmedzené možnosti tagovania zmlúv. Pozeranie komentárov a taktiež sledovanie a čítanie zmlúv bude bez prihlasovania.

Analýza problému

Pre splnenie tejto úlohy je potrebné naviazať sa na úlohu #814, ktorá vyriešila pridávanie používateľov a ich registráciu. Z Devise návodu sme získali konfiguračné premenne pre obmedzenie používateľovho konania na stránke bez prihlásenia.

Pre naše pohreby využívame

- `user_signed_in?`
- `before_filter :authenticate_user!`

Tieto metódy resp. filtre nám dostatočné postačujú pre splnenie úlohy.

Návrh riešenia

Do kódu webovej aplikácie na základe analýzy pridáme premenné a obmedzíme možnosti neprihlásených používateľov.

Opis implementácie

Implementácia spočíva v pridaní *before_filter :authenticate_user* do Comments Controleru a taktiež v ošetrení View Comment pri pridávaní komentuj , kde sa výplni používateľove meno a neumožní pridať komentár a vyzve používateľa ku prihlásenia.

Šprint číslo 8

Úloha #840 - Rebríček najaktívnejších používateľov

Stanislav Valachovič

Popis úlohy

Názov úlohy je nepresný, úlohou bolo vytvoriť hlavnú triedu pre ocenenia, ktorá bude pridávať

používateľom ocenenia podľa kritérii špecifikovaných pre dané ocenenie. Pre vznik nového ocenenia má byť potrebné len vytvorenie príslušnej triedy.

Analýza problému
Pri analýze som sa najmä zameriaval na naštudovanie ako dynamicky vytvoriť novú inštanciu triedy len podľa jej string názvu, a ako sa zapisujú rake tasky.

Návrh riešenia
V hlavnej triede pre ocenenia bude nejaká property, kam sa budú zapisovať názvy ocenení, ktoré sa majú spúšťať. Výpis získaných ocenení sa bude zobrazovať pri prehľade používateľa.

Opis implementácie
Vytvorila sa hlavná trieda badges, ktorá obsahuje metódu run, v ktorej sa spúšťajú všetky ocenenia, ktoré sa nachádzajú v property get_badges_to_run. V jednotlivých oceneniach sú následne metódy, či sa má dané ocenenie spustiť a metóda, ktorá vráti id používateľov, ktorý získavajú dané ocenenie. Bola vytvorená rake task badges, ktorá spúšťa hlavnú triedu pre ocenenia, v ktorej sa potom postupne spustia jednotlivé ocenenia.

Testovanie
Bolo vykonávané formou spúšťania rake tasky badge a kontrolou, či sa spustili ocenenia, ktoré sa mali spustiť a či vrátili správne id používateľov.

Úloha #838 - Pri komentovaní pridať komentár aj na facebook

Martin Molnár

Popis úlohy
Vytvoriť facebook účet pre projekt nasedane.sk. Zmeniť prihlasovanie cez facebook pomocou novej aplikácie. Vytvoriť novú aplikáciu, ktorá sa zaintegruje do ruby on rails a umožní pridať komentár na facebook stenu aktuálne prihláseného používateľa. Komentár na fb stene bude obsahovať: kto pridal skutočný komentár, úryvok z komentára a link na skutočný komentár.

Analýza problému
Facebook poskytuje API, ktoré umožnje pridávanie príspevkov na stenu. Je možné vložiť vlastný text a taktiež aj linku na externú URL. V našom prípade URL predstavuje link na komentovaný komentár. Pri testovaní je dôležité neprekročiť limit pridaných príspevkov, pretože facebook zablokuje pridávanie príspevkov pre daného facebook užívateľa pre danú aplikáciu. Facebook api funguje na základe spätných volaní, to znamená, že po kliknutí URL našej aplikácie, facebook overí užívateľa (užívateľ si musí pridať práva na aplikáciu) a presmeruje volanie na našu spätnú stránku, kde sa vykoná pridanie komentára na facebook s našim textom.

Návrh riešenia

Do controllera pridáme callback funkciu, kde sa vykoná pridanie príspevku na facebook. Ak používateľ nebol ešte prihlásený na facebook, tak sa presmeruje na stránku s našim prihlásením, kde sa užívateľ prihlási pomocou svojho facebook účtu. Užívateľ môže pridávať komentáre na facebook pre komentára pod zmluvou aj pre “inline” komentáre.

