

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Tím č. 9

Dokumentácia – Inžinierske dielo

Tímový Projekt II.

Členovia tímu: Bc. Martin Fukas, Bc. Ľubomír Samotný, Bc. Michal Selický, Bc. Michal Polák
Bc. Marek Števuliak, Bc. Michal Števuliak, Bc. Nghia Pham van

Vedúci tímu: Ing. Róbert Móro, PhD.

Máj 2018

Obsah

Úvod	3
Globálne ciele pre zimný semester	4
Globálne ciele pre letný semester	5
Celkový pohľad na systém	6
Technológie	6
Komponenty	6
Fyzická dátová vrstva	10
Logická dátová vrstva	16
Proces sťahovania zmlúv z CRZ	20
Proces sťahovania zmlúv z Egov	26
Opis modulov	29
Sledovanie aktivity prihláseného používateľa - vyhľadávanie	29
Zobrazenie údajov o subjekte	32
Sťahovanie zmlúv z centrálného registra zmlúv pomocou xml generovaného súboru.	36
Skrytie vybraných zmlúv pred verejnosťou	42
Editácia sprievodných dát v zmluve	46
Zobrazenie údajov o subjekte - samostatná stránka	48
Pridanie RPVS	51
Nahradenie používania regisu	52
Nové možnosti rozšíreného vyhľadávania zmlúv	54
Vylepšenie normalizácie názvu dodávateľov a objednávateľov	56
Editácia scanu a fulltextu zmluvy	57
Hromadné skrytie vybraných zmlúv pred verejnosťou	61
Prílohy:	65
Príloha A: Používateľská príručka	65
Príloha B: Návod na inštaláciu a nasadenie projektu	73

Príloha C: Návrhy používateľského rozhrania (mockupy) - zobrazenie informácií o subjekte	79
Príloha D: Technická dokumentácia od Michala Barlu	81
Príloha E: Návod na pridanie dát z RPO a RPVS	87
Príloha F: Zmeny a plán aktualizácie RoR 3.2 -> 4.0	89
Príloha G: Návod pre vývojové prostredie a rozbehanie projektu lokálne	93

Úvod

Otvorené Zmluvy je investigatívna stránka Aliancie Fair Play ktorej úlohou je pomôcť občanom čítať, vyhľadávať a posudzovať zmluvy uzatvorené štátom a štátnymi inštitúciami.

Ponúka občanom pokročilé vyhľadávanie, prezeranie, komentovanie a sledovanie zmlúv na centralizovanom portáli. Portál Otvorené Zmluvy čerpá zmluvy a iné dodatočné informácie z viacerých zdrojov ako napríklad z Centrálného Registra Zmlúv a portálu Egov

V rámci tímového projektu sme rozšírili tento projekt a aktualizovali jeho prvky. V nasledujúcich kapitolách sú opísané naše globálne ciele za zimný semester, architektúra existujúceho systému a moduly o ktoré sme systém rozšírili.

Globálne ciele pre zimný semester

Naším prvým cieľom bol setup projektu a procesu vývoja. V zimnom semestri sme chceli definovať všetky potrebné procesy a vytvoriť všetky potrebné metodiky ktoré budeme potrebovať pri vývoji. Taktiež sme chceli vytvoriť vhodné vývojové prostredie a oboznámiť sa s existujúcou verziou projektu pomocou analýz a prototypov.

Naše ďalšie ciele boli rozšírenie existujúceho portálu o novú funkcionality. Patrílo sem zlepšenie vyhľadávania a zvýšenie komfortu používateľa pri vyhľadávaní napríklad pomocou vylepšenia existujúceho fazetového vyhľadávania, evidovania častých dopytov, vytvorenia návrhov dopytov a iných.

Taktiež sme chceli rozšíriť informácie o subjektoch získavaním informácií z dostupných registrov ako napríklad register právnických osôb, register účtovných závierok a iných. Prepojením týchto dát s portálom sme chceli zvýšiť množstvo informácií ktoré budú poskytnuté používateľom. S týmto cieľom súviselo aj automatizované zlepšenie kvality dát a metadát, vo forme identifikovania a párovania súvisiacich zmlúv a ich dodatkov.

Medzi posledné ciele patrila údržba a aktualizácia technológií ktoré sú použité v projekte. V projekte boli použité veľmi staré verzie technológií ako napríklad Elasticsearch verzia 0.9 a Ruby verzia 1.9.3, ktoré sme chceli aktualizovať na novšie verzie.

Globálne ciele pre letný semester

V zimnom semestri sa nám podarilo splniť niektoré ciele opísané v predchádzajúcej časti. Vytvorili sme nový spôsob sťahovania zmlúv z Centrálného Registra Zmlúv pomocou publikovaného XML súboru. Do projektu sme pridali Register Právnických Osôb ako nový zdroj dát. Tieto informácie sme použili na rozšírenie informácií o subjektoch zmluvy. Podarilo sa nám čiastočne aktualizovať zastaralé verzie technológií ako napríklad Ruby, Rails a Elasticsearch, ale v tomto procese sme pokračovali aj v letnom semestri a dokončenie aktualizácií patrilo medzi naše najvyššie priority, pretože sme chceli zabrániť situácií kedy by sme museli prerobiť moduly ktoré sme vytvorili aby fungovali na novších verziách.

Medzi ďalšie ciele ktoré sme v letnom semestri určili patrilo ďalšie rozšírenie informácií o subjektoch pomocou Registra Partnerov Verejného Sektoru. Pre lepší prehľad sme tiež chceli vytvoriť samostatný pohľad na zobrazenie relevantných informácií o subjektoch zmluvy. Taktiež sme chceli nahradiť použitie nefunkčného Regisu v projekte pomocou informácií z Registra Právnických Osôb.

Po konzultácií s Alianciou Fair-Play sme chceli rozšíriť možnosti administrátora o možnosť skrývania zmlúv. Taktiež sme chceli pridať možnosť editácie zmlúv - sprievodných metadát, fulltextu zmluvy a aj scanu zmluvy.

Pre nové zdroje dát sme chceli taktiež rozšíriť možnosti vyhľadávania. Register Partnerov Verejného Sektoru obsahuje informácie o konečných užívateľov výhod a verejných funkcionároch. Pridať možnosť vyhľadávania podľa týchto parametrov patrilo medzi naše ciele.

Medzi posledné ciele patrili úpravy existujúcich modulov ako napríklad mapovanie ID rezortov na ich názov pre intuitívnejšie vyhľadávanie a opravovanie testov.

Celkový pohľad na systém

Technológie

Ruby 2.1.10

- programovací jazyk, používaný pre Otvorené zmluvy

Rails 4.2.10

- web framework pre Ruby

PostgreSQL 10.1

- relačná databáza pre uchovávanie údajov

Redis 3.0.1

- nerelačná databáza, ktorú využíva technológia resque - ukladanie názvov funkcií a príslušných parametrov

Resque 1.23.1 a Resque scheduler 2.0.0

- správa workerov, ktorí asynchrónne vyberajú z redisu uložená dáta v podobe funkcií s parametrami a spúšťajú ich

Elasticsearch 6.0.0

- nerelačná databáza pre uchovávanie documents a pages
- vyhľadávanie v zmluvách, fazetové vyhľadávanie
- tvorí veľkú časť logiky backendu

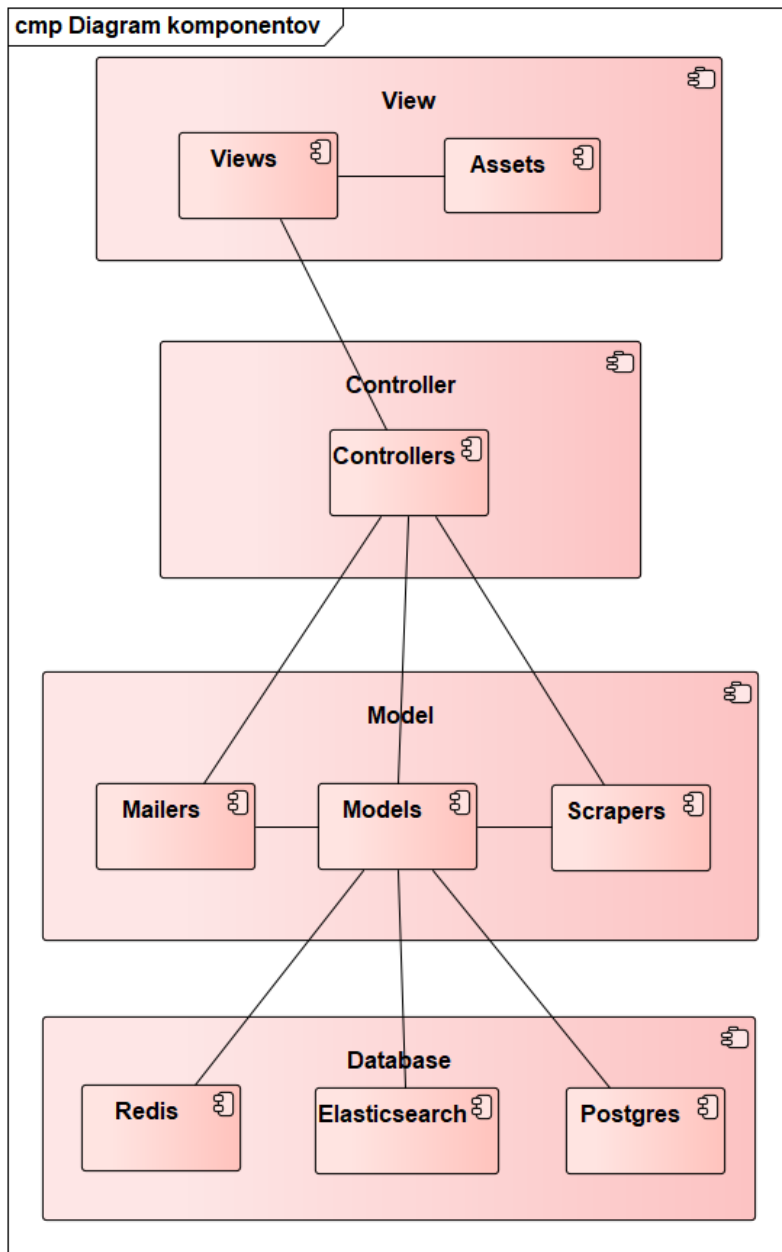
Komponenty

- Views
 - zahŕňa .html.erb, pomocou ktorých sa na portáli Otvorené zmluvy zobrazujú dáta
 - definuje webové rozhranie, jeho štruktúru
 - Verejné webové rozhranie
 - Administratívne rozhranie
- Assets
 - komponent používaný komponentom Views
 - obsahuje .js, .css súbory a obrázky využívané frontendom
- Controllers

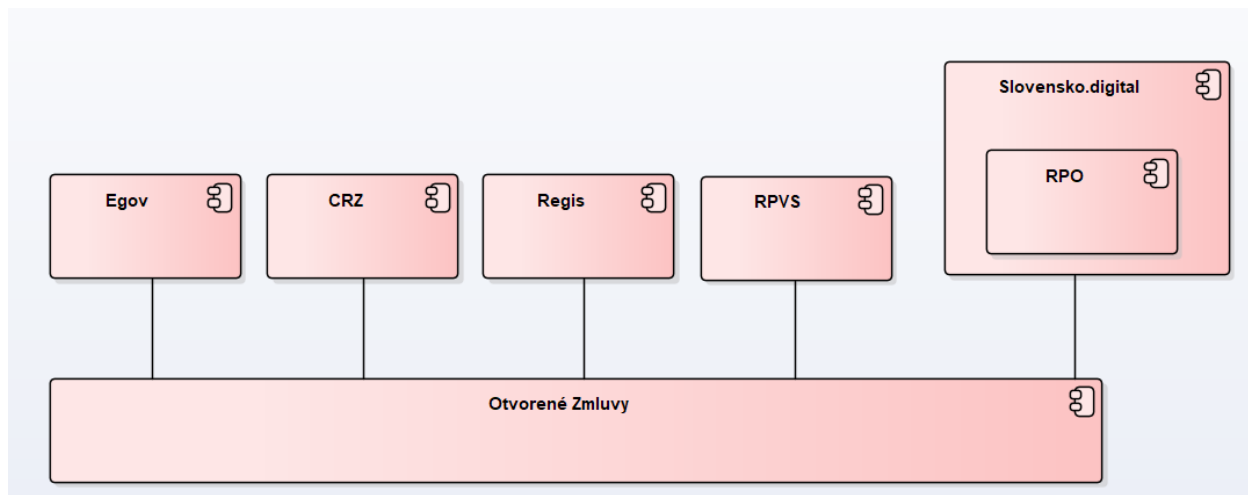
- Prepája Views s Models
- Obsahuje funkcie pre získavanie údajov, ktoré následne vyvolajú ich zobrazenie na stránke
- Volá funkcie z Mailers a Scrapers
- Mailers
 - funkcie pre notifikovanie cez email
- Scrapers
 - scapovanie dát z portálov CRZ, Egov, Regis
- Postgres
 - uchovávanie všetkých potrebných údajov
 - využíva ho komponent Models pre získavanie perzistentných dát
 - bližšie informácie o tabuľkách v časti **Fyzická dátová vrstva**
- Elasticsearch
 - využívaný veľkou časťou komponentu Models pre vyhľadávanie v zmluvách
 - má v projekte samostatné triedy pre definovanie, konfigurácie, mapovanie a prácu s indexami
 - nové zmluvy sú sem automaticky vkladané
- Redis
 - nerelačná databáza pre služby Resque

Diagram komponentov môžeme vidieť na obrázku 1. Ruby on Rails používa architektonický vzor model-view-controller.

Podstatnú časť systému tvoria dokumenty a entity s nimi spojené. V nasledujúcich diagramoch je možné vidieť pôvodný stav, ktorý navrhli predchádzajúci autori. Entity doplnené našim tímom sú zvýraznené odlišnou farbou (modrá) a ich význam je špeciálne popísaný v texte k danému obrázku.



Obr. 1: Diagram komponentov systému



Obr. 2: Kontext systému

Na obr. 2 môžeme vidieť externé zdroje, ktoré používajú otvorené zmluvy pre získavanie informácií. Aktualizovali sme tieto zdroje pridaním RPO (register právnických osôb), ktorý spadá pod Slovensko digital.

- Egov, CRZ (Centrálny register zmlúv) - zdroje sťahovania zmlúv
- Regis
 - zdroj informácií o právnických osobách
 - po analýze sa zistilo, že je nefunkčný, nahrádzame s RPO
- RPO (Register právnických osôb)
 - zdroj informácií o právnických osobách
 - informácie o dodávateľoch
- RPVS (Register Partnerov Verejného Sektoru)
 - zdroj informácií o konečných užívateľoch výhod a verejných funkcionároch

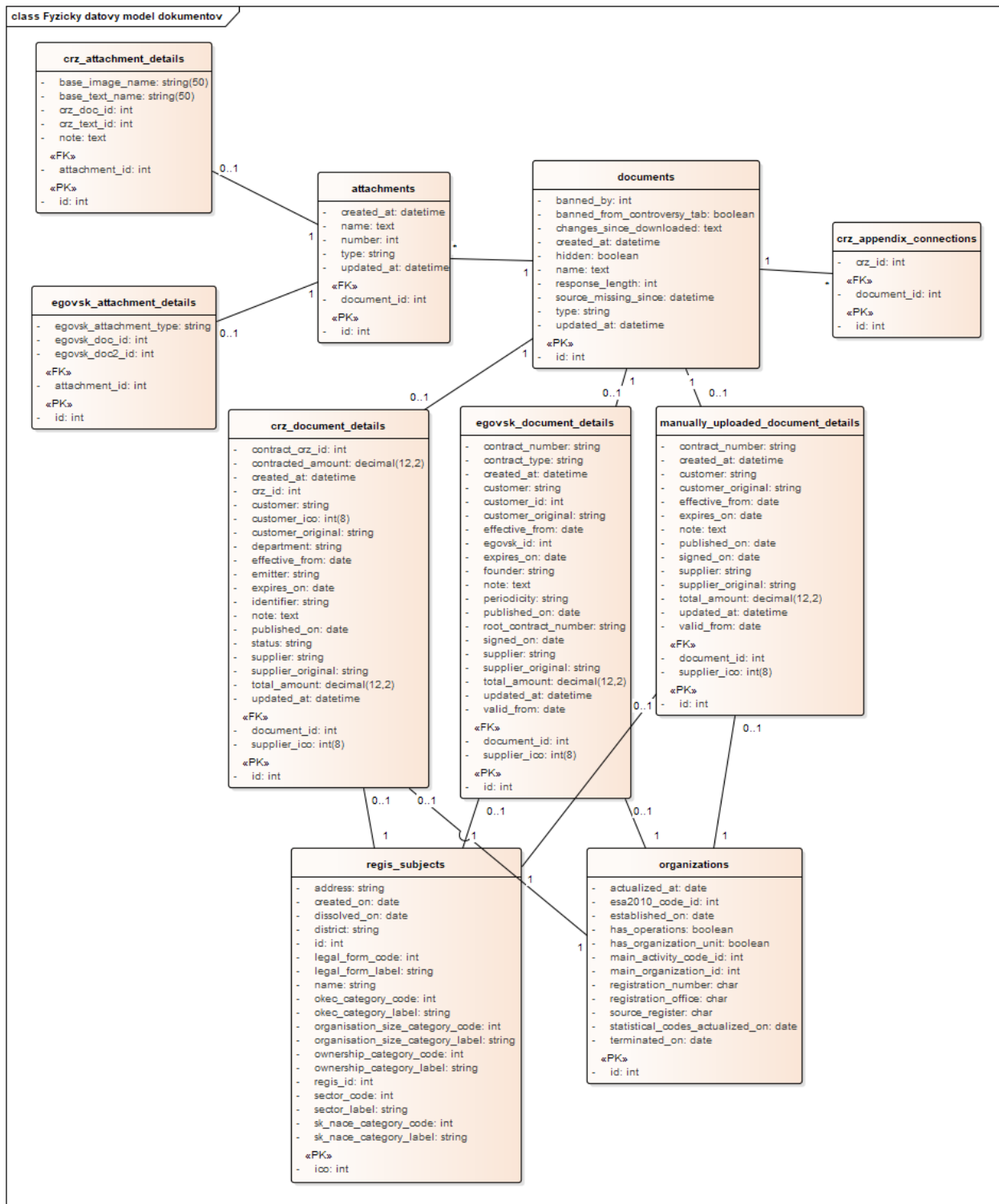
Fyzická dátová vrstva

Na obr. 3. je možné vidieť prepojenia rôznych typov dokumentov. Každý dokument má príbuzné dokumenty a prílohy (*attachments*). Prílohy delíme do dvoch logických celkov - CRZ a Egov. Toto delenie je prítomné kvôli spôsobu sťahovania zmlúv, keďže projekt využíva Centrálny register zmlúv a aj portál Egov. Nad rámec prílohy môže CRZ dokument odkazovať na ďalšie dokumenty (napríklad dodatky), takže dokument má väzbu aj na entitu *crz_appendix_connections*, ktorá takéto prepojenia uchováva.

Dokumenty majú základné informácie obsiahnuté v sebe, avšak v detaile dokumentu môžeme nájsť doplnujúce informácie. Detaily sa delia podobne ako prílohy na CRZ, Egov a manuálne uložené. Každý dokument teda na základe spôsobu stiahnutia môže obsahovať najviac jeden detail dokumentu. Tieto detaily sú nakoniec spárované s informáciami z portálu Regis, z dôvodu zjednotenia názvov spoločností opísaných v zmluve. Často sa totiž stávalo, že rovnaká firma mala svoj názov zapísaný v rozdielnom názve, čo práve spárovanie s Regisom vyriešilo.

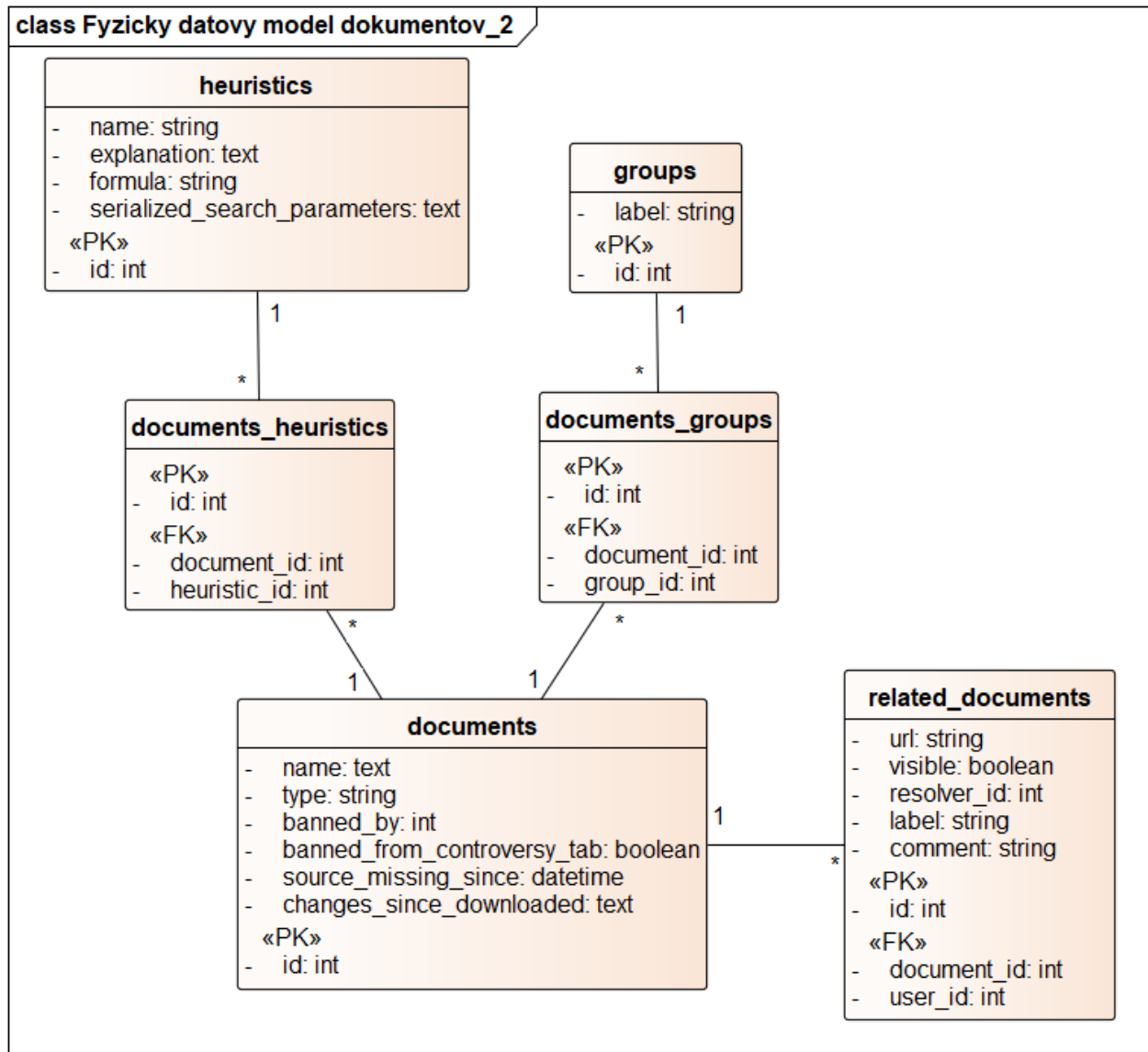
V rámci projektu bol v zimnom semestri pridaný modul RPO (register právnických osôb) ako náhrada za register Regis. Fyzický dátový model bol preto rozšírený o tabuľky zo slovensko.digital, kde sa práve modul RPO nachádza. Dokopy sa jedná o 24 nových tabuliek a bližšie informácie je možné nájsť v časti **Rozšírenie projektu o dáta zo Slovensko.digital RPO.**

V letnom projekte sme pridali modul RPVS (register partnerov verejného sektora) aby sme získali informácie o konečných užívateľoch výhod a verejných funkcionároch subjektov zmluvy a pomocou nich vedeli vyhľadávať zmluvy. Viac informácií je v časti **Pridanie RPVS.**



Obr. 3: Fyzický dátový model dokumentov, previazanie typov dokumentov

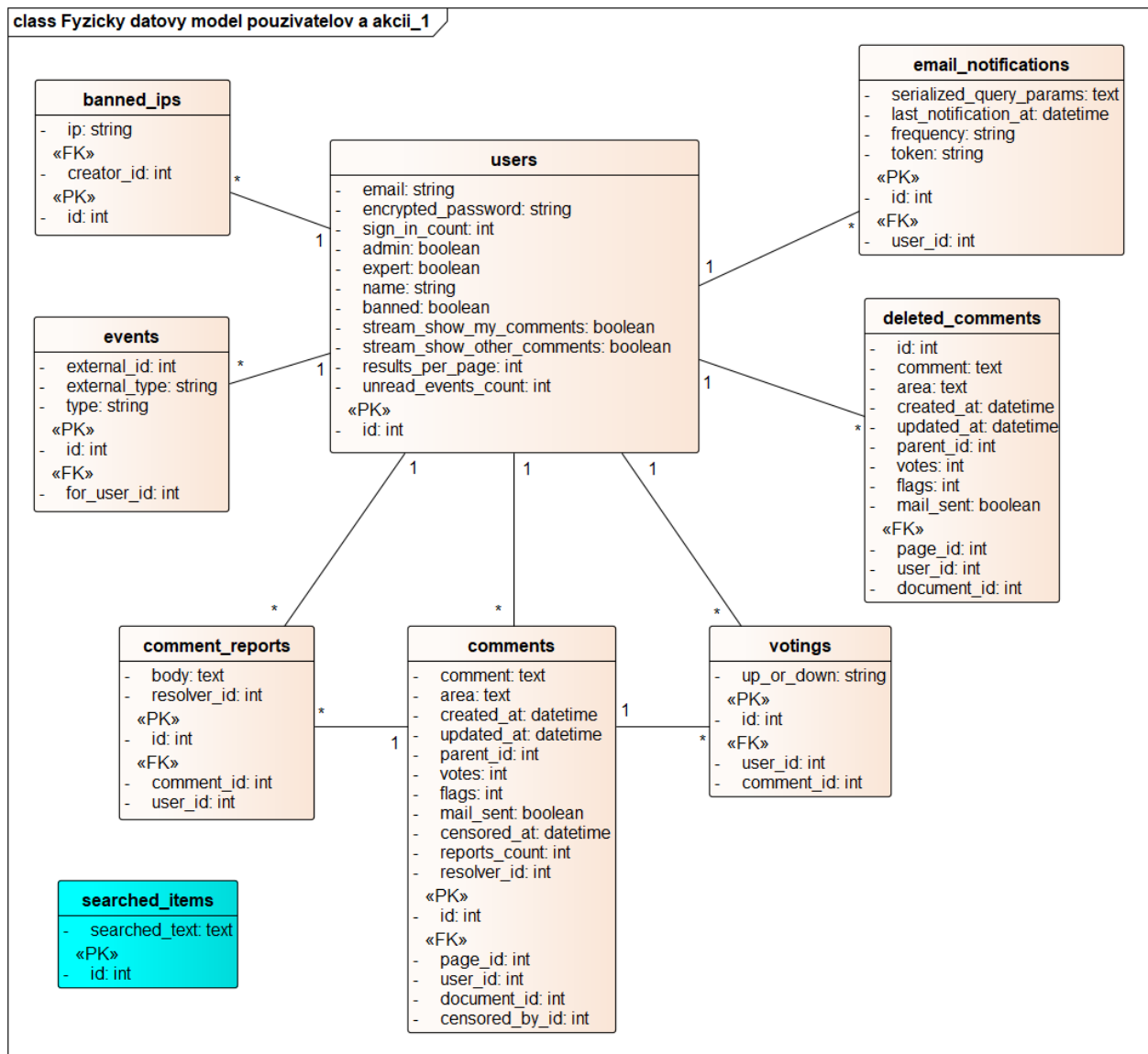
Nad rámec týchto prepojení sú dodatočné entity zobrazené na obr. 4. Každý dokument patrí do nejakej skupiny a môže mať pridelenú heuristiku. Heuristika slúži na automatické odhaľovanie podozrivých zmlúv na základe určených pravidiel (napríklad dlhé a nákladné zmluvy). Dokumenty je tiež možné spájať prostredníctvom entity *related_documents*.



Obr. 4: Fyzický dátový model dokumentov, doplnková časť

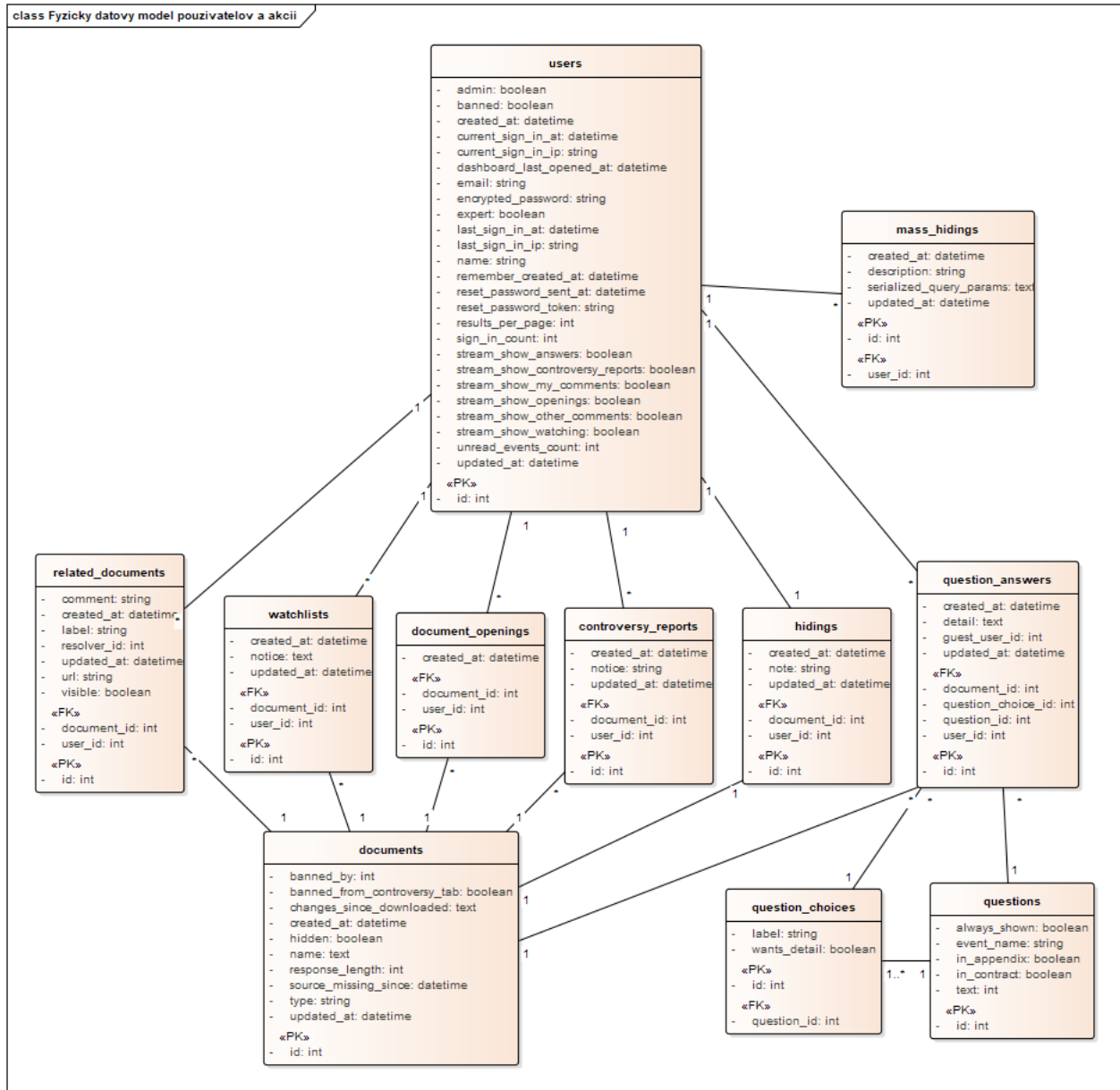
Dôležitou časťou systému sú aj používatelia. Medzi základné akcie používateľa patrí komentovanie zmlúv (dokumentov), hlasovanie za iné komentáre a nahlasovanie nevhodných komentárov, čo je zobrazené na obr. 5. Nedodržiavanie pravidiel môže viesť až k zabanovanej IP adrese. Používateľ si môže vyžiadať zasielanie notifikácií na emailovú adresu a udalosti s ním spojené sa uchovávajú v entite *events*. Podrobný rozbor používateľských akcií je možné vidieť v prílohe A.

