Šprint číslo 1

Úloha #719 - transformácia pdf súborov na obrázky Matej Maruš

Popis úlohy

Používateľ sa s touto funkcionalitou priamo nedostáva do kontaktu. Na zabezpečenie jednoduchej manipulácie so zmluvami potrebujeme rozdeliť pdf súbory na obrázky. Každá strana zmluvy uložená v pdf súbore bude transformovaná na samostatný obrázok.

Analýza problému

V analýze som sa venoval hlavne nájdeniu potrebných externých knižníc (gem) na realizáciu tejto funkcionality. Po konzultácii so zadávateľom projektu (vedúci) som objavil gem *Docsplit*. http://documentcloud.github.com/docsplit/. Tento nástroj plne podporuje všetky potrebné funkcie na vykonanie tejto úlohy.

Návrh riešenia

Použitím gemu *Docsplit* vytvorím na súborovom systéme adresár images, ktorý bude obsahovať všetky obrázky pre každú zmluvu.
Adresár *images* sa bude nachádzať pod adresárom, ktorého názov je primárny kľuč v tabuľke Contracts, pre jednoznačnú identifikáciu zmluvy.

Opis implementácie

Obrázky sú ukladané vo formáte png. Celá funkcionalita je implementovaná v jednej triede *Convertor.* Táto trieda obsahuje jednu inštančnú metódu convert, ktorá prijíma ako parameter cestu k root adresára, kde začne rekurzívne prehľadávať všetky podadresáre a následné transformovať pdf súbory na obrázky. Kontrolujem, či zmluva uložená v pdf súbore už bola transformovaná na obrázky. Ak nebola ešte transformácia uskutočnená, vykonám operáciu *convert*.

Testovanie

Testovanie je vykonávané na baze kontroly vytvorenia obrázkov na súborovom systéme.

Úloha #721 - Umožniť používateľom komentovať jednotlivé zmluvy Stanislav Valachovič

Popis úlohy

Používateľ má možnosť po označení špecifickej pasáži v zmluve - v obrázku pridať k tejto pasáži komentár, ktorý sa následne zobrazí pri obrázku a vyznačená pasáž sa zvýrazní. Meno používateľa, ktorý pridal komentár sa bude pamätať v cookie a predvypĺňať.

Analýza problému
V rámci analýzy som sa zameral na preskúmanie ruby a ajaxu, aká náročná je jeho implementácia do ruby a takisto tvorbu formulárov v ruby. Prezeral som aj možné riešenia zápisu do cookie.

Návrh riešenia
Na pridávanie komentárov využijem ajax. Meno používateľa, ktorý pridáva komentár a samotný text komentáru budú povinné. Po úspešnom pridaní komentáru sa zobrazí hneď pri obrázku. Na zapamätanie mena používateľa použijem cookie cez javascript.

Opis implementácie
Formulár na pridávanie komentáru sa vypisuje pri každej zmluve, len je hidden. Pri označení pasáže v zmluve sa zobrazí tento formulár pod vyznačenou časťou. Ak nie je vyplnené meno alebo text komentáru, táto nevyplnená časť sa zvýrazní červeným borderom. Ak bol kometár úspešne uložený - boli vyplnené všetky polia, zobrazí sa pri obrázku a vyznačí sa v obrázku vyznačená časť červeným borderom.

Testovanie
Testovanie bolo vykonávané kontrolou uložených komentárov do db, či majú správne hodnoty.

Úloha #758 - sťahovanie zmlúv a integrácia s parsermi — Martin Molnár

Popis úlohy
Táto úloha patrí do kategórie úloh určených pre backend. Jedná sa automatické sťahovanie pdf, doc zmlúv na základe url adresy. Taktiež k tejto úlohe patrí aj vytvorenie File systému a fyzického modelu pre ukladanie metadát k zmluvám pri sťahovaní.

Analýza problému
Pred implementáciou som musel návrhúť logický model databázy. Pri navrhovaní sťahovača som musel spolu s Matejom Marušom, ktorý mal na starosti konvertor pdf súborov na obrázky, navrhnúť vhodné rozhranie na komunikáciu medzi sťahovačom a konvertorom. Pri integrácií sťahovača s parsermi sme museli tiež dohodnúť spoločné rozhranie. Na sťahovanie som použil štandardné aplikačné rozhranie ruby.

Návrh riešenia
File system sme navrhli nasledovne: Sťahované pdf súbory sú organizované hierarchicky do priečinkov, pričom názov priečinku tvorí číslo zmluvy z databázy. Údaje o parseroch sa uložia do databázy. Taktiež sa pri sťahovaní uložia metadáta a údaje o zmluve do databázy. Sťahovač zabezpečí, že ak sa zmluva nepodarila stiahnúť tak sa neuloží ani do databázy. Samotné sťahovanie sa deje ako rake task.

Opis implementácie
Trieda Downloader implementuje funkcionalitu sťahovania do metódy download. Táto metóda najskôr prečíta údaje o parseroch, následne pre každý parser zavolá metódu Parse, ktorá vráti

údaje o zmluve. Tieto údaje sa následne overia a ak sú platné tak sa uložia do databázy a vytvorí sa na file systéme priečinok s číslom zmluvy a zaháji sa sťahovanie.

Testovanie

Testovanie bolo vykonávané kontrolou stiahnutých zmlúv na súborový systém ako aj kontrolou údajov v databáze.

Úloha #731 - Zvýrazňovanie špecifických pasáží v zmluvách

Stanislav Valachovič

Popis úlohy

Používateľ má možnosť v zmluve, v obrázku označit vybranú pasáž. Úlohou je zaznamenať pozíciu a rozmery vyznačenej pasáže.

Analýza problému

Preskúmal som existujúce javascriptové knižnice, ktoré boli zamerané na prácu s obrázkami a označovanie častí v obrázkoch.