Opis implementácie

Do pohľadov comment_inline a comment_unattached pridáme dynamickú linku na metódu write_to_wall, ktorá je implementovaná v controlleri. Táto metóda zavolá facebook API, kde ako parametre pôjdu autorizačná URL aplikácie, ktorá sa nachádza v triede FacebookOauthCredentials. Následne sa na strane facebooku (ak overienie dopadlo úspešne) zavolá naša callback metóda facebook_callback, ktorá je tiež implementovaná v controlleri. V tejto metóde sa vygeneruje nový facebook komentár a nastavia sa atribúty príspevku(autor komentára, úryvok komentára a link na komentár).

Testovanie

Testovanie bolo vykonané empiricky. Z localhost-u sa pridal komentár na zmluvu. Následne sa klikla akcia pridanie na facebook. Následne nás facebook požiadal o povolenie našej aplikácie. Po odsúhlasení je náš príspevok pridaný na facebook. Overíme to prihlásením sa na facebook a skontrolujeme nástenku. Pre ostrú prevádzku je potrebné zmeniť v nastaveniach facebook doménu localhost na doménu nasedane.sk

Úloha #839 - Emailová notifikácia všetkých zmien na dokumente

Michal Kundrát

Popis úlohy

Úlohou bolo navrhnúť a implementovať funkcionalitu sledovania zmien na konkrétnej zmluve zvolenej používateľom a doručovať informácie o týchto zmenách prostredníctvom emailu.

Analýza problému

Mailový server už nebolo potrebné konfigurovať, keďže sa už dlhšie využíva na odosielanie vzniknutých chýb. Aby sa používateľ nezahlcoval množstvom mailov pri každej zmene, mailovú notifikáciu dostane iba v prípade pridaného komentára k zmluve.

Návrh riešenia

Na stránke, kde sa nachádza aj zmluva bude existovať možnosť Prihlásiť sa pre odber notifikácií pri pridaní komentárov. Od tejto chvíle bude používateľ dostávať maily o udalosti pridania komentáru k zmluve, ktorý bude obsahovať meno komentátora, text komentu a informáciu, ku ktorej zmluve bol komentár pridaný.

Opis implementácie

Pri kliknutí na odber notifikácií sa vytvorí v tabuľke vzťah zmluva - používateľ. Následne pri vytvorení komentu sa prezrie , ktorí používatelia sú k danej zmluve prihlásený pre odber notifikácií a týmto sa následne odošlú maily asynchrónne, aby sa skrátila odozva pri pridaní komentáru.

Testovanie

Testovanie prebiehalo pokusom. Pri prihlásení na odber, som zkontroloval v databáze, či sa vytvoril potrebný vzťah zmluva - používateľ a následne pri pridaní komentáru, či sa odosielal mail.

Úloha #833 - Fazety cez autocomplete

Tomáš Háber

Popis úlohy

Prepracovať existujúce fazetové prehľadávanie tak aby bolo lepšie použiteľné, jednoduchšie pre použitie a zobrazovalo informácie v atraktívnej forme.

Analýza problému

Analyzoval som existujúce využívané fazetové prehľadávania na komerčných webových stránkach. Prečítal som si niekoľko článkov týkajúcich sa tejto problematiky a získal som prehľad v tejto oblasti.

Návrh riešenia

Navrhol som prepracovať rozhranie prehľadávania tak aby hlavným vizuálnym prvkom bol zoznam kritérií pre prehľadávanie (fazety) a aplikované kritéria. Vhodné by bolo obohatiť a zlepšiť použiteľnosť rozhrania doplnením animácií a zobrazovania iba relevatných informácií s čo najmenším počtom nutných kliknutí.

Opis implementácie

Vytvoril som dva UL zoznamy, v prvom sa zobrazujú aplikované kritéria v druhom fazety. Po kliknutí na fazetu sa pridá do zoznamu aplikovaných kritérií a aplikajú sa nastavené kritéria na základe ktorých, sa aktualizujú výsledky.

Úloha #842 - API ma pridávanie dát

Matej Maruš

Popis úlohy

Exkluzívny používatelia budú mať možnosť posielať vlastné údaje na základe vopred dohodnutom rozhraní , dokumenty pre ich zverejnenie v našej aplikácii.