V rámci projektu sme pridali entitu *searched_items* (bližšie opísané v časti **Sledovanie aktivity prihláseného používateľa**). Každá inštancia *searched_items* má odkaz v tabuľke *events* v atribúte *external_id*.



Obr. 5: Fyzický dátový model používateľov a akcií, udalosti a komentáre

Ako je však možné vidieť z obr. 6, medzi dokumentmi a používateľmi existuje viacero spojení. Používatelia si môžu vybrané dokumenty otvárať, uložiť do záložky, nahlasovať ich pochybnosť alebo sa zapájať pri zodpovedaní otázok. Administrátor dokáže skryť z zverejniť zmluvu. Otázky majú viacero možností odpovede a každý používateľ tak môže zodpovedať na vybranú otázku. Jeho voľba sa uchová v entite *question_answers* a viaže sa na konkrétny dokument.

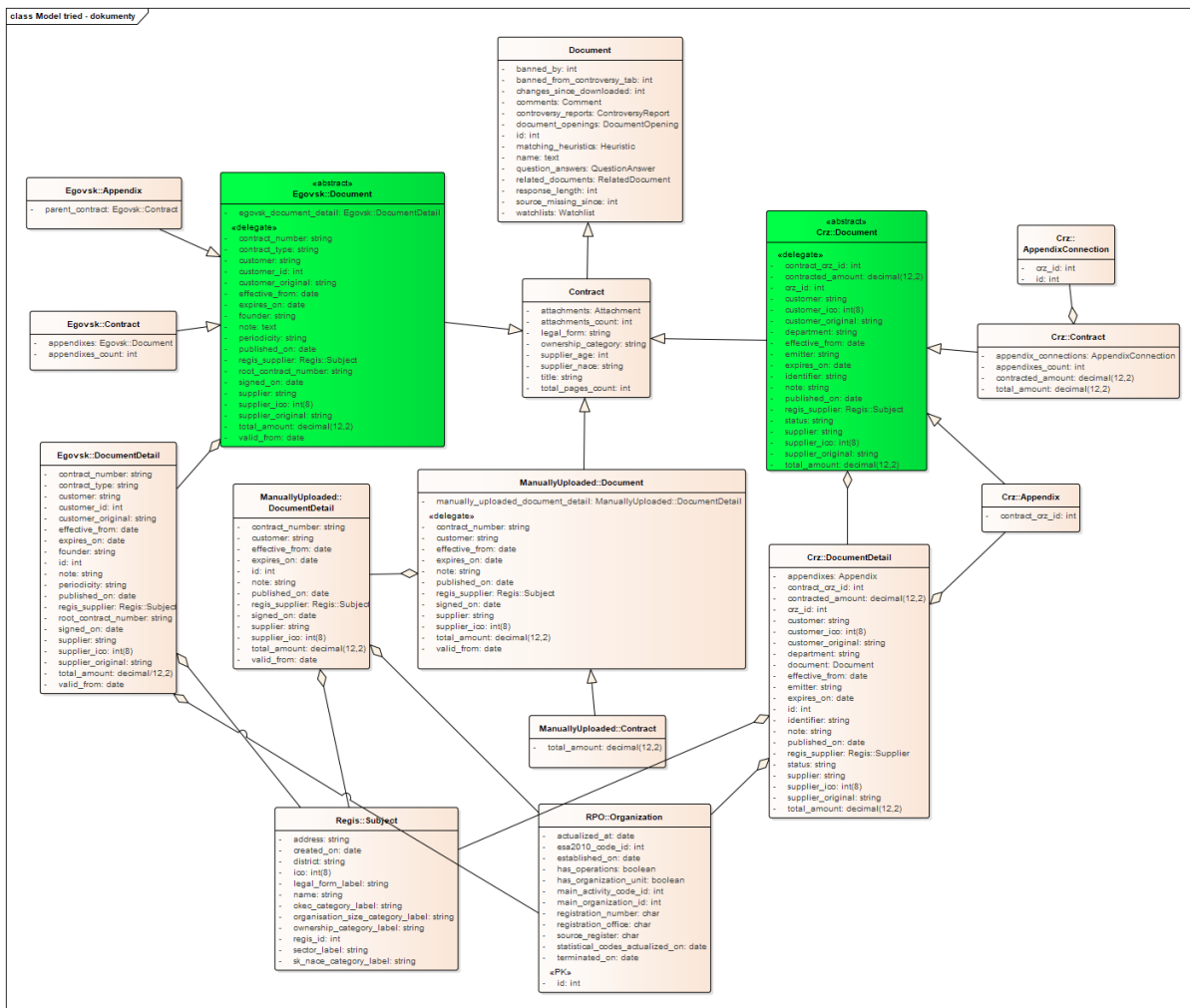


Obr. 6: Fyzický dátový model používateľov a akcií, prepojenia s dokumentmi

Logická dátová vrstva

Diagram na obr. 7 zobrazuje logický dátový model pre dokumenty a ich rôzne typy, teda Egovsk, CRZ alebo manuálne nahrané. Ako je vidieť, všetky typy sú odvodené od spoločnej triedy *Contract*, ktorá ako jediná priamo dedí vlastnosti a atribúty od materskej triedy *Document*.

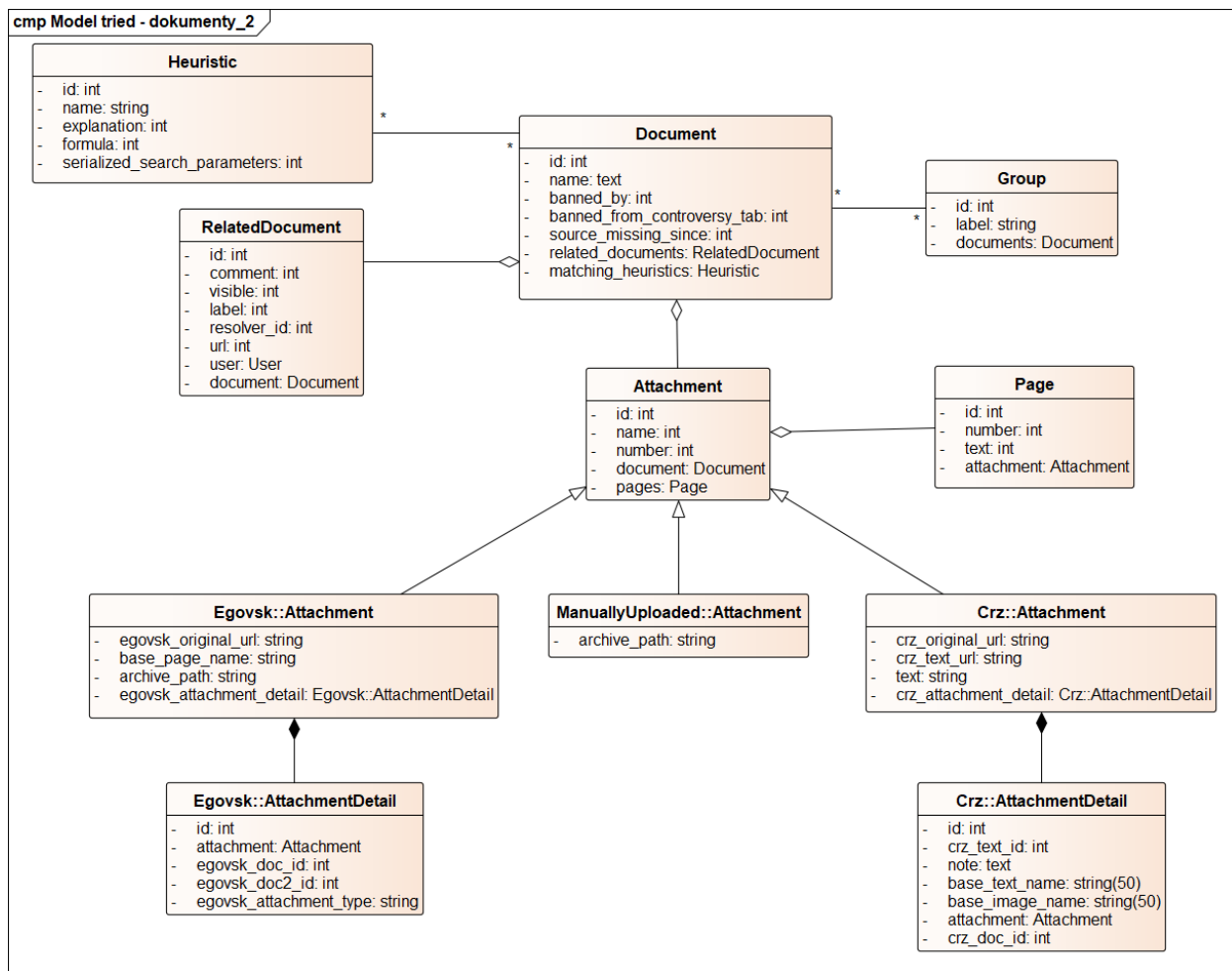
Každá vetva iného druhu dokumentu má vlastnú verziu *DocumentDetail*, kde sa nachádzajú mierne odlišné informácie o danej zmluve. Tento detail v sebe obsahuje aj informáciu o dodávateľovi (*Regis::Subject* a *RPO::Organizations*). Rovnako majú všetky typy dokumentu vlastný *Contract* a externe sťahované *Egovsk* a *Crz* dokumenty aj *Appendix*. Pre vetvu s *Crz* dokumentmi je existuje aj trieda *AppendixConnection*, ktorá jednoducho prepája rôzne prílohy.



Obr. 7: Diagram tried dokumentov, rôzne typy dokumentov

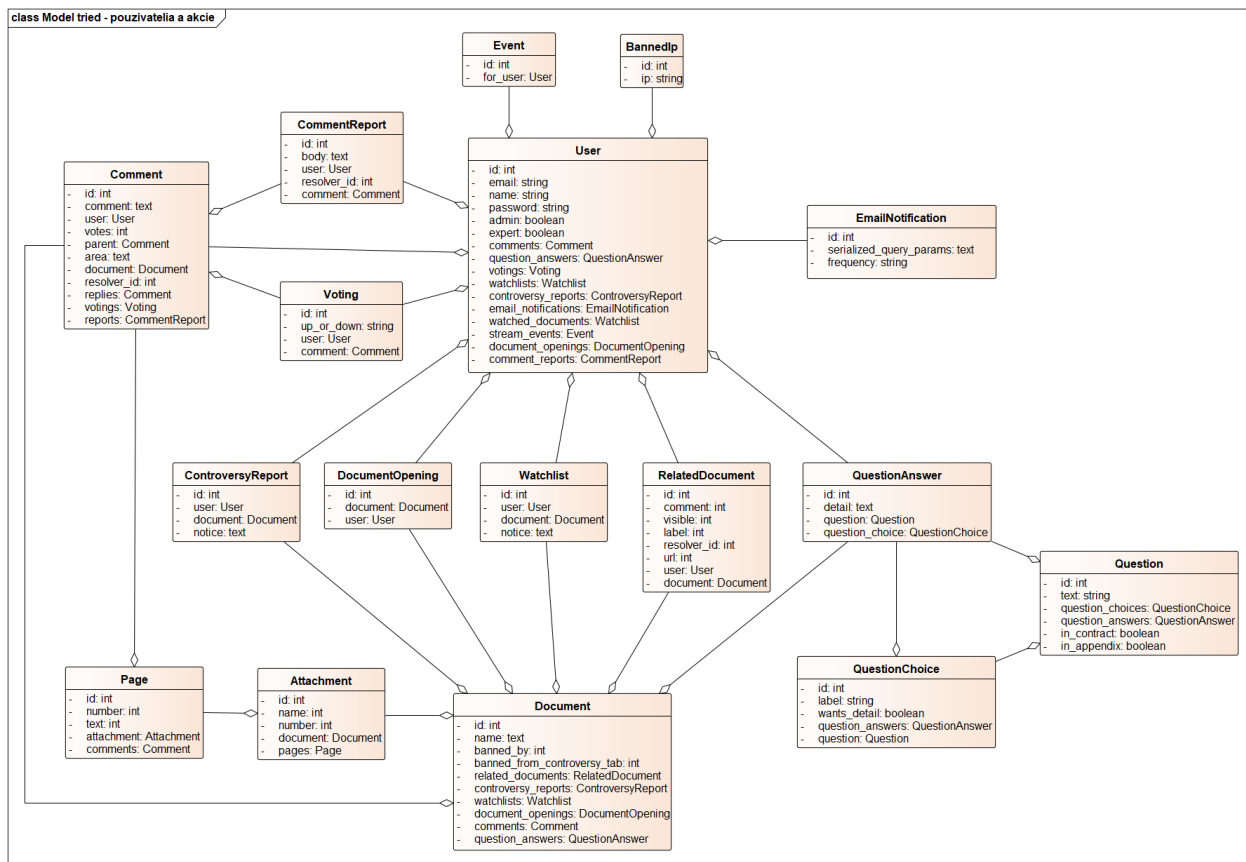
Na obr. 8 sa nachádza doplnkový diagram pre pohľad na logickú dátovú vrstvu zameraný na dokumenty. Tu je vidieť, že *Document* agreguje *Attachment*, príbuzné dokumenty cez triedu *RelatedDocument* a rovnako obsahuje aj väzby na heuristiky a skupiny.

Attachment následne agreguje list strán (trieda *Page*) a od tejto triedy sa opäť rozdeľujú 3 rôzne vetvy v závislosti na type materského dokumentu. Každý odvodený *Attachment* tak môže mať bez problémov rozdielne atribúty a externe sťahované dokumenty *Egovsk* a *Crz* majú aj vlastný *AttachmentDetail*, kde sú dodatočné informácie.



Obr. 8: Diagram tried dokumentov, rozdelenie príloh dokumentov

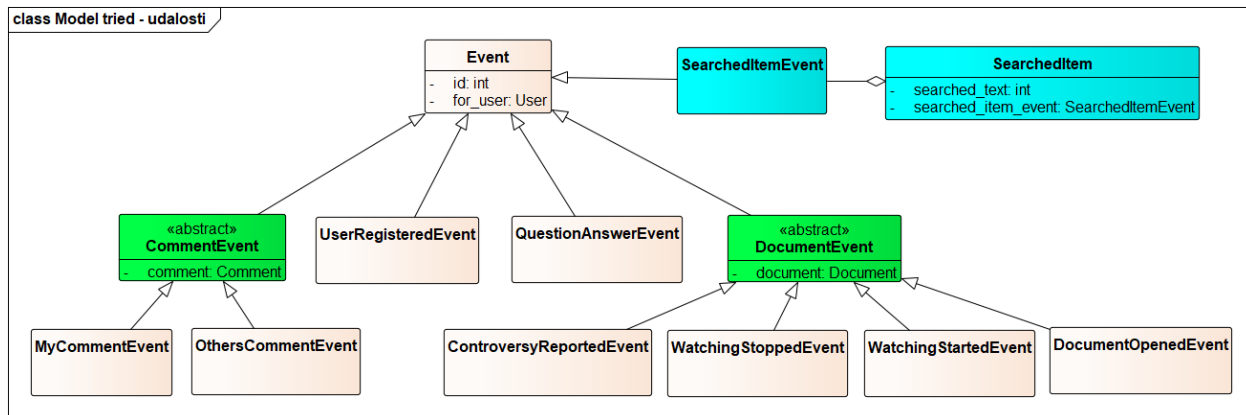
Na obr. 9 je možné vidieť rozpracovanie fyzického dátového modelu pre používateľov a akcie z obr. 5 a 6. V tomto prípade sa zachovala štruktúra daná fyzickým dátovým modelom a jednotlivé entity boli premenené na samostatné triedy. Základný popis je teda rovnaký ako v časti o fyzickej dátovej vrstve.



Obr. 9: Diagram tried použivateľov a akcií

Nakoľko rôzne udalosti v systéme sa pri diagrame tried rozrástli, boli umiestnené do samostatného diagramu na obr. 10. Tu je vidieť základné delenie udalostí, kde koncové udalosti (triedy *UserRegisteredEvent* a *QuestionAnswerEvent*) sú implementované ako konkrétne triedy a tie vetviace sa (triedy *CommentEvent* a *DocumentEvent*) sú iba abstraktnými a potrebujú ďalšie delenie.

Túto hierarchiu sme rozšírili o vlastné triedy, konkrétne sa jedná o triedy *SearchedItemEvent* a *SearchedItem*. Trieda *SearchedItemEvent* priamo dedí od materskej triedy *Event* a vďaka väzbe na *SearchedItem* máme uchované dodatočné informácie. Viac podrobností o tomto rozšírení sa nachádza v časti **Sledovanie aktivity prihláseného používateľa**.

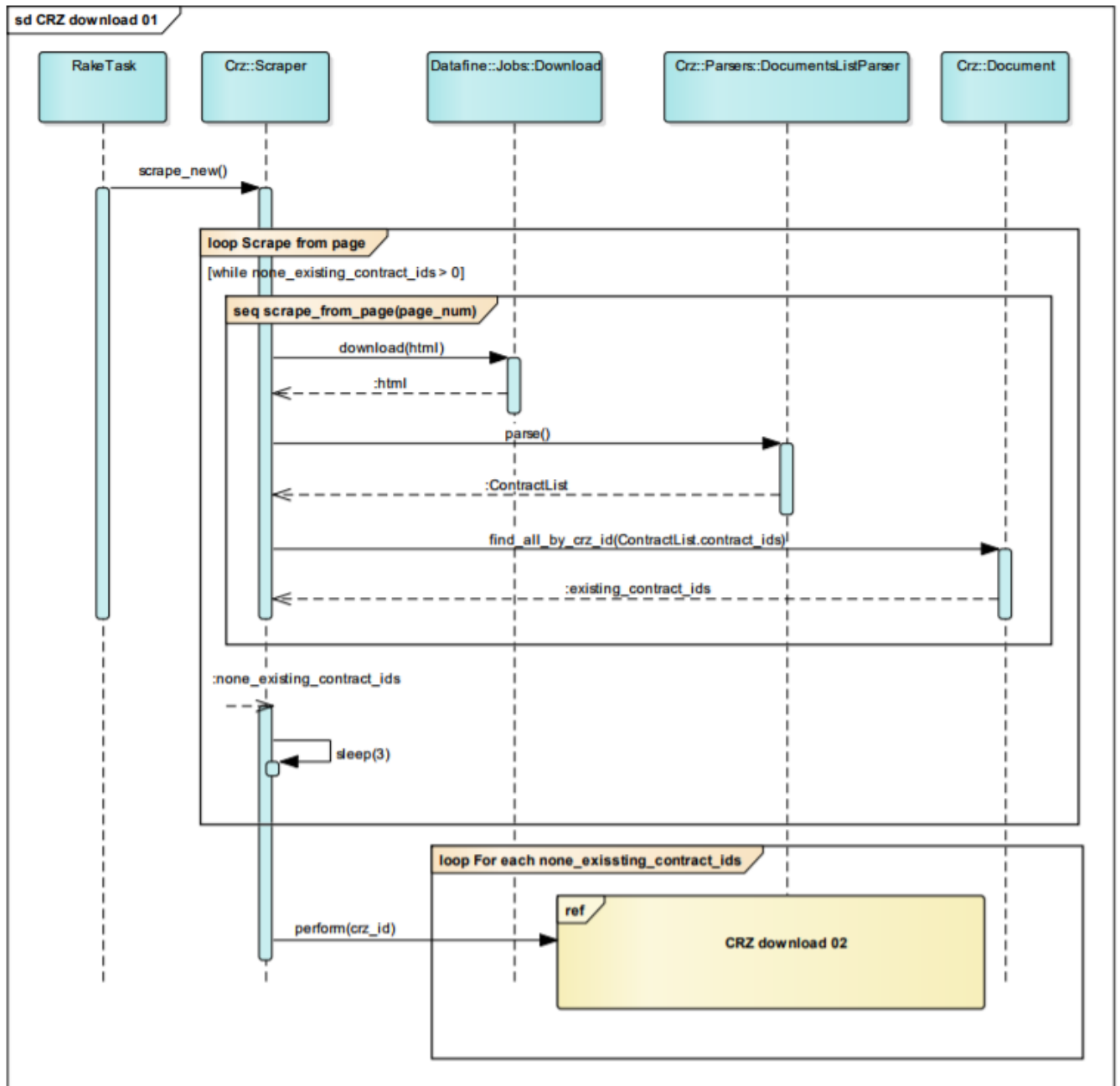


Obr. 10: Diagram tried - udalosti

Proces sťahovania zmlúv z CRZ

Jednou z najpodstatnejších častí portálu otvorených zmlúv je získavanie dát od tretích strán, ktorými sú štátne inštitúcie, ktoré zverejňujú dáta o uzavretých zmluvách medzi štátnymi inštitúciami navzájom alebo s tretím stranami. Výsledkom procesu je získanie takzvaných metadát, čo sú sprievodné dáta o zmluve (dohodnutá čiastka, zmluvné strany, ich ičo, dátum zverejnenia a platnosti, stav zmluvy). Najpodstatnejšie sú však informácie obsiahnuté v dokumentoch (zmluvách), ktoré sú nahrané na portál v podobe pdf dokumentu. Tento proces ich stiahne a pomocou OCR vyextrahuje text z pdf danej zmluvy pre lepšie vyhľadávanie. Pri procese sťahovania zmluvy sa môže jednať o novú zmluvu, alebo dodatok (apendix - úprava zmluvy iným dokumentom). Každá zmluva má prílohy (attachments), čo je vlastne pdf reálnej zmluvy, niekedy aj s ďalšími dodatkami a zmenami, ktoré pôvodnú zmluvu upravujú.

Niektoré procesy v tomto projekte sú spúšťané periodicky. To sa deje pomocou cron job (časový plánovač udalostí v systémoch založených na UNIXe). Tieto joby sú definované v súbore config/schedule.rb. Následne tieto cron joby vyvolajú tasky definované v súbore lib/tasks/crowdcloud.rake. Jeden z týchto taskov je sťahovanie nových (doteraz nestiahnutých) zmlúv z portálu <http://www.crz.gov.sk/>. Celé jeho vykonávanie je zobrazené na štyroch sekvenčných diagramoch uvedených nižšie. Zobrazený proces sťahovania bol len zdokumentovaný pre lepšie pochopenie fungovania systému a jedná sa o časť, ktorá už bola implementovaná predchádzajúcimi tvorcami tohto softvéru. Modifikácie tohto procesu, ktoré sme vykonali v rámci tímoveho projektu sú uvedené

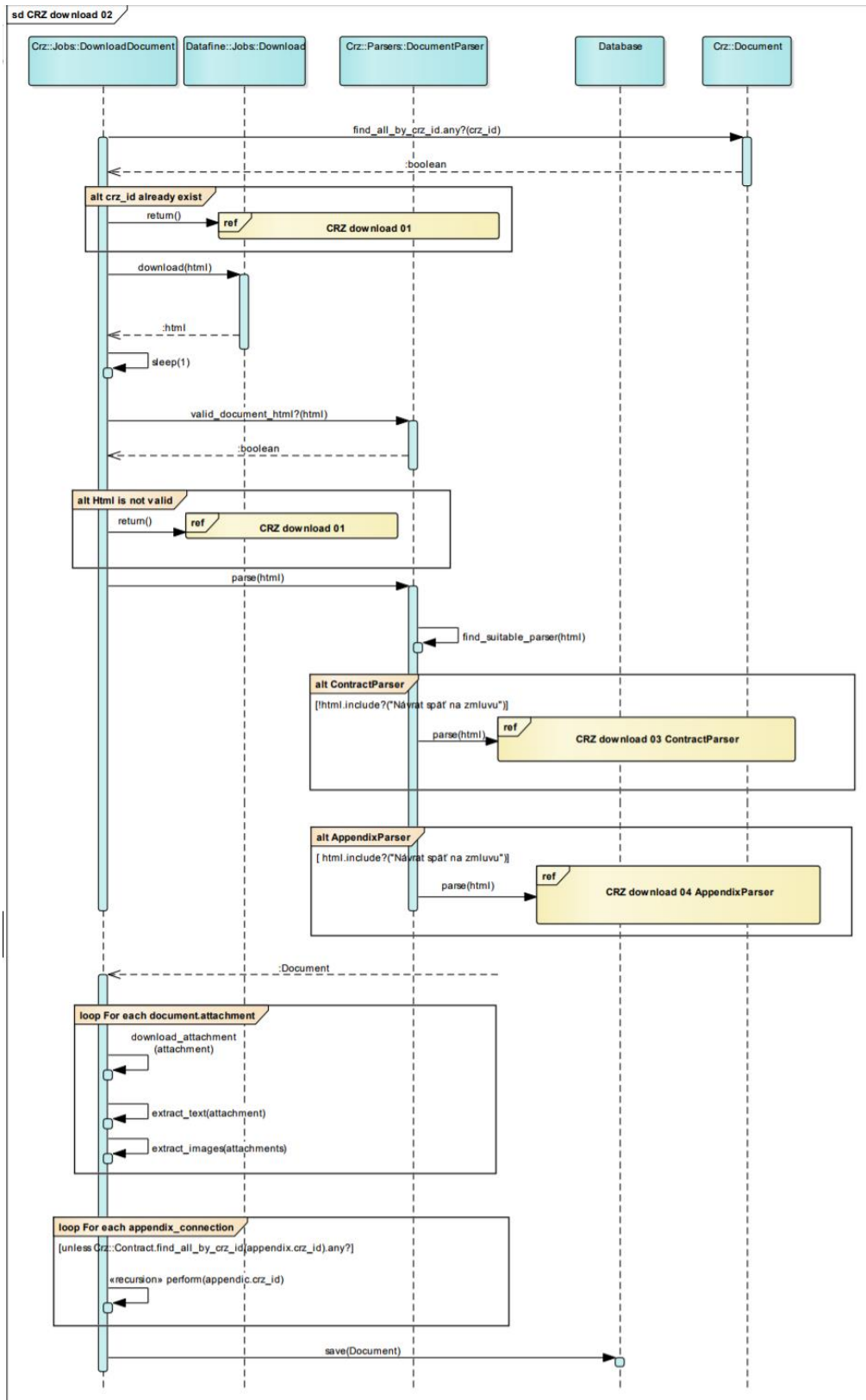


Obr. 11: Sekvenčný diagram sťahovania z CRZ 1

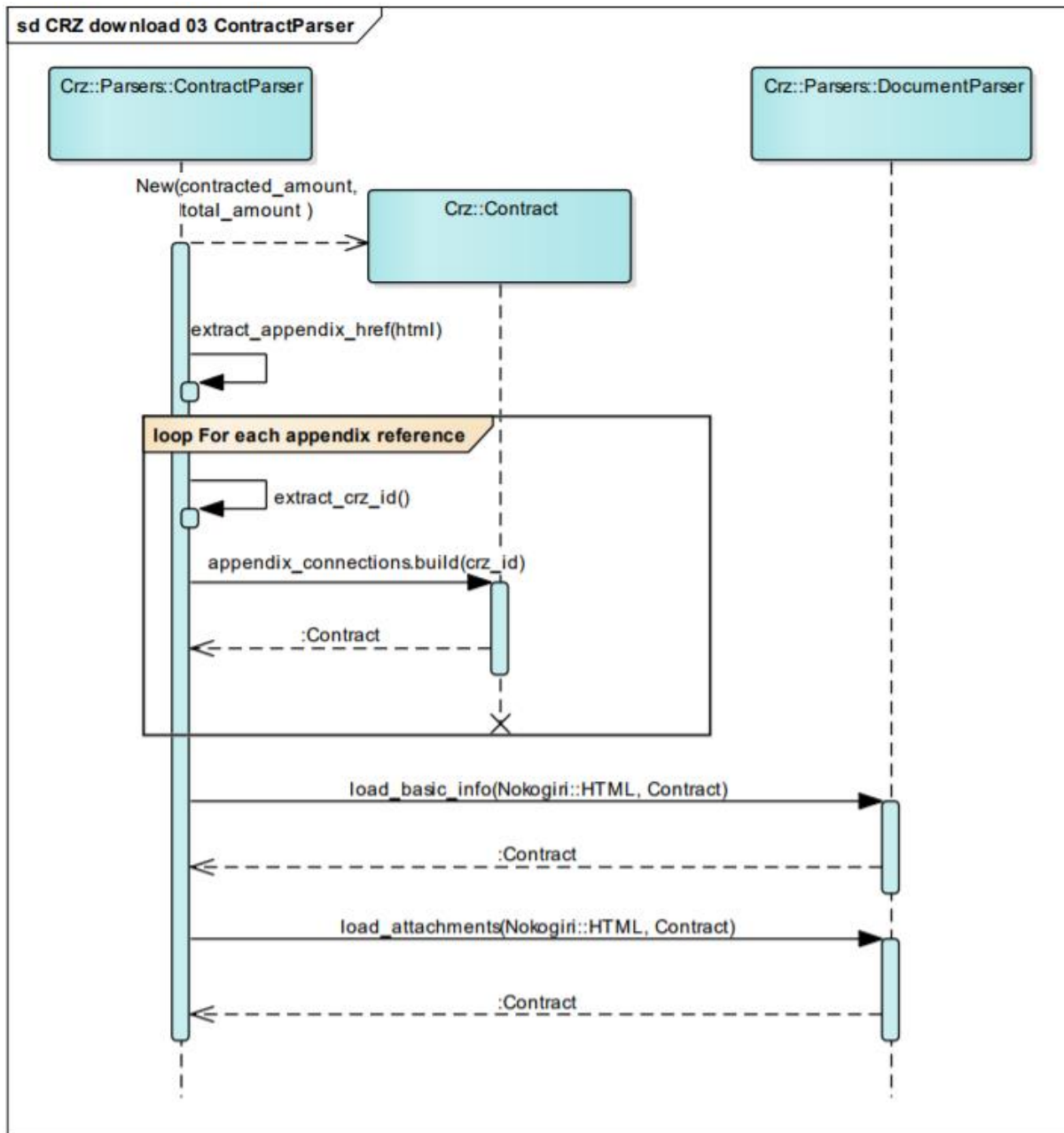
Zo stránky, kde sa nachádzajú zmluvy zoradené podľa dátumu najnovších zmlúv, sa stiahne zoznam s 20 zmluvami. Vyparsujú sa id daných zmlúv. Tento proces pokračuje, pokiaľ sa všetky zmluvy na danej stránke (teda 20), už nachádzajú v databáze. Sekvenčný diagram k tomuto procesu je možné vidieť na obrázku 7.