Návrh riešenia

Na označovanie pasáží v zmluvách - obrázkoch, použijem knižnicu rozširujúcu jquery - imgareaselect.js, sú v nej už implementované metódy na zvýraznenie určitej časti v obrázku ako aj funkcie na získanie označenej pozície v rámci obrázku a jej rozmerov.

Opis implementácie

Knižnicu na označovanie pasáží som trochu upravil pre potreby nášho projektu, preto je medzi skriptmi uložená jej plná verzia aj keď na stránke sa používa jej menšia - pack verzia. Do application.js, kde sa nachádza documentReady funkcia je vložený kód, ktorým sa inicializuje táto funkcia pre každý obrázok, ktorý obsahuje css classu “selectable”. Po označení pasáži v obrázku funkcia vráti pozíciu top a left v rámci označeného obrázku, výšku a šírku označenej časti a css id obrázku, v ktorom sa označovala pasáž.

Testovanie

Testovanie je vykonávané kontrolou, či vrátené údaje sú skutočne korektné - správne id obrázku, pozícia a rozmery.

Úloha #732 - Widget na externé webstránky

Miroslav Polgár

Popis úlohy

Používateľ má možnosť integrovať malý prehliadač - widget s konkrétnou zmluvou do externej (svojej) webstránky. Vo widgete bude možné si prehliadnuť jednu zmluvu s prislúchajúcimi komentármi.

Analýza problému

Preskúmal som existujúce riešenia. Vedúci objavil open-source projekt *Document cloud* http://www.documentcloud.org/ odkiaľ bolo možné použiť toto riešenie.

Návrh riešenia
Pre ukážku funkčnosti widgetu bude možné si zobraziť akúkoľvek zmluvu na našom webe. Avšak na vloženie do externej webstránky bude postačovať len krátky skript s adresou na náš server. Adresa ku konkrétnej zmluve - id z databázy - sa zadá na koniec cesty url (http://.../widget/1)

Opis implementácie
Do adresára views/widget je uložený vzorový widget. V /lib/widget sú uložené všetky javascripty a obrázky potrebné na zobrazenie widgetu. Trieda WidgetController obsahuje metódu pre vygenerovanie súboru formátu *js* s údajmi o zmluve. Zmluva je uložená v obrázkoch formátu *.png* a adresy k týmto obrázkom sú uložené v tomto súbore. Tento súbor načítava widget.

Testovanie
Prebehne zobrazením widgetu na stránke. Ak obrázky so zmluvou neexistujú, widget sa nezobrazí.

Úloha #724 – extrakcia dát zo zmluvy.gov **Michal Kundrát, Martin Molnár**

Popis ulohy
Na portáli zmluvy.gov.sk a na stránkach jednotlivých ministerstiev a úradov sa nachádzajú jednotlivé odkazy na súbory zmlúv spolu s niektorými doplňujúcimi informáciami k nim, ako napríklad číslo zmluvy, jej názov, dodávateľ, cena na celkové plnenie a podobne. Tieto informácie spolu s odkazmi na jednotlivé súbory zmlúv je potrebné extrahovať z týchto stránok.

Analyza problemu
V analýze riešenia bolo potrebné preskúmať a zvoliť vhodnú knižnicu, pomocou ktorej bude možné vyberať data z jednotlivých stránok. Na výber bolo viacero knižníc s dobrým hodnotením a použiteľnosťou ako napríklad Hpricot, Nokogiri či Rexml, pričom na implementáciu som si zvolil nakoniec Nokogiri.

Navrh riesenia
Na extrakciu dát bolo potrebné vytvoriť parser, ktorý bude parsovať html kód stránky ministerstiev. Keďže každé ministerstvo malo inú štruktúru stránky, bolo potrebné vytvoriť pre každú stránku iný parser. Na niektorých stránkach boli data uložené v jednej veľkej tabuľke, v iných zase boli rozdelené tieto tabuľky do viacerých stránok, alebo odkazy neboli priamo v danej tabuľke, ale boli uložené na stránkach kde daný odkaz smeroval.

Opis implementacia
Pre každý parser existuje samostatná trieda, obsahujúca jednotnú metódu parse(). Táto metóda použitím gemu Nokogiri postupne prechádza elementy zdrojového kódu stránky a ukladá potrebné údaje o zmluve do objektu, ktorý predstavuje informácie o jednej zmluve. Po každej zparsovanej zmluve tento objekt pridá do poľa, a po extrakcií všetkých informáciach o zmluvách

spolu s odkazmi na jednotlivé súbory zmlúv toto pole vráti ako návratovú hodnotu.

Testovanie
Testovanie je vykonávané pomocou unit test frameworku Rspec. Testy sú robené priamo na niektoré konkrétne zmluvy, pričom sa kontroluje či parser naozaj získal potrebné informácie zo stranky a teda či sa jednotlivé polia na stránke zhodujú s extrahovanými.

Úloha #754 Vytvorenie hlavnej stránky a šablóny — Tomáš Háber

Popis úlohy
Na základe dodaného dizajnu stránky v HTML formáte, vytvoriť šablónu pre aplikáciu aj s menu. Ďaľšou úlohou je vytvoriť hlavnú stránku, ktorá bude obsahovať zoznam všetkých zmlúv, naposledy komentované a najkomentovanejšie zmluvy.

Analýza problému
Šablónu som sa rozhodol vytvoriť pomocou ERB, nakoľko je integrovaná s Rails. Pre navigáciu som použil veľmi často využívaný gem simple-navigation.

Návrh riešenia
Vložiť HTML kód z dizajnu do application.html.erb. Následne do toho istého súboru vložím menu vygenerované cez simple-nagivation.

Opis implementácie

Spracoval som HTML dizajn do application.html.erb. Nastavil som v súbore navigation.rb menu, ktoré som potom vložil do dizajnu cez <% render_navigation%>.

Úloha #754 Fulltextové vyhladávanie — Tomáš Háber

Popis úlohy
Úlohou je umožniť fulltextovo vyhľadávať v textoch stiahnutých zmlúv.