Analýza problému

Po konzultáciách a získavaní informácii na internete sme zvolili formát JSON pre posielanie údajov k nám. Je to rozšírený spôsob prenosu informácii pomocou requestov a taktiež je to jednoduchý spôsob.
Pre obmedzenie posielania dát od každého bez jeho overenia je potrebné požívateľa autentifikovať. Po analýze databázy je optimálne využiť token, ktorý sa nachádza v tabuľke

user. Z tohto tokenu nie je možné získať používateľove heslo. Bez vloženia tokenu sa dáta nesmú nahrať do systému.

Návrh riešenia

Formát JSON požiadavky musí vyzerať nasledovne. Obsahuje hlavičku dokument: {} ktorý obsahuje základne a taktiež voliteľné parametre. Medzi základné esenciálne patri: *doc_name, doc_id, subject, ministry, taktiez supplier, pass, resort, customer, link.* Pass obsahuje token pre autentifikáciu. Ostatné údaje sú spracovane a pridane do DB. Návrh počíta s tým, že link na konkrétne PDF bude verejne dostupné na internete. Ďalšia možnosť je použiť Base64 pre posielanie dát v požiadavke. Dáta sú posielané pomocou POST requestu a v hlavičke musí byť nastavené *Content-type: application/json*

Opis implementácie

Pre posielanie dát za pomoci Base64 je aplikácia nedostatočné naimplementovana ale spôsob verejného posielania dát pomocou linku dostatočné funguje a všetky údaje sú spracované a zobrazené v aplikácii.

Úloha #834 - Spraviť wireframe

Miroslav Polgár

Popis úlohy

Nakresliť dizajn stránky. Navrhnúť rozmiestnenie funkcionalít.

Analýza problému

Zistiť čo je zaujímavé a to zvýrazniť. Menej zaujímavé veci umožniť prehliadať rozkliknutím. Upútať pozornosť návštevníka.

Testovanie

Členovia tímu súhlasili s daným návrhom a návrh bol posunutý do ďalšej úlohy - návrh a vytvorenie dizajnu.

Úloha #843 - Please donate!

Filip Lörinc

Popis úlohy

Spraviť konto na Paypal a vložit odkaz naň na stránku.

Analýza problému

Bolo treba založit konto na Paypal a následne vložit vygenerovaný embed kód do aplikácie

Opis implementácie
Vygenerovaný kód bol vložený na spodok stránky do stredu, ktorý sa zobrazoval ako ikonka s linkom priamo na dotačný formulár na Paypal.

Testovanie
Ikonku bolo možne ihneď vidieť na spodku stránky s tým, že naozaj presmerovala používateľa na dotačný formulár na Paypal.

Úloha #841 - Vymyslieť rôzne badge

Filip Lörinc

Popis úlohy
Podstatou motivácie používateľov je aj pridelovanie im rôznych badgeov za ich aktivitu alebo iný prínos pre webstránku. Úlohou bolo vymyslieť rôzne takéto badge (hodnotenia)..

Analýza problému
Boli vymyslené tieto badge, ktoré následne boli schválené:

- používateľ s najväčším počtom komentárov za mesiac
- používateľ s najväčším počtom odporúčaní zmlúv za mesiac
- používateľ s najväčším počtom pridaných tagov za mesiac

Šprint číslo 9

Úloha #846 - Vložiť do API info, že kto uploadol dokument

Matej Maruš

Popis úlohy

Triviálna úloha za 1 bod, kde je potrebné pridať do tabuľky Documents stlpec User_id ktorý bude obsahovať primárny kľuč používateľa, ktorý pridal za pomoci API zmluvu na server.

Analýza problému

Pridanie stĺpca si vyžaduje vytvoriť migráciu nad tabuľkou Documents. Keďže je táto tabuľka verzionovana ďalej je potrebné pridať stĺpec user_id taktiež do documents_versions tabuľky pre korektné fungovanie aplikácie.

Návrh riešenia

- Migrácia nad Dokument
- Migrácia nad Dokument_version
- v controleri Dokument api pridať zapisovanie id usera do databázy.

Opis implementácie

Implementácia si vyžadovala taktiež pozmenenie Dokument_metadata.rb, ktorý sa nachádza v lib a pridania hodnoty User, ktorý obsahuje id usera. Migrácie boli vykonané pomocou rake generate migration.