Pre každé id zmluvy sa následne spraví kontrola id, stiahne sa html stránka zmluvy a podľa obsahu sa použije buď AppendixParser alebo ContractParser, podľa toho, či sa jedná o novú zmluvu alebo nejaká

zmena či dodatok zmluvy. Po tomto sa pre každú prílohu stiahne každá príloha, extrahuje sa z nej text a obrázky. Ak sa jednalo o Appendix a vytvorili sa pripojenia na ďalšie appendixy tak sa rekurzívne vykonáva proces tohto sekvenčného diagramu pre každé prepojenie. Nakoniec sa celý dokument (metadata) uložia do databázy. Tento proces je možné vidieť na obrázku 8,

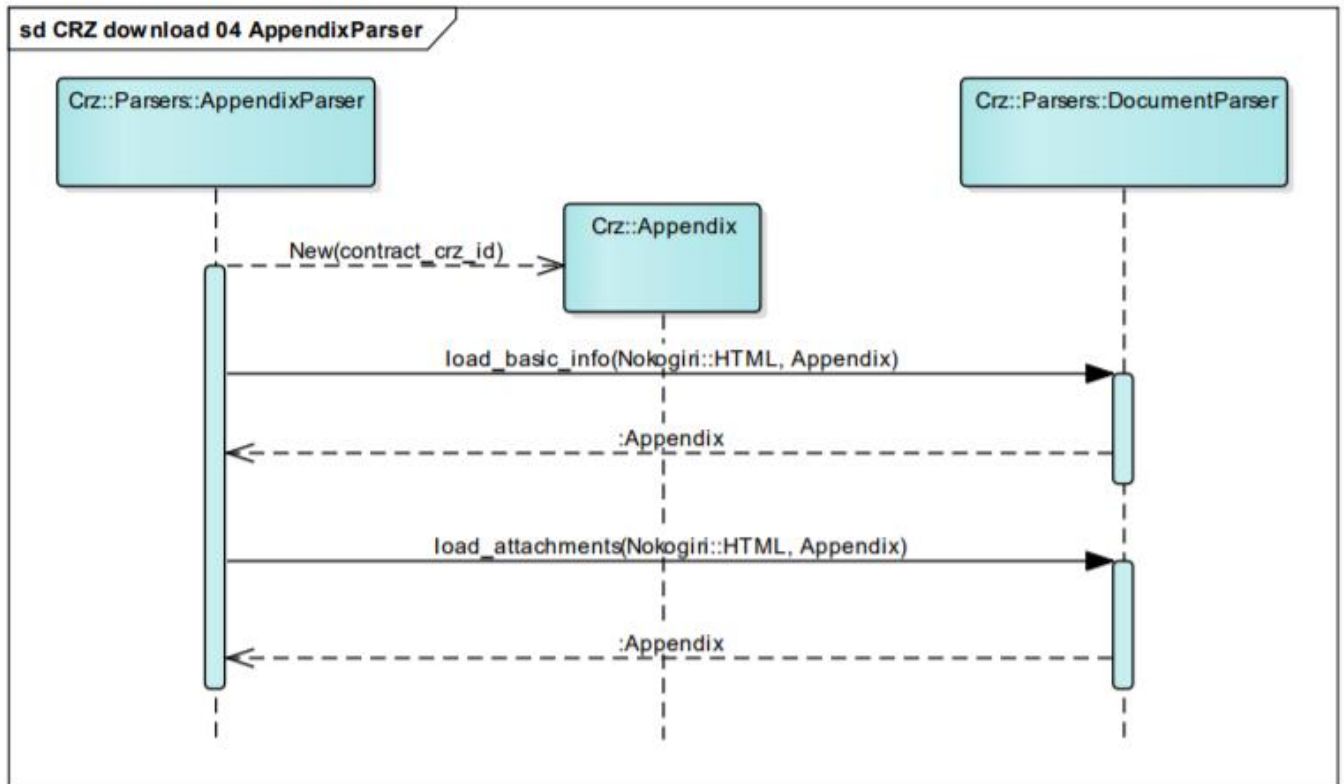


Obr. 12: Sekvenčný diagram sťahovania z CRZ 2



Obr. 13: Sekvenčný diagram sťahovania z CRZ 3

Ak sa jedná o novú zmluvu, tak ak sú k nej nejaké appendixy tak sa extrahujú url referencie, idečka a vytvoria sa prepojenia. Načítajú sa základné metadata z html stránky o zmluve a potom o prílohách k nej. Tento proces je možné vidieť na obrázku 9,

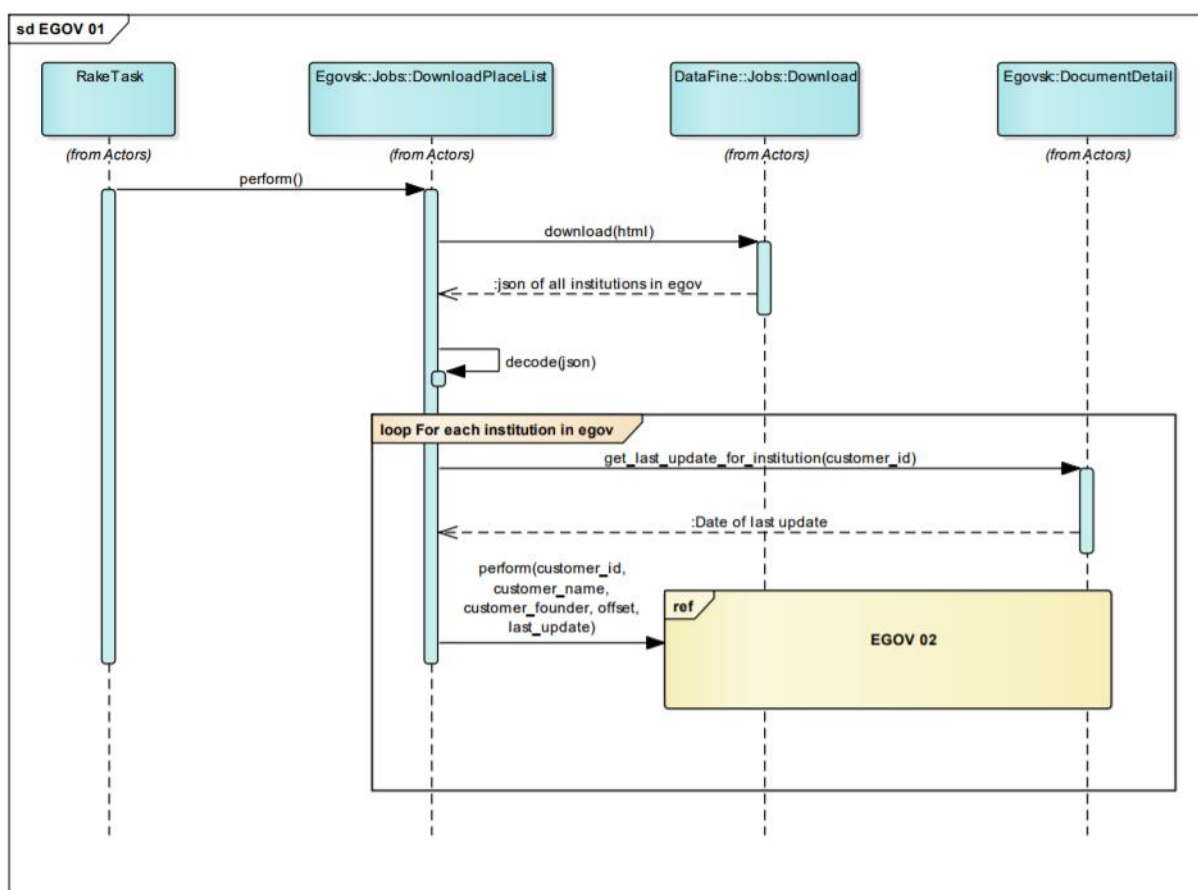


Obr. 14: Sekvenčný diagram ťťahovania z CRZ 4

Na obrázku 10 môžeme vidieť proces vytvorenia objektu `Appendix`, do ktorého sa načítajú o ňom základné metadáta a taktiež aj k jeho prílohám.

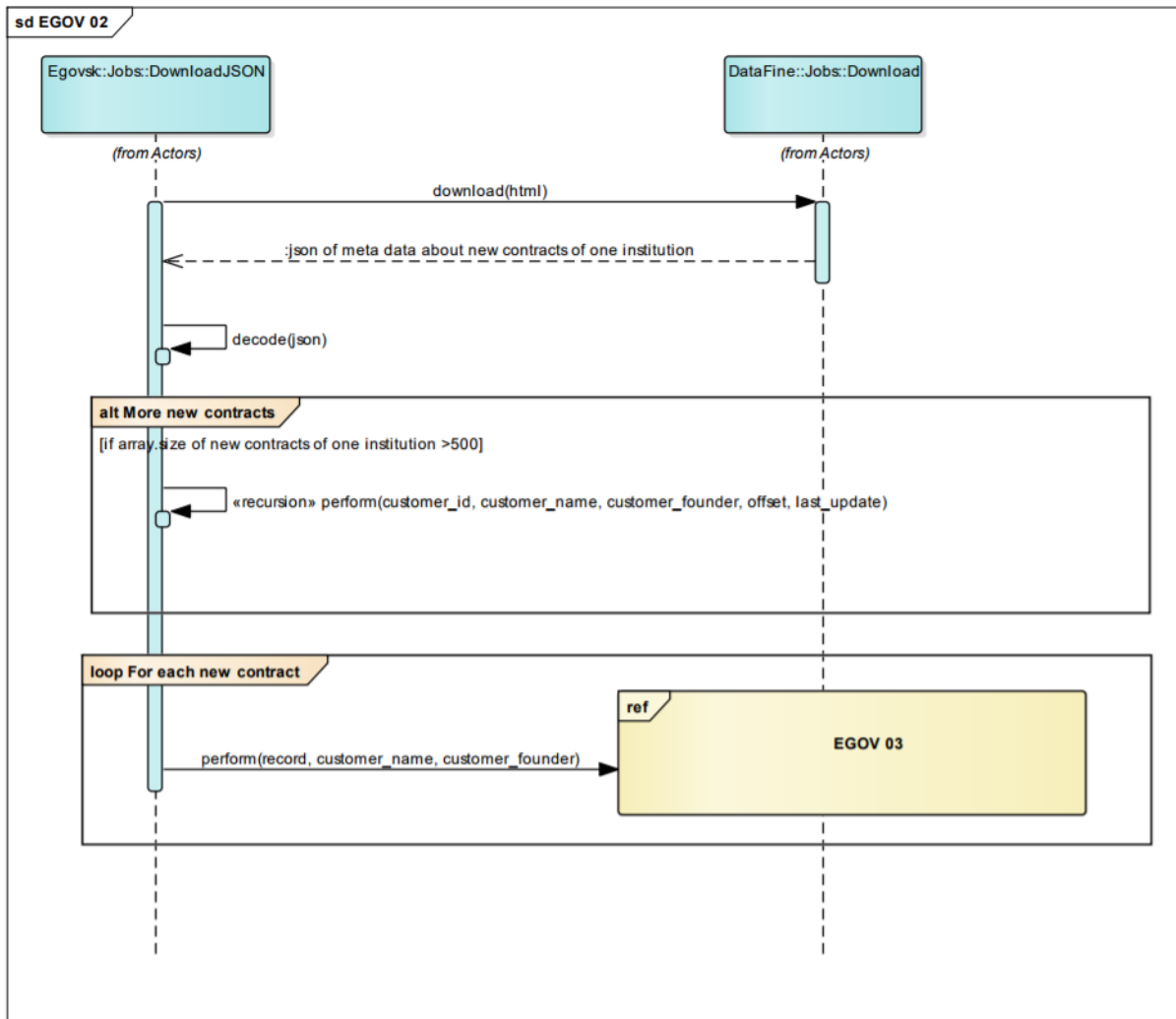
Proces sťahovania zmlúv z Egov

Ďalším procesom, ktorým sa získavajú zverejnené štátne zmluvy je ich sťahovanie z portálu <http://portal.egov.sk/>. Tento proces bol taktiež implementovaný predchádzajúcimi vývojarmi, a bol zdokumentovaný pre lepšie pochopenie fungovania. Základný princíp je veľmi podobný ako pri sťahovaní nových zmlúv z portálu CRZ. Stiahnu sa metadáta, ktoré su v tomto prípade vo formáte json, a k nim sa stiahnu pdf prílohy zmluvy (attachments). Taktiež buď sa jedná o novú zmluvu alebo o dodatok (appendix), ktorý nejakým spôsobom upravuje pôvodne znenie zmluvy.



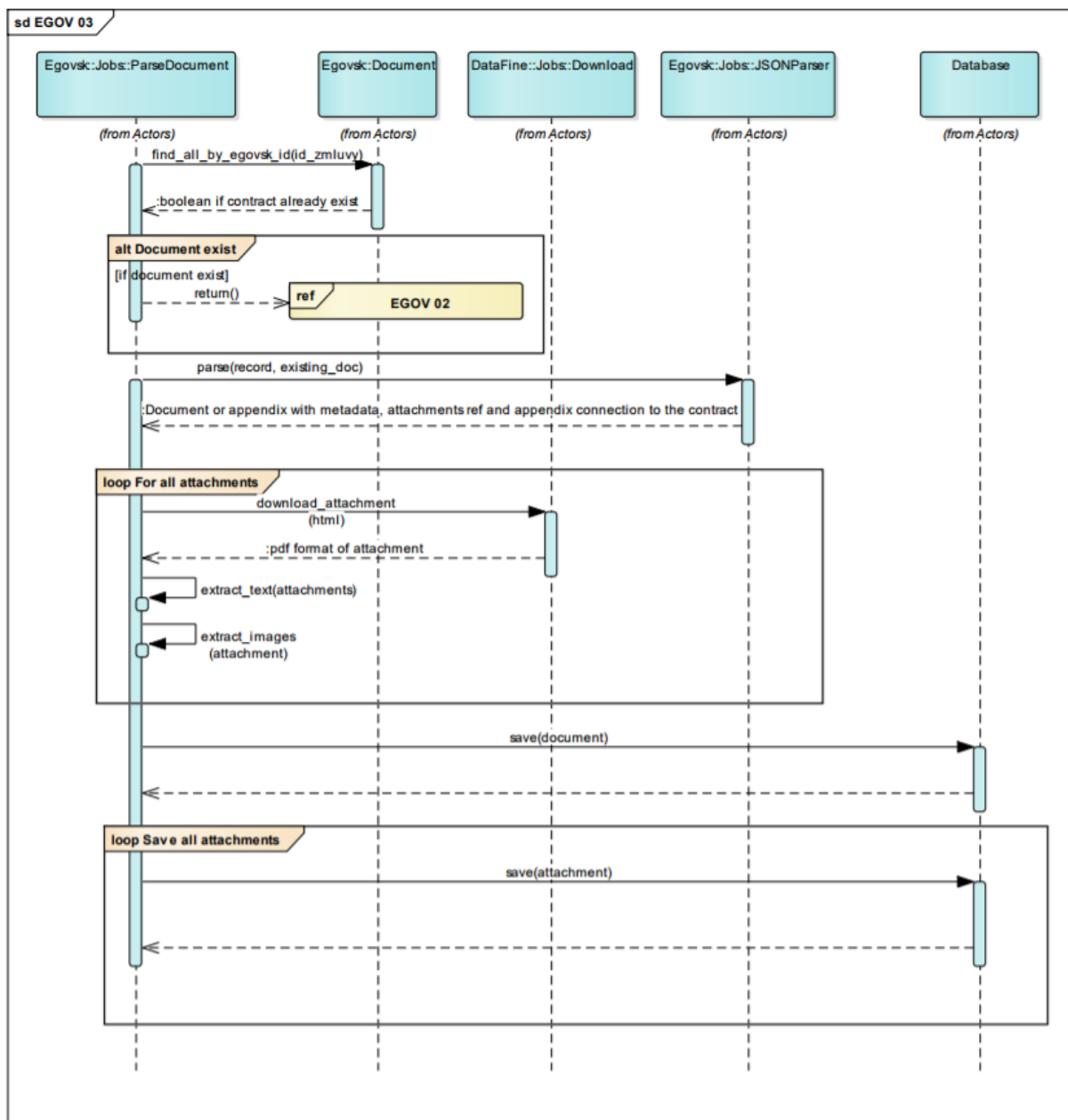
Obr. 15: Sekvenčný diagram sťahovania z Egov 1

V procese sťahovania nových zmlúv z portálu egov sa najprv stiahne JSON zoznám všetkých inštitúcií, ktoré uverejňujú zmluvy na tomto portáli. Z databázy sa načíta dátum posledného updatu pre každú inštitúciu. Tento proces je znázornený na obrázku 11.



Obr. 16: Sekvenčný diagram ťahovania z Egov 2

Následne sa vyskladá url adresa pre danú inštitúciu napríklad: <http://zmluvy.egov.sk/egov/jsonExport/place:105/offset:0/since:2017-01-01>. Stiahnu sa metadáta o zmluvách. Ak sa v danom json súbore nachádza viac ako 500 záznamov posunie sa offset o 500 a zavolá sa rekurzívne vykonanie tohto sekvenčného diagramu. Tento proces je možné vidieť na obrázku 12.



Obr. 17: Sekvenčný diagram sťahovania z Egov 3

Pre každú novú zmluvu sa uložia do objektu Document metadata a buď sa jedná o objekt Appendix alebo Contract podľa toho či sa jedná o novú zmluvu alebo dodatok a pod. V prípade Appendixu sa navyše vykoná len prepojenie na originál zmluvy. Načítajú sa url referencie na attachments z ktorých sa extrahuje text a obrázky. Tie sa uložia na filesystem a následne sa všetky metadata o zmluve a prílohách uložia do databázy. Tento proces je znázornený na obrázku 13.

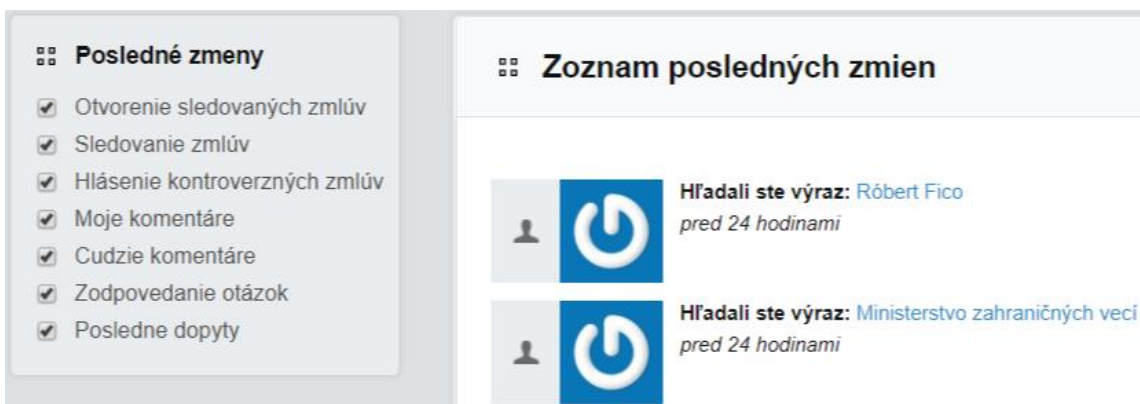
Opis modulov

V tejto časti opisujeme moduly a prototypy, o ktoré sme rozšírili systém.

Sledovanie aktivity prihláseného používateľa - vyhľadávanie

Analýza

Analýzou modulu personálneho “dashboardu” prihláseného používateľa vyplynula nová funkcionálnosť, ktorá dovoľí prihlásenému používateľovi sledovať vlastnú aktivitu vyhľadávania vo svojom osobnom profile. Na obrázku 15 je možné vidieť návrh pohľadu, ktorý pomôže lepšie pochopiť zamýšľaný zámer. Jedná sa o rozšírený pohľad v dashboarde prihláseného používateľa. Používateľ môže vo svojom profile zaškrtnúť/odškrtnúť možnosť zobrazovania poslednej aktivity vyhľadávania (*Posledné dopyty*), ktorú potom systém buď zobrazí alebo nezobrazí (*Hľadali ste výraz:*). Ak je používateľ prihlásený, systém automaticky zaznamenáva jeho dopyty vyhľadávania v zmluvách.

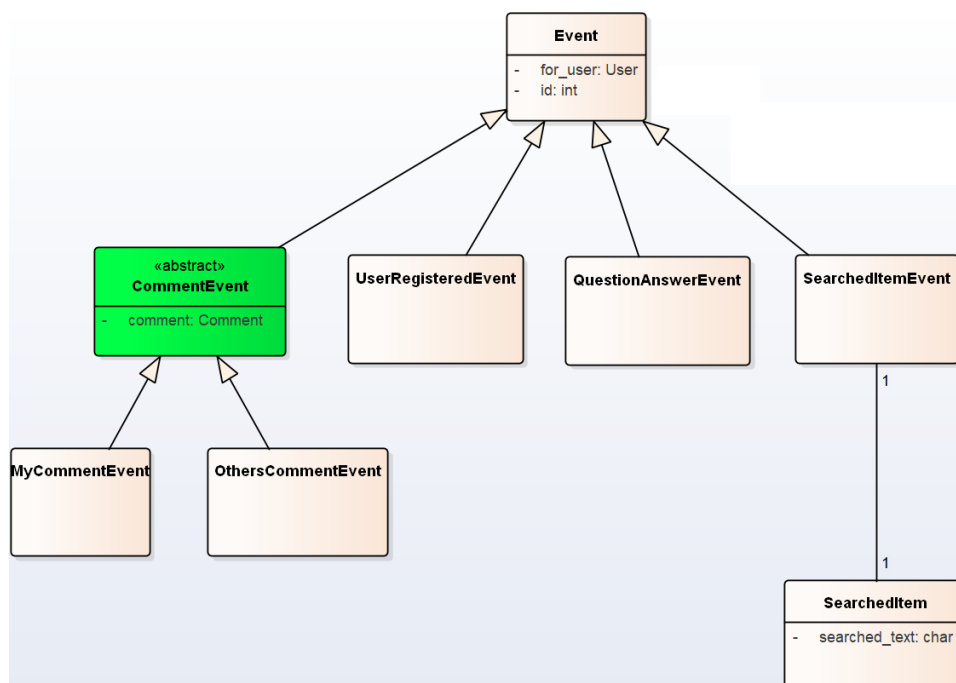


Obr. 18. Návrh vizualizácie vyhľadávaných dopytov v dashboarde prihláseného používateľa.

Návrh

Náš návrh zahŕňa rozšírenie existujúcich možností pre sledovanie aktivity v osobnom profile (registrácia, sledované zmluvy, atď.) o nový podmodul sledovania vyhľadávania. Používateľ tak bude mať možnosť sledovať vlastnú aktivitu podľa jeho uvážení kritérií. Je potrebné rozšíriť funkcionálnosť vyhľadávania o identifikáciu a perzistenciu týchto dopytov, taktiež dátový model o nové navrhnuté triedy a tabuľky v databáze, ktoré spoločne pomôžu zastrešiť túto funkcionálnosť a pridanie atribútu do triedy používateľa, ktorá bude reflektovať stav, či si praje zobrazovať vyhľadávané výrazy v osobnom dashboarde alebo nie. Konkrétne bude rozšírená trieda `/app/models/event` o špecifickú dcérsku triedu `/app/models/searched_item_event`, ktorá bude zastrešovať spomínanú funkcionálnosť - handlovanie vyhľadávania. Kvôli špecifickému princípu ORM frameworku RoR bude trieda

`/app/models/searched_item_event`, pre správne fungovanie, prepojená vzťahom One-to-One s triedou `/app/models/SearchedItem`. V tejto triede bude uložená informácia o tom, aký text daný používateľ vyhľadával (atribút `searched_text`). Lepšie pochopenie predchádzajúcich viet môže priniesť obrázok 16. Trieda používateľa - `/app/models/user` bude taktiež rozšírená o atribút `show_my_queries` (boolean), v ktorej bude uložená informácia o tom, či si používateľ praje renderovať vyhľadávané dopyty v osobnom dashboarde alebo nie.



Obr. 19. Výsek diagramu tried spolu s pridanými triedami, ktoré boli identifikované v návrhu a analýze.

Do modelu `/app/models/user` bude implementovaná funkcia `log_item_searched`, ktorá uloží hľadaný výraz pre špecifického používateľa. Táto funkcia bude potom volaná pri vyhľadávaní dopytu - funkcia `search`, ktorá sa nachádza v kontrolery `app/controllers/document_controller`.

Riešenie

Tak ako bolo avizované v časti analýzy a návrhu, modifikovali sme model používateľa aby bolo jasné, aké preferencie zobrazovania aktivity vyžaduje (pridanie možnosť sledovania aktivity) a pridali funkcionality na ukladanie týchto dát. Taktiež bol vytvorený model udalosti a hľadaného textu, ktoré pomáhajú spracovávať dopyty prihláseného používateľa. Tieto údaje sa potom ukladajú pri hľadaní.

Testovanie

Testovanie prebiehalo simuláciou vyhľadávania dopytov a následného sledovania/nasledovania týchto hľadání v osobnom profile. Zobrazené výsledky a vykonané dopyty boli navzájom konfrontované. Taktiež boli napísané unit testy *spec/models/user_spec*, ktoré sme sa snažili koncipovať tak, aby pokryli všetky možné hraničné situácie pri hľadání (prázdny reťazec, biele znaky, atď.).

Zobrazenie údajov o subjekte

Backend

Analýza

Register právnických osôb je jedným z kľúčových referenčných registrov vo verejnej správe. Obsahuje informácie o právnických osobách z viacerých zdrojových registrov (napr. obchodný register, živnostenský register, register občianskych združení...). Na portáli by sa mohli získať informácie zobrazovať ako rozšírené informácie o jednotlivých subjektoch a tiež ako zdroj pomocných informácií na zistenie podozrivej zmluvy.

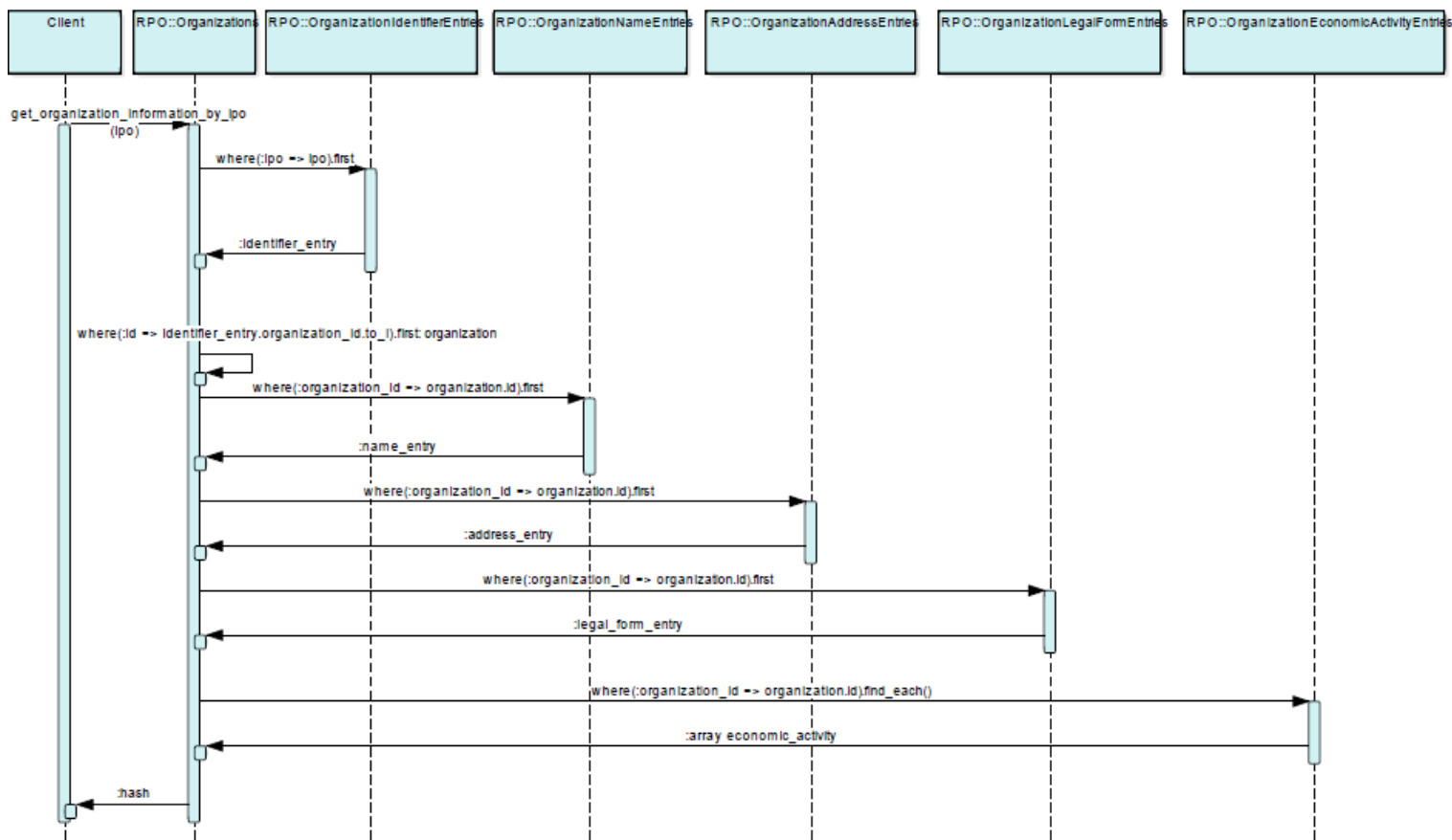
Slovensko digital poskytuje prístup k ich dátam právnických osôb, pomocou dumpov databázy, ktoré aktualizujú každú nedeľu. Opis jednotlivých tabuliek sa nachádza na stránke [Slovensko.digital](#). Na priebežné aktualizovanie zmien v dátach RPO slúži [synchronizačné API](#). Parametre synchronizačného API sú Čas poslednej zmeny, Posledný identifikátor a parameter `only_ids` slúžiaci na získanie iba zmenených identifikátorov záznamov pre zrýchlenie odpovede. Volanie API je limitované na 60 dopytov za minútu a jeden dopyt vracia 100 záznamov. Napríklad volanie žiadosti: https://datahub.ekosystem.slovensko.digital/api/data/rpo/organizations/sync?since=2017-03-12T01:00:29.004649Z&last_id=9119664 nám vráti zmenené záznamy od 3.12.2017 a posledného id spoločnosti 9119664.

Návrh

Pre rozšírenie projektu o register právnických osôb je najskôr potrebné vytvoriť si v databáze tabuľky zodpovedajúce schéme z dumpu Slovensko.digital. Na aktualizovanie projektu o nové tabuľky budú použité migrácie. Následne je potrebné vložiť dump z RPO databázy do našej vlastnej.

Pre priebežné aktualizovanie našej RPO databázy je potrebné napísať rake task, ktorý bude získavať zmeny cez synchronizačné API. Zmeny sa uskutočňujú každý deň a preto je potrebné aby bol spustený každý deň, na čo bude slúžiť CRON job a aby získal zmeny od včerajšku, čiže volanie synchronizačného API bude obsahovať včerajší dátum. Posledný záznam z odpovede API bude slúžiť ako parameter nového volania (čas a posledné id) a týmto spôsobom sa bude iterovať až kým neaktualizujeme všetky zmenené záznamy (nedostaneme prázdnu odpoveď). V každej iterácii sa bude kontrolovať hlavička odpovede, ktorá obsahuje parameter `x_ratelimit_remaining`, ktorý obsahuje informáciu o dostupných zostávajúcich volaniach. Ak už sme všetky vyčerpali, hodnota je rovná 0, rake task sa uspí na 60 sekúnd a pokračuje ďalej.

Pre získanie informácií o subjekte podľa iča je potrebné napísať metódu, ktorá z našej RPO databázy získa potrebné informácie o subjekte. Najskôr je potrebné získať z tabuľky `organization_identifier_entries` id spoločnosti v databáze na základe iča. Následne vieme na základe získaného id pristupovať k ďalším údajom o spoločnosti: jej vzniku, zániku, oficiálnemu názvu, adrese, právnej forme a ekonomických aktivitách. Výsledok je potrebné vrátiť vložiť do hashu a vrátiť pomocou návratovej hodnoty metódy. Proces získavania informácií je zobrazený na nasledujúcom obrázku 20 sekvenčného diagramu.