Analýza problému
Najjednoduchším riešením je použiť fulltextové vyhladavanie cez MySQL. Pre zvyrazovanie textu a najdeného useku textu budem pouzivat helper excerpt a highlight.

Návrh riešenia
Riesenim je pouzit MySQL fulltext indexovanie, pridat index na stplec s textom zmluvy do tabulky contracts.

Opis implementácie
Implementoval som controller a view podla navrhu.

Úloha #729 Minimalistický a prehľadný dizajn Filip Lörinc

Popis úlohy
Podstatou úlohy je pripraviť grafický návrh celej stránky včítane grafických a estetických prvkov.

Analýza problému
Na vytvorenie dizajnu bolo nutné vytvoriť grafické elementy a následne ich poskladať tak, aby bol výsledok čo najviac pohľadný a zároveň praktický.

Návrh riešenia
Vzhľadom na fakt, že ako prvá bola vytvorená stránka projektu, dizajn samotnej aplikácie sa bude inšpirovať práve dizajnom tímovej webstránky.

Opis implementácie
Ako prvé som som vytvoril statickú stránku, ktorá obsahovala predpokladané rozloženie HTML a grafických prvkov. Následne jednotlivé užitočné komponenty boli zahrnuté do aplikácie, pričom sa dané elementy upravujú podľa aktuálnych potrieb.

Šprint číslo 2

Úloha #737 - Komentáre pod zmluvou - neštrukturovaná verzia
Stanislav Valachovič

Popis úlohy
Umožniť používateľom pridávať jednoduché komentáre pod zmuvu.

Analýza problému
Väčšia analýza nebola potrebná kedže je to podobná úloha ako úloha #721.

Návrh riešenia
Využijem existujúce riešenie pridávania komentárov z úlohy #721.

Opis implementácie
Využil som už exitujúci formulár na pridávanie komentárov. Rozdiel je len v zobrazení pridaného komentáru, ktorý sa zobrazí pod zmluvou. Na rozlíšenie, či je to komentár k obrázku alebo pod zmluvou je použitý page_number komentáru, ktorý je v prípade ak sa jedná o komentár pod zmluvou rovný 0.

Testovanie
Testovanie bolo vykonávané kontrolou uložených komentárov do db, či majú správne hodnoty.

Úloha #741 - 2 módy zobrazovania zmlúv Stanislav Valachovič

Popis úlohy

Pri zobrazení zmluvy budú na výber 2 módy zobrazenia - zobrazenie všetkých strán zmluvy alebo zobrazenie len strán, ktoré majú komentáre. Strany bez komentárov budú zbalené, po kliku sa rozbalia.

Analýza problému
V rámci anayzy som sa pozrel na zachytávanie parametrov z url v ruby a javascriptové knižnice, ktoré podporujú rozbalovanie a zbalovanie elementov - niečo na spôsob rozbalovacieho menu.

Návrh riešenia
Pri zobrazení zmluvy budú pri nej linky na zobrazenie všetkých strán alebo len strán s komentármi. Pre rozbalovanie a zbalovanie strán bez komentárov použijem rozširujúcu jquery knižnicu accordion.js.

Opis implementácie
Pri výpise obrázkov, ak nemá obrázok komentár vloží sa tam div tag s css classou “nocomment”. Do application.js, kde sa nachádza documentReady funkcia je vložený kód, ktorým sa inicializuje táto funkcia pre každý tento div tag. Táto funkcia následne zbalý prvého parenta tohto divu (v našom prípade obrázok) a pridá classu “selected”, čo umožňuje zmenu obrázka plus na mínus a opačne.

Testovanie
Testovanie bolo vykonávané, kontrolou či sa obrázky bez komentárov naozaj zbalujú a či sa po kliku rozbalia, po ďalšom kliku zbalia a pod.

Úloha #746 - transformácia zmlúv na čistý text pomocou OCR technológie

Matej Maruš

Popis úlohy
Používateľ priamo nepristupuje k tejto funkcionalite. Táto funkcionalita je potrebná na zabezpečenie vyhľadávania v texte zmlúv. Verzia podporuje čítanie textu napísaného v slovenčine.

Analýza problému
Po vykonaní analýzy existujúcich nástrojov zaoberajúcich sa OCR technológiou som si zvolil projekt *tesseract* (http://code.google.com/p/tesseract-ocr/). V čase hľadania vhodného externého programu vyšla nová verzia *tesseract*, ktorá spľňa všetky kritéria na implementáciu tejto úlohy. Obsahuje slovník na prácu so slovenskými znakmi. *Tesseract* je vydávaný pod OpenSource licenciou. Nevýhody v použití novej verzie *tesseract* sú hlavne v tom, že ešte nebol implementovaný v projekte *docsplit*, ktorý zatiaľ pracuje len s verziou *tesseract* 2.0. Táto verzia nepodporuje slovenský jazyk a tak možnosti použitia v tejto úlohe sú minimálne.
Napriek všetkým nevýhodám som zvolil *tesseract* 3.0 kôli jeho vysokej úspešnosti rozoznávania slovenských znakov.

Návrh riešenia

Na realizáciu tejto funkcionality potrebujeme vykonať nasledujúce kroky:

- spustenie OCR extraktora pre každý obrázok a následné uloženie do textového súboru
- načítanie textu zo všetkých textových súborov patriacich jednej zmluve
- opakovanie tohto procesu pre každú zmluvu.

Opis implementácie

Konkrétna realizácia tejto úlohy je vykonaná pomocou ruby metódy *system()*, ktorá volá programy z príkazového riadku. Implementácie úlohy si vyžaduje kompiláciu zdrojových súborov programu tesseract pod operačným systémom GNU/Linux. Predtým ako môžeme spustiť skompilovaný program tesseract pomocou system() metódy, potrebujeme poznať počet obrázkov(počet strán zmluvy). To je vykonané jednoducho prehľadaním súborového systému a spočítaním počtu súborov s príponou .png v adresári *images*.
Túto funkcionalitu som pridal do triedy *Converter*.