Úloha #849 - Aktualizácia zmlúv

Martin Molnár

Popis úlohy

Táto úloha je pokračovaním úlohy #819. Po refaktorovaní modelu a parsovaním centrálneho registra je potrebné zaistiť aktualizovanie nových zmlúv, ktoré zahŕňa aj párovanie dodatkov na zmluvy.

Analýza problému

Na stránke Crz.gov sa zmluvy pridávajú v pravidelných intervaloch(cca každých 20 min). Najskôr sa pridávajú dokumenty(zmluvy a dodatky) sekvenčne. Následke sa v nepravidelných a nedeterministických intervaloch pridá doplnená zmluva, ktorá napáruje zmluvy na dodatky. Táto doplnená zmluva má rovnaké id ako pôvodná napárovaná zmluva. Dodatok má tiež rovnaké id ako pôvodný dodatok.
Je potrebné rozlišovať pri parsovaní medzi normálnym dokumentom a doplnenou zmluvou. Centrálny register obsahuje relatívne veľa dát a nie je vhodné zakaždým parsovať celý register ale len nové zmluvy.

Návrh riešenia

Parsovanie centrálneho registra sa bude realizovať asynchrónne a paralelne. Z toho plynie problém pri určení poradia pri spracovaní. Je nevyhnutné navrhnúť sťahovanie tak aby bolo nezávislé na poradí prichádzajúcich zmlúv(dodatok-zmluva/zmluva-dodatok). Na demonštráciu riešenia je nutné vytvoriť testy. Rovnako je treba implementovať do parsera centrálneho registra možnosť sťahovať iba nové zmluvy. T.j pri opätovnom spustení sťahovania sa stiahnu iba najnovšie zmluvy.

Opis implementácie

Do metódy create_from_metadata sa urobí dotaz na databázu s id dokumentu. Ak sa nájde taký dokument, tak sa zistí podľa atribútu(doplena) či sa jedná doplnenú zmluvu, ak áno tak sa napárujú dodatky na zmluvu. Ak nie tak sa tento dokument ďalej nespracuje, vtedy sa jedná o aktualizáciu nových zmlúv - stiahnú sa len tie zmluvy, ktoré niesú ešte v databáze.

Testovanie

Testovanie bolo vykonané v prostredí Rspec. Test je vytvorený nad modelom Document. V teste sa testuje, či platí, že sa nestiahne už existujúca zmluva ak prídu dve rovnaké zmluvy za sebou. Následne sa testuje či správanie pri doplnených zmluvách je podľa očakávaní, a síce sa testuje sekvencia prichádzajúcich zmlúv a porovnanie voči databáze.

Úloha #850 - Sťahovanie starých zmlúv

Matej Maruš

Popis úlohy

Kolegovia v tíme vytvorili parser pre ministerstva, ktoré neboli spustene nasadene na server. Úlohou je tieto parsre zaregistrovať a spustiť sťahovanie a kompletnú procedúru spracovania dokumentu(vytvorenie obrázkov a textu za pomoci OCR)

Analýza problému

Keďže od vytvorenia parsrov pre jednotlivé ministerstva prešlo dostatok času. Polovica z nich už nebola funkčná. Po analýze problému sme sa zhodli, že všetky zmluvy, resp. ich linky boli presunuté na server zmluvy.gov.sk. Tam sa nachádzajú všetky staré zmluvy, ktoré boli v minulosti zverejnene na každom ministerstve zvlášť.

Návrh riešenia

- Vytvoriť parter pre zmluvy.gov.sk (úloha #855)
- Spustiť sťahovanie týchto zmlúv zo stránky na serveri nasedane.sk
- Overiť korektne sťahovanie zmlúv

Opis implementáci

V spolupráci s Michalom Kundrátom sme vytvorili parser na stránku zmluvy.gov.sk a následne sme dali sťahovať tieto zmluvy na produkčný (testovací) server nasedane.sk

Úloha #851 - Sťahovanie nových zmlúv

Michal Kundrát

Popis úlohy

Úlohou bolo prerobiť existujúci parser pre centrálny register zmlúv tak, aby sťahoval nové zmluvy.

Analýza problému

Zmluvy na centrálny register prichádzajú v nepravidelných intervaloch. Je potrebné, aby pri spustení parsera neprechádzal opätovne cez veľké množstvo zmlúv, avšak aby iba zparsoval tie zmluvy, ktoré ešte neprešiel.