Obr. 20: Diagram získavania informácií o subjekte

Implementácia

Funkcionalita bola implementovaná podľa návrhu a pri implementácii boli vytvorené migrácie na pridanie RPO modelu do databázy, pre každú tabuľku bola tiež vytvorená trieda, umiestnená v priečinku `app/models/rpo`. Na volanie synchronizačného API bol vytvorený rake task `slovensko_digital_rpo.rake` a bol tiež pridaný do cron jobu na spúšťanie každý deň o jednej hodine. Pre prácu s HTTP odpoveďou bolo potrebné do gemfile pridať nový gem: `rest-client`, ktorý si vyžadoval ruby verziu vyššiu ako 2. Na

získanie informácií o subjekte bola do triedy `RPO::Organizations` pridaná metóda `get_organization_information_by_ipo(ipo)`, ktorá na základe `ipo` (identifikátora právnickej osoby - ičo), vráti hash s informáciami o subjekte.

Testovanie

Na testovanie volania synchronizačného API a správneho prístupu k údajom v odpovedi z API bol vytvorený test `spec/models/RPO/synchronise_spec.rb`, ktorý pre zadaný dátum získa zmeny a skontroluje sa správnosť údajov v prvom zázname.

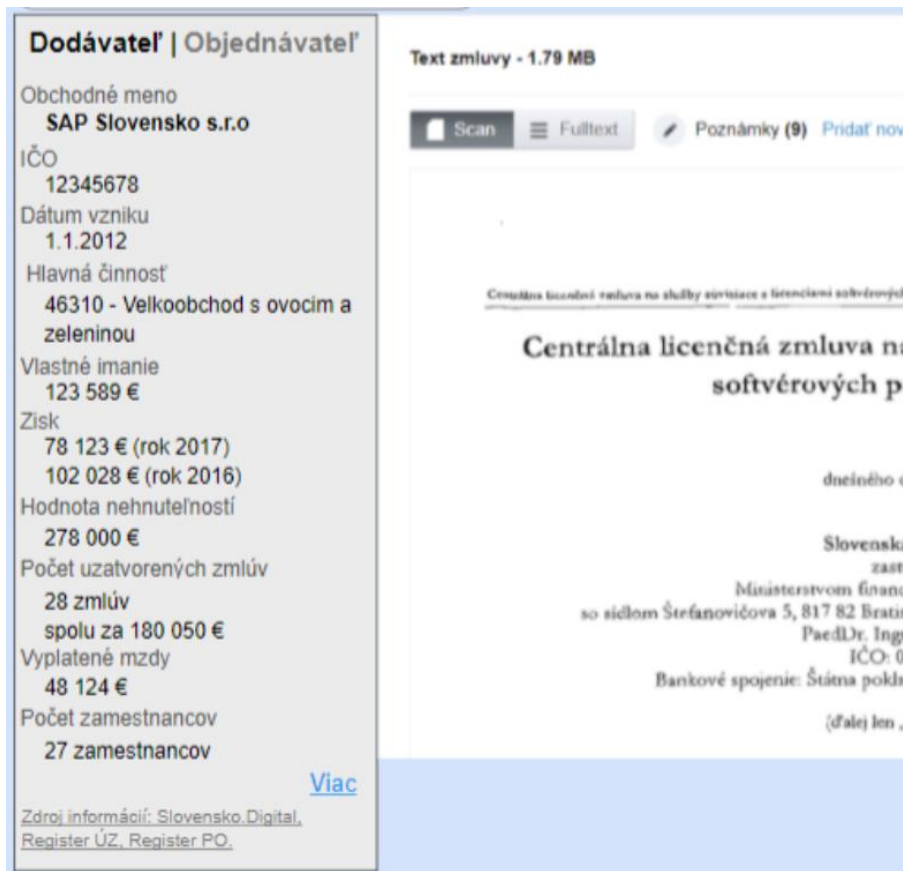
Frontend

Analýza

Po diskusii s produktovým vlastníkom sme identifikovali novú funkcionálnosť - zobrazenie informácií o subjekte. S produktovým vlastníkom sme diskutovali možné pohľady (používateľské rozhranie.), ktoré by zobrazovali dané informácie. Tieto pohľady sú na rôznych miestach, preto sme sa rozhodli spraviť low fidelity návrh, resp. mockup týchto pohľadov a spoločnou diskusiou s produktovým vlastníkom dospieť ku konkrétnemu výsledku.

Návrh

Na základe analýzy sme vytvorili mockupy podľa dohody s produktovým vlastníkom. Všetky vytvorené mockupy sú vizualizované v prílohe C. Po spoločnej diskusii sme sa rozhodli zobraziť tento pohľad pri pohľade na otvorenú zmluvu (upravením `app/views/documents/show.html.erb`) a na samostatnej stránke (pridaním `app/views/subjects/show.html.erb`). Z diskusie vyplynulo, že zobrazovanie otázok pri otvorenej zmluve (ľavý bočný panel) nie je až tak relevantné, a teda môže byť nahradené informáciami o subjekte. Na obrázku 14 môžeme vidieť detail zobrazovania informácií o subjekte v ľavom bočnom paneli pri prezeraní zmluvy. V prvej iterácii sme sa rozhodli implementovať zobrazovanie informácií len v tomto paneli a vytvorenie samostatnej stránky bude výsledkom ďalších iterácií. Položky od "Vlastné imanie" nadol budú pripravené, ale keďže ešte nedisponujeme týmito informáciami nebudú použité. Po kliknutí na políčko "Viac" bude používateľ presmerovaný na samostatný pohľad, kde budú podrobné informácie o zvolenom subjekte (ako už bolo avizované, táto funkcionálnosť bude implementovaná v ďalších iteráciách).



Obr. 21: Detail z mockupu, kde su zobrazene informacie o subjekte.

Riešenie

Funkcionalita bola implementovaná podľa dohodnutého návrhu opísaného v predošlom bloku a výsledok je totožný s mockupom. Zatiaľ zobrazované dáta sú ako placeholdre a v blízkej dobe sa informácie prepoja s relevantnými, ktoré sú dostupné z implementovaného backendu. Z pohľadu `app/views/subjects/show.html.erb` bolo odstránené zobrazovanie otázok a nahradené touto funkcionalitou. Do súboru `app/assets/stylesheets/standard/title.css.scss.erb` bolo pridané CSS pre zobrazenie informácií v takej štruktúre ako je zobrazené v návrhu. Bol vytvorený súbor `app/assets/javascripts/public/change_subject.js`, ktorý pomocou jQuery zabezpečuje prepínané informácii medzi dodávateľom a objednávateľom.

St'ahovanie zmlúv z centrálneho registra zmlúv pomocou xml generovaného súboru.

Z dôvodu, že Portál CRZ obmedzuje počet dopytov na ich stránku sme sa rozhodli využiť odporúčaný spôsob s'ahovania zmlúv a to pomocou xml súboru, ktorým by sme nahradili súčasný spôsob s'ahovania parsovaním html stránok jednotlivých zmlúv. Takýmto spôsobom taktiež znížime počet dopytov na stránku, kde potrebné metadáta o zmluve sú obsiahnuté v jednom xml súbore, a tak by sme potrebovali pristupovať na stránku CRZ len v prípade stiahnutia pdf príloh k zmluve.

Analýza

St'ahovanie zmlúv, ktoré je odporúčané portálom CRZ je možné urobiť stiahnutím .zip súboru, ktorý podľa popisu na stránke obsahuje všetky zmeny, ktoré sa udiali v zmluvách za posledných 24 hodín. Tento rozdielový súbor je generovaný systémom okolo 2:00. Adresa pre stiahnutie rozdielového súboru je <http://www.crz.gov.sk/export/rrrr-mm-dd.zip>. Výraz rrrr-mm-dd sa nahradí dátumom daného dňa, ktorého rozdielový súbor chceme vygenerovať.

Rozdielový súbor je vo formáte XML, kde zmluva vyzerá nasledovne:

```
<zmluva>
    <nazov>Z201749380_Z</nazov>
    <ID>3121040</ID>
    <zs1>Dopravný podnik mesta Košice, akciová spoločnosť</zs1>
    <zs2>Mercedes-Benz Slovakia s.r.o.</zs2>
    <predmet>Kúpna zmluva</predmet>
    <datum_ucinnost>2017-10-03</datum_ucinnost>
    <datum_platnost_do>0000-00-00</datum_platnost_do>
    <suma_zmluva>122400</suma_zmluva>
    <suma_spolu>122400</suma_spolu>
    <id>92031</id>
    <poznamka></poznamka>
    <rezort>1552144</rezort>
    <datum_zverejnene>2017-10-02 18:59:49</datum_zverejnene>
    <ico>35780754</ico>
    <stav>4</stav>
```

<potv_ziadost>1</potv_ziadost>
<potv_datum>0000-00-00</potv_datum>
<zdroj>1</zdroj>
<text_ucinnost></text_ucinnost>
<sidlo>Tuhovská 5, 83107 Bratislava SVK</sidlo>
<ico1>31701914</ico1>
<sidlo1>Bardejovská 6, 04329 Košice SVK</sidlo1>
<potvrdenie></potvrdenie>
<typ>1</typ>
<datum>2017-10-02</datum>
<popis></popis>
<druh>1</druh>
<ref></ref>
<internapozn>TID 31376</internapozn>
<popis_predmetu></popis_predmetu>
<poznamka_zmena></poznamka_zmena>
<chan>2017-10-20 08:37:38</chan>
<prilohy>
 <priloha>
 <ID>3121041</ID>
 <nazov>Z201749380_Z</nazov>
 <dokument>3121041_dokument.</dokument>
 <velkost>126933</velkost><dokument1>
 </dokument1><velkost1>0</velkost1>
 <chan>2017-10-02 18:59:49</chan>
 </priloha>
 <priloha>
 <ID>3157032</ID>
 <nazov>Odstúpenie od zmluvy</nazov>
 <dokument></dokument>
 <velkost>0</velkost>
 <dokument1>3157032_dokument1.pdf</dokument1>
 <velkost1>185066</velkost1>
 <chan>2017-10-20 08:37:15</chan>

</priloha>
</prilohy>
</zmluva>

Bližšie sa analyzoval aj XSD súbor, ktorý však nanešťastie existuje len k zmluvám, ktoré sa nahrávajú na portál vo formáte XML a s XML rozdielovým súborom má tento XSD súbor veľmi málo spoločné. Zmluva teda obsahuje <zs1> a <zs2> čo nám hovorí o zmluvných stranách danej zmluvy, kde <zs1> je objednávateľ, teda štátna inštitúcia a <zs2> je dodávateľ. Ďalej súbor obsahuje dátumy platnosti a účinnosti a informácie o sume danej zmluvy a celkovej sumy. Informácia o rezorte, ktorý zastrešuje danú zmluvu je uvedený prostredníctvom ID tohto rezortu, taktiež je tu aj dátum zverejnenia zmluvy. Element <id> a jeho obsah je zatiaľ neznámy a nenašli sme nič, čo by vysvetľovalo a objasnilo tento element. <predmet> označuje názov zmluvy (kúpna zmluva, príkazná zmluva atď). Ďalej nasledujú informácie o IČE a sídle dodávateľa a objednávateľa, ktoré sú oproti zmluvným stranám prehodené. <ico> a <sidlo> opisujú dodávateľa a <ico1> a <sidlo1> objednávateľa.

<stav> v danej množine, ktorú som skúmal (asi 1000 zmlúv) nadobúdala zmluva stav len 2, 3 alebo 4, kde 2 bola zmluva vytvorená a uverejnená, 3 znamená, že je zmluva doplnená (existuje k nej aspoň jeden dopĺňujúci dodatok) a 4 zmluva bola zrušená. Ďalej sa v súbore nachádzajú číselníky <potv_ziadost>, ktorý nadobúdal hodnoty 1, 2 a 5, kde nebolo zistené, čo daná hodnota znamená. Ďalším neznámym číselníkom je <zdroj>, ktorý nadobúda hodnoty 1, 2 a 3, v tomto prípade sa asi jedná o zdroj alebo spôsob akým bola zmluva nahraná (predpoklad). Číselník <druh> a nadobúda hodnotu 1 a 2. Pri druhu je zmluva označená 1-zmluva a 2-dodatok. <chan> pravdepodobne označuje dátum poslednej vykonanej zmeny.

V ďalšej sekcii sa nachádzajú prílohy. Príloha je definovaná svojím ID a názvom. Ďalšie elementy, ktoré ju opisujú sú <dokument> <velkost> a <dokument1> <velkost1>, kde je zase veľmi nejasné v ktorej sekcii sa nachádzajú informácie o prílohe, keďže niekedy je to v <dokument> <velkost> a niekedy v <dokument1> <velkost1>, v takom prípade druhé elementy sú prázdne a alebo obsahuje veľkosť 0. Tak ako tak je možné takto stiahnuť pdf prílohy o zmluve. Sťahovanie pdf súborov je možné vďaka ID prílohy, avšak je rozdiel pri sťahovaní textovej verzie pdf a naskenovanej verzie. Textová verzia má príponu za názvom súboru .pdf a skenovaná verzia má len . (bodku za názvom súboru). Potom podľa ID stiahneme textovú verziu http://www.crz.gov.sk//index.php?ID=603&doc={priloha_id}&text=1 a skenovanú verziu http://www.crz.gov.sk//index.php?ID=603&doc={priloha_id}.

Vo výsledku je takýmto spôsobom možné vykonať sťahovanie zmlúv pomocou XML súboru, čo by bola menšia záťaž pre CRZ portál a taktiež aj proces sťahovania by mohol prebiehať oveľa rýchlejšie, keďže by sa musel stiahnuť len jeden .zip súbor a následne prílohy ku danej zmluve. Obmedzili by sa tak dopyty na CRZ portál pri parsovaní html stránok.

Návrh riešenia

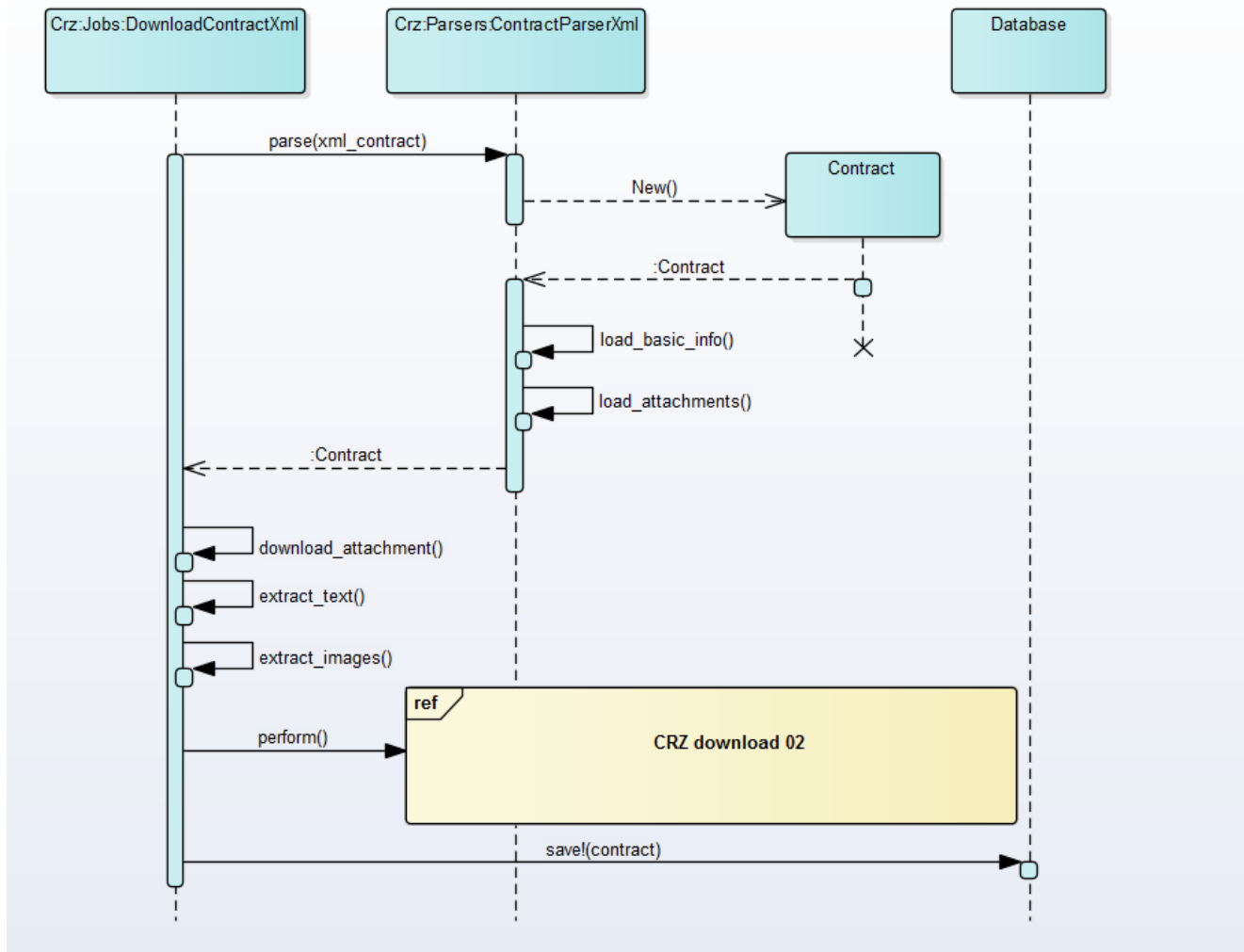
Pri pridaní procesu sťahovania pomocou xml, je potrebné aby sme nezasahovali do aktuálne funkčného riešenia na sťahovanie zmlúv z tohto portálu pre potreby neskoršieho porovnania týchto sťahovaní a taktiež aby sa nenarušil funkčný chod produkčnej verzie.

V takom prípade je potrebné vytvoriť rake task v súbore */lib/tasks/crowdcloud.rake*, cez ktorý sa bude tento nový modul spúšťať. Rake task bude spúšťať metódu *download_zip* v triede *Crz::ZipDownload*. V tejto triede sa stiahne xml súbor, ktorý ako parameter potrebuje dátum (pre ktorý ma stiahnuť rozdielový dokumnt). Na evidovanie dátumu sťahovania potrebujeme tabuľku, kde budeme evidovať dátum a úspešnosť sťahovania. Ak sa xml súbor úspešne stiahne a vyextrahujú sa z neho všetky zmluvy zapíše sa aktuálny dátum +1 s príznakom true (úspešne sťahovanie). inak sa zaeviduje aktuálny dátum s príznakom false (neúspešne sťahovanie). Ak však v novovytvorenej tabuľke nebudeme mať žiadne úspešne sťahovanie, zoberie sa dátum poslednej stiahnutej zmluvy, ktorá sa nastaví ako inializačný dátum sťahovania.

Keďže potrebujeme spracovávať zmluvy asynchrónne a na pozadí vytvoríme background job, ako aj v prípade sťahovania z CRZ od pôvodných autorov. Na vytvorenie background jobu použijeme Resque gem, ktorý využíva technológiu redis. Takýmto spôsobom vyberáme zmluvy zo stiahnutého xml súboru a postupne ich dávame do queue v Resque. postupne iterujeme dátum pre sťahovanie xml rozdielového súboru, až kým nenarazíme na dnešný dátum. V takomto prípade budú už všetky novo zverejnené zmluvy zaradené do queue na stiahnutie.

Vytvorené budú dva background joby. Jeden pri pridaní novej zmluvy (Ak sa zmluva ešte nenachádza v databáze, zistenie podľa unikátnej hodnoty *crz_id*). Ak sa už zmluva nachádza v databáze urobí sa jej update (táto funkcionlita sa bude dopĺňať v neskorších fázach).

Pre pridanie novej zmluvy sa zavolá funkcia s parametrom, ktorý obsahuje metadáta o zmluve vo formáte xml. Volanie tejto funkcie aj s parametrom je uložené v redise, odkiaľ ho voľný Resque worker vyberie. Pre tento background job bude vytvorený súbor *CrzDownloadContractXml*, ktorý skontroluje údaje ako *crz_id* a obsah xml zmluvy aby nedošlo k tomu, že daná xml zmluva bude nadobúdať hodnotu null, alebo že sa dané *crz_id* už nachádza v databáze, čo sa pri asynchrónnom spracovaní môže stať. Následne vyextrahujeme metadáta z xml súboru, pomocou funkcie *parse* triedy *CrzParsersContractParserXml*, v ktorej sa načítajú metadáta a zmluve a jej prílohách. následne sa v triede *CrzDownloadContractXml* stiahnu pdf prílohy na základe ich metadát vyextrahuje sa text a obrázky. Proces stiahnutia novej zmluvy od momentu prebratia xml zmluvy Resque workerom je znázornený na sekvenčnom diagrame na Obr. 22.



Obr. 22: Proces sťahovania novej zmluvy z CRZ pomocou xml

V jednom špeciálnom prípade keď je zmluva doplnená, teda obsahuje nejaké úpravy (dodatky), tak v takomto prípade pomocou xml súboru nie je možné vytvoriť prepojenia a tak použije už implementovaný spôsob parsovania zmluvy pomocou html, čo je znázornené volaním procesu sekvenčného diagramu znázorneného na Obr. 12.

Takýmto spôsobom budeme ukladať nové zmluvy z portálu CRZ.

Implementácia

Implementácia riešenia vychádzala z analýzy a návrhu riešenia. Pre potreby evidovania sťahovania sme vytvorili novú tabuľku v databáze s názvom *crz_downlaod_stuses*, ktorá obsahuje dva stĺpce a to *date* typu date, pre evidovanie dátum a *success*, typu boolean, označuje príznak úspešného sťahovania. ďalej

sme pri implementácii pouzili gemy *rubyzip* pre sťahovanie a rozbalenie zip súboru, v ktorom sa nachádza xml súbor. Na manipuláciu s xml súborom sme pouzili gem *Nokogiri*. Pri extrahovaní textu a obrázkov sme pouzili ako aj v doterajšom module gem *docsplit* a jeho metódy na to určené. Pre pouzitie *docsplit* gemu sme potrebovali nainštalovať tieto dependencies:

```
sudo apt install aptitude
```

```
sudo aptitude install poppler-utils poppler-data
```

```
sudo apt install graphicsmagick
```

```
sudo aptitude install tesseract-ocr
```

V neskorších fázach manuálneho testovania sme odhalili ešte nedostatky riešenia, ktoré bolo dovtedy navrhnuté a implementované a tým pádom sme pridávali dodatočnú funkcionálnu na vyriešenie zistených problémov. Jednou z nich bolo zistené nedostatočné ošetrovanie pri parsovaní niektorých atribútov z xml súboru. Išlo o atribúty *ico* pre dodávateľa aj objednávateľa, ktoré mohlo obsahovať prázdne znaky tabulátoary *entre* a podobne, a taktiež sme ošetrovali aj cenu. Toto sme dosiahli implementovaním funkcií v *parse_ico* a *parse_price* v súbore *contract_parser_xml.rb*. Tiež sme vytvorili novú tabuľku v databáze *crz_resort_ids*, ktorá je len číselníkom a obsahuje *id* rezortu a jeho názov v atribúte *name*. Dôvodom bolo, to že xml súbor obsahoval len ID rezortu, keďže týchto rezortov bolo len 84 implementovalo sa riešenie číselníkovej tabuľky, ktorá sa naplní spustením migrácie a pri sťahovaní sa nahradí ID, ktoré je v xml súbore jeho názvom z tejto novej tabuľky.

d'alšia funkcionálna bola naimplementovaná podľa analýzy a návrhu.

Testovanie

Jednotkové testy boli napísané pre implementovanú funkcionálnu. testovala sa základná funkcionálna sťahovania novej zmluvy ale taktiež aj niektoré špeciálne prípady, ako napríklad absencie IČA subjektu alebo nesprávny formát dátum, čo sa pri zadávaní do CRZ portálu často stáva. Na testy bol pouzítý framework RSpec a nachádzajú sa v súboroch *spec/scrapers/crz/jobs/download_contract_xml_spec.rb* a *spec/scrapers/crz/parsers/contract_parser_xml_spec.rb*.

V neskoršej fáze projektu bolo pridané aj väčšie integračné testovanie tohto modulu, pre ktorý sa vytvoril rake task, spustiteľný príkazom *rake crowdcloud:crz:download:xml:control*, ktorý spustí proces sťahovania pomocou xml aj pomocou html parsovania. Vyextrahujú sa však len ID-ečka za predchádzajúci deň oboma spôsobmi. Následne sa porovná, či sa všetky ID-ečka, ktoré sme dostali html parsovaním nachádzajú aj v xml súbore. Ak sa tak nestalo tak sa vypíše ID, ktoré sa v xml súbore nenachádza. Takto sme spúšťali tento rake task niekoľko dní, kde sa nám v pár dňoch stálo, že sa tam nejaké ID nenachádzali. Z tohto dôvodu je xml sťahovanie zatiaľ nespoľahlivé, avšak dá sa vyriešiť tým,

že tých niekoľko zmlúv, ktoré v xml súbore nie sú sa stiahnu pomocou existujúceho html parsovania, a tak stále ušetríme zbytočné časté pristupovanie na stránky CRZ. Rake task na toto testovanie je definovaný v súbore *lib/tasks/crowdcloud.rake* , ktorý volá vykonávanie funkcie nachádzajúcej sa v *app/scrapers/crz/xml_control*.

Skrytie vybraných zmlúv pred verejnosťou

Samotný portál zoskupuje veľké množstvo dát vo forme zmlúv. Údaje obsiahnuté v týchto dokumentoch musia byť v súlade so zákonom, ktorý zabraňuje zverejňovanie osobných údajov ako napr. rodné číslo. Kvôli množstvu dát je priam nemožné načas skontrolovať spomínaný obsah. Nakoľko dochádza k situáciám, kedy na zmluvách figurujú osobné údaje aj napriek tomu, že pôvodne by takéto údaje nemali byť zverejňované, vznikla požiadavka skrývania daných dokumentov pred verejnosťou.

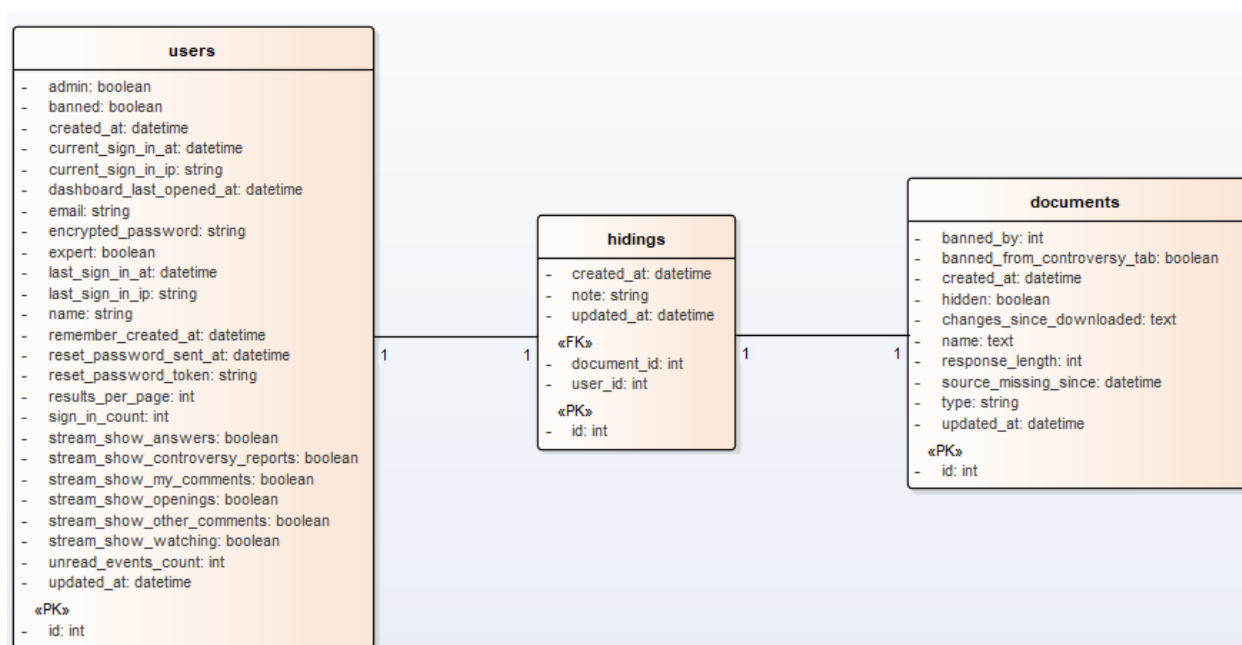
Analýza

Samotná požiadavka nevyžaduje prácu s dodatočnými zdrojmi alebo externými rozhraniami. Hovoríme skôr o úprave používateľských práv v rámci zobrazovania konkrétnych dokumentov. Je vhodné uvažovať aj analógiu v implementácii medzi skrytím zmluvy a jej sledovaním resp. označením za kontroverznú. V našom prípade teda práca bude podliehať úprave grafického rozhrania z pohľadu vyhľadávania všetkých zmlúv, ako aj z pohľadu detailu dokumentu. Konkrétne, je potrebné tieto pohľady rozšíriť o tlačidlo, ktoré zastrešuje požadovanú funkcionálnu skrytia pre aktuálny dokument. Dodatočne, požiadavkou je aj spätná akcia - odkrytie zmluvy a jednotné miesto, kde budú všetky skryté dokumenty zoskupené z dôvodu prehľadnosti, spolu s danou akciou odkrytia. Záverom bude aj úprava štatistík pre dané zmlúvy, kde skryté zmlúvy nebudú zahrnuté do finalných výpočtov pre bežného používateľa. Predošlá požiadavka teda znamená aj rozšírenie indexu vyhľadávania.

Návrh riešenia

Na zastrešenie požiadavky bude potrebné rozšíriť databázovú entitu *documents* o príznak *hidden*, ktorý indikuje, či je daný dokument momentálne verejný alebo nie. Toto však nebude jediná modifikácia v databáze, z dôvodu, že vznikla požiadavka pre manažment skrytých faktúr priamo v administrátorskom rozhraní formou tabuľky. Nakoľko v tomto prehľade bude možnosť pre každý skrytý záznam urobiť inverznú akciu odkrytia spolu s informáciami, kto modifikoval viditeľnosť dokumentu, s akým časom a finálnou poznámkou, potrebná bude aj väzobná entita *hidings*, ktorá si udržiava väzby na používateľa *users* (ktorý skryl dokument) a skrytého dokumentu *documents* spolu s ostatnými informáciami (Obr. 23). K tejto entite budeme realizovať aj separátny controller pre zastrešenie odkrytia. Podobne aj pri obrazovke vyhľadávania dokumentov, nakoľko pracujeme s MVC architektúrou, samotné stlačenie

tlačidla pre skrytie resp. odkrytie bude vyvolávať akciu skrytia v *documents_controller*. Akcia najprv vyhľadá daný dokument, nad ktorým je daná akcia vyvolaná. Nasleduje aktualizácia daného príznaku *hidden* a samotné vytvorenie resp. zmazanie príslušného záznamu vo väzobnej tabuľke *hidings* medzi používateľom a dokumentom. Dodatočne, z hľadiska vyhľadávacieho nástroja elasticsearch, bude potrebné doplniť do mapovania dokumentu pre indexovanie logický atribút *hidden*, ktorý bude použitý pri vyhľadávaní. Tento atribút bude konkrétne využitý pri filtroch, čím zabezpečíme vyhľadávanie iba odkrytých zmlúv, ktoré majú byť prístupné bežnému používateľovi. Pri odkrývaní a skrývaní zmlúv sa bude tento atribút aktualizovať. Okrem filtrovania konkrétnych dokumentov v prehľade zmlúv sa tento atribút využije aj pri štatistikách na úvodnej stránke.




Obr. 23: Väzba medzi skrytím dokumentu a používateľom

Z hľadiska grafického rozhrania, ktoré bolo spomenuté v časti analýzy, nami navrhované skrývanie resp. odkrývanie zmlúv je zobrazené na Obr. 24, 25.