Testovanie

Testovanie je vykonávané zobrazením súborového systému a overením existencie textových súborov pre každú zmluvu uloženú v pdf súbore.

Úloha #746 Cron job - sťahovanie nových zmlúv automaticky Matej Maruš

Popis úlohy

Proces sťahovania zmlúv bude plne automatizovaný. Taktiež následná transformácia pdf súborov a extrakcia textu z obrázkov sa bude vykonávať automaticky v stanovený čas pod operačným systémom GNU/Linux.

Analýza problému

Hľadanie existujúcich riešení na internete ma priviedlo k projektu *whenever*. Je to gem do prostredia ruby, ktorý slúži na spúšťanie rake úloh v presne špecifikovaný čas. Na realizáciu tejto funkcionality nám stačí iba tento gem.

Návrh riešenia

Na realizáciu tejto funkcionality potrebujeme vykonať nasledujúce kroky:

- vytvoriť rake úlohy pre každú funkciu zvlášť (sťahovač, pridať parsre, transformácia)
- nastaviť čas vykonávania úloh
- exportovať zmeny z ruby do operačného systému, ktorý vykonáva spúšťanie funkcionalít nezáväzne od *ruby on rails* projektu.

Opis implementácie

Vytvoril som v adresári lib/task textové súbory, ktoré som nastavil podľa vzoru pre vytváranie rake úloh. Následne som pridal do projektu gem *whenever*, ktorý vygeneroval súbor *schedule.rb.*
V tomto súbore sa nastavujú časy spúšťania úloh. Ošetrenie chybových stavov je implementovane v samostatných triedach, ktoré sa starajú o vykonanie tej konkrétnej

funkcionality (sťahovač...).

Testovanie
Testovanie je realizované na základe kontroly programátora po spustení úlohy a sledovaní zmien v projekte**.**

Úloha #750 Monitorovanie služby — Tomáš Háber

Popis úlohy
Nastaviť monitorovanie aplikácie, tak aby pri vyhlásenej výnimke v aplikácií bol notifikovaný realizačný tím.

Analýza problému
Rozhodol som sa použiť Hoptoad, nakoľko bola táto služba jednoduchšia na integráciu.
Návrh riešenia
Podľa návodu na hoptoad.com integrovať aplikáciu s touto službou.

Opis implementácie

Do Gemfile som doplnil gem hoptoad-notifier. Spustil príkaz rails generate hoptoad -api, ktorý nastavil autorizačný klúč

Úloha #749 Info o projekte — Tomáš Háber

Popis úlohy
Vytvoriť stránku s informáciami o projekte a kontakte na tím.

Analýza problému
Na vytvorenie stačí jednoduchá textová stránka.

Návrh riešenia
Vytvoriť controller a view pre informácie o projekte a doplniť túto stránku do menu.

Opis implementácie

Vygeneroval som controller a view pre informácie o projekte. Upravil som HTML stránku pre view a doplnil do nej informácie o projekte a kontakt na spoločný e-mail.

Úloha #747 Kategórie ku zmluvám — Martin Molnár

Popis úlohy

Vytvoriť ku zmluve editovacie políčka, ktoré umožnia užívateľom editovať kategórie ku zmluvám. Taktiež vytvoriť verziovanie týchto kategórií v databáze.

Analýza problému
Na editovanie som použil plugin do ruby on rails, a síce plugin rest_in_place, ktorý j e možno získať na adrese http://github.com/janv/rest_in_place. Tento plugin umožňuje editáciu a následnú perzistenciu polí z databázi.
Na verziovanie som použil taktiež plugin, ktorý je možné získať z adresy /http://github.com/technoweenie/acts_as_versioned/tree/master. Tento plugin sleduje zmeny nad zvolenou tabuľkou a ak sa zmení ľubovolné pole, tak sa do verziovacej tabuľky zapíše predchádzajúca hodnota.

Návrh riešenia
Nainštalovať pluginy a použiť nad tabuľkou Contract.

Opis implementácie

Inštalácia obidvoch pluginov sa robí pomocou ruby script/plugin install . git://adresa . Po nainštalovaní pluginu na editovanie som v contract controlleri vytvoril metódu update, ktorá sa spustí po vykonaní zmien. Následne som v contract view-e doplnil tabuľku, ktorá obsahuje kategórie zmluvy.
Pri verziovaní sa musí pridať migrácia,, ktorá obsahuje väzbu na danú tabuľku , ktorá sa má verziovať. ďalej sa musí dopisať do modelu verziovanej tabuľky dopísať informácia o verziovaní.

Úloha #749 Info o projekte — Tomáš Háber

Popis úlohy
Vytvoriť stránku s informáciami o projekte a kontakte na tím.

Analýza problému
Na vytvorenie stačí jednoduchá textová stránka.

Návrh riešenia
Vytvoriť controller a view pre informácie o projekte a doplnť túto stránku do menu.

Opis implementácie

Vygeneroval som controller a view pre informácie o projekte. Upravil som HTML stránku pre view a doplnil do nej informácie o projekte a kontakt na spoločný e-mail.

Úloha #748 Fazetové vyhľadávanie — Tomáš Háber

Popis úlohy
Úlohou je umožniť vyhľadávanie podľa kritérií: ceny / obstaraváteľa / / dátumu / oblasti (hardvér,

nábytok, diaľnica..) pomocou fazetového vyhľadávania.

Analýza problému
Pre fazetové vyhľadávanie som sa rozhodol použiť už existujúcu technológiu, ktorá implementáciu vyhľadávania výrazne uľahčuje. Konkrétne som sa rozhodol použiť indexovací engine Sphinx, a gem thinking_sphinx.