Návrh riešenia

Pri postupnom prechádzaní zmlúv sa trieda, ktorá sa prezrie, či už sa daná zmluva

nespracovala.

Opis implementácie
Pri postupnom parsovaní sa trieda, ktorá sa stará o logiku parsovania pozrie do databázy, či sa už daná zmluva tam nenachádza, ak sa nachádza, nasledujúce zmluvy sa už nepridajú do delay jobu a parsovanie sa ukonči.

Testovanie
Testovanie bolo vykonané empiricky. Nechalo sa zparsovať niekoľko zmlúv, a po pridaní nových sa zisťovalo, či sa naozaj zparsovali iba zmluvy, ktoré ešte neboliv databáze.

Úloha #852 - Zoznam badgeov a ako ich získať

Stanislav Valachovč

Popis úlohy
Úlohou bolo vytvoriť samostatnú stránku, kde budú vysvetlená ocenenia s popisom ako ich získať.

Opis riešenia
Do jednotlivých tried ocenení bola vložená property explain, ktorá obsahuje text, ktorý popisuje daný badge. Bola vytvorná samostatná stránka, kde sa iba prechádzajú jednotlivé triedy ocenení a ku každému sa vypíše obrázok, jeho názov a popis.

Úloha #855 - Nový parser pre stránku zmluvy.gov.sk

Michal Kundrát

Popis úlohy
Je potrebné vytvoriť parser, ktorý bude extrahovať data zo stránky zmluvy.gov.sk.

Analýza problému
Na stránke zmluvy.gov.sk sa nachádza veľké množstvo zmlúv, a aj v poslednej dobe mnohé ministerstvá presunuli zmluvy zo svojich stránok práve sem. Stránka obsahuje stránkovanie, a každá zmluva je na samostatnej podstránke.

Návrh riešenia
Parser bude obsahovať metódy pre parsovanie každej úrovne stránky. Nad týtmo parserom bude operovať trieda, ktorá sa bude starať o logiku parsovania.

Opis implementácie
Pre prácu s xml dokumentom je využitá knižnica Nokogiri. Každé volanie parsovanie jednotlivej

zmluvy bude vykonávané asynchrónne.

Testovanie
Testy boli robené pomocou unit testu RSpec. Test porovnáva, či vrátený objekt naozaj obsahuje také isté údaje, ako sa nachádzajú na vstupnom dokumente.

Úloha #848 - Implementácia badgeov

Tomáš Háber

Popis úlohy
Implementovať nasledovné “badges”.

- najaktivnejsi používatel (počet prihlásení za mesiac)
- najviac tagov
- ten kto dostane najviac likes za mesiac

Analýza problému

Preštudoval som implementáciu existujúceho badge a na základe toho sa inšpiroval pri implementácií ostatných.

Návrh riešenia

Skopirovať a upraviť existujúcu implementáciu a zmeniť SQL príkaz pre každý badge tak aby zodpovedal názvu.

Opis implementácie

Napísal som SQL príkazy pre každý badge a upravil časy ich spúštania.

Úloha #847 - Otestovať widget

Miroslav Polgár

Popis úlohy
Otestovať widget pri množstve súčasných požiadaviek a zdokumentovať.

Návrh riešenia
Využitie apache **ab** pre otestovanie zvládnutia záťaže widgetu. Bolo potrebné ako root nastaviť vyšší limit pre “open files”. Následne som spúšťal príkaz ab s rôznymi parametrami pre adresu /

documents/1/widget na serveri nasedane.sk.
Výsledky pre priemerné časy na požiadavku som spracoval v grafe.

Úloha #844 - Spraviť novú grafiku a dizajn

Filip Lörinc

Popis úlohy

Spraviť novú grafiku a dizajn na už produkčnú verziu, ktorá by bola zverejniteľná.

Návrh riešenia

Bolo treba zobrať už nakreslený wireframe a pretransformovať nakreslený obsah na statickú webstránku, ktorá už bude obsahovať všetky relevantné grafické komponenty a rozloženie prvkov. Následný výsledný produkt bol posunutý na schválenie a pripomienky boli zapracované do výsledného produktu, ktorý bol posunutý ďalej programátorovi, ktorý do dizajn a jednotlivé prvky zakomponoval do dynamickej aplikácie.