3. **Zmluva o poskytnutí finančných prostriedkov č. 18-143-00011** 5 000 €

Objednávateľ: Fond na podporu umenia
 Dodávateľ: Galéria Nedbalka, n.o.
 Dátum zverejnenia: 16.04.2018



← Späť na vyhľadávanie

Názov **Zmluva o poskytnutí finančných prostriedkov č. 18-143-00011**

Rezort [Fond na podporu umenia](#)

Objednávateľ [Fond na podporu umenia](#)

Dodávateľ [Galéria Nedbalka, n.o.](#) (vek 5 rokov 5 mesiacov 27 dni)

Cena **5 000 €**


Dátum zverejnenia 16.04.2018


Dátum účinnosti 17.04.2018


Dátum platnosti do nezistený


Pôvodný dokument [Odkaz na pôvodný dokument](#)


Súvisiace dokumenty


 **Skryt**


 Editovať


 Sledovať


 Diskutovať 0

 Nahlásiť 0

 Vložiť zmluvu

 Páči sa mi to


 Tweet


 G+


Obr. 24: Návrh rozhrania z pohľadu vyhľadávania a detailu dokumentov


otvorenezmluvy.sk Administrácia ▾ Heuristiky Nastavenia Resque


Dashboard


 Používatelia


 Komentáre

 Kontroverznosť

 **Skryté dokumenty**

 Súvisiace dokumenty

 Uploadni zmluvu

 Výsledky kontroly CRZ

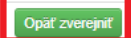
Nevyriešené komentárov
Žiadne nevyriešené hlásenia

Nevyriešené komentáre
Žiadne nevyriešené oznámenia

Nevyriešené dokumenty
Žiadne nevyriešené súvisiace dokumenty

otvorenezmluvy.sk Administrácia ▾ Heuristiky Nastavenia Resque admin Odhlásiť

Skryté zmluvy

Názov zmluvy	Komentáre	Zmenené	Kto skryl
Zmluva o poskytnutí finančn...	 Opäť zverejniť	Poznámka	pred necelou minútou
			admin

Obr. 25: Návrh rozhrania z pohľadu administrátorského rozhrania

Implementácia

Implementácia prebiehala v súlade s návrhom. Konkrétne, boli vytvorené alebo rozšírené nasledujúce súbory:

- `app/controllers/documents_controller.rb` - pridané akcie *hide* a *unhide* pre skrytie resp. odkrytie dokumentu (SOD). Úprava akcií *index*, *search* a *show*.
 - *index* - pridanie filtra pre vytváranie štatistík iba z odkrytých zmlúv na úvodnej stránke
 - *search* - nastavenie parametra ohľadom používateľových práv (konkrétne či je administrátorom), čo sa využíva pri skladaní dopytu do Elasticsearch
 - *show* - ošetrenie stavu, že používateľ zadá URL skrytej zmluvy. Doplnené presmerovanie na 404 šablónu s adekvátnou chybovou hláškou
- `app/controllers/admin/hidings_controller.rb` - zastrešuje naplnenie a prehľad skrytých zmlúv v admin rozhraní
- `app/models/hiding.rb` - model väzobnej entity
- `app/views/dashboard/_hide.html.erb` - tlačidlo pre SOD, volané ako partial pre admina v súbore `app/views/documents/show.html.erb`
- `app/views/admin/hidings/_hiding.html.erb` - partial pre riadok záznamu v prehľade skrytých zmlúv admin rozhrania, kde tento prehľad je reprezentovaný ako `app/views/admin/hidings/index.html.erb`
- `app/assets/stylesheets/standard/document.css.scss.erb`,
`app/assets/javascripts/public/hide.js.coffee` - príslušné css a javascript subory
- `public/404.html.erb` - dynamická verzia 404 šablóny. V `app/controllers/documents_controller.rb` sa vyvolá jej zobrazenie, ak používateľ, ktorý nie je administrátorom, otvorí URL so skrytou zmluvou. Taktiež sa tu nastaví obsah chybovej hlášky a link na originálny dokument.
- `app/lib/factic.rb` - úprava skladania filtrov pre zobrazovanie iba odkrytých zmlúv bežnému používateľovi. Taktiež pridanie kontroly, či je používateľ administrátorom

Testovanie

Riešenie bolo otestované vo viacerých iteráciach. V každom prípade prebiehala kontrola zobrazovania skrytých dokumentov v rámci rozhraní bežného používateľa a administrátora. Interakcia si vyžadovala vyššiu réžiu z hľadiska skrývania a odkrývania jednotlivých dokumentov. Z tohto dôvodu sme zvolili tento spôsob evaluácie na úkor jednotkových testov.

Editácia sprievodných dát v zmluve

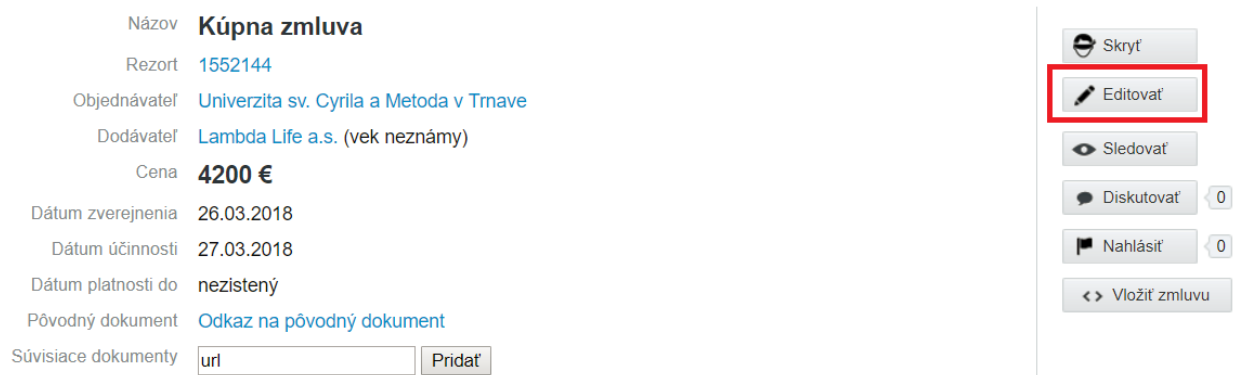
Analýza

Niektoré informácie, ktoré súvisia s konkrétnym dokumentom, resp. zmluvou nemusia byť aktuálne alebo korektne zadané. Taktiež môžeme identifikovať údaje, ktorými aktuálne nedisponujeme a chceme ich zahrnúť. Medzi takéto informácie môže napríklad patriť: cena, alebo názov zmluvy. Na základe tohto faktu sme identifikovali potrebu pridať funkcionalitu, ktorá by tento problém vyriešila.

Frontend

Návrh

Po spoločnej diskusii sme dospeli k tomu, že najlepšie a najvýhodnejšie bude túto funkcionalitu implementovať práve pri detaile zmluvy (detaily zmluvy na samostatnej stránke), a to k skupine tlačidiel, ktoré spadajú do rovnakého kontextuálneho obsahu (napr. sledovanie alebo skrytie zmluvy). Na obrázku nižšie môžeme vidieť detail návrhu tlačidla pre editáciu metadát.



Obr. 26: Detail návrhu editácie zmluvy.

Pre tieto účely chceme vedieť editovať atribúty zmluvy ako: názov, cenu, dátum zverejnenia, účinnosti a platnosti. Takisto tieto údaje bude môcť editovať len prihlásený admin používateľ. Z hľadiska implementácie bude treba editovať existujúci súbor detailu zmluvy a vytvoriť javascriptový súbor, ktorý bude handlovať túto funkcionalitu. Po stlačení editácie zmluvy sa sprístupnia zadané polia pre úpravu a tlačidlo sa zmení na tlačidlo uloženia. Po skončení editácie a uložení informácií sa údaje odošlú do controllera pre uloženie.

Implementácia

Implementácia funkcionality priamo vychádza z návrhu. Obohatili sme súbor `app\views\documents\show.html.erb` o tlačidlo editácie zmluvy s príslušnou ikonou pre jednoznačné rozoznanie funkcionality, skrývajúcej sa pod týmto tlačidlom. Vytvorili sme javascriptový súbor `app\assets\javascripts\public\edit_document.js`, ktorý handluje stlačenie tohto tlačidla. Po stlačení tlačidla editovať sa toto tlačidlo zmení na tlačidlo uložiť a vytvorený JS súbor prejde zmieňované polia a substituje ich za editovateľné elementy ako datepicker a inputfield. Do týchto editovateľných elementov sa defaultne vložia údaje, ktoré prislúchajú konkrétnemu poľu. Po zmene atribútov a stlačení tlačidla uložiť sa nové údaje odošlú do controllera za účelom uloženia (POST požiadavka z JS súboru). Pohľad sa po uložení vráti do pôvodného stavu (ako pred editáciou) no s novými (zmenenými) údajmi.

Testovanie

Vizuálnu stránku funkcionality (frontend) sme testovali manuálne, pričom sme sa snažili pokryť všetky situácie, ktoré môžu nastať (aj hraničné) - prázdne údaje, reťazec znakov namiesto čísla, atď. Prípadné detegované nedostatky sme zakomponovali.

Backend

Návrh

Po implementácii frontendovej časti, kde sa zmenené údaje o zmluve poslali na controller, konkrétne na `documents_controller.rb` do funkcie `edit`, bolo potrebné tieto údaje skontrolovať a zmeniť aj v databáze aby sa konečná zmena mohla prejaviť tak ako bolo zadané v úlohe tohto používateľského príbehu.

Implementácia

Do spomínanej funkcie `edit` v controlleri nám ako parametre prišli zmenené údaje a ID zmluvy, pre ktoré sa majú zmeny vykonať. Na základe ID zmluvy sme si z databázy vytiahli celý dokument. Keďže typy dokumentov, môžu byť rôzne a nie každý má definované všetky parametre, ktoré je možné meniť, bolo potrebné najprv ošetriť, či zmenený parameter daný dokument obsahuje. Ak áno tak následne prebehla validácia jeho hodnoty, kde sme použili validáciu hodnôt používaných pri sťahovaní zmluvy. Konkrétne sa jedná o kontrolu dátumu funkciou `Crz::Parsers::ContractParserXml.parse_date()` a na kontrolu zmenenej ceny `Crz::Parsers::ContractParserXml.parse_price()`. Ak došlo k nejakej zmene a validácie boli úspešné tak sa document updatuje pomocou funkcie vykonanou na dokumentom `update_document`, ktorý uloží zmenené údaje do databázy a taktiež aj dokument v rámci elasticu pre možnosť vyhľadávania na základe nových zmenených údajov.

Testovanie

Testovanie prebiehalo ručným testovaním frontendovej časti ako už bolo spomenuté. taktiež boli napísané testy pre použité funkcie validácie, ktoré sme spomenuli v časti implementácia. Tieto testy sa nachádzajú v priečinku *spec/scrapers/crz/parsers/contract_parser_xml_spec.rb*.

Zobrazenie údajov o subjekte - samostatná stránka

Analýza

Funkcionalitu zobrazenia údajov o subjekte sme implementovali pri detaile zmluvy a je bližšie opísaná v tomto dokumente. Priestor pre zobrazované informácie je však limitovaný a nezmestia sa doň všetky informácie, ktorými máme k dispozícii. Na základe tohto zistenia sme identifikovali potrebu vytvoriť samostatný pohľad, v ktorom by boli všetky dôležité informácie, ktorými disponujeme.

Frontend

Návrh

Ku zvolenému pohľadu (samostatná stránka údajov o subjekte) by bolo najlepšie sa dostať z panelu, kde sú základné informácie o subjekte (pohľad detailu zmluvy). Na základe toho bude nutné obohatiť tento pohľad (súbor *app\views\documents\show.html.erb*) o link na samostatnú stránku. Na obrázku nižšie môžeme vidieť bližšie detaily.

Dodávateľ | Objednávateľ

Obchodné meno
i-firma P15 s.r.o. v likvidácii

IČO
36174289

Dátum vzniku
04.06.1997

Predmet činnosti

- veľkoobchod s chemickými a farmaceutickými výrobkami
- vykonávanie inžinierskych stavieb (vrátane technickej vybavenosti sídlisk)
- vykonávanie priemyselných stavieb
- a ďalšie (6)

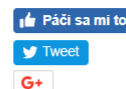
[+Viac detailov](#)

Zdroj informácií: Slovensko.Digital, Register PO, Register PVS.

Obr. 27: Detail návrhu zobrazenia detailov subjektu - link pre zobrazenie.

Po spoločnej diskusii sme dospeli k záveru, že na samostatnej stránke budeme zobrazovať nasledovné údaje o subjekte: všetky dostupné názvy subjektu, alternatívny názov, IČ, všetky dostupné právne formy, aktuálnu adresu, hlavnú ekonomickú činnosť, všetky dostupné predmety činnosti, všetkých dostupných konečných užívateľov, verejných funkcionárov, predchodcov a následníkov právnických osôb, taktiež všetky dostupné zainteresované osoby a štatutárne orgány. Na obrázku nižšie môžeme vidieť detail návrhu samostatného pohľadu na detail subjektu.

Názov	i-firma P15 s.r.o. v likvidácii (platný od 26.05.2015 po súčasnosť) Michalovské pozemné stavby, s.r.o. v likvidácii (platný od 29.01.2015 po 25.05.2015) Michalovské pozemné stavby, s.r.o. (platný od 03.11.1999 po 28.01.2015) Stavebný závod Michalovce, s.r.o. (platný od 04.06.1997 po 02.11.1999)
Alternatívny názov	neexistujúci alebo neznámy
IČO	36174289
Právna forma	neznáma
Právny stav	V likvidácii (platný od 29.12.2014 po súčasnosť)
Adresa	Bitúnková 13, 940 01 Nové Zámky, Slovenská republika
Hlavná ekonomická činnosť	neznáma
Predmet činnosti	- veľkoobchod a maloobchod s potravinami, tabakovými, textilnými, odevnými, obuvníckymi, kožiarskymi, papierenskými, hutníckymi, železiarskymi, elektronickými, elektrotechnickými, sklárskymi, keramickými, drevospracujúcimi, stavebnými, automobilovými výrobkami, poľnohospodárskymi produktami rastlinnej a živočíšnej výroby, zariadením, technickými potrebami a náhradnými dielmi pre obchod, služby, priemysel, dopravu, poľnohospodárstvo, stavebníctvo, nerastnými surovinami, výrobkami z dreva, kovov, plastov, ropnými produktami, ovocím, zeleninou, nápojmi, mäsom, mäsovými výrobkami, drogériou, kozmetikou, nábytkom, hračkami, domácimi potrebami (platný od 04.06.1997 po súčasnosť) - sprostredkovanie obchodu (platný od 04.06.1997 po súčasnosť) - podnikateľské poradenstvo (platný od 04.06.1997 po súčasnosť) - správa nehnuteľností na základe honoráru alebo kontraktu (platný od 04.06.1997 po súčasnosť) - staviteľ-vykonávanie jednoduchých stavieb a poddodávok (platný od 04.06.1997 po súčasnosť) +Viac
Aktívny konečný užívateľ	neexistujúci alebo neznámy
Neaktívny konečný užívateľ	neexistujúci alebo neznámy
Aktívny verejný funkcionár	neexistujúci alebo neznámy
Neaktívny verejný funkcionár	neexistujúci alebo neznámy
Predchodca právnických osôb	neexistujúci alebo neznámy
Následník právnických osôb	neexistujúci alebo neznámy
Zainteresovaná osoba	i-firma s.r.o. (platný od 26.05.2015 po súčasnosť) Ing. Ján Žoffčák (platný od 12.02.2008 po 25.05.2015) Ing. Ján Žoffčák (platný od 03.11.1999 po 11.02.2008) Ing. Daniel Čičák (platný od 04.06.1997 po 02.11.1999) Ing. Ján Žoffčák (platný od 04.06.1997 po 02.11.1999)
Štatutárny orgán	Miroslav Fitala (platný od 18.03.2015 po súčasnosť) Miroslav Fitala (platný od 18.03.2015 po súčasnosť) Ing. Ján Žoffčák (platný od 29.12.2014 po 25.05.2015) Ing. Ján Žoffčák (platný od 01.02.2008 po 25.05.2015) Ing. Ján Žoffčák (platný od 01.02.2008 po 18.03.2015) +Viac



Obr. 28: Detail návrhu zobrazenia detailov subjektu - samostatná stránka.

Implementácia

Implementácia funkcionality priamo vychádza z návrhu. Ako už bolo avizované, bol obohatený súbor `app\views\documents\show.html.erb`, a to za účelom prelinkovania - link viac detailov (prenáša sa ID subjektu). Prislúchajúci controller `app\controllers\subject_details_controller.erb` prečíta prijaté ID subjektu a vytiahne dostupné údaje z databázy. V tomto súbore sú implementované metódy pre parsovanie a prehľadné zobrazenie jednotlivých informácií, ktoré volá pohľad (súbor `app\views\subject_details\show.html.erb`). Vizuálna štruktúra pohľadu je koncipovaná tak, aby zapadala do zaužívaného konceptu (jednotný vzhľad stránok). Ako už bolo spomínané, samostatný pohľad využíva

metódy controllera, a to za účelom prehľadného renderovania dostupných informácií. V podstate sú všetky metódy koncipované tak aby prehľadne zobrazovali dostupné údaje - každý údaj v samostatnom riadku, kde je zobrazené aj primárne trvanie daného atribútu. Za účelom prehľadnosti bola implementovaná funkcionálna, ktorá skrýva obsah niektorých informácií, ak je ich mnoho. Defaultná konfigurácia je 5, to znamená, že ak disponujeme viac ako 5-timi údajmi pre konkrétny atribút subjektu, tak sa tento obsah skryje a ponúkne sa možnosť zobrazenia viacerých detailov. Táto funkcionálna je implementovaná v súbore `app\assets\javascripts\public\subject_detail.js`. Príslušný JS kód sa teda stará o odkrývanie a skrývanie údajov tam, kde je to potrebné. V prípade, že nedisponujeme konkrétnymi atribútmi subjektu, vypíše sa generická hodnota ako napríklad neexistujúci alebo neznámy a pod.

Testovanie

Vizuálnu stránku funkcionality (frontend) sme testovali manuálne. Testovali sme korektnosť zobrazenia pre rôzne subjekty - väčšina subjektov je špecifická a obsahuje rôzny počet údajov pre konkrétny atribút subjektu, ak vôbec. Týmto sme sa snažili pokryť všetky hraničné situácie. Prípadné detegované nedostatky sme zakomponovali.

Pridanie RPVS

Analýza

Register partnerov verejného sektora obsahuje údaje o konečných užívateľov výhod, pre firmy, ktoré figurujú ako partner verejného sektora. Obsahuje takisto záznamy o verejných funkcionároch v riadiacej štruktúre firmy. Štát poskytuje prístup k týmto záznamom pomocou svojho [API](#). Neposkytuje však žiaden spôsob na synchronizáciu. V dátach sa takisto niektoré záznamy vyskytujú viacnásobne s rôznym identifikačným číslom. Pridanie RPVS do projektu by umožnilo prepojiť firmy zo zmlúv s ich konečnými užívateľmi výhod a bolo by možné vyhľadávať podľa týchto údajov.

Návrh

Na pridanie registra do projektu je potrebné najskôr rozšíriť model databázy o nové tabuľky. Na aktualizovanie projektu o nové tabuľky budú použité migrácie. Je potrebné pridať tabuľku pre konečných užívateľov výhod a takisto pre verejných funkcionárov firmy. Takisto je potrebné zaručiť prepojenie medzi organizáciou (RPO) a RPVS pridaním cudzieho kľúča na organizáciu do tabuliek RPVS. Na naplnenie našej RPVS databázy je potrebné napísať rake task, ktorý bude získavať záznamy pre organizácie z RPO. Keďže oficiálne API neposkytuje spôsob synchronizácie je potrebné pravidelne získavať zmeny. To by sa mohlo diať pri pravidelnej aktualizácii RPO.

Implementácia

Pomocou migrácií boli pridané dve nové tabuľky. Tabuľka reprezentujúca konečných užívateľov výhod sa nazýva `end_users`. Tabuľka pre verejných funkcionárov firmy sa nazýva `public_office_workers`. Atribúty tabuliek kopírujú atribúty vrátené z API. Takisto boli vytvorené triedy pre dané tabuľky v balíku `models/rpvs`. Na prepojenie medzi organizáciou a RPVS bol do oboch tabuliek pridaný cudzí kľúč na tabuľku `Organizations` z RPO.

Pre získanie údajov na naplnenie tabuliek bol napísaný rake task `rpvs.rake`, ktorý získava údaje pomocou RPVS API. Boli vytvorené dva tasky, jeden pre získanie údajov pre všetky záznamy a druhý pre aktualizovanie iba jednej firmy počas aktualizácie RPO. Obe dva fungujú rovnakým princípom. Najskôr je pre dané ičo, ktoré je získané z tabuľky `organization_identifier_entries` získané číslo vložky volaním requestu z RPVS API. Z odpovede sa získané číslo vložky pre firmu použije ako parameter pre druhý request. Odpoveď z druhého requestu je zoznam záznamov konečných užívateľov výhod a verejných funkcionárov pre danú firmu. Záznamy sú uložené do tabuliek, spolu s id organizácie, aby existovalo prepojenie na firmu. Tabuľka `organizations` bola pomocou migrácie obohatená o atribút `cislo_vlozky`, ktoré sa pri získavaní údajov pomocou `rpvs rake` tasku uloží. Na frontende slúži ako odkaz na použitý zdroj, odkiaľ máme údaje o danej firme.

Keďže pre niektoré firmy sa vyskytuje viacero duplikátov s rovnakým menom konečného užívateľa ako aktuálny koneční užívateľia sú považovaní tí ktorí majú atribút `effective_to` rovný null. Do tabuliek RPVS boli napísané metódy na získanie aktuálnych a všetkých konečných užívateľov a verejných funkcionárov.

Testovanie

Testovanie prebiehalo manuálne keďže dáta sa môžu zmeniť a automatizovaný test nezaručuje správnosť porovnaných údajov v budúcnosti. Boli porovnávané získaní koneční užívateľia pomocou API a uložení v databáze s informáciami na oficiálnom [vyhladávači štátu](#) pre RPVS. Otestované boli rôzne prípady ako napríklad keď mala firma viacerých konečných užívateľov s rovnakým menom, líšiacim sa iba dátum efektivity. Skontrolované bolo aj prepojenie v databáze, či RPVS odkazuje na správne organizácie.

Nahradenie používania regisu

Analýza

Pôvodný projekt získaval informácie o firmách z Regisu. Táto služba už však skončila a sú k dispozícii iba historické dáta uložené v databáze otvorených zmlúv. Je preto potrebné nahradiť regis RPO, ktoré

bolo do systému pridané. Najskôr je potrebné spraviť prepojenie medzi zmluvami a rpo. Následne je potrebné upraviť všetku funkcionálnosť na ktorú sa využíval regis aby už využívala RPO. Regis sa používal pri detaile zmluvy na zobrazenie veku firmu v čase uzatvorenia zmluvy. Tiež sa používa pri filtrovaní zmlúv na základe veku firmy, právnej formy, druhu vlastníctva a hlavnej činnosti. Všetky tieto údaje vieme získať aj z RPO, okrem druhu vlastníctva.

Návrh

Najskôr je potrebné prepojiť dokument a RPO. Hlavnou tabuľkou RPO je tabuľka organizations a z nej sa vieme dostať k ostatným tabuľkám. Je preto potrebné ju previazať s dokumentmi. Dokument obsahuje číslo dodávateľa a zákazníka. Najskôr preto potrebujeme získať z tabuľky organization_identifier_entries id organizácie a spojiť Document a Organization. Ďalej je potrebné aby funkcie v Contract.rb využívali RPO a nie regis.

Implementácia

Pri prepájaní dokumentov a RPO sa vyskytol problém, že v tabuľke organization_identifier_entries sa vyskytovalo viacero rovnakých záznamov IČO na rôzne záznamy v tabuľke organizations. Bolo preto potrebné navrhnuť vlastný SELECT, ktorý by získal správny záznam organizácie. Do triedy organization bola implementovaná metóda na získanie správnej organizácie podľa zadaného iča. Organizácie, ktoré mali vyplnený stĺpec source_register obsahovali správne údaje. V prípade, že bolo takýchto organizácií viac boli vrátené záznamy usporiadané podľa doby platnosti a následne podľa poslednej aktualizácie.

Do triedy Contract boli pridané metódy rpo_supplier a rpo_customer, ktoré pre číslo zo zmluvy volali metódu z triedy Organization na získanie správnej organizácie. Následne boli upravené metódy supplier_age, supplier_nace, a legal_form tak aby získavali údaje z RPO. Tieto metódy preto používajú novo pridané metódy rpo_supplier a rpo_customer. Informáciu o druhu vlastníctva nemáme v RPO k dispozícii, preto bola ponechaná metóda ownership_category, ktorá v prípade historických údajov na otvorenie zmluvy.sk použije údaje z regisu a v opačnom prípade vráti null. Upravením týchto metód je možné vyhľadávať podľa daných informácií o organizáciách a takisto sa zobrazuje vek organizácie pri podpise zmluvy.

Testovanie

Keďže dáta sa môžu meniť, nie je možné napísať automatizovaný test, ktorý by zaručoval správne skontrolovanie prepojenia RPO s dokumentami a jeho využívanie. Implementácia sa preto testovala

manuálne. V rails konzole boli otestované prepojenia medzi týmito časťami. Takisto bolo na stránke otestované správne vyhľadávanie podľa údajov o firme.

Nové možnosti rozšíreného vyhľadávania zmlúv

Analýza

Vďaka rozšíreniu využívaných dátových zdrojov o RPO a RPVS je možné obohatiť projekt o možnosti rozšíreného vyhľadávania v zmluvách vďaka prepojeniu nových dát s doteraz sťahovanými zmluvami. Údaje z vyššie spomenutých registrov umožňujú prepájať základné údaje o zmluvách z CRZ pomocou IČO dodávateľa danej zmluvy. Takto je vhodné k dodávateľovi pripojiť nové údaje, konkrétne sme sa zamerali na nasledujúce možnosti:

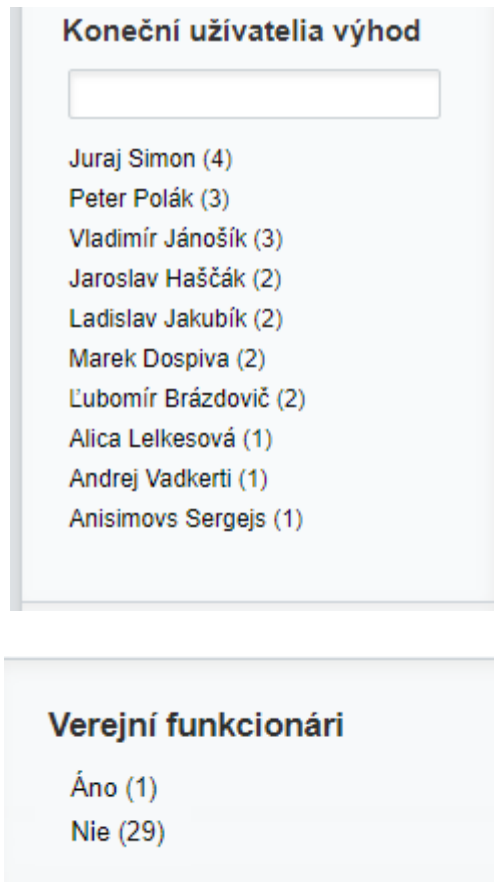
- *Koneční užívatelia výhod* - Podľa zákona je nutné pre každú štátnu zákazku zverejniť tzv. konečných užívateľov výhod, ktorí budú profitovať z danej zmluvy. Takto je možné vyhľadať konkrétne osoby podľa užívania výhod, aj pokiaľ nefigurujú ako vlastníci firiem.
- *Verejní funkcionári* - Vďaka údajom o osobách z RPVS sa dá zistiť, či v danej firme zabezpečujúcej zákazku pôsobí alebo v minulosti pôsobil verejný funkcionár, čo môže poukázať na ich podozrivosť.

Návrh

Existujúca architektúra projektu umožňuje pomerne jednoducho pridávať nové možnosti vyhľadávania nad zvolenými atribútmi, pokiaľ sa jedná o už implementované typy atribútov. V tomto prípade však musí prísť prídanie nových druhov atribútov na strane frontendu aj backendu. Modulárnosť zdrojového kódu týkajúceho sa vyhľadávania cez Elasticsearch (kde sa odohrávajú všetky aktivity rozšíreného vyhľadávania nad zmluvami) je však na veľmi dobrej úrovni, takže stačí pridať nové triedy a šablóny bez potreby do veľkej miery upravovať alebo mazať existujúce riešenie.

Implementácia

Vyhľadávanie sme rozšírili vo forme faziet, teda sme zachovali architektúru existujúceho riešenia a rovnako aj dizajn zobrazenia na stránke. Nové atribúty sa získavajú cez metódu v modelovej triede dokumentu. V nej sa cez IČO dodávateľa získajú údaje z tabuliek RPO. Príklad rozšírených možností vyhľadávania je na obr. 29.



Obr 29: Ukážka nových faziet v rozšírenom vyhľadávaní

Konkrétne sme pre jednotlivé rozšírenia vyhľadávania pridali nasledovné:

- *Koneční užívatelia výhod* - Tento atribút sme implementovali ako pole reťazcov znakov, kde jeden reťazec zodpovedá menu a priezvisku jedného konečného užívateľa výhod pre danú zmluvu. Vďaka tomu sme mohli na backende použiť existujúcu fazetu *SeachableTermsFacet*, ktorá je implementovaná práve pre reťazec znakov. Museli sme však vykonať miernu úpravu kódu, nakoľko pôvodná implementácia nepočítala s poľami, ale iba s jedným atribútom. Naše zmeny teraz umožňujú do budúcnosti využívať túto fazetu aj pre iné atribúty vo forme polí, čo drasticky zrýchli pridávanie nových faziet tohto typu v budúcnosti. Zdieľaná šablóna pre *SearchableTermsFacet* na frontende dokonca nemusela byť upravená a jej funkčnosť zostala zachovaná.
- *Verejní funkcionári* - Tento údaj sme implementovali ako boolean fazetu podľa toho, či medzi zainteresovanými osobami u dodávateľa pôsobí alebo pôsobil súčasný alebo minulý verejný funkcionár. Riešenie sme implementovali tak, aby rozširovanie o iné boolean atribúty bolo jednoduché a vyžadovalo si iba minimum pridaného kódu na frontende aj backende. Z hľadiska

frontendu sme vytvorili novú šablónu pre nový druh atribútu pri vyhľadávaní (boolean). Z hľadiska backendu sme vytvorili novú podtriedu materskej triedy *Facet*, ktorá je spoločná pre všetky rozširujúce fazety a poskytuje základnú zdieľanú funkcionálnu. Novovytvorená *BooleanFacet* potom umožňuje vytvárať akýkoľvek nový atribút pre rozšírené vyhľadávanie po pridaní mena atribútu v konfigurácii.

Testovanie

V rámci globálnych úprav starých neudržiavaných testov sme upravili všetky existujúce testy týkajúce sa fazetového vyhľadávania. Vďaka modulárnosti bolo potrebné pridať iba 1 nový unit test pre boolean fazetu. Okrem automatického testovania sme vykonali aj manuálne testovanie všetkých možných scenárov a kombinácií hodnôt nových atribútov. Funkcionalita bola prezentovaná na stretnutí a po schválení nasadená na tímový server.

Vylepšenie normalizácie názvu dodávateľov a objednávateľov

Analýza

Využitie nových registrov RPO a RPVS (modul Pridanie RPVS) umožňuje získavať aktuálne informácie o firmách, okrem iného aj o zmene názvu firmy. Vďaka synchronizácii popísanej v časti Pridanie RPVS tak môžeme každý deň udržiavať aktuálne informácie o mene firmy pre každú zmluvu v systéme. V minulosti totiž bolo bežné, že firma zmenila svoje meno a údaje pripojené k starším zmluvám zostali neaktuálne.