Návrh riešenia
Použiť funkcie poskytované gem-om thinking_sphinx pre fazetové prehliadanie. Zároveň upraviť fulltextové prehľadávanie tak aby používalo tiež sphinx, ktorý je schopný poskytovať relevantnejšie výsledky ako mysql vyhľadávanie.

Opis implementácie

Cez balíčkovací systém operačného systému som nainstaloval Sphinx. Následne som v konfiguračnom súbore config/sphinx.yml nastavil cesty k sphinx. Ďalším krokom bolo vygenerovanie konfiguračného súboru pomocou rake tasku thinking_sphinx:configure a spustenia sphinx pomocou tasku thinking_sphinx:start. Upravil som model tak aby obsahoval nastavenie indexovaných polí, a označil ako fazetu ministerstvo, ku ktorému zmluva patrí.

Ďalej som upravil controller pre vyĥladávanie tak aby používal pre vyhľadávanie fazety, a prislušne k tomu upravil aj view.

Úloha #742 Čo pribudlo od poslednej návštevy — Tomáš Háber

Popis úlohy
Úlohou je zvýrazniť komentáre, ktoré používateľ ešte nevidel, a ktoré pribudli od jeho poslednej návštevy. Tieto údaje je potrebné uložiť do cookie.

Analýza problému
Potrebné je ukladať si do cookie vždy pri zobrazení danej zmluvy informáciu o tom, ktoré komentáre už používateľ videl. Tým ktoré ešte nevidel a sú pre používateľa nové, je potrebné zvýrazniť pridaným css triedy.

Návrh riešenia
Aby používateľ nemaľ priveľký počet cookies uložených v prehliadači, rozhodol som sa pre všetky zmluvy používať jednu cookie. Do tejto cookie ukladám informácie v YAML formáte. Konktrétne ukladám hash tabuľku, kde je klúčom primárny klúč zmluvy a hodnotou je primárny kľuč najnovšieho komentára patriaceho k zmluve, ktorý už používateľ videl.

Pri zobrazení zmluvy sa získa toto id, a nastaví aktuálnou hodnotou, a zvýraznia sa všetky komentáre, ktoré majú vyššie id, ako získaná hodnota.

Opis implementácie

Na základe návrhu som upravil contract_contoller.rb a contract/show.html.erb. Zároveň som upravil aj vytvorenie nového komentára, tak aby nový používateľom vytvorené komentár mu nebol zvyraznený ako nový.

Úloha #739 - Zobrazenie dvoch "naj-like" komentárov Miroslav Polgár

Popis úlohy
Používateľovi sa pri zobrazení zmluvy zobrazia 2 najobľúbenejšie komentáre.

Analýza problému
Je potrebné, aby v tabuľke "comments" v databáze bol stĺpec "like".

Návrh riešenia
Je treba načítať z databázy komentáre, ktoré patria k danej zmluve. Potom treba nájsť dva, ktoré majú najvyššie číslo v stĺpci "like" tabuľky "comments" v databáze.
Napokon zobraziť komentáre. V prípade, že komentár má tento atribút nulový, alebo neexistujú komentáre, nezobrazí sa nič.

Opis implementácie
Funkcionalitu som implementoval priamo do views/contract/show.html, pretože to nie je kľúčová funkcionalita, je to len pohľad do databázy, takže v modeli MVC sa nachádza vo "View".

Testovanie
Dôležité je, aby sa nevyskytla chyba pri neexistujúcich komentároch, alebo v prípade, že je len jeden, či žiaden obľúbený.

Úloha #738 Like / dislike komentárov Filip Lörinc

Popis úlohy
Podstatou úlohy je umožniť používateľom hodnotiť jednotlivé komentáre v štýle - páči sa mi / nepáči sa mi.

Analýza problému
Bude nutné vytvoriť v databáze ďalšie dva stĺpce, kde jeden bude obsahovať celočíselnú hodnotu pozitívnych hodnotení a ďalší bude obsahovať celočíselné hodnoty. Následne doplniť obrázky na ktoré bude možné kliknúť a tým možnosť vyjadriť svoj súhlas alebo nesúhlas s komentárom.

Návrh riešenia

Po kliknutí na obrázok s pozitívnym alebo negatívnym ohlasom sa bude ajaxom volať externý skript, ktorý zapíše do databázy buď pozitívny alebo negatívny ohlas podľa toho či už človek hlasoval alebo nie. Následne sa na stránke dynamicky objaví aktualizovaný počet hlasov pre a proti.

Opis implementácie

Pri výpise jednotlivých komentárov boli doplnené obrázky v tvare zdvihnutého/nezdvihnutého palca. Ktoré obsahovali link na externý script. Po jeho zavolaní sa upraví dané pole v databáze pre like alebo dislike.

Šprint číslo 3

Úloha #757 - zobrazenie zmlúv - less mód Stanislav Valachovič

Popis úlohy

Úloha nadväzuje na úlohu #741 - 2 módy zobrazenia zmlúv. Pri obrázku zmluvy, ktorá má komentáre sa zobrazí z tohto obrázku len tá časť, ktorá je okomentovaná, teda nie celý obrázok.

Analýza problému

V rámci anaýzy som sa zameral na už používanú javascriptovú knižnicu imgareaselect.js. Analyzoval som kde sa nastavuje veľkosť obrázku, v ktorom sa dá vyznačovať pasáž a kde sa inicializuje veľkosť a umiestnenie tejto pasáže.

Návrh riešenia

Bude potrebné upraviť imgareaselect.js, tak aby sa dala zobrazovať len určitá časť obrázku. Bude na to potrebné využiť aj css.

Opis implementácie

Pri výpise komentárov k obrázku, v prípade ak je zapnutý tzv. less mode (zobrazenie strán s komentármi), sa zistí najvyššie a najnižšie umiestnený komentár v rámci obrázku. Z toho sa vypočíta veľkosť časti obrázku, ktorá bude viditeľná (na ktorej sa nachádzajú komentáre). Do html sa pri obrázok následne vloží hidden element, ktorý obsahuje údaj o posunutí obrázku. O zvyšok sa postará javascript, ktorý na základe tohto čísla zobrazí len požadovanú časť obrázku.