Teraz je však možné podľa IČO zmeniť aktuálny názov firmy pre nové aj staršie zmluvy, pokiaľ si zmení meno. Toto spôsobí jednoduchšie vyhľadávanie vďaka unifikovaným názvom firiem a vyvarovanie sa podobných tvarov mena tej istej firmy, čo bolo identifikované ako jeden z problémov pôvodného systému. Zároveň však v databáze zostáva informácia o pôvodnom názve firmy pri prvom uložení zmluvy do našej databázy Postgres, čo bolo implementované už v pôvodnom systéme.

Návrh

Doterajšia normalizácia mien dodávateľov a objednávateľov fungovala na základe jednoduchej heuristiky, ktorá spočíta počet odlišných tvarov tej istej firmy a ten najčastejší zvolí za normalizovaný.

Táto metóda bude nahradená novou, ktorá bude využívať tabuľku RPO, konkrétne *organization_name_entries*. V nej vďaka dátumu vieme zistiť najaktuálnejší názov firmy a prepojením s tabuľkou *organization_identifier_entries* IČO danej firmy. Pomocou tohto IČO sa potom dajú vyhľadať

všetky zmluvy v našej Postgres databáze a upraviť tak nový názov dodávateľa a objednávateľa danej firmy, čo taktiež upravíme aj v ElasticSearch pri vyhľadávaní zmlúv.

Implementácia

Upravením metódy na normalizáciu mena dodávateľa a objednávateľa a nahradením jej použitia v kóde novo implementovanou logikou sme docielili nahradenie zastaralej normalizačnej heuristiky popísanej v návrhu. Táto normalizácia prebehne vždy pri ukladaní zmluvy do Postgres a rovnako aj v Elasticsearch, aby vyhľadávanie zostalo aktuálne. Nové zmluvy teda prebehnú normalizáciou automaticky, v prípade nesprávne vyplnenej kolónky úradníkom (čo sa historicky ukázalo ako pomerne pravidelný jav v existujúcom systéme v produkcii).

Okrem nových zmlúv bolo tiež potrebné upravovať priebežne staršie zmluvy. Toto sme docielili rozšírením synchronizačného skriptu pre RPO, takže vždy pri pridaní nového záznamu o zmene názvu firmy sa hneď vzápätí nájdu všetky dotknuté zmluvy podľa IČO firmy meniacej názov. Tieto zmluvy sa pridajú do Resque fronty na asynchrónne spracovanie. Resque na serveri pracuje priebežne, takže keď prídu dané úlohy na radu, pre každú zmluvu sa vykoná update mien v Postgres aj v Elasticsearch.

Testovanie

Pre novú funkcionálnosť sme pridali niekoľko unit testov. Rovnako sme vykonali manuálne testovanie. Takto sme pokryli prípad, kedy sa ukladá nová zmluva, aj prípad upravenia existujúcich zmlúv a zobrazenie nových informácií vo vyhľadávaní aj v detaile zmluvy.

Editácia scanu a fulltextu zmluvy

Analýza

Informácie, ktoré sú obsiahnuté v zmluvách, môžu častokrát obsahovať aj citlivé informácie, ktoré napríklad daný subjekt dožaduje odstrániť. Je v našom záujme byť legislatívne chránený, a teda je nutné implementovať funkcionálnosť, ktorá nám tento problém pomôže potlačiť. Konkrétne editáciou vybraných častí (napr. s citlivými údajmi), a to vo fulltextovej aj obrázkovej podobe.

Frontend

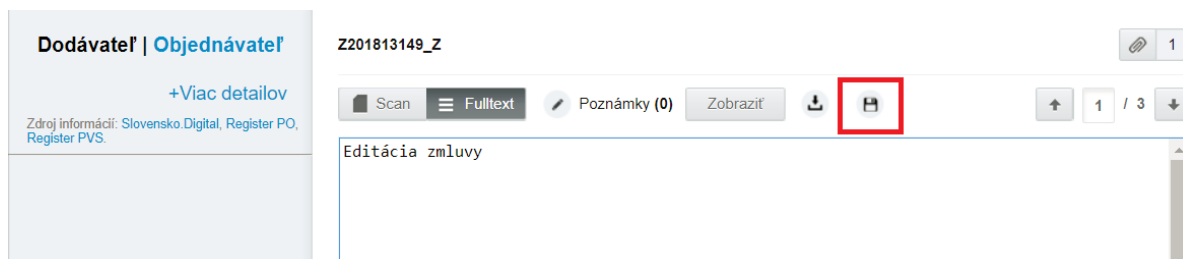
Návrh

Spomínanú funkcionálnosť je najrozumnejšie umiestniť do kontextu, ku ktorému patrí - detail zmluvy popri existujúcich nastaveniach. Po spoločnej diskusii vyplynulo, že v prípade prezerania scanu zmluvy budeme schopný stiahnuť scan zmluvy v podobe obrázka. V prípade, že prezeráme fulltext zmluvy, budeme schopný stiahnuť text zmluvy. V prípade scanu zmluvy, tento obrázok upravíme lokálne v nejakom editore - začiernenie citlivých, resp. vybraných informácií a uploadneme tento upravený scan, pričom prepíšeme už existujúci. Ak sa jedná o fulltext, je rozumné a jednoduché upravovať tento text priamo v pohľade (webovej stránke). Po skončení úprav, tento modifikovaný text jednoducho nahráme na server, pričom sa prepíše existujúci fulltext. Na obrázku nižšie môžeme vidieť návrh pre stiahnutie, resp. upload obrázka alebo fulltextu za účelom úpravy.



Obr. 30: Detail návrhu editácie alebo fulltextu obsahu zmluvy.

V prípade, že sme nastavený na fulltextové zobrazenie, môžeme editovať text priamo v pohľade a prípadné zmeny uložiť. Na obrázku nižšie môžeme vidieť detail.



Obr. 31: Detail návrhu editácie prípadne ukladanie fulltextu zmluvy.

Je treba poznamenať, že túto editáciu bude môcť vykonať výlučne prihlásený admin používateľ. Scenár pre úpravu scanu zmluvy je nasledovný:

- nastavíme sa na aktuálnu stranu, podľa toho aký scan chceme editovať

- stiahneme tento scan pomocou tlačidla
- upravíme obrázok
- nahráme tento editovaný obrázok pomocou príslušného tlačidla, pričom sa prepíše pôvodný scan

Scenár pre úpravy fulltextu je nasledovný:

- prepneme sa do fulltextového pohľadu
- nastavíme sa na aktuálnu stranu, podľa toho aký text chceme editovať
- priamo na stránke editujeme text
- uložíme editovaný text pomocou príslušného tlačidla, pričom sa prepíše pôvodný

Implementácia

Na začiatku bolo nutné upraviť konfiguračný objekt, ktorý je definovaný v súbore *app\views\documents\show.html.erb*, kde sme doplnili informáciu, či je aktuálne prihlásený používateľ administrátor. Podľa tejto informácie potom budeme vedieť, či budeme renderovať tlačidlá pre možnosti editácie scanu prípadne fulltextu alebo nie. Rozšírili sme súbor *app/assets/javascripts/document_viewer/views/document_control_view.js.coffee* o funkcionality, ktorá renderuje tlačidlá pre editáciu, prípadne stiahnutie scanu alebo fulltextu (ak sa jedná o administrátora). V tomto súbore sú implementované aj všetky metódy, ktoré sa starajú o prípadné stiahnutie alebo upload, prípadne uloženie fulltextu alebo scanu zmluvy, a to presne podľa zadaného scenára v návrhu. Ďalej sme upravili súbor *app/assets/javascripts/document_viewer/views/page_view.js.coffee*, kde v prípade prihláseného admin používateľa meníme html element do ktorého je zasadený fulltext zmluvy, na editovateľný element - za účelom možnosti editácie fulltextu priamo na stránke.

Testovanie

Vizuálnu stránku funkcionality (frontend) sme testovali manuálne. Funkcionality sme primárne testovali podľa zadaného scenára v časti návrhu. Týmto sme sa snažili pokryť všetky hraničné situácie. Prípadné detegované nedostatky sme zakomponovali.

Backend

Návrh

Z pohľadu backendu sme aj pri editácii scanu a fulltextu zmluvy postupovali analogicky ako pri editácii metadát. Najprv sa pripravil na toto doplnenie funkcionality frontend, ktorý posiela zmenené informácie, ktoré chce používateľ zmeniť na controller, Informácie, ktoré potrebujeme na realizáciu tejto zmeny sú

samozrejme zmenené súbory, ktorými chceme nahradiť staré súbory. Keďže sa reálne súbory o dokumente nevidujú priamo v databáze (v nej sa nachádza iba cesta k nim), ale sú na uložené na disku. Je teda potrebné aby sa na controller posielala aj táto cesta. Následne sa na danej ceste vymaže žiadaný dokument a nahradí sa novým súborom s rovnakým menom aby nenastal problém s cestou, ktorá je uložená v databáze.

Implementácia

V súbore *documents_controller.rb*, sme analogicky vytvorili funkciu *edit_scan*, do ktorej vstupujú parametre z frontendu, kde prichádza informácia o tom, či sa jedná o zmenu obrázka alebo fulltextu zmluvy v atribúte *uploaded_file_type*. Potom sme si podľa toho a aký súbor sa jednalo, vyextrahovali cestu z ďalšieho parametra. Samozrejme bolo potrebné najprv skontrolovať, či daná cesta, ktorú sme obržali je relevantná. Ak to tak nie je tak nedôjde k vymazaniu súboru, avšak ak áno tak sa najprv vymaže súbor z disku a následne sa nahradí nový, ktorý prišiel v argumente *uploaded_file*, buď sa jedná o gif súbor alebo textový súbor (keď sa jedná o úpravu fulltextu zmluvy). V prípade, že by došlo k chybe pri vymazaní novú súbor sa nenahrá.

Testovanie

V prípade backendu sa výsledná funkcionálnosť testovala zväčša manuálne, kontrolou vymazania a nahratia nových rozdielných súborov na dané miesto na disku. Toto bolo testované na viacerých súboroch, či už išlo o súbory typu *.gif* (ak sa jednalo o obrázky zmluvy) alebo *.txt* (pri fultexte zmluvy). Dôležitou časťou bolo aj vizuálne testovanie zmeny zmluvy na administrátorskom rozhraní, ktoré prebehlo na frontendovej časti.

Hromadné skrytie vybraných zmlúv pred verejnosťou

So skrývaním zmlúv nastala ďalšia požiadavka z hľadiska UX. Doteraz bolo možné dokumenty skrývať resp. odkrývať len po jednom, čo nie je používateľsky prívetivé v prípadoch, kedy bude potrebné zamedziť pre verejnosť viacero dokumentov.

Analýza

Z analytického hľadiska, hovoríme o úlohe, ktorá má už čiastočne zastrešenú funkcionálnosť. Napriek tomu, z hromadného hľadiska je dôležité počítať so skutočnosťou, že skrývaných zmlúv môže byť v danom čase veľké množstvo, čo by mohlo mať za následok zahltenie aplikácie, a teda znemožnenie práce s ňou. Preto je vhodné počas návrhu uvažovať asynchrónny proces, ktorý bude zodpovedný za takéto hromadné skrytie. Je vhodné zamyslieť sa aj nad spätnou akciou, teda hromadné odkrytie. Odhliadnúc na funkcionálnosť vytvárania heuristik zisťujeme, že ako rozumné riešenie tohto problému je vytvorenie novej entity v databáze, ktorá uchováva vyhľadávacie parametre, ktoré boli použité pri skrývaní. Takéto riešenie nám umožní následne hromadne dané dokumenty zverejniť, ktoré vyhovujú parametrom vyhľadávania.

Návrh

Z pohľadu používateľského rozhrania navrhujeme umiestniť možnosť hromadného skrývania zmlúv do prehľadu zmlúv, konkrétne vedľa tlačidla pre vytváranie heuristik a zasielanie e-mail notifikácií. Administrátor bude môcť vložiť poznámku pre hromadné skrytie. Taktiež sa používateľovi zobrazí potvrdzovacie okno (Obr. 32). Obdobne, spätná akcia bude jednoduchou inverziou voči doteraz popísanému postupu (Obr. 33). Z pohľadu backendu navrhujeme vytvoriť resque job, ktorý bude asynchrónne, hromadne skrývať zmluvy. Bude potrebné vytvoriť v databáze tabuľku *mass_hidings*, ktorá bude zahŕňať okrem poznámky aj parametre z URL, keďže sa očakáva možnosť aj hromadného odkrytia zmlúv (podobná logika ako heuristiky). Daná väzba je zobrazená na Obr. 34. Resque job okrem nastavenia atribútov zmlúv pri skrývaní vytvorí záznam v tabuľke *hidding* pre každú zmluvu kvôli prehľadu skrytých zmlúv zoradených v prehľade administrátorského rozhrania.

Otvorené zmluvy Administrácia | Výhody Otvorených zmlúv | O projekte | FAQ | RSS | admin | Odhláste sa

Vyhľadavanie v zmluvách

Objednávateľ

Úrad práce, sociálnych... (33)
 LESY Slovenskej republ... (16)
 Národná diaľničná spol... (8)
 Národný žrebčín Topoľč... (6)
 Fakultná nemocnica s p... (2)
 Ministerstvo obrany (2)
 Národná tranšúžna slu... (2)
 Obec Lovce (2)
 Slovenská poľnohospodá... (2)
 Úrad práce, sociálnych... (2)

Spolu 2 330 855 € v 153 zmluvách. Zoradiť podľa Dátumu zverejnenia

Vytvoriť heuristiku | Odoberať zmluvy cez e-mail **Hromadne skryť zmluvy**

1. Zmluva o poskytovaní služieb Objednávateľ: Ústav na výkon trestu odňatia slobody, Železovce Dodávateľ: Ecoder s.r.o. Dátum zverejnenia: 23.04.2018	7 652 €
Dôvod skrytia zmlúv: <input type="text"/> Skryť zmluvy	
2. Objednávka z Portálu produktov a služieb Objednávateľ: Obec Lovce Dodávateľ: Geodetický a kartografický ústav Bratislava Dátum zverejnenia: 23.04.2018	0 €

Obr. 32: Detail návrhu rozhrania hromadného skryvania zmluvy.

Otvorené zmluvy Administrácia | Výhody Otvorených zmlúv | O projekte | FAQ | RSS | admin | Odhláste sa

Vyhľadavanie v zmluvách

Objednávateľ

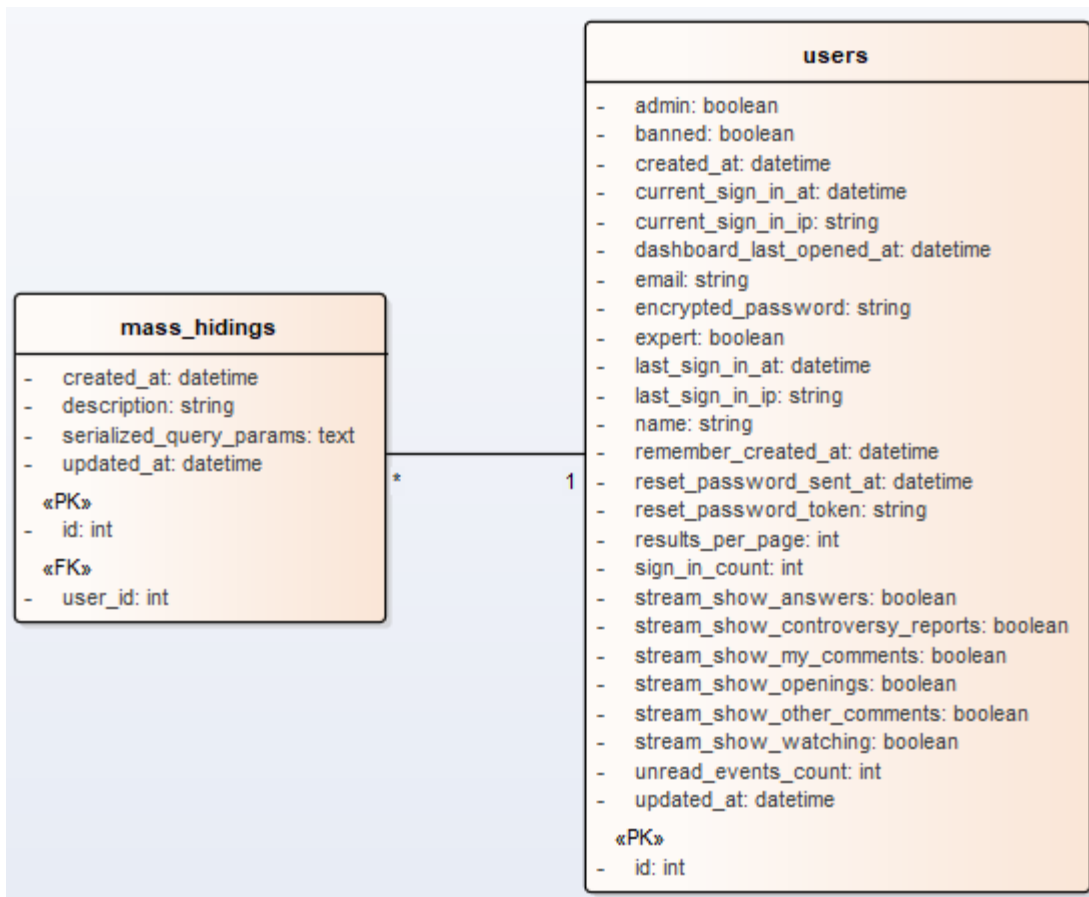
Úrad práce, sociálnych... (33)
 LESY Slovenskej republ... (16)
 Národná diaľničná spol... (8)
 Národný žrebčín Topoľč... (6)
 Fakultná nemocnica s p... (2)

Spolu 2 330 855 € v 153 zmluvách. Zoradiť podľa Celkovej sumy

Vytvoriť heuristiku | Odoberať zmluvy cez e-mail **Hromadne odokrýť zmluvy**

1. Kúpna zmluva č. H/18/063/2018 Objednávateľ: Národný ústav srdcových a cievnych chorôb, a.s. Bratislava Dodávateľ: Siemens Healthcare s.r.o. Dátum zverejnenia: 23.04.2018	1 620 000 €
--	-------------

Obr. 33: Detail návrhu rozhrania hromadného odkrývania zmluvy.



Obr. 34: Väzba medzi používateľom a hromadným skrytím dokumentu

Implementácia

Implementovali sme danú funkcionality v súlade s návrhom. Súbor, ktoré boli modifikované a vznikli počas tejto fázy uvádzame nižšie:

- `app/models/document/jobs/hide_documents.rb` - rescue job zastrešujúci hromadné skrytie
- `app/controllers/admin/mass_hidings_controller.rb` - controller pre akciu skrývania a odkývania v hromadnom kontexte
- `app/models/mass_hiding.rb` - model pre záznam hromadného skrytia a jeho parametrov
- `app/controllers/documents_controller.rb` - volanie vytvorenia rescue jobu
- `app/views/documents/_search_results.html.erb` - používateľské rozhranie skrývania
- `app/models/ability.rb` - používateľské práva pre hromadné skrývanie

Testovanie

Podobne ako v prípade skrývania dokumentov jednotlivu, riešenie bolo otestované vo viacerých iteráciach. V každom prípade prebiehala kontrola zobrazovania skrytých dokumentov v rámci rozhraní bežného používateľa a administrátora. Interakcia si vyžadovala vyššiu réžiu z hľadiska skrývania a odkrývania jednotlivých dokumentov. Z tohto dôvodu sme zvolili tento spôsob evaluácie na úkor jednotkových testov.

Prílohy:

Príloha A: Používateľská príručka

1. Bežný pohľad

Používatelia sa delia do skupín:

- *Neprihlásený*
- *Prihlásený*
 - Bežný prihlásený používateľ
 - Admin
 - Expert

Ako *neprihlásený používateľ* môžem:

- V doméne projektu *byť informovaný* [Príloha 1]:
 - Možnostiach, ktoré ponúka aplikácia - vyhľadávať a posudzovať výhodnosť zmlúv, upozorniť na zmluvy hodné preskúmania, možnosť priamo sa k nim vyjadriť na verejnom fóre, atď.
 - Podmienkach používania aplikácie a Cookies
 - Základných možnostiach odberu najnovších a najkomentovanejších zmlúv cez RSS
 - Kódexe komentujúceho, spôsobe ako pomôcť a najčastejšie kladených otázkach
- *Vyhľadávať v zmluvách* [Príloha 2]:
 - *Základné vyhľadávanie* v zmluvách - samostatné slová alebo skupiny slov, ktoré sa nachádzajú v sledovaných poliach zmlúv
 - *Rozšírené vyhľadávanie* - špecifikácia kritérií vyhľadávania na základe:
 - Objednávateľa, dodávateľa, celkovej sumy, rezortu a dátumu zverejnenia
 - Konečných užívateľov výhod a či sú verejným funkcionárom
 - Počtu komentárov, dodatkov, príloh a strán
 - Splatnosti, veku a kategórie dodávateľa
 - Druhu vlastníctva, právnej formy a zdroja
 - *Pokročilé vyhľadávanie* - v základnom vyhľadávaní, a to za pomoci logických operátorov v sledovaných poliach zmlúv
 - Výsledky vyhľadávania zoradzovať podľa celkovej sumy, dátumu zverejnenia a počtu komentárov
- *V kontexte zmlúv* [Príloha 4]:
 - *Rýchlo sprístupniť nové zaujímavé zmluvy, najdiskutovanejšie zmluvy, posledné komentáre a kontroverzné zmluvy*
 - *Prezerať obsahy a detaily zmlúv* (sledované polia zmlúv)
 - Skopírovať odkaz na zmluvu (napr. na svoju stránku - HTML)
 - *Vypĺňať dotazníky k zmluvám*, ktoré pomáhajú dozvedieť sa o zmluvách viac - pripraviť do budúca kvalitnejšiu analytiku
 - Kvalita vypracovania, resp. prevedenia a informačná kvalita, ktorú ponúka zmluva

- **Prezerat' komentáre k zmluvám** a zobrazovanie prípadných anotácií, ktoré sú prepojené so zmluvami
 - **Prezerat' si prípadné prílohy k zmluvám**
 - **Prezerat' informácie o dodávateľoch a objednávateľoch zmluvy**
- **Stat' sa prihláseným použív.** v aplikácii a tak využívať určité výhody opísané nižšie

Ako **prihlásený bežný používateľ**, vrátane ponúkanej funkcionality pre neprihláseného používateľa môžem:

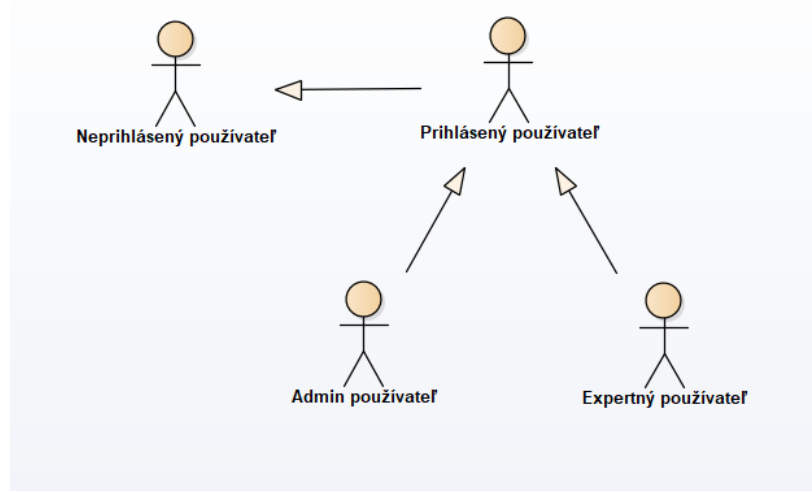
- V doméne projektu **byť informovaný** o [Príloha 3]:
 - Mojej aktivite a zmenách vykonaných v aplikácii
 - Sledované, nahlásené zmluvy a vyplnené doplň. otázky k nim
 - Moje komentáre a apelovanie na ne ďalšími prihl. používateľmi
- **Vyhľadávat' v zmluvách**
 - **Rozšírené vyhľadávanie** - špecifikácia kritérií vyhľadávania na základe:
 - Mnou sledovaných zmlúv
- **V kontexte zmlúv** [Príloha 4]:
 - **Označiť/sledovat' vybrané zmluvy** za účelom sledovanie aktivity v zmluvách (napr. nové komentáre, atď.)
 - Pridávanie poznámok k zmluvám - **vyznačovanie kontroverzných pasáží** (aby si ich ostatný použ. všimli)
 - **Pridávat' komentáre k zmluvám**, ktoré môžu pomôcť upozorniť na nezrovnalosti
 - Pridávanie súvisiace dokumenty k zmluve
 - **Nahlásiť zmluvu ako kontroverznú**
 - **Označovat' komentáre ako nevhodné a odpovedať, hlasovat' za existujúce komentáre**

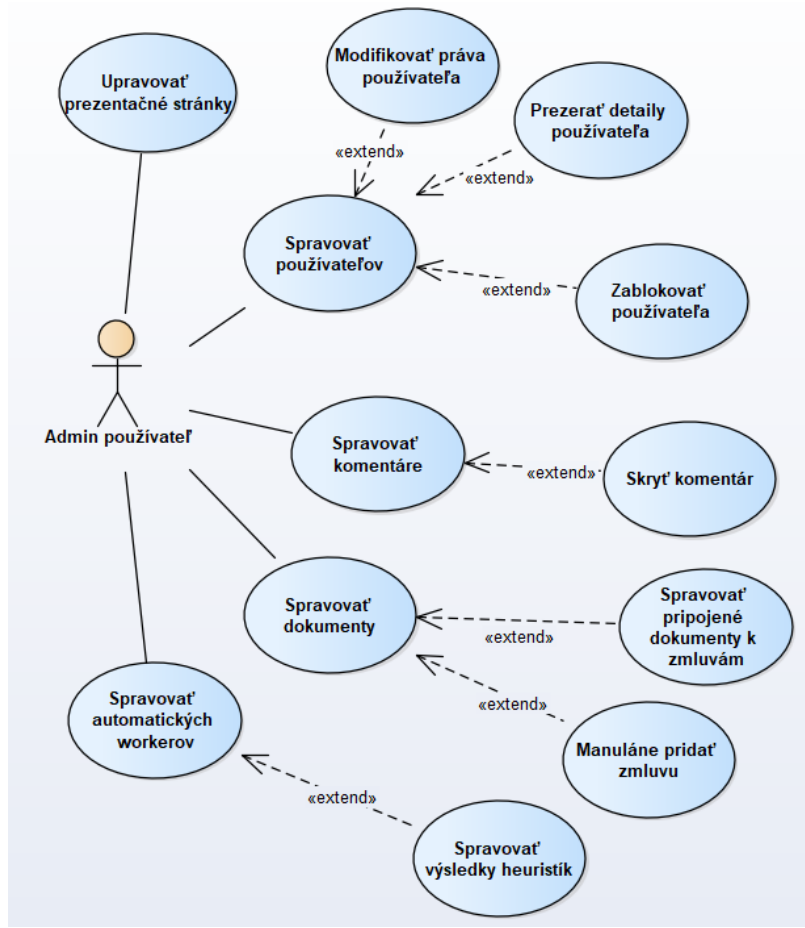
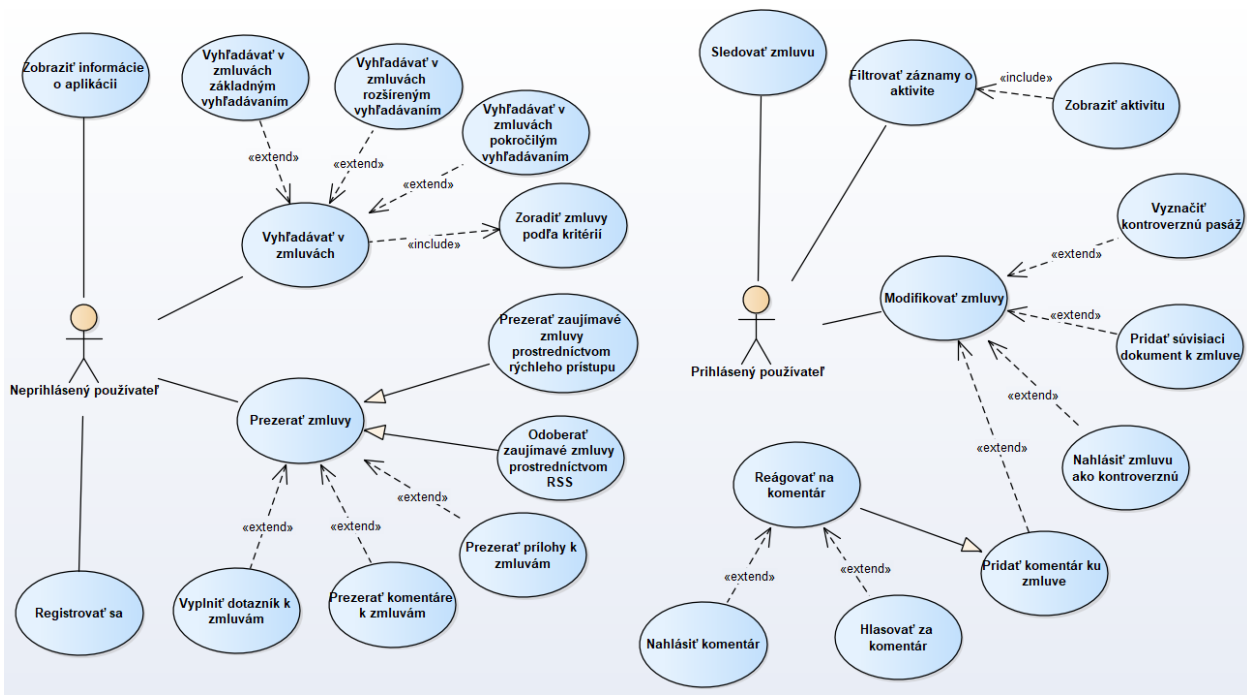
Ako **prihlásený administrátor**, vrátane ponúkanej funkcionality pre prihláseného bežného používateľa môžem:

- V doméne **používateľov** [Príloha 5]:
 - Prezerat' detaily všetkých prihlásených používateľov
 - Sťahovať tieto informácie
 - Sledovat' aktivitu vybraných používateľov
 - Udeľovat' práva používateľom - admin, expert
 - Banovat' vybraných používateľom v kontexte IP adresy alebo celkového účtu
- V doméne **komentárov** [Príloha 6]:
 - Spravovat' komentáre
 - Novopridané, skryté a nahlásené
- V kontexte **zmlúv** [Príloha 8]:
 - **Skryť alebo zverejniť zmluvu**

- *Editovať metadáta zmluvy (názov, cena, dátum zverejnenia, dátum platnosti do, dátum účinnosti, súvisiace dokumenty)*
- V kontexte *vyhľadávania zmlúv* [Príloha 9]:
 - *Skryť alebo zverejniť zmluvu*
- V doméne *dokumentov/zmlúv* [Príloha 6]:
 - Sledovať nahlásené *kontroverzné* zmluvy, ktoré boli označené používateľmi
 - Spravovať a *schvaľovať pridané dokumenty* ku zmluvám (od ostatných používateľov)
 - Manuálne *pridávať zmluvy* do databázy zmlúv
 - Prezeráť výsledky kontroly CRZ - porovnanie počtu zmlúv v aplikácii a na CRZ za účelom identifikovania odstraňovania/zmeny zmlúv v CRZ (s najväčšou pravdep. nefunkčné).
 - Prezeráť existujúce heuristiky (domnievame sa, že sú ťahané z DB a dajú sa len prezeráť)
 - Prezeráť zoznam skrytých zmlúv a znovu ich zverejniť
 - Nastavenia (nevieme však presne čo niektoré políčka znamenajú) - nastavenia zobrazenia počtu kontroverzných zmlúv na hlavnej strane, ostatné sme nedokázali identifikovať.
 - Spravovať workerov (Resque) pre automatické sťahovanie dokumentov - presmerovanie na samostatné webové rozhranie pre správu automatických jobov knižnice Resque.
- V doméne *statických prezentačných stránok* [Príloha 7]:
 - Upravovať statické prezentačné stránky, ktoré majú prevažne informačný charakter - o projekte, často kladené otázky, atď.
 - Editor pre úpravu textu podporuje Markdown syntax (<https://daringfireball.net/projects/markdown/syntax>)

2. Technický pohľad





3. Prílohy

Príloha 1

Otvorené zmluvy

Výhody Otvorených zmlúv | O projekte | FAQ | RSS | Martin | Odhláste sa

Ako môžem pomôcť?
Kontakt
Často kladené otázky
Podmienky používania
O projekte
Kódex diskutujúceho
Autorské práva
Výhody Otvorených zmlúv
Čo sú cookies?