Testovanie

Testovanie bolo vykonávané kontrolou či sa zobrazuje správna časť obrázkov - časť kde sú komentáre.

Úloha #762 - anglické názvy do slovenčiny **Stanislav Valachovič**

Popis úlohy

Úlohou bolo prejsť celý projekt a poprepisovať anglické názvy za slovenské ekvivalenty.

Úloha #767 - autofocus pre komenty **Stanislav Valachovič**

Popis úlohy

Pri zobrazení formuláru sa má nastaviť kurzor do inputu meno, v prípade ak už je meno vyplnené (z predošlého zadávania komentáru alebo z cookies), tak sa kurzor nastaví do inputu pre text komentáru.

Analýza problému

Pri analýze som sa pozrel na to ako sa dáva focus na elementy v html pomocou javascriptu.

Návrh riešenia

Pre focus na element sa použije javascript.

Opis implementácie

Do application.js bola pridaná funkcia setFocusOnComment(), ktorá sa zavolá pri zobrazení formulára a nastaví focus na input meno, v prípade ak je vyplnené, tak na text komentáru.

Testovanie

Testovanie bolo vykonávané pri zobrazovaní formulára na pridanie komentáru, či sa správne nastaví focus na meno resp. text.

Úloha #767 - Vyhladavanie - okienko bude priamo napravo hore + autocomplete Tomáš Háber

Popis úlohy

Presunúť vstup pre vyhľadávanie tak aby bol stále viditeľný na stránke. Obohatiť ho o

automatické doplňovanie.

Analýza problému

Kedže aplikácia už obsahuje fulltextové vyhľadávanie rozhodol som sa ho využiť aj pre účely automatického doplňovania.

Návrh riešenia

Použije sa existujúce vyȟladávanie pričom sa pre automatické doplnovanie bude z výsledkov vracať iba číslo zmluvy.

Opis implementácie

Do application.js som pridat funkciu autocomplete pre vstup vyhľadávania. Upravil som upravil som metódu vracajúcu výsledky tak aby pre formát js vracala ibá čísla nájdených zmlúv, každé na novom riadku.

Testovanie

Testovanie bolo vykonávané zadaním výrazu, ktorý sa nachádza v texte zmluvy a overením či je automaticky doplnené číslo správnej zmluvy.

#771 Klikátko na widget pri zmluve - Miroslav Polgár

Popis úlohy

Vytvoriť link, ktorý si používateľ vloží na stránku a zobrazí sa mu widget so zmluvou, ktorú mal používateľ zobrazenú.

Analýza problému

Je možné urobiť to viacerými spôsobmi. Najprv som spravil link, na ktorý používateľ klikne a zobrazí sa *textarea* s linkom. Potom sa členovia rozhodli, že bude vhodnejšie zobraziť *modálne dialógové okienko.*

Návrh riešenia

Samotný link na widget sa bude nachádzať v elemente „iframe“. Pre zobrazenie modal dialogu využijem knižnicu jQuery.

Opis implementácie

Samotný link na widget sa bude v „src“ parameteri HTML elementu iframe, čiže <iframe src=http:/ /...>. Link musí obsahovať id aktuálnej zmluvy. Link na *modal dialog* bude href="javascript:" .
V súbore application.js pridám funkciu $(#id).click(), kde #id je id linku. V tejto funkciu zavolám $(#id_textarea).modal(), kde id_textarea je id textového poľa, ktoré nie je na stránke zobrazené a zobrazí sa priamo v dialogu.

Testovanie

Funkčnosť overím kliknutím na link a vložením linku na widget do súboru html. Súbor otvorím vo webovom prehliadači. Musí sa zobraziť widget so zmluvou.

Úloha #760 - menu: hore nad horizontálnou čiarou - Filip Lörinc

Popis úlohy

Bude nutné presunúť zvislé menu z ľavého panelu na horizontálnu formu pod nadpisom.

Analýza problému

Táto úprava si vyžaduje jednoduchú úpravu HTML kódu a css štýlov.

Návrh riešenia

Riešenie spočíva v presune HTML tagov pridružených k menu na iné miesto a úpravu css štýlov im proslúchajúcich.

Opis implementácie

Menu bolo presunuté na iné miesto v tabuľke tak ako to bolo naznačené vyššie.

Testovanie

Zmena sa prejavila okamžite, preto testovanie nebolo nutné.

Úloha #763 - Úprava designu tak aby sa dali zväčšiť obrázky aspoň na šírku 800px - Filip Lörinc

Popis úlohy

Zväčšiť veľkosť obrázkov zmluvy na 800x600px.

Analýza problému

Táto úprava si vyžaduje jednoduchú úpravu HTML kódu.

Návrh riešenia

Úprava šírky v bloku kde sú zobrazené zmluvy.

Opis implementácie

Veľkosť obrázka bola zmenená v tagu zmenením width a height.

Testovanie
Zmena sa prejavila okamžite, preto testovanie nebolo nutné.

Šprint číslo 4

Úloha #788 - možnosť zmeniť názor na komentár like <->dislike
Stanislav Valachovič

Popis úlohy
Za komentáre sa dá hlasovať len 1x na každého používateľa (páči sami/nepáči sa mi). V prípade ak používateľ hlasoval za páči sa mi, bude mať možnosť zmeniť svoj názor na nepáči sa mi a naopak.

Analýza problému
V rámci analýzy som sa zamýšlal nad ukladaním jednotlivých hlasovaní.

Návrh riešenia
Do db bude pri hlasovaniach potrebné ukladať aj samotnú hodnotu (like/dislike). Následne pri pridávaní nového hlasovania, kontrolovať ak už používateľ hlasoval a teraz hlasuje sa opačnú možnosť, povoliť mu zmenu hlasu.

Opis implementácie
Do tabuľky comment_ips v db sa vytvoril nový stĺpec ‘rating’, kde sa ukladá hodnota hlasovania (like/dislike). Ak používateľ zahlasuje a už predtým hlasoval za tento komentár, skontroluje sa ešte, či nezadal opačnú hodnotu hlasu ako predtým. Ak áno hodnota hlasu sa zmení na opačnú.

Testovanie
Testovanie bolo vykonávané zmenami v hlasovaní s like na dislike a opačne a kontrolou či sa tieto zmeny vykonali správne.

Úloha #790 - mouse over na komentár Stanislav Valachovič

Popis úlohy
Po prechode myškou cez zvýraznený obdĺžnik na obrázku sa zvýrazní k nemu prislúchajúci komentár, takisto po prechode cez komentár sa zvýrazní k nemu prislúchajúci obdĺžnik.

Analýza problému

Pri analýze som sa pozrel na jquery funckiu hover().

Návrh riešenia

Každému obdĺžniku a komentáru sa pridá špecifický attribút podľa ktorého sa budú dať na seba naviazať. Následne sa im pridá hover funkcia.

Opis implementácie

Obdĺžniku bol pridaný attribút ‘data-comment’, ktorý obsahuje id komentáru, na ktorý je naviazaný a naopak komentáru bol pridaný atribút ‘data-s_area’. Následne do application.js do fukncie document.ready bola pridaná hover funkcia naviazaná na komentár a obdĺžnik, ktorá zvýrazní príslušný komentár/obdĺžnik.

Testovanie

Testovanie bolo vykonávané kontrolou, či sa po prejdení myškou cez obdĺžnik zvýrazní príslušný komentár a naopak.

Úloha #780 - Ukladanie info o zhliadnutých komentároch do databázy namiesto do cookie

Tomáš Háber

Popis úlohy

Zmeniť spôsob ukladania informácie o zhliadnutých komentárov tak aby sa už neuklade do *cookies* hodnoty ale do databázy.

Analýza problému

Rozdohodol som sa zachovať čo najviac už vytvorenej funkcie a zmenil iba spôsob ukladania informácií.

Návrh riešenia

Pre nového používateľa vygenerujem pernamentnú cookie, do ktorej uložím jednoznačný identifikátor používateľa. Tento identifikátor používam ďalej v databáze, kde k nemu ukladám informáciu o poslednom videnom komentári.

Opis implementácie

Vytvoril som model visits a upravil metódu, ktorá zobrazuje zmluvy. Doplnil som do nej vytváranie unikátnej cookie hodnoty a ukladanie zobrazených komentárov.

Testovanie

Testovanie bolo vykonávané vytvorením komentáru z iného prehliadača a následnej kontroly, či je nový komentár príslušne zvýraznený.

#770 Tagovanie: freeform + autocomplete existujucich tagov - Miroslav Polgár

Popis úlohy

V záhlaví zmluvy sa bude nachádzať textové pole s možnosťou napísať tag. Pri zadávaní sa zobrazia už existujúce tagy. Po potvrdení sa vedľa zobrazí príslušný tag.

Analýza problému

Problém sa týka databázy, keďže je potrebné si zapamätať už vložené tagy. Tagy by mali byť nezávisle dostupné a zároveň naviazané na konkrétnu zmluvu.

Návrh riešenia

V databáze vytvorím novú tabuľku s tagmi a ďalšiu s kľúčmi tagu a zmluvy. Vytvorím model Tag a model Contract_Tag. Modely pospájam vzťahmi a prepojím aj s modelom Contract . Na stránku vložím textové pole. Bude potrebný javascript aby sa tag zobrazil hneď po pridaní, bez reloadovania celej stránky. Pre autocomplete využijem funkciu jqueryui autocomplete().

Opis implementácie

Vytvorím migráciu databázy nasledovne:

- tabuľka Tag bude obsahovať id_tag a text. Budú v nej všetky tagy.
- prepojovacia tabuľka Contracts_tags bude obsahovať contract_id a tag_id.
- vytvorím tag_controller s metódami create a destroy
- do .html pridám textfield, ktorý bude volať metódu create. Tá pridá tag do databázy (ak neexistuje) a pridá vzťah do prepojovacej tabuľky. Následne metóda zavolá javascript, ktorý na stránku pridá nový tag.
- Textovému poľu v súbore application.js priradím funkciu autocomplete.

Testovanie

Do textového poľa napíšem nový tag. Stlačím enter a tag sa musí zobraziť na stránke. Znova začnem písať rovnaký tag. Musí sa zaobraziť roletka s predchádzajúcim tagom. Pokúsim sa ho pridať. Zobrazí sa upozornenie, že už existuje. Nakoniec stránku dám zobraziť znovu. Pokiaľ tag zostal na stránke, úloha je implementovaná správne.

Úloha #792 - integrácia s FOAF.sk **Filip Lörinc**

Popis úlohy

Bude nutné, pri vykreslovaní zmlúv, pridať linku pri názve firmy, ktorá bude odkazovať na vyhľadávanie na stránke foak.sk

Analýza problému

Pri vykreslovaní danej zmluvy sa vypisuje len názov dodávateľa. Miesto samotného výpisu bude nutné vykresliť link odkazujúc na www.foaf.sk/?q=meno_dodavatela s tým, že textový obsah linky bude korešpondovať s názvom dodávateľa.

Návrh riešenia

Bude nutné zmeniť <%= @contract.deliver %> v súbore contract/show.html.erb na <%=link_to @contract.deliver, "http://www.foaf.sk/?q="+@contract.deliver, :target => "_blank"%>.

Opis implementácie

Viď návrh riešenia.

Testovanie

Testovanie bolo vykonané na malom počte zmlúv. Pri niektorých dlhších a komplokovanejších názvoch firiem, sa linka prejavila ako nefunkčná respektíve vyhľadávač nedokázal rozpoznať napríklad medzeru ako reťazec %20. Tento defekt bol opravený zmenou reťazca %20 za reťazec +.