O projekte

OtvorenéZmluvy.sk sú investigatívna stránka Aliancie Fair-play a Transparency International Slovensko, ktorej úlohou je pomôcť občanom čítať, vyhľadávať a posudzovať výhodnosť zmlúv uzatvorených štátom a štátnymi inštitúciami.

Prostredníctvom automatizovaných analýz a zapojenia verejnosti do hodnotenia zmlúv (crowdsourcingu) sa snaží upozorniť na zmluvy hodné preskúmania - od zmlúv uzavretých na vysoké sumy až po zmluvy obsahujúce nejasnosti alebo formálne chyby.

Od januára 2011 je na Slovensku povinné zverejňovať všetky zmluvy týkajúce sa nakladania s verejnými

Príloha 2

Otvorené zmluvy

Výhody Otvorených zmlúv | O projekte | FAQ | RSS | Martin | Odhláste sa

FIIT STU

Základné vyhľadavanie na základe kľúčových slov

Spolu 21 907 € v 172 zmluvách

Zoradiť podľa Celkovej sumy

Pokročilé vyhľadavanie definovaním kombinácie výrazov, operátorov a sledovaných polí
Operátor: AND (a súčasne), OR (alebo), TO (ohraničenie množs.)
? (ľubovoľný znak), ! (negácia), * (ľubovoľná komb.)
Sledované polia: customer, total_amount, department, atď.
Príklad: customer:F*T ST? AND total_amount:[1 TO 100000]

Rozšírené vyhľadavanie na základe sledovaných polí - zaškrtnutím jednotlivých kritérií

Objednávateľ	Objednávateľ	Celková suma
Slovenská technická un... (99)	1. Zmluva o dielo č. 1100034706/2013/5400/025	56 943 984 €
SLOVENSKÁ TECHNICKÁ ... (13)	Objednávateľ: Železnice Slovenskej republiky	5 688 118 €
Ministerstvo školstva,... (7)	Dodávateľ: ELZA - Elektromontážny závod Bratislava a.s. ELZA - Elektromontážny závod Bratislava, a.s.	3 818 531 €
Slovenská technická un... (3)	Dátum zverejnenia: 17.01.2014	
Ministerstvo financií SR (3)	2. Sprístupenie ďalších sledovaných polí sa zobrazí v rozpočtu Ministerstva školstva, vedy, vysokej školy a športu SR	
QBSW, a.s. (2)	Objednávateľ: Ministerstvo školstva, vedy, výskumu a športu SR	
MONOGRAM Technologies,... (2)	Dodávateľ: Slovenská technická univerzita v Bratislave	
Železnice Slovenskej r... (1)	Dátum zverejnenia: 28.04.2011	
Úrad pre verejné obsta... (1)	Zmluva spĺňa kritériá: stavebné práce	
enprovia, s.r.o. (1)	4. Medzinárodné centrum excelentnosti pre výskum inteligentných a bezpečných informačnokomunikačných technológií a systémov	2 991 975 €
	Objednávateľ: Agentúra Ministerstva školstva, vedy, výskumu a športu Slovenskej republiky pre štrukturálne fondy EÚ	
	Dodávateľ: Atos IT Solutions and Services s.r.o. Atos IT Solutions and Services, s.r.o.	
	Dátum zverejnenia: 04.02.2014	

Príloha 3

Otvorené zmluvy Výhody Otvorených zmlúv | O projekte | FAQ | RSS | Martin | [Odhláste sa](#)

Moje zmluvy

Po stlačení profilu
si prezerat aktivitu

Posledné zmeny

- Otvorenie sledovaných zmlúv
- Sledovanie zmlúv
- Hlásenie kontroverzných zmlúv
- Moje komentáre
- Cudzie komentáre
- Zodpovedanie otázok

Zoznam posledných zmien Načítať po 5 záznamoch ▾

Zmluva o dielo

Obstarávateľ: DataCentrum, Bratislava
Dodávateľ: PosAm, spol. s r.o.
Dátum zverejnenia: 16.03.2012

867 504 €

Prestali ste sledovať zmluvu
pred asi 10 hodinami

Otvorili ste si zmluvu
pred asi 11 hodinami

Príloha 4

Otvorené zmluvy Výhody Otvorených zmlúv | O projekte | FAQ | RSS | Martin | [Odhláste sa](#)

[← Späť na vyhľadávanie](#)

Názov	Kúpna zmluva č. 5/OZ07/IV.Q.2013/EAD	Sledovať zmluvu
Rezort	Ministerstvo pôdohospodárstva a rozvoja vidieka SR	<input type="checkbox"/> Sledovať
Objednávateľ	FOREST OBCHODNÍ s.r.o.	Prezerat komentáre <input type="checkbox"/> Diskutovať <input type="checkbox"/>
Dodávateľ	Lesy Slovenskej republiky, štátny podnik, Odštepny závod Trenčín (dni)	Nahlásiť zmluvu <input type="checkbox"/> Nahlásiť <input type="checkbox"/>
Cena	91 800 €	<input type="button" value="Vložiť zmluvu"/>
Dátum zverejnenia	10.10.2013	Prezeranie detailov
Dátum účinnosti	11.10.2013	Skopírovanie odkazu na zmluvu
Dátum platnosti do	31.12.2013	<input type="button" value="Páči sa mi to"/>
Pôvodný dokument	Odkaz na pôvodný dokument	<input type="button" value="Tweet"/>
Súvisiace dokumenty	<input type="text" value="url"/> <input type="button" value="Pridať"/>	<input type="button" value="G+"/>

Prepínanie

Hľadať v dokumente **medzi dodávateľom a objednávateľom**

Dodávateľ | Objednávateľ

Obchodné meno
RODEX CAR, s.r.o.

IČO
35860251

Dátum vzniku
20.06.2003

Hlavná činnosť
opravy cestných motorových vozidiel,
autoelektrikárstvo,
oprava karosérií, lakovačské

Kúpna zmluva č. 5/OZ07/IV.Q.2013/EAD - 242.83 kB

Strana 1 z 8

Prezerat informácie o dodávateľovi a objednávateľovi

Kúpna zmluva č. 5/OZ07/ IV.Q.2013/EAD

uzatvorená podľa § 409 a nasledujúcich Obchodného zákonníka,
na základe výsledkov EAD č. 129/IV.1./O/OZ07/2013
medzi

Zmluvná strana

Prezerat prílohy k zmluvám

1

Príloha 5

Otvorené zmluvy Administrácia | Výhody Otvorených zmlúv | O projekte | FAQ | RSS | admin | Odhláste sa

Admin rozhranie

Používatelia

Hľadaj

Uloženie informácií o používateľoch →

Aktivita používateľa →

Používateľ	Admin Expert	Registrácia	Posledné prihlásenie (IP adresa)	Štatistiky posledných 30 dní / spolu	Akcie
Kom.	Sled.	Otvo.	Odpo.	Kont.	
admin admin@crowdcloud.com	<input checked="" type="checkbox"/>	16.10.2017 - 10:09	17.11.2017 - 18:32 (147.175.185.2)	2 / 2 0 / 0 12 / 7 / 7 1 / 1	ban účtu ban IP

Zmena práv používateľa

Zabanovanie používateľa

Príloha 6

otvorenezmluvy.sk **Administrácia** | Heuristiky | Nastavenia | Resque | **Správa autom. workerov** | Odhlásiť

- Správa komentárov →
- Správa pridaných súvisiacich dokumentov →
- Správa automatických workerov →
- Správa autom. heuristik →
- Správa používateľov →
- Správa označených zmlúv →
- Manuálne pridanie zmluvy →

Príloha 7

Otvorené zmluvy **Výber stránky** | Administrácia | Výhody Otvorených zmlúv | O projekte | FAQ | RSS | admin | Odhláste sa

Ako môžem pomôcť?

Kontakt

Často kladené otázky

Podmienky používania

O projekte

Kódex diskutujúceho

Autorské práva

Výhody Otvorených zmlúv

Čo sú cookies?

Úprava textu → Upraviť stránku

Nadpis: O projekte

Obsah: #O projekte

OtvorenéZmluvy.sk sú investigatívna stránka [Aliancia Fair-play] (http://fair-play.sk "Aliancia Fair-play") a [Transparency International Slovensko] (http://www.transparency.sk "Transparency International Slovensko"), ktorej úlohou je pomôcť občanom čítať, vyhľadávať a posudzovať výhodnosť zmlúv uzatvorených štátom a štátnymi inštitúciami.

Prostredníctvom automatizovaných analýz a zapojenia verejnosti do hodnotenia zmlúv (crowdsourcingu) sa snaží upozorniť na zmluvy hodné preskúmania - od zmlúv uzavretých na vysoké sumy až po zmluvy obsahujúce nejasnosti alebo formálne chyby.

Od januára 2011 je na Slovensku povinné zverejňovať všetky zmluvy týkajúce sa nakladania s verejnými zdrojmi, majetkom štátu alebo samospráv. Tento krok robí verejnú kontrolu lepšie, no sám osebe nestačí.

Použite Markdown Uložiť

Príloha 8

Názov **Zmluva o poskytnutí finančných prostriedkov z rozpočtovej kapitoly MDV SR pre ŽSR na projekty modernizácie a rozvoja železničnej dopravnej cesty** [Skryť zmluvu](#)
 Editovať metadáta

Rezort [Ministerstvo dopravy, výstavby a regionálneho rozvoja SR](#)

Objednávateľ [Ministerstvo dopravy a výstavby SR](#)

Dodávateľ [Železnice SR \(vek 24 rokov 3 mesiace 22 dní\)](#)
 Železnice Slovenskej republiky

Cena **46 220 000 €**

Dátum zverejnenia 16.04.2018

Skryť

Editovať

Sledovať

Diskutovať 0

Nahlásiť 0

<> Vložiť zmluvu

Príloha 9

Otvorené zmluvy Administrácia | Výhody Otvorených zmlúv | O projekte | FAQ | RSS | [admin](#) | [Odhláste sa](#)

Vyhľadávanie v zmluvách

[Zmazať všetky obmedzenia](#)

Spolu 49 967 196 € v 34 zmluvách. Zoradiť podľa Celkovej sumy

[Vytvoriť heuristiku](#) | [Odoberať zmluvy cez e-mail](#)

<p>Objednávateľ</p> <input type="text"/>	<p>1. Zmluva o poskytnutí finančných prostriedkov z rozpočtovej kapitoly MDV SR pre ŽSR na projekty modernizácie a rozvoja železničnej dopravnej cesty 46 220 000 €</p> <p>Objednávateľ: Ministerstvo dopravy a výstavby SR Dodávateľ: Železnice SR Železnice Slovenskej republiky Dátum zverejnenia: 16.04.2018</p> <p style="text-align: right;">Skryť zmluvu <input type="button" value="👁"/></p>
	<p>2. zmluva o dielo 2 671 200 €</p> <p>Objednávateľ: Železnice Slovenskej republiky Dodávateľ: TSS GRADE, a.s. Dátum zverejnenia: 16.04.2018</p> <p style="text-align: right;"><input type="button" value="👁"/></p>

Objednávateľ:

- Slovenská správa ciest... (8)
- Železnice Slovenskej r... (8)
- Národná diaľničná spol... (4)
- Ministerstvo dopravy a... (2)
- Slovenská pošta, a. s. (2)
- Štátny fond rozvoja bý... (2)
- ARMYSHOP TEAM s.r.o. (1)
- Letisko Poprad-Tatry, ... (1)
- Letové prevádzkové slu... (1)
- Obec Birkovce (1)

Príloha B: Návod na inštaláciu a nasadenie projektu

Následovný postup je uvedený pre server s nainštalovaným OS Ubuntu Server 17.04 :

V prípade že prostredie je už pripravené, môžeme začať krokom 10: Nasadenie pomocou Capistrano

1. Krok - Inštalácia RVM:

- pre účel inštalácie rvm je potrebné zadať do príkazového riadku nasledujúcu kombináciu príkazov:

```
curl -L get.rvm.io | bash -s stable
```

- ak príkaz vyhodí chybovú hlášku, pravdepodobne nedisponuje stroj verejným kľúčom alebo nemáme nainštalovaný program **curl**

- chybajúce doplnky vieme doinštalovať klasicky cez:

```
sudo apt-get install <meno_programu>
```

- v prípade chybajúceho verejného kľúča, kľúč jednoducho doinštalujeme príkazom (program gnupg2):
gpg2 --recv-keys <retazec_alphanumznakov_v_chyb_hlaske>

- naštartujeme rvm:

```
source ~/.rvm/scripts/rvm
```

2. Krok - inštalácia Ruby verzie 1.9.3

- ešte pred samotnou inštaláciou ruby cez rvm, potrebujeme nainštalovať závislosti pre systém príkazom:

```
sudo apt-get install build-essential openssl libreadline6 libreadline6-dev curl git-core zlib1g zlib1g-dev libssl-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt-dev autoconf libc6-dev ncurses-dev automake libtool bison subversion pkg-config libcurl4-openssl-dev python-software-properties libffi-dev nodejs
```

- Budeme potrebovať aj libread knižnicu

```
sudo apt-get install libreadline-gplv2-dev
```

- Nainštalujeme ruby verziu 1.9.3:

```
rvm install ruby-1.9.3-p545
```

- Nainštalujeme bundler aj dokumentáciu:

```
gem install bundler  
rvm docs generate-ri
```

- **Aktualizácia Ruby verzie na 2.1.10**

```
rvm install 2.1  
rvm use --default 2.1
```

- pridáme bundler a nainštalujeme gemy zo súboru

```
gem install bundler  
bundle
```

3. Krok - inštalácia Rails verzie 3.2.8

- Nainštalujeme nodejs

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -  
sudo apt-get install -y nodejs
```

- Nainštalujeme rails verziu 3.2.8:

```
gem install rails -v 3.2.8
```

4. Krok - Inštalácia PostgreSQL verzie 9.2 - 10 pre Zesty (Ubuntu 17.04):

- vytvoríme súbor:

```
vim /etc/apt/sources.list.d/pgdg.list
```

- vložíme do súboru nasledujúci riadok:

```
deb http://apt.postgresql.org/pub/repos/apt/ zesty-pgdg main
```

- importujeme repozitár pre postgresql verzie:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | \  
sudo apt-key add -  
sudo apt-get update
```

- Nainštalujeme požadovanú verziu postgresql:

```
sudo apt-get install postgresql postgresql-contrib libpq-dev
```

5. Krok - Inštalácia Redis

- nainštalujeme redis server a php redis

```
sudo apt-get install redis-server  
sudo apt-get install php-redis
```

- upravíme konfiguráciu redisu

```
vim /etc/redis/redis.conf
```

- pridáme nastavenia, uložíme zmeny, zavrieme súbor:

```
maxmemory 128mb  
maxmemory-policy allkeys-lru
```

- reštartujeme a znovu povolíme redis: príkazmi:

```
systemctl restart redis-server.service  
systemctl enable redis-server.service
```

6. Krok - Inštalácia Elastic Search (dole 0.9.0, novšia verzia zatiaľ nefunguje)

- V prípade, že nemáme nainštalovanú javu:

```
sudo apt-get install default-jre
```

pridáme JAVA_HOME do premenných prostredia:

```
sudo vim /etc/environment
```

pridáme riadok a uložíme súbor:

```
JAVA_HOME="<cesta_k_jvm>"
```

znovu načítame premenné prostredia:

```
source /etc/environment
```

- stiahneme elasticsearch inštaláciu z linku na oficiálnej stránke vo formáte .deb:

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.6.2.deb
```

- nainštalujeme stiahnutý súbor a spustíme elastic:

```
dpkg -i elasticsearch-5.6.2.deb
```

```
sudo systemctl start elasticsearch
```

ELASTIC 6.0.0

- stiahneme a nainštalujeme elasticsearch 6.0.0

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.0.0.deb  
sudo dpkg -i elasticsearch-6.0.0.deb
```

- počas inštalácie **vyhodí chybu** ohľadom nenájdenia JDK, aj keď máme JAVA_HOME nastavenú
- je potrebné upraviť obsah súboru **/etc/init.d/elasticsearch**, konkrétne treba pridať do množiny foldrov, kde sa hľadá JVM defaultne aj našu absolútnu cestu k aktuálnej verzii JDK
- pokúsime sa spustiť elastic -> `sudo systemctl start elasticsearch` (ak chyba, opravíme podľa hintu v chybovej hláske, reštartovanie daemonov)

7. Krok - Inštalácia OCR a ImageMagick

- programy nainštalujeme dvoma jednoduchými príkazmi

```
sudo apt-get install tesseract-ocr-slk tesseract-ocr-slk-frak  
sudo apt install imagemagick
```

8. Krok - Inštalácia HTTP servera Apache 2.4.25

- Aplikáciu nainštalujeme nasledovným príkazom:

```
sudo apt-get install apache2
```

- Spustíme kontrolu správnosti syntaxe konfiguračného súboru Apache:

```
sudo apache2ctl configtest
```

- Ak je všetko v poriadku, povolíme porty na prenos údajov pre Apache vo firewallle operačného systému a skontrolujeme stav:

```
sudo ufw allow in "Apache Full"  
sudo ufw app info "Apache Full"
```

- Reštartujeme server Apache:

```
sudo systemctl restart apache2
```

- Pre pohodlnejší prenos (neregulárnych) súborov (napr. zdrojový kód stránky) na server nainštalujeme ZIP package

```
sudo apt-get install unzip
```

9. Krok - Inštalácia dependencies pre aplikáciu

- gem curb potrebuje predtým nainštalovať libcurl príkazmi:

```
sudo apt-get install libcurl4-openssl-dev  
alebo  
sudo apt-get install libcurl4-gnutls-dev
```

10. Krok - setup projektu

projekt sa dá nasadiť z lokálneho stroja na server pomocou Capistrano:

```
bundle exec cap staging deploy
```

alebo pre deploy na produkciu

```
bundle exec cap production deploy
```

Server treba nakonfigurovať v config/deploy/production.rb a staging.rb. V týchto súboroch si nakonfigurujeme branchu ktorú chceme deploynúť a server kam to chceme deploynúť.

V /config/deploy.rb nastavíme repozitár do atribútu:

```
set :repository, "git@github.com:[repozitár]"
```

Ak chceme aby od nás Capistrano nepýtalo heslo, môžeme taktiež vytvoriť SSH kľúč podľa návodu na: <https://help.github.com/articles/connecting-to-github-with-ssh/>

následne použijeme príkazy na serveri na setup projektu

```
bundle install  
bundle exec rake db:create  
bundle exec rake db:schema:load  
bundle exec rake db:migrate  
bundle exec rake db:seed  
bundle exec rake db:migrate  
bundle exec rake crowdcloud:index:rebuild
```

V prípade že nasadzujeme iba novšiu verziu alebo sa na serveri už nachádza staršia verzia projektu, spustíme iba príkazy:

```
bundle install  
bundle exec rake db:migrate  
bundle exec rake crowdcloud:index:rebuild
```

Sťahovanie zmlúv z CRZ sa dá spustiť nasledujúcimi príkazmi:

crowdcloud:crz:download alebo crowdcloud:crz:download:xml
rake resque:work

Príloha C: Návrhy používateľského rozhrania (mockupy) - zobrazenie informácií o subjekte

The screenshot shows a web browser window with the address bar containing the URL <http://otvorenezmluvy.sk/documents/123-foobar>. The page title is 'Otvorené zmluvy'. The main content area displays contract details for 'Centrálna licenčná zmluva na služby'.

Späť na vyhľadávanie

Názov	Centrálna licenčná zmluva na služby
Rezort	Ministerstvo financií SR
Objednávateľ	Ministerstvo financií Slovenskej Republiky
Dodávateľ	+ SAP Slovensko s.r.o. (vek 13 rokov 11 mesiacov 17 dni)

Obchodné meno	SAP Slovensko s.r.o
IČO	12345678
Dátum vzniku	1.1.2012
Dátum zániku	
Adresa sídla	Dlhá 123/47, Poprad, 05801
Okres	SK0416 - Okres Poprad
Obec	SK0416523381 - Poprad
Právna forma	121 - Akciová spoločnosť
Hlavná činnosť	46310 - Veľkoobchod s ovocím a zel.
Inštituc. sektor	11002 - NSNK
Druh vlastníctva	2 - Súkromné tuzemské
Kateg. veľkosti org.	00 - Nezistený

Cena	12 523 €
Dátum zverejnenia	22.12.2011
Dátum účinnosti	23.12.2011
Zmluva spĺňa kritériá	Nákladné zmluvy
Pôvodný dokument	Odkaz na pôvodný dokumenty
Súvisiace dokumenty	Dodatok č.1

[Sledovať](#)

[Diskutovať](#)

[Nahlásiť](#)

[Vložiť zmluvu](#)

← Späť na vyhľadávanie

Centrálna licenčná zmluva na služby

Názov	Centrálna licenčná zmluva na služby
Rezort	Ministerstvo financií SR
Objedávateľ	Ministerstvo financií Slovenskej Republiky
Dodávateľ	SAP Slovensko s.r.o. (vek: 13 rokov 11 mesiacov 17 dní)
Cena	12 523 €
Dátum zverejnenia	22.12.2011
Dátum účinnosti	23.12.2011
Zmluva spĺňa kritériá	Nákladná zmluva
Pôvodný dokument	Odkaz na pôvodný dokument
Súvisiace dokumenty	Dodatok č.1

[Sledovať](#)
[Diskutovať](#)
[Nahásiť](#)
[Vložiť zmluvu](#)

Hľadať v dokumente...

Dodávateľ | Objedávateľ

SAP Slovensko s.r.o.

Obchodné meno: SAP Slovensko s.r.o.
 IČO: 12345678
 Dátum vzniku: 1.1.2012
 Hlavná činnosť: 46310 - Veľkoobchod s ovocím a zeleninou
 Vlastné imanie: 123 589 €
 Zisk: 78 123 € (rok 2017)
 102 028 € (rok 2016)
 Hodnota nehmotných: 278 000 €
 Počet uzatvorených zmlúv: 28 zmlúv
 spolu za 180 050 €
 Vyplatené mzdy: 48 124 €
 Počet zamestnancov: 27 zamestnancov

[Viac](#)

Zdroj informácií: Slovenská Digitalizácia, Register ÚV, Register FOD, Poskytovateľ nemajetku, Register právnických osôb, súd - právne informácie

Text zmluvy - 1,79 MB

[Sledovať](#)
[Diskutovať](#)
[Nahásiť](#)

Centrálna licenčná zmluva na služby súvisiace s licenciami softvérových produktov SAP

Slovenská republika
 Bratislava
 Miestomesto: Bratislava, Slovenská republika
 so sídlom Štefánikova 5, 817 82 Bratislava, komárce: vedecov sketubalio úndu
 Právny úrad: Digitalizácia
 IČO: 90151742
 Bankové spojenie: Súčas podnikateľa, č ústa: 7000001400/8180
 (faktúra „Základná“)

← Späť

Informácie o subjekte

Obchodné meno	DodávateľJán s.r.o.
IČO	12345678
Dátum vzniku	1.1.2012
Adresa sídla	Diňa 12347, Poprad, 05801
Okres	SK0416 - Okres Poprad
Obec	SK0416523381 - Poprad
Právna forma	121 - Akciová spoločnosť
Hlavná činnosť	46310 - Veľkoobchod s ovocím a zeleninou
Inštitúcia	11002 - NISNK
Druh vlastníctva	2 - Súkromné tuzemské
Kateg. veľkosti org.	00 - Neuzistený

[Sledovať](#)
[Diskutovať](#)
[Nahásiť](#)

Účtovné informácie

Vlastné imanie	123 589 €
Zisk	78 123 € (rok 2017) 102 028 € (rok 2016)
Hodnota nehmotných	278 000 €
Počet uzatvorených zmlúv	28 zmlúv spolu za 180 050 €
Vyplatené mzdy	48 124 €
Počet zamestnancov	27 zamestnancov

Dátum poslednej zmeny: 28.03.2017
 Zdroj informácií: Slovenská Digitalizácia, Register ÚV, Register FOD
 Poskytovateľ nemajetku, Register právnických osôb, súd - právne informácie

Diskusia

Jozko Mrovička, pred 4 dňami
 Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

▲ 2 ▼ | [Odpovedať](#) | [Trvalý odkaz](#) | [Upozorniť na nevhodný komentár](#)

Jozko Karota, pred 7 dňami
 Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

▲ 2 ▼ | [Odpovedať](#) | [Trvalý odkaz](#) | [Upozorniť na nevhodný komentár](#)

[Pridať príspevok](#)

Príloha D: Technická dokumentácia od Michala Barlu

Introduction

CrowdCloud is an application that helps to collect, search, analyze vast amounts of documents and through power of crowdsourcing find the interesting ones. We use powerful features that help users to slice-and-dice documents using faceted search and notify users when documents matching their criteria are added. Users can interact with documents by commenting, marking them as suspicious and even highlighting parts of documents. Documents are automatically downloaded from public sources and processed through OCR that extracts text even from non-text documents.

The pilot application running on www.otvorenezmluvy.sk collects Slovak government contracts (documents) that are published at various official government sites.

Installation

CrowdCloud is a classic Ruby on Rails 3.2 application, gems are managed using Bundler, application is deployed via Capistrano. Database schema is managed using raw SQL schema dumper (not Rails default).

Full source code is available at <https://github.com/otvorenezmluvy>

Prerequisites

- Linux environment (debian-based is preferred)
- MRI Ruby 1.9.3+
- PostgreSQL database 9.1+
- ElasticSearch
- Redis
- Tesseract OCR
- GraphicMagick
- MemCache (optional)

Installation guide

1. Install and start prerequisites if needed (e.g. elasticsearch, redis).
2. Add eulang analyzer (for handling accented characters) to elasticsearch.yml config

index:

analysis:

analyzer:

eulang:

type: custom

tokenizer: standard

filter: [standard, lowercase, asciifolding, stop]

3. Application settings for database and elasticsearch can be found in database.yml and crowdcloud.yml
4. Run default Rails application bootstrap (bundle install, rake db:create, rake db:schema:load, rake db:seed)

5. Create elasticsearch indexes & index data (if available) with rake crowdcloud:index:rebuild
6. Optional. Start downloading data from CRZ source by running rake crowdcloud:crz:download and rake resque:work

Project structure

Top level architecture

Project consists of these three main parts:

- Wrappers/parsers
- Public web interface
- Administration interface

Data model

The core model works with arbitrary documents and attachments as defined in the Documents Core module, the pilot implementation (www.otvorenezmluvy.sk) deals with contracts (a specialization of document) from two sources with corresponding modules CRZ and Egov.

Documents Core

- Document - represents a single published document
 - has_many Attachments
 - has_many Comments
 - has_many Heuristics
- Attachment - represents an attachment (pdf/text) file related to a Document
 - has_many Pages
- Page - represents a single page of an Attachment
 - has_many Comments

CRZ Specific Documents

- Crz::Document < Document - abstract class represents a document from CRZ portal
 - has_one Crz::DocumentDetail
- Crz::DocumentDetail - represents various specific fields for a Crz::Document
- Crz::Contract < Crz::Document - represents a contract from CRZ portal
 - has_many Crz::AppendixConnections
- Crz::Appendix < Crz::Document - represents an appendix from CRZ portal
- Crz::AppendixConnection - represents a connection to an appendix from a Crz::Contract

EGov Specific Documents

- Egovsk::Document < Document - abstract class represents a document from Egov.sk portal
 - has_one Egovsk::DocumentDetail
- Egovsk::DocumentDetail - represents various specific fields for a Egovsk::Document
- Egovsk::Contract < Egovsk::Document - represents a contract from Egov.sk portal
- Egovsk::Appendix < Egovsk::Document - represents an appendix from EGov.sk portal

Intelligence

- Heuristic - represents an automatic heuristic for scoring document suspiciousness.
 - has_many Documents

Questionnaire

- Question - represents a question from questionnaire.
 - has_many QuestionChoices
- QuestionChoice - represents a question choice for a particular Question
 - has_many QuestionAnswers

User activities

- Comment - represents a user comment on a Document and/or particular Page area of Document.
 - has_many Votings
 - has_many CommentReports
- CommentReport - represents a user reporting of a Comment (e.g. malicious, abusive...)
- Voting - represents users voting for/against a particular Comment
- Watchlist - represents a Document watched by a User
- DocumentOpening - represents a Document opened by a User
- QuestionAnswer - represents an user answer to a Question

Stream

- Event - represents an event in user's activity stream and has following subclasses
 - UserRegisteredEvent - represents an event when user has registered
 - QuestionAnswerEvent - represents an event when a question is answered
 - DocumentEvent - abstract class representing event related to a document
- DocumentOpenedEvent - represents an opening of document by user
- WatchingStartedEvent/WatchingStoppedEvent- represents an event by user that starts/stops watching a document
- ControversyReportedEvent - represents an event when users marks a document as controversial
 - CommentEvent - abstract class representing event related to a comment
- MyCommentEvent - represents an event related to users comment
- OthersCommentEvent - represents an event related to comments by other users (e.g. on a watched document)

Processes

Downloading and preprocessing contracts

New contracts are downloaded periodically, from a cron job. The cron job is defined in config/schedule.rb. This file is automatically picked up by the whenever gem which

translates it into a cron configuration and updates local crontab on each deploy. Jobs are then managed by unix daemon cron and run at the specified intervals. The cron job invokes rake tasks defined in lib/tasks/crowdcloud.rake which start the download process. The rake tasks check the remote websites (currently CRZ and egov) for new updates and schedule downloads for each new contract. The rake tasks themselves do not download data, they just parse remote websites/invoke third-party APIs and schedule Resque jobs. The actual downloading and preprocessing happens in Resque jobs. The actual logic and execution depends on the scraped website, but these steps are common among all scraping

jobs:

- download the HTML with contract metadata
- parse the HTML, extract metadata and links to attachments
- download the pdf attachments
- split pdfs into images (one image per page) using a tool graphicsmagick
- extract text, either using pdftotext tool or Tesseract OCR

The metadata is saved in database and the images and texts on filesystem in public/documents.

Note that in order to process the queued jobs, separate resque workers must be running. Resque workers can be started by invoking rake resque:work.

Components

Document viewer

Document viewer is a components responsible for displaying the contracts. It can display scanned contract, its extracted plain text and provides tools for interactive annotating. It is implemented as a standard Backbone.js application in app/assets/javascripts/document_viewer. The same viewer is used both on portal to show contracts and in the widget, although with different configuration. Document viewer is instantiated as shown in the example and accepts several options.

```
var dv = DV.DocumentViewer.init({
  container: '.document',

  attachments: <%== @attachments.to_json %>,
  currentUser: "<%= current_user.label %>",
  onReply: CrowdCloud.Comments.replyTo,
  onShowReplies: CrowdCloud.Comments.showComment,
  width: '673px',
  height: '990px',
  annotationsAllowed: true,
  zoom: false,
  commentList: false
});
```

Option name	Type	Description
container	string	CSS selector for the wrapping DOM element where the document viewer shall be rendered.
attachments	array	JSON-encoded array of attachments. Each attachment should provide: name, number and array of pages. Each page should provide: scanUrl, textUrl and optionally an array of annotations.
comments	array	JSON-encoded array of comments related to the document.
currentUser	string	Name of the user. It is displayed when adding new annotation.
onReply	function	Callback invoked when user chooses to reply to annotation.
onShowReplies	function	Callback invoked when user chooses to see replies to an annotation.
width	integer	Width of the document viewer in pixels. parseInt'd before usage.
height	integer	Height of the document viewer in pixels. parseInt'd before usage.
annotationsAllowed	boolean	Should we allow to annotate parts of the contract?
zoom	boolean	Can we zoom?
commentList	boolean	Should we display a toggleable tab which shows all comments? If set to true, the comments are pulled from the comments option.