Šprint číslo 5

Bug #797 - zvýrazňovanie komentárov **Stanislav Valachovič**

Popis úlohy

Upraviť terajšie zvýrazňovanie komentárov a obdĺžnikov na iné - zmeniť farbu, zrušiť blikanie, na obdĺžniku nastaviť opacity.

Analýza problému

V rámci analýzy som hľadal vhodnú farbu pre zvýraznenie.

Návrh riešenia

Použije sa nová farba a namiesto blikania sa zvýrazní komentár borderom.

Opis implementácie

Po prechode cez obdĺžnik/komentár sa pridé class ‘hover’, v ktorej je nastavený nový border-color prostredníctvom css.

Testovanie

Testovanie bolo vykonávané kontrolou, či sa po prejdení myškou cez komentár/obdĺžnik zmení border.

Bug #802 - komentár sa dá do popredia Stanislav Valachovič

Popis úlohy

Pri zvýraznení komentára sa komentár dá do popredia (po prechode cez obdĺžnik), ale po prechode cez komentár sa komentár samotný nedá do popredia.

Analýza problému

Po prechode cez komentár sa mu nenastavuje vyšší z-index, ako pri prechode cez obdĺžnik.

Návrh riešenia

Po prechode cez komentár sa mu nastavý vyšší z-index.

Opis implementácie

Po prechode cez komentár (alebo po prechode cez obdĺžnik) sa všetkým komentárom nastaví z-index:1 a komentáru, ktorý sa má zvýrazniť z-index:2.

Testovanie

Testovanie bolo vykonávané kontrolou, či sa po prejdení myškou cez komentár dá do popredia.

Bug #805 - autocomplete vyhľadávanie - zobrazovať viac info o zmluve

Tomáš Háber

Popis úlohy

Pri automatickom doplnovaní sa aktuálne zobrazí iba číslo zmluvy, čo neposkytuje dostatočné množstvon informácií o zmluve pre používateľa.

Analýza problému

Potrebnou zmenou je doplniť riadok s číslom zmluvy o dalšie informácie. A to kokrétne o dodávateľa a klienta. Nepochybne zaújmavou informáciou je určite aj cena uvedené k zmluve. Pre zlepšenie použiteľnosti navrhujem aby sa po kliknutí na riadok v autocomplete presmerovala stránka priamo na konkrétnu zmluvu a nie na výsledky vyhľadávania tak ako je to teraz.

Návrh riešenia

Potrebné je použiť autocomplete plugin z JQuery UI, nakoľko aktuálne používaný nie je možné dostatočne jednoducho prispôsobiť. Potrebné je upraviť funkciu, ktorá zabezpečuje zobrazenie riadku v autocomplete.

Opis implementácie

Upravil som funkciu _renderItem, tak aby zobrazila okrem čísla zmluvy aj informácie o dodávateľovi a klientovi danej zmluvy. Zobrazuje sa aj cena, ktorá je zadaná k zmluve. Tieto informácie su zasielané vo formáte JSON a získavané sú z ako klasického vyhľadávanie pomocou ThinkingSphinx.

Testovanie

Testovanie bolo vykonávané kontrolou, či sa po prejdení myškou cez obdĺžnik zvýrazní príslušný komentár a naopak.

#806 Bug Možnosť vymazať tag Miroslav Polgár

Popis úlohy

Po prejdení myšou cez tag sa zobrazí krížik, po ktorého stlačení sa tag vymaže.

Analýza problému

Problém sa týka databázy. Vymazanie tagu by ale malo vymazať len prepojenie tagu so zmluvou, keďže môže byť používaný inými zmluvami. Tag bude treba zmeniť na link.

Návrh riešenia

Využijem metódu destroy v Tag_controller-i. Tagom priradím id, aby ich bolo možné identifikovať. Bude potrebné zadefinovať funkciu .hover v javascripte, ktorá zobrazí krížik. Krížik bude linkom na metódu destroy v tag_controller-i. V metóde sa vymaže vzťah z databázy

zavolá sa aj javascript, ktorý vymaže tag zo stránky.

Opis implementácie

Tagy na stránku budem vkladať každý do samostatného elemetnu . Jeho id bude identifikovať prislúchajúci tag. Do neho vložím aj obrázok (krížik) s linkom na metódu destroy v tag_controlleri. Do application.js pridám funkciu .hover(), v ktorej bude zobrazenie a skrytie obrázka. V metóde destroy vymažem záznam v prepojovacej tabuľke. Následne zavolám javascript, ktorý vymaže element s príslušným tagom podľa id tagu.

Testovanie

Po pridaní nového tagu overím zobrazovanie a schovávanie krížiku vedľa všetkých tagov. Po kliknutí na krížik musí tag zmiznúť zo stránky. Po znovu-zobrazení stránky sa tam už nesmie nachádzať.

Bug #804 Zmeniť link na modal dialog - Miroslav Polgár

Popis úlohy

Po kliknutí na link „Vložte si zmluvu na stránku“ sa zobrazí okienko s linkom na widget.

Analýza problému

Problém sa týka stránky so zmluvou.

Návrh riešenia

Využijem funkciu dialog() knižnice jqueryUI.

Opis implementácie

Samotný link na widget sa bude v „src“ parameteri HTML elementu iframe, čiže <iframe src=http://...>. Link musí obsahovať id aktuálnej zmluvy. Link na *modal dialog* bude href="javascript:" . V súbore application.js pridám funkciu $(#id).click(), kde #id je id linku. V tejto funkciu zavolám $(#id_textarea).modal(), kde id_textarea je id textového poľa, ktoré nie je na stránke zobrazené a zobrazí sa priamo v dialogu.

Testovanie

Funkčnosť overím kliknutím na link a vložením linku na widget do súboru html. Súbor otvorím vo webovom prehliadači. Musí sa zobraziť widget so zmluvou.