Extending the project

Adding a new source parser / wrapper

Before adding a parser/wrapper for a new source you will probably need to add a model for holding custom fields for your documents. The best way to start is to look how existing models work and extend Document class (e.g. Crz::Document, Crz::DocumentDetail or Egovsk::Document, Egovsk::DocumentDetail).

A word of warning: These models don't use classic single table inheritance that is common for RubyOnRails, but for the sake of extensibility pull custom fields to separate models

*DocumentDetail.

If you want your documents custom fields to be indexed (to fully leverage the power of faceted fulltext searching) make sure you add them to Document::Indexable mixin and refer to the next

section of how to add a new facet to the interface for such fields.

There are no restrictions on how to download, parse and process your own documents. Look at existing parsers for Crz and Egovsk for example production ready parsers.

Warning: Make sure documents are saved using `Configuration.document_repository.save(document)` method instead of the default `ActiveRecord::Base.save`. Using a repository object decouples model from various callbacks (indexing, heuristics calculations, etc.) and is considered a best practice for easier testing.

Extending faceted search

Adding a facet

Facets for search GUI are defined in `Settings.facets` method and there are multiple ready-to-use types of facets:

- `FulltextFacet` - fulltext search in document (`_all` field in ElasticSearch) you probably need only one of this facet.
- `SearchableTermsFacet` - facet containing strings with autocomplete feature
- `DateFacet` - facet for fields containing date, default bucketing by month.
- `RangeFacet` - facet for arbitrary ranges
- `StatisticalFacet` - facet for calculating statistical properties on fields (e.g. sum)

After adding new facet/s to this definition you might need to supply some translations and templates for customizing the look&feel to make it fully working, but the process is standard and

should be straightforward for any Rails developer.

Adding a sort field

Adding a new field that should be used for sorting is done by adding it to definition defined in `Configuration.factic[:sort_fields]`. Again, some translations need to be added.

Contact information

Authors: minio, s.r.o. (kontakt@minio.sk), Aliancia Fair-Play (www.fair-play.sk), Transparency International Slovensko (www.transparency.sk)

Feel free to contact us anytime.

Príloha E: Návod na pridanie dát z RPO a RPVS

1. Ako prvé je potrebné stiahnuť si posledný dump RPO zo stránky slovensko.digital kliknutím na tlačidlo **rpo.sql.gz**
2. Dump následne rozbalíme.
3. RPO dáta vkladáme v troch krokoch 1. číselníky, 2. tabuľka organizations 3. zvyšné tabuľky organization_
4. Príkaz **sed -n '/INSERT\ INTO\ esa2010_codes/,/);/p' rpo.sql >> ciselniky.sql** vyfiltruje všetky INSERT príkazy pre tabuľku esa2010_codes z dumpu a uloží ich do súboru ciselniky.sql
5. Toto opakujeme pre zvyšné číselníky (legal_forms, main_activity_codes, share_forms, share_types, stakeholder_types)
6. Príkazom **psql -h localhost -d crowdcloud_development -U postgres -f ciselniky.sql** načítame údaje do databázy, môžeme použiť aj inú databázu ako development V prípade, že nechceme aby nám do konzoly vypisovalo po každom inserte správu, pripíšeme prepínač **-q**
7. Príkazom **sed -n '/INSERT\ INTO\ organizations\ ,/);/p' rpo.sql > organizations.sql** získame všetky údaje pre tabuľku organizations
8. Do pôvodnej tabuľky organizations sme ako posledný stĺpec pridali cislo_vlozky, ktoré v dumpe nemáme, preto na koniec každého riadka vložíme NULL príkazom: **sed -i 's/]];/, \ NULL\);/g' organizations.sql**
9. Keďže tabuľka organizations obsahuje cudzí kľúč na seba a dáta v dumpe nie sú v správnom poradí, prihlásime sa do našej databázy a dočasne vypneme všetky skryté triggery: **ALTER TABLE organizations DISABLE TRIGGER ALL;**
10. Príkazom **psql -h localhost -d crowdcloud_development -U postgres -f organizations.sql** vložime organizations data do databázy
11. Znovu zapneme triggery: **ALTER TABLE organizations ENABLE TRIGGER ALL;**
12. Príkazom **sed -n '/INSERT\ INTO\ organization_/,/);/p' rpo.sql > insert.sql** si vyfiltrujeme zvyšné inserty do súboru insert.sql
13. Príkazom **psql -h localhost -d crowdcloud_development -U postgres -f insert.sql** vložíme do databázy
14. Nepotrebné sql súbory môžeme vymazať aby nám zbytočne nezaberali diskový priestor

Ak si chceme vyfiltrovať z dumpu, všetky príkazy okrem insertov, použijeme príkaz:


```
sed -n '/INSERT\ INTO/,/);!p' rpo.sql > tables.sql
```

Synchronizácia RPO dát sa deje pomocou rake tasku, ktorý je spustený ako cron job. Pre manuálne spustenie rake tasku použijeme príkaz: `rake rpo:synchronise` Tento rake task získa všetky zmeny v RPO od včerajšieho dátumu a aktualizuje ich.

Na pridanie dát z RPVS je najskôr potrebné mať pridané dáta z RPO. Následne je potrebné spustiť rake task pomocou príkazu: **rake rpv:synchronise** ktorý pre všetky záznamy v tabuľke `OrganizationIdentifierEntries` získa dáta z RPVS. Pri priebežnom aktualizovaní dát RPO sa automaticky volá rake task `rpvs:synchronise_by_ico`, ktorý pre každú aktualizovanú firmu obsahujúcu ICO získa údaje z RPVS.

Príloha F: Zmeny a plán aktualizácie RoR 3.2 -> 4.0

UPDATE action

- Rails 4 používa pri UPDATE akcii HTTP **patch** namiesto **put**, ako bolo tomu v predošlej verzii
 - skontrolovať config routes.rb, ak su nejaké custom routes definované
 - skontrolovať form for statements, kde bude potrebné pridať
 - s patchom súvisí aj JSON objekt v počas UPDATE volania, skontrolovať sofistikované volania

Gemfile

- bude potrebné vyhodíť assets skupinu ak je
- pridať rails groups do `config/application.rb`

vendor/plugins

- rails už neťahá pluginy odtiaľ
- všetko pôjde do Gemfile alebo si vytvoríť explicitny konfigurač, odkiaľ sa to poťahá

ACTIVE RECORD

- Identity map už nie je podporovaná
- Pribudli nové dátové argumenty string a integer pre *delete* akciu
- Nie je viac potrebné premenovanie indexov
- zmena `serialized_attributes attr_readonly` na class metódy
- boli vyhodené `attr_accessible attr_protected`, namiesto toho to rieši nový gem [Protected Attributes gem](#)
- scope používa volateľný objekt -> {}
- zastaralý `ActiveRecord::Fixtures` na `ActiveRecord::FixtureSet`
- zastaralý `ActiveRecord::TestCase` nahradený `ActiveSupport::TestCase`.
- zastaralé hashové vyhľadávanie `Book.find(:all, conditions: { name: '1984' })` bude treba zameniť za `Book.where(name: '1984')`
- zastaralé dynamické metódy

- All dynamic methods except for `find_by_...` and `find_by_...!` are deprecated. Here's how you can handle the changes:
 - `find_all_by_...` becomes `where(...)`.
 - `find_last_by_...` becomes `where(...).last`.
 - `scoped_by_...` becomes `where(...)`.
 - `find_or_initialize_by_...` becomes `find_or_initialize_by(...)`.
 - `find_or_create_by_...` becomes `find_or_create_by(...)`.
- Note that `where(...)` returns a relation, not an array like the old finders. If you require an Array, use `where(...).to_a`.
- These equivalent methods may not execute the same SQL as the previous implementation.
 - možno bude treba [activerecord-deprecated finders gem](#), ktorý podporuje staršie dopyty, ak by bol problém so staršími dopytmi, ktoré riešia niečo špecifické
 - zmena implicitného join table pre vzťah `has_and_belongs_to_many`
 - `join_table` možnosť bude pridaná pre spoločne prefixy

Active Resource

- extrahovaná do vlastného gemu, ak je potrebná, stačí pridať [Active Resource gem](#)

Active Model

- zmenený prístup k chybovým hláškam *active modelu*
- `config/initializers/wrap_parameters.rb` (root in json defaultne na false)

```
# Disable root element in JSON by default.
# ActiveSupport.on_load(:active_record) do
#   self.include_root_in_json = false
# end
```

Action Pack

- došlo k šifrovaniu cookie session
- nový generátor, slúži ako základ pre generáciu cookies a verifikáciu
- už vygenerované signed cookies budú aktualizované ak necháme starý token a pridáme novú bázu pre privátny kľúč v súbore `config/initializers/secret_token.rb`
 - jedine ak je dôvod a nemáme závislosti v externom javascripte pre cookie read
- vyhodnený `ActionController::Base.asset_path` / assets pipeline
- zastaralý `ActionController::Base.page_cache_extension` / `ActionController::Base.default_static_extension`
- vyhodnený caching Action a Page / `actionpack-action_caching` a `actionpack-page_caching`

- vyhodený parser XML parametrov / `actionpack-xml_parser`
- no layout false vracať miesto nil
- mamcached client teraz cez gem dalli
- **zastaralé**
 - `dom_id` a `dom_class` v kontroleroch, treba doplniť (`ActionView::RecordIdentifier`)
 - `link_to` nepoužíva možnosť `:confirm`, namiesto používať `data:...`
 - Argument error ak sú clashing name routes definované
 - nonJS/CSS do `app/assets`
 - nepotrebné pri url do skriptov už `default_url_options[:script_name]`
 - `ActionController::Integration` / `ActionDispatch::Integration`
 - `ActionController::IntegrationTest` / `ActionDispatch::IntegrationTest`
 - `ActionController::PerformanceTest` / `ActionDispatch::PerformanceTest`
 - `ActionController::AbstractRequest` / `ActionDispatch::Request`
 - `ActionController::Request` / `ActionDispatch::Request`
 - `ActionController::AbstractResponse` / `ActionDispatch::Response`
 - `ActionController::Response` / `ActionDispatch::Response`
 - `ActionController::Routing` / `ActionDispatch::Routing`

Cache

- prebehla zmena namespaces pre cacheovanie

Active Record Observer a Action Controller Sweeper

- teraz v geme `rails-observers`

Sprockety

- `assets:precompile:primary` a `assets:precompile:all` vyhodené, použiť `assets:precompile`
- Zmena `config.assets.compress` na `config.assets.js_compressor`

Sass

- `asset-url("rails.png", image)` na `asset-url("rails.png")`

Ktoré súbory bude potrebné zmeniť ? (Plán)

1. Prvým krokom bude aktualizácia Gemfile, teda pridanie príslušnej verzie RoR, na ktorú chceme aktualizovať. Následne bude potrebné aktualizovať ostatné gemy závislé od RoR verzie. Automaticky by sme mali toto zabezpečiť cez **bundle update**.

2. Po update bude potrebné vyhodit **strong_parameters** gem, nakoľko ho RoR 4 už obsahuje.
3. Následne skontrolujeme **config/routes.rb** kde došlo k zmenám, kedy http používa patch. Teda všetky custom routes používajúce PUT bude treba upraviť. V našom prípade pôjde maximálne o jednu zmenu. PUT volania prepíšeme na PATCH a asociované **form for** definujeme metódu **method: put**. Potrebné bude aj upraviť aj definovanú cestu **static_pages** nakoľko je tam daná 2x
4. Public a private key v config priečinkoch bude potrebné premenovať na `config.site_key` `config.secret_key` (`initializers/recaptcha.rb` subor, uprava `initializers/devise.rb`
5. ActiveSupport::Memoizable nie je podporované, bude potrebné integrovať memoist <https://github.com/matthewrudy/memoist>
6. `config/initializers/wrap_parameters.rb` zakomentovať root in json
7. Následne skontrolovať mapovania a dynamické metódy

Príloha G: Návod pre vývojové prostredie a rozbehanie projektu lokálne

PREREQUISITIES:

- Nainštalovaný VirtualBox verzia <= 5.1.28
 - <http://download.virtualbox.org/virtualbox/5.1.28/>
- Nainštalovaný Vagrant (nástroj pre buildovanie a manažovanie prostredia virtuálneho stroja)
 - <https://www.vagrantup.com/downloads.html>
- Návod je pre **Windows Hosta** a **RubyMine 2017.2.4**

Ako na to ?

- spustíme Git bash a vytvoríme si ľubovoľný workspace priečinok na disku, kde budeme mať uložený projekt **crowdcloud**
- nakonfigurujeme si meno, email -> musí byť rovnaký, ako máme primárny na github.com (mail zvolíme primárny, ako máme na gite, aby nás pri každej operácii s remote repo nežiadalo o meno a heslo do githubu)

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

- vygenerujeme si ssh kľúč a pridáme tento kľúč do svojho git účtu podľa návodu <https://help.github.com/articles/connecting-to-github-with-ssh/>
- ak sme vo windowse, nastavíme si credential helper

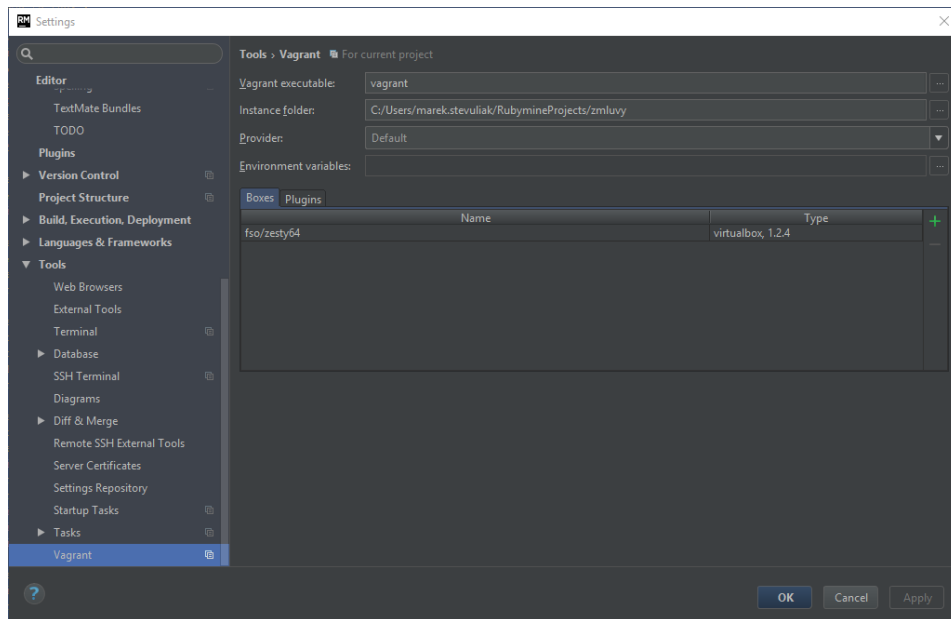
```
git config --global credential.helper manager
```

- keď sme hotoví, naklonujeme si repozitár cez jeden z príkazov (nachádzame sa vo zvolenom workspace)

```
git clone git@github.com:robom/crowdcloud.git
git clone https://github.com/robom/crowdcloud.git // pre windows, kvôli firewallu
```

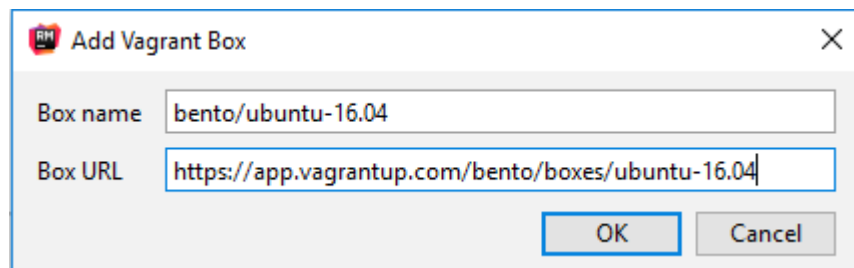
- presunieme sa do vnútra naklonovaného repa **cd crowdcloud**
- zmeníme vetvu na **git checkout dev**
- otvoríme **RubyMine** a zvolíme možnosť **open project**
 - nasmerujeme cestu k naklonovanému **crowdcloud** projektu
 - import projektu do IDE by mal prebehnúť úspešne
- potrebujeme správne nastaviť Vagrant

- Horná lišta : **File -> Settings -> Tools ->Vagrant** v okne boxes vidime dostupné

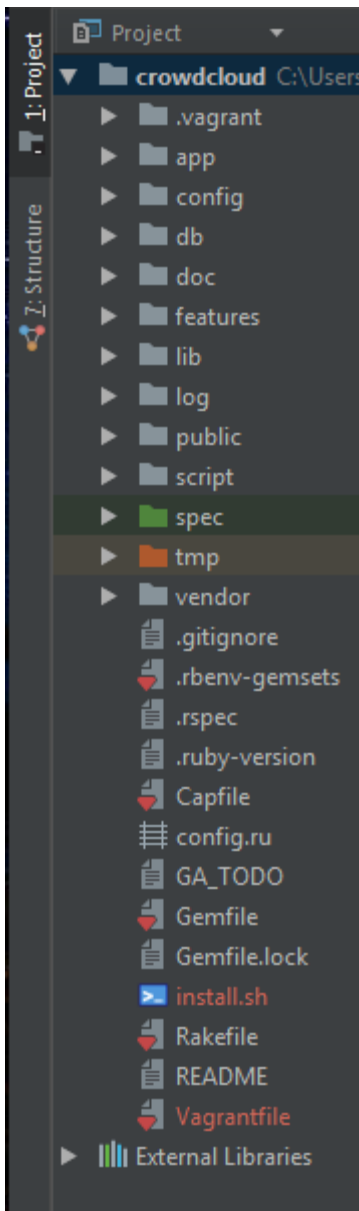


vagrant boxy

- v prípade, že nemáme nainštalovaný **bento/ubuntu-16.04** box, volíme zelené plusko a do dialógového okna zadávame tieto údaje a potvrdíme aktuálne + okno settings tlačidlom OK



- Ak sme hotoví, v hornom menu vygenerujeme **Vagrantfile** pre projekt tak, že zvolíme: **Tools -> Vagrant -> Init in Project root**
 - tento subor sa nám objaví vo foldri projektu, na ľavo v tree si ho vieme všimnúť, otvoríme ho a jeho obsah nahradíme obsahom súboru **Vagrantfile**, ktorý je uložený na tímovom Drive v priečinku | **Zdielane/setup_files**
- Skopírujeme si do priečinka vytvoreného projektu súbor **install.sh** dostupný na Drive v priečinku | **Zdielane/setup_files**



Náš projekt tree by mal vyzerat' nasledovne:

- Po týchto krokoch spustíme virtuálny stroj, horná lišta **Tools -> Vagrant -> Vagrant Up**
- Do stroja z Hosta pristúpime cez **Tools -> Start SSH session -> vyberieme aktualny vagrant pre akt. projekt**
- Po úspešnom pripojení disponujeme konzolou v prostredí Ubuntu 17.04

- Odpalíme skript, ktorý nainštaluje potrebné technológie pre projekt (nezaspať!!! **skript vyžaduje našu interakciu, ak je potrebné niečo potvrdiť, dávame ENTER, ak je potrebné heslo volíme 'vagrant'**)
 - ak sa nam nechce neustále potvrdzovať, pred zavolaním skriptu môžeme

If you want to have these settings permanent, create a file in `/etc/apt/apt.conf.d/`, like `/etc/apt/apt.conf.d/90forceyes` with the following content:

```
APT::Get::Assume-Yes "true";  
APT::Get::force-yes "true";
```

nastaviť forceyes subor

```
./crowdcloud/install.sh
```

- po zbehnutí skriptu refreshneme premenné prostredia a pridáme rovnako do zdrojov nainštalovaného RVM príkazmi

```
source /etc/environment  
source ~/.rvm/scripts/rvm
```

- nastavíme heslo pre postgres

```
sudo -i -u postgres  
psql  
alter user postgres with password 'postgres';  
\q  
ctrl+d
```

- presunieme sa do projektu, ktorý je zároveň zdieľaným priečinkom medzi HOST-GUEST príkazom ***cd crowdcloud/***

- nainštalujeme potrebné gemy

```
bundle install
```

- v projekte nahradíme obsah súboru `/config/database.yml` obsahom súboru s rovnakým názvom dostupnom na Drive | ***Zdielane/setup_files***

pridáme do suboru crowdcloud/db/seeds.rb pomocný objekt, ktorý sa volá pri *index* akcií aplikačného controllera

```
SpaceshipSetting.create!(
  :identifier => 'searchd_activated_flag',
  :label => 'Min points',
  :value => 0
)
```

- mali by sme teraz byť schopní vytvoriť databázu, postupne do konzoly zadávame príkazy

rake db:create

rake db:schema:load

rake db:seed

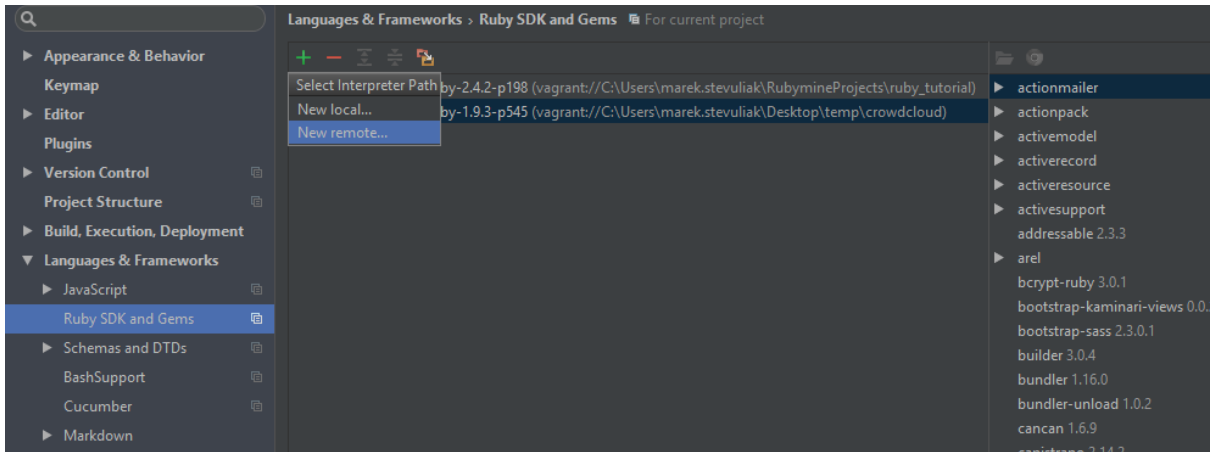
UPDATE for rails 3.2.12:

I just checked the source and the dependencies are like this now:

- **db:create** creates the database for the current env
- **db:create:all** creates the databases for all envs
- **db:drop** drops the database for the current env
- **db:drop:all** drops the databases for all envs
- **db:migrate** runs migrations for the current env that have not run yet
- **db:migrate:up** runs one specific migration
- **db:migrate:down** rolls back one specific migration
- **db:migrate:status** shows current migration status
- **db:rollback** rolls back the last migration
- **db:forward** advances the current schema version to the next one
- **db:seed** (only) runs the db/seed.rb file
- **db:schema:load** loads the schema into the current env's database
- **db:schema:dump** dumps the current env's schema (and seems to create the db as well)
- **db:setup** runs db:schema:load, db:seed
- **db:reset** runs db:drop db:setup
- **db:migrate:redo** runs (db:migrate:down db:migrate:up) or (db:rollback db:migrate) depending on the specified migration
- **db:migrate:reset** runs db:drop db:create db:migrate

`rake db:migrate`

- vytvoríme indexy pre elasticsearch, naindexujeme data
`rake crowdcloud:index:rebuild`
- za predpokladu, že nám beží virtuálka potrebujeme nastaviť **remote ruby sdk**, kvôli spúšťaniu, debugovaniu a inštalácii gemov priamo z IDE
- horná lišta **File -> Settings -> Languages & Frameworks -> RubySDK and**



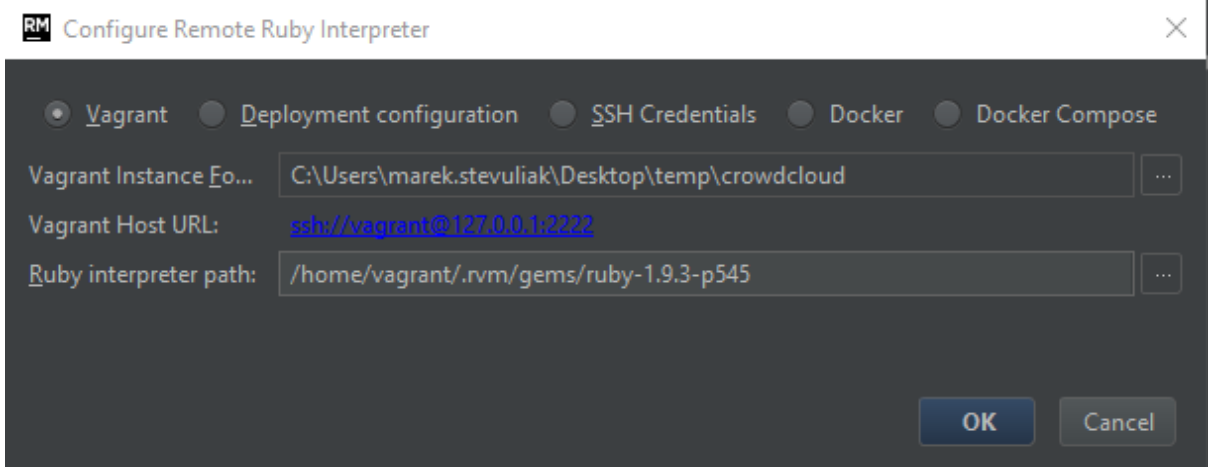
- **db:migrate** runs (single) migrations that have not run yet.
- **db:create** creates the database
- **db:drop** deletes the database
- **db:schema:load** creates tables and columns within the (existing) database following schema.rb
- **db:setup** does db:create, db:schema:load, db:seed
- **db:reset** does db:drop, db:setup

Typically, you would use db:migrate after having made changes to the schema via new migration files (this makes sense only if there is already data in the database). db:schema:load is used when you setup a new instance of your app.

I hope that helps.

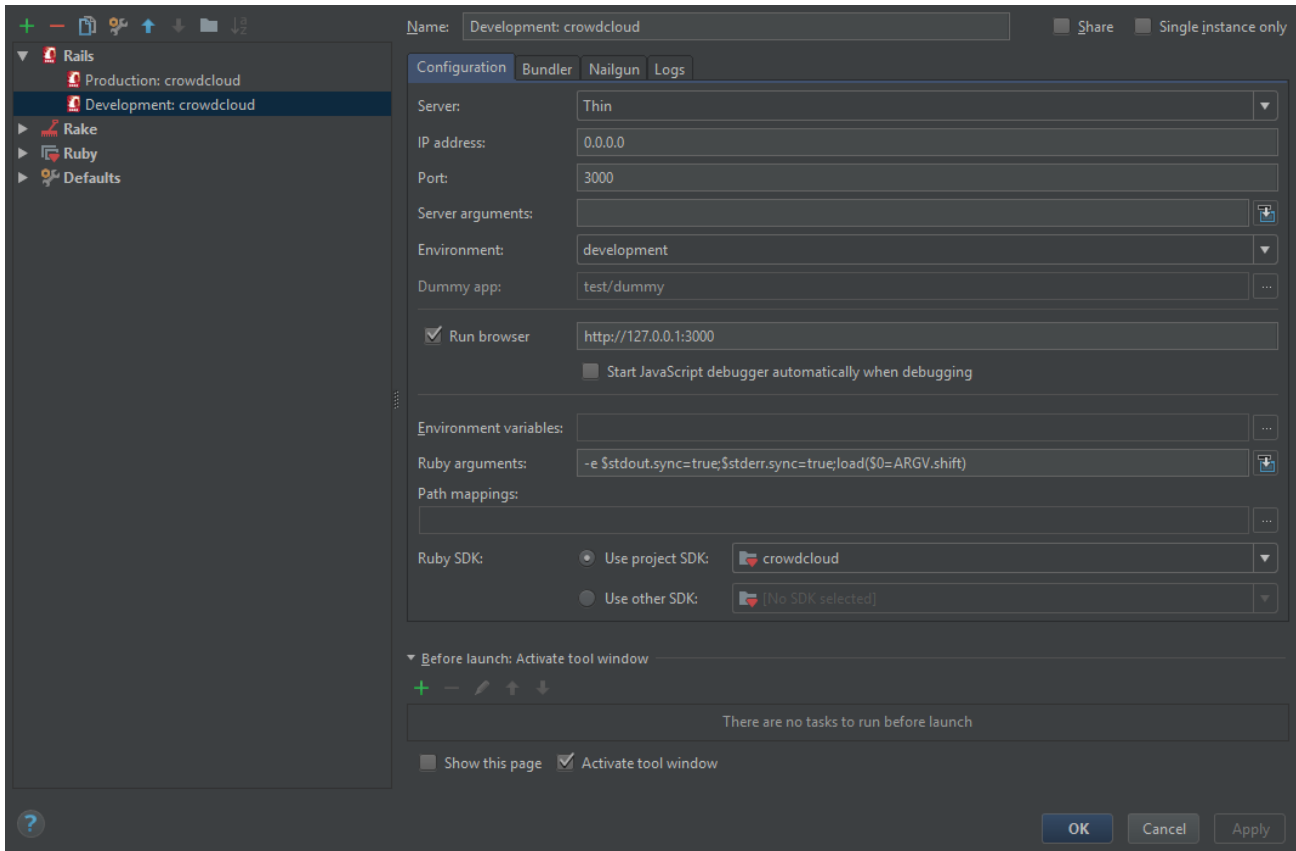
gems

- stlačením zelenej plusky, volíme *new remote*
- konfiguruje nás nasledovne (zaujima nas len radio *vagrant, ruby interpreter path*)



- potvrdíme a počkáme, kým RubyMine nastaví SDK
- radio button na práve nasetovaný SDK a potvrdíme tlačidlom **OK**

- Nakonfigurujeme spúšťanie aplikácie pre development cez **Run-> Settings -> Run configurations** a nastavíme IP adresu, port a adresu pre spúšťanie v prehliadači, ako je uvedené nižšie



- ešte potrebujeme upraviť súbor v projekte `/config/environments/development.rb` (riadok **36 -> debug na false**) za účelom zrýchlenia renderovania layoutu na HOSTOVI

```
# Expands the lines which load the assets
config.assets.debug = false
```

- ostáva nám spustiť projekt, horné menu **Run-> Run**
 - vyberieme *Development: crowdcloud* a mod (run, debug, cover)
 - malo by nám spustiť *Thin* server a automaticky otvoriť prehliadač na adrese <http://127.0.0.1:3000>, kde by mala úspešne bežať aplikácia otvorených zmlúv
- **sme hotoví, máme pripravené lokálne prostredie pre vývoj**

- Statické stránky portálu Otvorené Zmluvy sú uložené v DB. Dump sa nachádza tu <https://drive.google.com/open?id=1ITqPYwHkiTAHWgBai9RmeCoRK2DYVSCO>
- Na načítanie statických stránok do DB lokálne je potrebné spustiť príkaz:
 - `psql -h localhost -d crowdcloud_development -U postgres -f static_pages.sql`
- Namiesto `crowdcloud_development` môžeme použiť aj inú databázu (`development`, `test` alebo `production`)